Package: Rminibwa (via r-universe)

July 22, 2026

Title Native R Bindings and SIMD Dispatch for 'minibwa'

Version 0.4.0-0.0.1.9000

Description Provides native R bindings to the 'minibwa' genomic read aligner with raw-vector query input, external-pointer alignment batches, ALTREP column views, and an installed C API for downstream packages. The package builds staged Single Instruction Multiple Data ('SIMD') KSW backends and selects portable scalar, SSE4, or AVX2 code at runtime without global instruction-set compiler flags.

License GPL (>= 2)

Depends R (>= 4.0.0)

SystemRequirements GNU make, zlib

LinkingTo RsimdDispatch (>= 0.1.2.9001)

Suggests knitr, rmarkdown, roxygen2, Rtinycc, tinytest

VignetteBuilder knitr

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Additional_repositories https://sounkou-bioinfo.r-universe.dev

URL https://github.com/sounkou-bioinfo/Rminibwa, https://sounkou-bioinfo.github.io/Rminibwa/

BugReports https://github.com/sounkou-bioinfo/Rminibwa/issues

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

Date/Publication 2026-07-22 22:58:17 UTC

RemoteUrl https://github.com/RGenomicsETL/Rminibwa

RemoteRef HEAD

RemoteSha a752739320be7c641c46afb28beeca1bc661f460

Contents

mb_align_n . . . 2
mb_fastx_iter . . . 3
mb_index_build . . . 4
mb_index_contigs . . . 5
mb_index_load . . . 5
mb_map . . . 6
mb_map_batch . . . 6
mb_map_pair_batch . . . 7
mb_opts . . . 8
mb_query_stream . . . 8
minibwa_available . . . 10
minibwa_cli . . . 10
minibwa_index . . . 11
minibwa_map . . . 12
minibwa_path . . . 13
minibwa_upstream_info . . . 13
minibwa_version . . . 14
simd_backend . . . 14
simd_info . . . 15
simd_set_backend . . . 15

Index **16**

`mb_align_n` *Inspect native minibwa alignment batches*

Description

`mb_align_col()` returns ALTREP views over native alignment columns. These views do not copy columns unless R forces materialization. `mb_align_read_col()` returns ALTREP views over the read-level sidecar with one row per input read, including reads without hits. C consumers should prefer the installed `Rminibwa.h` API.

Usage

```
mb_align_n(x)

mb_align_read_n(x)

mb_align_col(x, name)

mb_align_read_col(x, name)

mb_align_cigar_words(x)
```

Arguments

x	Native alignment batch returned by mb_map() or mb_map_batch().
name	Column name. Alignment columns include read_id, tid, qlen, qs, qe, tlen, ts, te, strand, mapq, score, matches, blen, flags, n_sub, cigar_offset, and cigar_n. Read-sidecar columns include read_id, length, fragment_id, mate, hit_offset, and hit_count.

Value

mb_align_n() returns the number of alignment records. mb_align_read_n() returns the number of input reads represented in the batch. mb_align_col() and mb_align_read_col() return ALTREP integer or real vectors. mb_align_cigar_words() returns a raw debug copy of packed minibwa CIGAR words.

---

mb_fastx_iter *Iterate FASTA/FASTQ batches without R sequence materialization*

Description

mb_fastx_iter() opens one FASTA/FASTQ file, or two mate files, using the vendored minibwa reader. mb_fastx_next() returns a native FASTX batch, and mb_map_fastx_batch() maps that batch directly from native sequence buffers.

Usage

```
mb_fastx_iter(
  path,
  paired = c("auto", "none", "by_name", "two_file"),
  batch_bases = 100000000L,
  max_batch_bases = batch_bases,
  with_qual = FALSE,
  with_comment = FALSE
)

mb_fastx_next(iter)

mb_fastx_n(batch)

mb_map_fastx_batch(batch, index, opt = mb_opts())
```

Arguments

path	One FASTA/FASTQ path, or two paths for paired = "two_file".
paired	Pairing mode.
batch_bases	Approximate number of sequence bases to read per batch.

max_batch_bases
: Maximum bases to read when extending a batch to keep a short paired fragment together.

with_qual, with_comment
: Preserve qualities/comments in the native batch.

iter
: Native FASTX iterator returned by mb_fastx_iter().

batch
: Native FASTX batch returned by mb_fastx_next().

index
: Index handle returned by mb_index_load().

opt
: Options list from mb_opts().

Details

Pairing modes follow the CLI reader shape: "none" treats each record as a fragment, "by_name" groups adjacent records in one file with matching query names, and "two_file" reads one record from each of two files per fragment. "auto" selects "two_file" for two paths and "none" for one path.

Value

mb_fastx_iter() returns a native iterator. mb_fastx_next() returns a native FASTX batch or NULL at EOF. mb_fastx_n() returns the number of records in a FASTX batch. mb_map_fastx_batch() returns a native alignment batch.

---

mb_index_build *Build a minibwa index with the native library*

Description

Builds index files at prefix using the vendored minibwa C library. The package is GPL, so the native build includes minibwa's GPL-compatible low-memory indexing path; low_memory = FALSE uses the default libsais path.

Usage

```
mb_index_build(fasta, prefix, meth = FALSE, threads = 1L, low_memory = FALSE)
```

Arguments

fasta
: Reference FASTA path. Gzip-compressed input is supported by minibwa/zlib.

prefix
: Output prefix for .l2b, .mbw, and optionally .meth.mbw.

meth
: Build the directional methylation index as well.

threads
: Number of index-build threads requested by minibwa.

low_memory
: Use minibwa's low-memory BWT builder.

Value

prefix, invisibly.

mb_index_contigs *Return minibwa index contigs*

Description

Names are returned as raw byte vectors to avoid forced string materialization.

Usage

```
mb_index_contigs(index)
```

Arguments

index	Index handle returned by mb_index_load().

Value

A list with raw-vector name entries and numeric length values.

mb_index_load *Load a native minibwa index*

Description

Load a native minibwa index

Usage

```
mb_index_load(prefix, meth = FALSE)
```

Arguments

prefix	Index prefix.
meth	Load the methylation BWT index.

Value

An external pointer handle to a native minibwa index.

mb_map — *Map raw sequence bytes with native minibwa*

Description

`mb_map()` is pipe-friendly: the sequence bytes are the first argument. The result is a native alignment batch external pointer. Use `mb_align_col()` for ALTREP column views or pass the batch to downstream C consumers.

Usage

```
mb_map(x, index, opt = mb_opts(), name = NULL, meth = c("none", "c2t", "g2a"))
```

Arguments

`x`	Raw vector of query sequence bytes.
`index`	Index handle returned by `mb_index_load()`.
`opt`	Options list from `mb_opts()`.
`name`	Optional read name as raw bytes. Character names are accepted for convenience but raw bytes are preferred.
`meth`	Methylation code: `"none"`, `"c2t"`, or `"g2a"`.

Value

A native alignment batch external pointer.

Examples

```
# See vignettes for full index-backed examples.
```

mb_map_batch — *Map a batch of query sequences with native minibwa*

Description

`mb_map_batch()` maps many reads in one native call and returns the same columnar alignment-batch object as `mb_map()`. Input may be a character vector of sequence bytes or a list whose elements are raw vectors or character scalars. Unlike the CLI reader, this direct batch API defaults to unpaired mapping unless `opt = mb_opts(paired = TRUE)`. In paired mode, order records as `R1, R2, R1, R2, ...`; this call pairs adjacent records and does not group FASTQ records by name for you.

Usage

```
mb_map_batch(x, index, opt = mb_opts(), name = NULL)
```

Arguments

x	Character vector or list of raw vectors / character scalars.
index	Index handle returned by mb_index_load().
opt	Options list from mb_opts().
name	Optional character vector of read names, or a list of raw vectors / character scalars. If supplied, it must have the same length as x.

Value

A native alignment batch external pointer.

---

mb_map_pair_batch *Map paired query sequence batches with native minibwa*

Description

mb_map_pair_batch() is the explicit two-batch paired-end interface. r1[i] and r2[i] are flattened as adjacent records before mapping, matching the two-file CLI layout. The read sidecar records fragment_id and mate.

Usage

```
mb_map_pair_batch(r1, r2, index, opt = mb_opts(), name1 = NULL, name2 = NULL)
```

Arguments

r1, r2	Character vectors or lists of raw vectors / character scalars. They must have the same non-zero length.
index	Index handle returned by mb_index_load().
opt	Options list from mb_opts().
name1, name2	Optional read-name vectors for r1 and r2.

Value

A native alignment batch external pointer.

mb_opts *Native minibwa options*

Description

Returns a plain list consumed by native mapping functions. Only the selected options are exposed initially; all other fields use minibwa defaults for the chosen preset.

Usage

```
mb_opts(
  preset = c("adap", "sr", "lr"),
  paired = NULL,
  methylation = NULL,
  out_n = NULL,
  min_seed_len = NULL,
  threads = NULL,
  match_score = NULL,
  mismatch_penalty = NULL,
  gap_open = NULL,
  gap_extend = NULL
)
```

Arguments

`preset`	One of `"adap"`, `"sr"`, or `"lr"`.
`paired`, `methylation`	Optional logical overrides for minibwa flags.
`out_n`, `min_seed_len`, `threads`, `match_score`, `mismatch_penalty`, `gap_open`, `gap_extend`	Optional integer overrides for common minibwa fields.

Value

A plain list.

mb_query_stream *Open a lossless native FASTQ query-group stream*

Description

Opens a one-template-at-a-time, bounded-memory native stream for a downstream C/C++/Rcpp BAM producer. Each `mb_query_stream_next()` result is one complete normalized QNAME group and remains valid only until the next call on the stream or cancellation. The normal producer path neither creates SAM text nor calls R once per alignment record; downstream native consumers should use the versioned API in `Rminibwa.h`.

Usage

```
mb_query_stream(
  path,
  index,
  opt = mb_opts(),
  mode,
  threads,
  read_group = NULL,
  max_seen_qnames = 1000000L
)

mb_query_stream_next(stream)

mb_query_stream_cancel(stream)

mb_query_stream_error(stream)

mb_query_group_n_reads(group)

mb_query_group_n_records(group)

mb_query_group_name(group)

mb_query_group_input_order(group)
```

Arguments

path	One FASTQ path for mode = "single", or two ordered mate FASTQ paths for mode = "paired".
index	A native minibwa index returned by mb_index_load().
opt	Mapping options from mb_opts(). Its threads entry is replaced by threads; its paired flag is set from mode.
mode	Explicit "single" or "paired" input mode.
threads	Positive total thread budget owned by the caller.
read_group	NULL or one exact @RG record with a non-empty ID:.
max_seen_qnames	Positive maximum number of normalized QNAMEs tracked for exact repeated-name detection.
stream	A native stream returned by mb_query_stream().
group	A native query group returned by mb_query_stream_next().

Details

mode and threads are deliberately required. threads is the complete caller-owned minibwa thread budget: the stream creates no additional mapping worker pool. read_group is either NULL or an exact literal @RG record containing an ID: tag. It is retained byte-for-byte; library and sample

metadata are never inferred. The native header view declares `SO:unsorted` and `GO:query`: groups are contiguous input templates, not lexicographically query-name sorted.

Exact repeated-QNAME validation uses a caller-bounded name tracker. If the limit is reached, the stream fails with a reason code rather than emitting an ambiguous later group. Consumers must treat any stream error as an incomplete downstream output.

Value

`mb_query_stream()` returns a native query stream. `mb_query_stream_next()` returns a native query group or `NULL` at EOF or after cancellation. Group helper functions expose only debug accounting; use the installed C API for record views.

`minibwa_available` *Test whether the minibwa CLI is available*

Description

Test whether the minibwa CLI is available

Usage

```
minibwa_available(path = NULL)
```

Arguments

`path`	Optional executable name or path. When `NULL`, use Rminibwa's packaged minibwa executable.

Value

`TRUE` when an executable can be resolved, otherwise `FALSE`.

`minibwa_cli` *Run the minibwa CLI*

Description

Low-level helper for invoking arbitrary `minibwa` commands from R. Prefer `minibwa_index()` and `minibwa_map()` for common workflows.

Usage

```
minibwa_cli(args = character(), stdout = TRUE, path = minibwa_path())
```

Arguments

args	Character vector of command-line arguments passed to minibwa.
stdout	Passed to system2(). Use TRUE to capture standard output, FALSE to inherit it, or a file path to redirect it.
path	Executable name or path.

Value

If stdout = TRUE, a character vector of captured output; otherwise an invisible process status.

---

minibwa_index *Build a minibwa index through the CLI*

Description

Build a minibwa index through the CLI

Usage

```
minibwa_index(
  reference,
  prefix = NULL,
  threads = NULL,
  meth = FALSE,
  low_memory = FALSE,
  extra_args = character(),
  path = minibwa_path()
)
```

Arguments

reference	Path to a FASTA reference.
prefix	Optional output prefix. When NULL, minibwa uses reference as the prefix.
threads	Optional number of worker threads.
meth	If TRUE, build a directional bisulfite sequencing index via --meth.
low_memory	If TRUE, pass -l to minibwa index.
extra_args	Additional CLI arguments inserted before positional arguments.
path	Executable name or path.

Value

Invisibly, the expected index prefix.

`minibwa_map` *Map reads through the minibwa CLI*

Description

Map reads through the minibwa CLI

Usage

```
minibwa_map(
  index,
  reads,
  output = NULL,
  format = c("sam", "paf"),
  threads = NULL,
  meth = FALSE,
  hic = FALSE,
  extra_args = character(),
  path = minibwa_path()
)
```

Arguments

`index`	Index prefix produced by `minibwa_index()` or the minibwa CLI.
`reads`	One or more FASTA/FASTQ input paths. The common cases are one single-end file or two paired-end files.
`output`	Optional output file. When `NULL`, standard output is captured and returned as a character vector.
`format`	Output format: `"sam"` by default, or `"paf"` to pass `-f`.
`threads`	Optional number of worker threads.
`meth`	If `TRUE`, pass `--meth`.
`hic`	If `TRUE`, pass `--hic`.
`extra_args`	Additional CLI arguments inserted before positional arguments.
`path`	Executable name or path.

Value

If `output = NULL`, captured alignment records as a character vector; otherwise the output path, invisibly.

minibwa_path *Locate the packaged minibwa executable*

Description

minibwa_path() resolves the command-line executable used by the CLI-backed helpers. By default it returns the executable built from the vendored minibwa sources and installed with Rminibwa. Supplying path is mainly for developer comparisons against another minibwa build.

Usage

```
minibwa_path(path = NULL, must_work = TRUE)
```

Arguments

path	Optional executable name or path. When NULL, use Rminibwa's packaged minibwa executable.
must_work	If TRUE, throw an error when the executable cannot be found.

Value

A scalar character path, or NA_character_ when must_work = FALSE and no executable is found.

minibwa_upstream_info *Report the pinned upstream minibwa source*

Description

Rminibwa pins a specific upstream minibwa commit for native-library work. minibwa_upstream_info() reports the installed copy of that provenance metadata.

Usage

```
minibwa_upstream_info()
```

Value

A named list with fields such as Component, Version, Repository, Commit, and PatchDirectory.

Examples

```
minibwa_upstream_info()[c("Version", "Commit")]
```

minibwa_version *Report the minibwa CLI version*

Description

Report the minibwa CLI version

Usage

```
minibwa_version(path = minibwa_path())
```

Arguments

path	Optional executable name or path. When NULL, use Rminibwa's packaged minibwa executable.

Value

A scalar character version string.

simd_backend *Report the selected SIMD backend*

Description

Report the selected SIMD backend

Usage

```
simd_backend()
```

Value

A character scalar.

simd_info *Report SIMD dispatch diagnostics*

Description

Report SIMD dispatch diagnostics

Usage

```
simd_info()
```

Value

An object of class `rminibwa_simd_info`: a named list with dispatch mode, requested backend, selected backend, compiled backends, CPU-supported backends, available backends, and target.

simd_set_backend *Select the runtime SIMD backend*

Description

Rminibwa compiles staged KSW backend objects and switches among them at runtime. Use `"auto"` for the best available backend, or force one of `"scalar"`, `"sse4"`, or `"avx2"` for diagnostics and benchmarks. The `"scalar"` backend is the portable SIMDe baseline with native intrinsics disabled; compilers may still lower portable vector types efficiently.

Usage

```
simd_set_backend(backend = "auto")
```

Arguments

`backend`	Character scalar backend name.

Value

The selected backend, invisibly.

Index

mb_align_cigar_words *(mb_align_n)*, 2
mb_align_col *(mb_align_n)*, 2
mb_align_n, 2
mb_align_read_col *(mb_align_n)*, 2
mb_align_read_n *(mb_align_n)*, 2
mb_fastx_iter, 3
mb_fastx_n *(mb_fastx_iter)*, 3
mb_fastx_next *(mb_fastx_iter)*, 3
mb_index_build, 4
mb_index_contigs, 5
mb_index_load, 5
mb_index_load(), *9*
mb_map, 6
mb_map_batch, 6
mb_map_fastx_batch *(mb_fastx_iter)*, 3
mb_map_pair_batch, 7
mb_opts, 8
mb_opts(), *9*
mb_query_group_input_order *(mb_query_stream)*, 8
mb_query_group_n_reads *(mb_query_stream)*, 8
mb_query_group_n_records *(mb_query_stream)*, 8
mb_query_group_name *(mb_query_stream)*, 8
mb_query_stream, 8
mb_query_stream_cancel *(mb_query_stream)*, 8
mb_query_stream_error *(mb_query_stream)*, 8
mb_query_stream_next *(mb_query_stream)*, 8
minibwa_available, 10
minibwa_cli, 10
minibwa_index, 11
minibwa_map, 12
minibwa_path, 13
minibwa_upstream_info, 13
minibwa_version, 14
simd_backend, 14
simd_info, 15
simd_set_backend, 15
system2(), *11*