

Package ‘megadepth’

May 2, 2026

Title megadepth: BigWig and BAM related utilities

Version 1.23.0

Date 2021-08-05

Description This package provides an R interface to Megadepth by Christopher Wilks available at <https://github.com/ChristopherWilks/megadepth>. It is particularly useful for computing the coverage of a set of genomic regions across bigWig or BAM files. With this package, you can build base-pair coverage matrices for regions or annotations of your choice from BigWig files. Megadepth was used to create the raw files provided by <https://bioconductor.org/packages/recount3>.

License Artistic-2.0

URL <https://github.com/LieberInstitute/megadepth>

BugReports <https://support.bioconductor.org/t/megadepth>

biocViews Software, Coverage, DataImport, Transcriptomics, RNASeq, Preprocessing

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Suggests covr, knitr, BiocStyle, sessioninfo, rmarkdown, rtracklayer, derfinder, GenomeInfoDb, tools, RefManageR, testthat

SystemRequirements megadepth
(<<https://github.com/ChristopherWilks/megadepth>>)

VignetteBuilder knitr

Imports xfun, utils, fs, GenomicRanges, readr, cmdfun, dplyr, magrittr

Config/testthat/edition 3

git_url <https://git.bioconductor.org/packages/megadepth>

git_branch devel

git_last_commit 73f027d

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-01

Author Leonardo Collado-Torres [aut] (ORCID: <https://orcid.org/0000-0003-2140-308X>),
David Zhang [aut, cre] (ORCID: <https://orcid.org/0000-0003-2382-8460>)
Maintainer David Zhang <david.zhang.12@ucl.ac.uk>

Contents

bam_to_bigwig	2
bam_to_junctions	3
get_coverage	5
install_megadepth	6
megadepth_cmd	7
process_junction_table	8
read_coverage	9
read_junction_table	10

Index	12
--------------	-----------

bam_to_bigwig	<i>Convert a BAM file to a BigWig</i>
---------------	---------------------------------------

Description

Given an input BAM file, convert this to the BigWig format which can then be used in `get_coverage()`.

Usage

```
bam_to_bigwig(
  bam_file,
  prefix = file.path(tempdir(), basename(bam_file)),
  min_unique_qual = FALSE,
  double_count = FALSE,
  overwrite = FALSE
)
```

Arguments

<code>bam_file</code>	A character(1) with the path to the input BAM file.
<code>prefix</code>	A character(1) specifying the output file prefix. This function creates a BigWig file called <code>{prefix}.all.bw</code> . By default, the prefix is the BAM file name and the file is created in the <code>tempdir()</code> and will be deleted after you close your R session.
<code>min_unique_qual</code>	A integer(1) specifying a mapping quality threshold and only bases above this will be used to generate the BigWig. If set to <code>FALSE</code> this argument is not used by <code>Megadepth</code> . Otherwise it will generate files <code>{prefix}.unique.bw</code> and <code>{prefix}.unique.tsv</code> .
<code>double_count</code>	A logical(1) determining whether to count the overlapping ends of paired ends reads twice.
<code>overwrite</code>	A logical(1) specifying whether to overwrite the output file(s), if they exist already.

Details

Note that this functionality is currently not supported on Windows by Megadepth.

Value

A character() with the path to the output files(s).

Examples

```
## Install the latest version if necessary
install_megadepth(force = TRUE)

## Find the example BAM file
example_bam <- system.file("tests", "test.bam",
  package = "megadepth", mustWork = TRUE
)

## Create the BigWig file
## Currently Megadepth does not support this on Windows
if (!xfun::is_windows()) {
  example_bw <- bam_to_bigwig(example_bam, overwrite = TRUE)

  ## Path to the output file(s) generated by bam_to_bigwig()
  example_bw

  ## Use the all.bw file in get_coverage(), first find an annotation file
  annotation_file <- system.file("tests", "testbw2.bed",
    package = "megadepth", mustWork = TRUE
  )

  ## Compute the coverage
  bw_cov <- get_coverage(
    example_bw["all.bw"],
    op = "mean",
    annotation = annotation_file
  )
  bw_cov
}
```

bam_to_junctions

Extract junctions from a BAM file

Description

Given a BAM file, extract junction information including co-ordinates, strand, anchor length for each junction read. For details on the format of the output TSV file, check <https://github.com/ChristopherWilks/megadepth#junctions>.

Usage

```
bam_to_junctions(
  bam_file,
  prefix = file.path(tempdir(), basename(bam_file)),
```

```

    all_junctions = TRUE,
    junctions = FALSE,
    long_reads = FALSE,
    filter_in = 65535,
    filter_out = 260,
    overwrite = FALSE
  )

```

Arguments

bam_file	A character(1) with the path to the input BAM file.
prefix	A character(1) specifying the output file prefix. This function creates a file called <code>prefix.jxs.tsv</code> . By default, the prefix is the BAM file name and the file is created in the <code>tempdir()</code> and will be deleted after you close your R session.
all_junctions	A logical(1) indicating whether to obtain all junctions.
junctions	A logical(1) indicating whether to obtain co-occurring jx coordinates.
long_reads	A logical(1) indicating whether to increase the buffer size to accommodate for long-read RNA-sequencing.
filter_in	A integer(1) used to filter in read alignments. See https://github.com/ChristopherWilks/megadepth#processing and https://samtools.github.io/hts-specs/SAMv1.pdf for further documentation on how to apply this parameter.
filter_out	A integer(1) used to filter out read alignments. See https://github.com/ChristopherWilks/megadepth#processing and https://samtools.github.io/hts-specs/SAMv1.pdf for further documentation on how to apply this parameter.
overwrite	A logical(1) specifying whether to overwrite the output file(s), if they exist already.

Value

A character(1) with the path to the output junction tsv file.

Examples

```

## Install if necessary
install_megadepth()

## Find the example BAM file
example_bam <- system.file("tests", "test.bam",
  package = "megadepth", mustWork = TRUE
)

## Run bam_to_junctions()
example_jxs <- bam_to_junctions(example_bam, overwrite = TRUE)

## Path to the output file generated by bam_to_junctions()
example_jxs

```

`get_coverage`*Compute coverage summarizations across a set of regions*

Description

Given an input set of annotation regions, compute coverage summarizations using Megadepth for a given BigWig file.

Usage

```
get_coverage(  
  bigwig_file,  
  op = c("sum", "mean", "max", "min"),  
  annotation,  
  prefix = file.path(tempdir(), "bw.mean")  
)
```

Arguments

<code>bigwig_file</code>	A character(1) with the path to the input BigWig file.
<code>op</code>	A character(1) specifying the summarization operation to perform.
<code>annotation</code>	A character(1) path to a BED file with the genomic coordinates you are interested in.
<code>prefix</code>	A character(1) specifying the output file prefix. This function creates a file called <code>prefix.annotation.tsv</code> that can be read again later with <code>read_coverage()</code> . By default the file is created in the <code>tempdir()</code> and will be deleted after you close your R session.

Details

Note that the chromosome names (seqnames) in the BigWig file and the annotation file should use the same format. Otherwise, Megadepth will return 0 counts.

Value

A [GRanges-class](#) object with the coverage summarization across the annotation ranges.

See Also

Other Coverage functions: [read_coverage\(\)](#)

Examples

```
## Install if necessary  
install_megadepth()  
  
## Next, we locate the example BigWig and annotation files  
example_bw <- system.file("tests", "test.bam.all.bw",  
  package = "megadepth", mustWork = TRUE  
)  
annotation_file <- system.file("tests", "testbw2.bed",  
  package = "megadepth", mustWork = TRUE
```

```

)

## Compute the coverage
bw_cov <- get_coverage(example_bw, op = "mean", annotation = annotation_file)
bw_cov

## If you want to cast this into a RleList object use the following code:
## (it's equivalent to rtracklayer::import.bw(as = "RleList"))
## although in the megadepth case the data has been summarized
GenomicRanges::coverage(bw_cov)

## Checking with derfinder and rtracklayer
bed <- rtracklayer::import(annotation_file)

## The file needs a name
names(example_bw) <- "example"

## Read in the base-pair coverage data
if (!xfun::is_windows()) {
  regionCov <- derfinder::getRegionCoverage(
    regions = bed,
    files = example_bw,
    verbose = FALSE
  )

  ## Summarize the base-pair coverage data.
  ## Note that we have to round the mean to make them comparable.
  testthat::expect_equivalent(
    round(sapply(regionCov[c(1, 3:4, 2)], function(x) mean(x$value)), 2),
    bw_cov$score,
  )

  ## If we compute the sum, there's no need to round
  testthat::expect_equivalent(
    sapply(regionCov[c(1, 3:4, 2)], function(x) sum(x$value)),
    get_coverage(example_bw, op = "sum", annotation = annotation_file)$score,
  )
}

```

install_megadepth *Install Megadepth*

Description

Download the appropriate Megadepth executable for your platform from Github and try to copy it to a system directory so **megadepth** can run the megadepth command.

Usage

```
install_megadepth(version = "latest", force = FALSE)
```

Arguments

version A character()' specifying the Megadepth version number, e.g., 1.0.4; the special value latest means the latest version (fetched from Github releases).

`force` A logical(1) specifying whether to install megadepth even if it has already been installed.

Details

If this function is run in a non-interactive session such as R CMD Check, it will install Megadepth to `tempdir()`. If this function is run interactively, the user will be prompted to agree to allow Megadepth to be installed at `Sys.getenv('APPDATA')` on Windows, `~/Library/Application Support` on macOS, and `~/bin/` on other platforms (such as Linux). If these directories are not writable, the package directory `'Megadepth'` of **megadepth** will be used. If it still fails, you have to install Megadepth by yourself and make sure it can be found via the environment variable `PATH`.

If you want to install Megadepth to a custom path, you can set the global option `megadepth.dir` to a directory to store the Megadepth executable before you call `install_megadepth()`, e.g., `options(megadepth.hugo.dir = '~/Downloads/Megadepth_1.0.4/')`. This may be useful for you to use a specific version of Megadepth for a specific project. You can set this option per project, similar to how `blogdown.hugo.dir` is used for specifying the directory for Hugo in the **blogdown** package.. See [Section 1.4 Global options](#) for details, or store a copy of Megadepth on a USB Flash drive along with your project code.

Value

Returns NULL. The main use is to install Megadepth.

References

This function is based on `blogdown::install_hugo()` which is available from <https://github.com/rstudio/blogdown/blob/master/R/install.R>.

Examples

```
## Install megadepth
install_megadepth()
```

<code>megadepth_cmd</code>	<i>Run Megadepth commands</i>
----------------------------	-------------------------------

Description

Wrapper functions to run Megadepth commands via `system2('megadepth', ...)`.

Usage

```
megadepth_cmd(...)

megadepth_shell(input = ".", ...)
```

Arguments

`...` Arguments to be passed to `system2('megadepth', ...)`, e.g. `annotation(path)` is basically `megadepth_cmd(c('--annotation', path))` (i.e. run the command `megadepth --annotation path`).

`input` A character(1) with the path to the input BAM, BigWig or text file for Megadepth.

Value

See `base::system2()` for the types of output you can generate.

A `character()` with the capture of the standard output stream generated by Megadepth.

Functions

- `megadepth_cmd()`: Run an arbitrary Megadepth command.
- `megadepth_shell()`: Run an arbitrary Megadepth command.

References

`megadepth_cmd()` is based on `blogdown::hugo_cmd()` which is available at <https://github.com/rstudio/blogdown/blob/master/R/hugo.R>.

`megadepth_shell()` is based on the `shell_ls()` example from `cmdfun` which is available at <https://snystrom.github.io/cmdfun/index.html>.

Examples

```
## Install if necessary
install_megadepth()

## Find version
## megadepth_shell() provides an interface more familiar to R users
megadepth_shell(version = TRUE)
## megadepth_cmd() requires using directly the command line syntax for
## Megadepth
megadepth_cmd("--version", stdout = TRUE)

## Compare the help files:
# megadepth_shell() captures the standard output and returns a character()
# megadepth_cmd() shows the standard output on the console
megadepth_shell("--help")
megadepth_cmd("--help")
```

process_junction_table

Process junctions into a STAR compatible format

Description

Parses the junctions outputted from `process_junction_table()` into an STAR compatible format (SJ.out) for more convenient use in downstream analyses. The columns `strand`, `intron_motif` and `annotated` will always be 0 (undefined) but can be derived through extracting the dinucleotide motifs for the given reference coordinates for canonical motifs. This function is an R-implementation of the Megadepth helper script, on which further details of column definitions can be found: <https://github.com/ChristopherWilks/megadepth#junctions>.

Usage

```
process_junction_table(all_jxs)
```

Arguments

`all_jxs` A `tibble::tibble()` containing junction data ("all.jxs.tsv") generated by `bam_to_junctions(all_jxs = TRUE)` and imported through `megadepth::read_junction_table()`.

Value

Processed junctions in a STAR-compatible format.

Examples

```
## Install if necessary
install_megadepth()

## Find the example BAM file
example_bam <- system.file("tests", "test.bam",
  package = "megadepth", mustWork = TRUE
)

## Run bam_to_junctions()
example_jxs <- bam_to_junctions(example_bam, overwrite = TRUE)

## Read the junctions in as a tibble
all_jxs <- read_junction_table(example_jxs[["all_jxs.tsv"]])

## Process junctions into a STAR-compatible format
processed_jxs <- process_junction_table(all_jxs)

processed_jxs
```

`read_coverage` *Read a coverage TSV file created by Megadepth*

Description

Read an `*annotation.tsv` file created by `get_coverage()` or manually by the user using Megadepth.

Usage

```
read_coverage(tsv_file, verbose = TRUE)

read_coverage_table(tsv_file)
```

Arguments

`tsv_file` A `character(1)` specifying the path to the tab-separated (TSV) file created manually using `megadepth_shell()` or on a previous `get_coverage()` run.

`verbose` A `logical(1)` controlling whether to suppress messages when reading the data.

Value

A [GRanges-class](#) object with the coverage summarization across the annotation ranges.

A `tibble::tibble()` with columns `chr`, `start`, `end` and `score`.

Functions

- `read_coverage_table()`: Read a coverage TSV file created by Megadepth as a table

See Also

Other Coverage functions: [get_coverage\(\)](#)

Examples

```
## Install if necessary
install_megadepth()

## Locate example BigWig and annotation files
example_bw <- system.file("tests", "test.bam.all.bw",
  package = "megadepth", mustWork = TRUE
)
annotation_file <- system.file("tests", "testbw2.bed",
  package = "megadepth", mustWork = TRUE
)

## Compute the coverage
bw_cov <- get_coverage(example_bw, op = "mean", annotation = annotation_file)
bw_cov

## Read in the coverage file again, using read_coverage()
## First, lets locate the tsv file that was generated by get_coverage()
tsv_file <- file.path(tempdir(), "bw.mean.annotation.tsv")
bw_cov_manual <- read_coverage(tsv_file)
stopifnot(identical(bw_cov, bw_cov_manual))

## To get an RleList object, just like the one you would get
## from using rtracklayer::import.bw(as = "RleList") directly on the
## BigWig file, use:
GenomicRanges::coverage(bw_cov_manual)

## The coverage data can also be read as a `tibble::tibble()`
read_coverage_table(tsv_file)
```

`read_junction_table` *Read a junction TSV file created by Megadepth as a table*

Description

Read an `*all_jxs.tsv` or `*jxs.tsv` file created by `bam_to_junctions()` or manually by the user using Megadepth. The rows of a `*jxs.tsv` can have either 7 or 14 columns, which can lead to warnings when reading in - these are safe to ignore. For details on the format of the input TSV file, check <https://github.com/ChristopherWilks/megadepth#junctions>.

Usage

```
read_junction_table(tsv_file)
```

Arguments

`tsv_file` A character(1) specifying the path to the tab-separated (TSV) file created manually using `megadepth_shell()` or on a previous `bam_to_junctions()` run.

Value

A `tibble::tibble()` with the junction data that follows the format specified at <https://github.com/ChristopherWilks/megadepth#junctions>.

Examples

```
## Install if necessary
install_megadepth()

## Find the example BAM file
example_bam <- system.file("tests", "test.bam",
  package = "megadepth", mustWork = TRUE
)

## Run bam_to_junctions()
example_jxs <- bam_to_junctions(example_bam, overwrite = TRUE)

## Read the junctions in as a tibble
all_jxs <- read_junction_table(example_jxs[["all_jxs.tsv"]])

all_jxs
```

Index

* Coverage functions

get_coverage, [5](#)
read_coverage, [9](#)

bam_to_bigwig, [2](#)
bam_to_junctions, [3](#)

get_coverage, [5](#), [10](#)
GRanges-class, [5](#), [9](#)

install_megadepth, [6](#)

megadepth_cmd, [7](#)
megadepth_shell (megadepth_cmd), [7](#)

process_junction_table, [8](#)

read_coverage, [5](#), [9](#)
read_coverage_table (read_coverage), [9](#)
read_junction_table, [10](#)

system2, [7](#)