

# Package ‘fRagmentomics’

October 13, 2025

**Title** Extract Fragmentomics Features and Mutational Status

**Version** 0.99.0

**Description** A user-friendly R package that enables the characterization of each cfDNA fragment overlapping one or multiple mutations of interest, starting from a sequencing file containing aligned reads (BAM file). fRagmentomics supports multiple mutation input formats (e.g., VCF, TSV, or string ``chr:pos:ref:alt" representation), accommodates one-based and zero-based genomic conventions, handles mutation representation ambiguities, and accepts any reference file and species in FASTA format. For each cfDNA fragment, fRagmentomics outputs its size, its 3' and 5' sequences, and its mutational status.

**URL** <https://github.com/ElsaB-Lab/fRagmentomics>

**BugReports** <https://github.com/ElsaB-Lab/fRagmentomics/issues>

**License** GPL (>= 3)

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**biocViews** Software,  
Genetics,  
VariantDetection,  
IndelDetection,  
Sequencing,  
DNASeq,  
Alignment,  
MultipleSequenceAlignment

**Suggests** ragg,  
covr,  
testthat (>= 3.0.0),  
knitr,  
rmarkdown,  
BiocStyle

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Imports** Biostrings,  
dplyr,

future,  
future.apply,  
GenomeInfoDb,  
GenomicRanges,  
ggh4x,  
ggplot2,  
ggseqlogo,  
IRanges,  
magrittr,  
progressr,  
purrr,  
RColorBrewer,  
readr,  
Rsamtools,  
scales,  
stringr,  
tibble,  
tidyr

**SystemRequirements** bcftools ( $\geq 1.21$ )

**Contents**

plot_freq_barplot . . . . .	2
plot_motif_barplot . . . . .	3
plot_qqseqlogo_meme . . . . .	6
plot_size_distribution . . . . .	7
run_fRagmentomics . . . . .	10

<b>Index</b>	<b>13</b>
--------------	-----------

---

plot_freq_barplot	<i>Plot Overall Nucleotide Frequency</i>
-------------------	--

---

**Description**

Creates a bar plot to compare the overall proportion of each nucleotide (A/C/G/T; optional 'Other') in the end motifs. Error bars show 95% confidence intervals.

**Usage**

```
plot_freq_barplot(  
  df_fragments,  
  end_motif_5p = "Fragment_Bases_5p",  
  end_motif_3p = "Fragment_Bases_3p",  
  motif_type = "Both",  
  motif_size = 3,  
  col_z = "Fragment_Status_Simple",  
  vals_z = NULL,  
  ...,  
  colors_z = "Dark2",  
  title = NULL,
```

```

    output_path = NA_character_,
    ggsave_params = list(width = 14, height = 5, units = "in", dpi = 300, bg = "white"),
    show_pvalue = FALSE,
    drop_non_acgt = TRUE
  )

```

### Arguments

df_fragments	The input data frame containing fragment sequence data.
end_motif_5p	Character string. Column name for 5' end sequences.
end_motif_3p	Character string. Column name for 3' end sequences.
motif_type	Character string. Which ends to analyze: 'Start', 'End', or 'Both'.
motif_size	A single integer ( $\geq 1$ ) specifying the k-mer length to analyze.
col_z	Character string or 'NULL'. Column name for grouping. If 'NULL', all fragments are pooled.
vals_z	A character vector of group names from 'col_z' to include. If 'NULL', all unique groups in 'col_z' are used.
...	Additional aesthetics/arguments passed to <code>ggplot2::geom_col()</code> and <code>ggplot2::geom_errorbar()</code> (e.g., 'alpha', 'position', or 'width').
colors_z	A character vector of colors for the groups, or a single RColorBrewer palette name (e.g., 'Set2'). Named vectors are aligned to 'vals_z'.
title	Character or 'NA'. Plot title. If 'NULL', 'NA', or empty, a default title is used.
output_path	Character or 'NA'. If provided and non-empty, the plot is saved to this path.
ggsave_params	A named list of arguments passed to <code>ggplot2::ggsave()</code> .
show_pvalue	Logical. If 'TRUE' and there are at least two groups, append a global Chi-squared p-value to the caption.
drop_non_acgt	Logical. If 'FALSE', characters other than A/C/G/T are tallied into an 'Other' category.

### Value

A 'ggplot' object. If 'output\_path' is provided and non-empty, the plot is saved to file and the function returns 'invisible(NULL)'.

---

plot_motif_barplot	<i>Plot 3-base motif proportions with various representations</i>
--------------------	---

---

### Description

Creates a bar plot to visualize the proportion of 3-base motifs at fragment ends. Supports grouped analysis and three different visual representations: hierarchical faceting by base, log2 fold change, or side-by-side motifs.

**Usage**

```
plot_motif_barplot(
  df_fragments,
  end_motif_5p = "Fragment_Bases_5p",
  end_motif_3p = "Fragment_Bases_3p",
  motif_type = "Both",
  motif_start = NULL,
  col_z = "Fragment_Status_Simple",
  vals_z = NULL,
  representation = "split_by_base",
  ...,
  colors_z = NULL,
  title = NULL,
  output_path = NA_character_,
  ggsave_params = list(width = 10, height = 6, units = "in", dpi = 300, bg = "white")
)
```

**Arguments**

<code>df_fragments</code>	The input dataframe containing fragment sequence data.
<code>end_motif_5p</code>	Character string. Column name for 5' end sequences.
<code>end_motif_3p</code>	Character string. Column name for 3' end sequences.
<code>motif_type</code>	Character string. Which ends to analyze: 'Start', 'End', or 'Both'.
<code>motif_start</code>	Optional character vector ('A','C','G','T') to filter motifs by their starting base.
<code>col_z</code>	Character string. Column name for grouping. If NULL, no grouping is applied.
<code>vals_z</code>	A character vector of group names from 'col_z' to include. If NULL, all unique groups in 'col_z' are used.
<code>representation</code>	Character string. The type of plot to generate. <ul style="list-style-type: none"> <li>• 'split_by_base' (default): A proportion plot with hierarchical axes, splitting motifs by each base position into facets.</li> <li>• 'differential': A log2 fold change plot comparing two groups.</li> <li>• 'split_by_motif': A proportion plot with motifs on the x-axis, with bars for different groups placed side-by-side.</li> </ul>
<code>...</code>	Additional arguments passed on to 'ggplot2::geom_bar()'.
<code>colors_z</code>	Colors for the representation: <ul style="list-style-type: none"> <li>• For 'split_by_base': 4 colors for A/C/G/T, or a single RColorBrewer palette name.</li> <li>• For 'differential': 2 colors for 'Positive'/'Negative' (named vector or palette).</li> <li>• For 'split_by_motif': colors per group (palette, unnamed vector, or a vector named by group names).</li> </ul>
<code>title</code>	Character or NA. Plot title; if NULL/NA/'NA'/empty, a default title is used.
<code>output_path</code>	Character or NA. If provided and non-empty, the plot is saved to this file.
<code>ggsave_params</code>	A named list of arguments passed to 'ggplot2::ggsave()'.

**Value**

A ggplot object.

**Examples**

```
## --- Create a dataset for demonstration ---
set.seed(42)

# Helper function to generate random DNA sequences with a bias
generate_biased_dna <- function(n_seq, len, prob) {
  bases <- c('A', 'C', 'G', 'T')
  replicate(n_seq, paste(sample(bases, len, replace = TRUE, prob = prob), collapse = ''))
}

# Create 50 'MUT' fragments with a high proportion of motifs starting with 'C'
df_mut <- data.frame(
  Fragment_Bases_5p = generate_biased_dna(50, 10, prob = c(0.2, 0.5, 0.15, 0.15)),
  Fragment_Bases_3p = generate_biased_dna(50, 10, prob = c(0.2, 0.5, 0.15, 0.15)),
  Fragment_Status_Simple = 'MUT'
)

# Create 50 'WT' fragments with a high proportion of motifs starting with 'G'
df_wt <- data.frame(
  Fragment_Bases_5p = generate_biased_dna(50, 10, prob = c(0.15, 0.15, 0.5, 0.2)),
  Fragment_Bases_3p = generate_biased_dna(50, 10, prob = c(0.15, 0.15, 0.5, 0.2)),
  Fragment_Status_Simple = 'WT'
)

# Combine into a single dataframe
example_df <- rbind(df_mut, df_wt)

## --- Function Calls for Each Representation ---

# 1. Hierarchical Plot (representation = 'split_by_base')
# This is the default. It creates nested facets for each base position.
p1 <- plot_motif_barplot(
  df_fragments = example_df,
  representation = 'split_by_base'
)
print(p1)

# You can also filter this plot to show only motifs starting with certain bases.
p1_filtered <- plot_motif_barplot(
  df_fragments = example_df,
  representation = 'split_by_base',
  motif_start = c('C', 'G'),
  title = 'Motifs starting with C/G'
)
print(p1_filtered)

# Optional: customize colors for the 2nd base (A/C/G/T) in split_by_base
p1_colors <- plot_motif_barplot(
  df_fragments = example_df,
  representation = 'split_by_base',
  colors_z = c(A = '#FD96A9', C = '#E88B00', G = '#0D539E', T = '#6CAE75')
)
print(p1_colors)

# 2. Differential Plot (representation = 'differential')
# Shows log2 fold-change in motif proportions between two groups (needs exactly two groups).
```

```

p2 <- plot_motif_barplot(
  df_fragments = example_df,
  representation = 'differential',
  vals_z = c('MUT', 'WT'),
  colors_z = c(Positive = '#66C2A5', Negative = '#E78AC3'),
  title = 'MUT vs WT (log2FC)'
)
print(p2)

# 3. Side-by-side Motif Plot (representation = 'split_by_motif')
# Motifs on the x-axis; bars for each group shown side-by-side.
p3 <- plot_motif_barplot(
  df_fragments = example_df,
  representation = 'split_by_motif',
  colors_z = 'Set2' # or a vector named by group names
)
print(p3)

# 4. Save the default hierarchical plot (commented for CRAN)
# out_file1 <- file.path(tempdir(), 'motif_split_by_base.png')
# plot_motif_barplot(
#   df_fragments = example_df,
#   representation = 'split_by_base',
#   title = 'Motif proportions (hierarchical)',
#   output_path = out_file1,
#   ggsave_params = list(width = 8, height = 6, units = 'in', dpi = 300, bg = 'white')
# )

# 5. Save the differential plot with custom dimensions (commented for CRAN)
# out_file2 <- file.path(tempdir(), 'motif_differential.png')
# plot_motif_barplot(
#   df_fragments = example_df,
#   representation = 'differential',
#   vals_z = c('MUT', 'WT'),
#   title = 'Differential motif usage',
#   output_path = out_file2,
#   ggsave_params = list(width = 12, height = 8, units = 'in', dpi = 300, bg = 'white')
# )

```

---

plot\_qqseqlogo\_meme      *Plot sequence motif composition*

---

## Description

Creates a sequence logo plot showing the proportion of each nucleotide at each position, with flexible grouping/faceting.

## Usage

```

plot_qqseqlogo_meme(
  df_fragments,
  end_motif_5p = "Fragment_Bases_5p",
  end_motif_3p = "Fragment_Bases_3p",
  motif_type = "Both",

```

```

motif_size = 3,
col_z = "Fragment_Status_Simple",
vals_z = NULL,
colors_z = NULL,
title = NULL,
output_path = NA_character_,
ggsave_params = list(width = 12, height = 6, units = "in", dpi = 300, bg = "white"),
...
)

```

### Arguments

df_fragments	Data frame containing fragment sequence data.
end_motif_5p	Character. Column name for 5' end sequences.
end_motif_3p	Character. Column name for 3' end sequences.
motif_type	Character. One of 'Start', 'End', or 'Both'.
motif_size	Integer ( $\geq 1$ ). Length of the motif to analyze.
col_z	Character or NULL. Grouping/faceting column. If NULL, all fragments are pooled.
vals_z	Character vector or NULL. Subset of groups from 'col_z' to include. If NULL, all unique groups are used.
colors_z	NULL (use ggseqlogo defaults), a single RColorBrewer palette name (e.g., 'Dark2'), or a named vector for 'A/C/G/T', e.g. 'c(A='#1B9E77', C='#D95F02', G='#7570B3', T='#E7298A')'.
title	Character or NA. Plot title; if NULL/NA/'NA'/empty, a default title is used.
output_path	Character or NA. If provided and non-empty, the plot is saved to this file.
ggsave_params	Named list passed to <code>ggplot2::ggsave()</code> .
...	Extra arguments forwarded to <code>ggseqlogo::ggseqlogo()</code> (e.g., 'stack_width', 'font', or 'col_scheme').

### Value

A 'ggplot' object (invisibly NULL if saved).

---

```
plot_size_distribution
```

*Plot Fragment Size Distribution*

---

### Description

Generates a plot visualizing the distribution of fragment lengths. Supports grouping by a categorical variable and can represent the distribution as a histogram, a density plot, or an overlay of both. The legend displays the sample size (N) per group. Groups with fewer than 2 observations are automatically removed when `show_density = TRUE`.

**Usage**

```
plot_size_distribution(
  df_fragments,
  size_col = "Fragment_Size",
  col_z = "Fragment_Status_Simple",
  vals_z = NULL,
  histo_args = list(),
  density_args = list(),
  colors_z = NULL,
  show_histogram = FALSE,
  show_density = TRUE,
  x_limits = c(0, 750),
  histogram_binwidth = 5,
  show_nuc_peaks = TRUE,
  title = NULL,
  output_path = NA_character_,
  ggsave_params = list(width = 10, height = 7, units = "in", dpi = 300, bg = "white")
)
```

**Arguments**

<code>df_fragments</code>	The input dataframe containing the fragment data.
<code>size_col</code>	A character string specifying the name of the numeric column that contains the fragment lengths.
<code>col_z</code>	A character string specifying the name of the column to use for grouping the data. If <code>NULL</code> , no grouping is applied.
<code>vals_z</code>	An optional character vector to filter and display only specific groups from <code>col_z</code> . If <code>NULL</code> , all groups are used.
<code>histo_args</code>	A named list of additional arguments passed to <code>ggplot2::geom_histogram()</code> .
<code>density_args</code>	A named list of additional arguments passed to <code>ggplot2::geom_density()</code> .
<code>colors_z</code>	A character vector of colors for the groups (named or unnamed), or a single string naming an <code>RColorBrewer</code> palette.
<code>show_histogram</code>	Logical. If <code>TRUE</code> , a histogram layer is added.
<code>show_density</code>	Logical. If <code>TRUE</code> , a density plot layer is added.
<code>x_limits</code>	Optional numeric vector of length 2 to set the x-axis limits (e.g., <code>c(0, 700)</code> ).
<code>histogram_binwidth</code>	Numeric value specifying the bin width for the histogram.
<code>show_nuc_peaks</code>	Logical. If <code>TRUE</code> , adds dashed vertical lines for nucleosome peaks (mono/di/tri).
<code>title</code>	Character or <code>NA</code> . Plot title; if <code>NULL/NA/''NA''/empty</code> , a default title is used.
<code>output_path</code>	Character or <code>NA</code> . If provided and non-empty, the plot is saved to this file.
<code>ggsave_params</code>	A named list of arguments passed to <code>ggplot2::ggsave()</code> .

**Value**

A `ggplot` object representing the size distribution plot (invisibly `NULL` if saved).



**Examples**

```
## --- Create a dataset for demonstration ---
set.seed(42)

# Generate fragment sizes for two groups with different distributions
# 'MUT' group: N=100, shorter fragments
mut_sizes <- rnorm(100, mean = 150, sd = 20)

# 'WT' group: N=150, centered around the mononucleosome peak
wt_sizes <- rnorm(150, mean = 170, sd = 25)

# Add some larger, dinucleosomal fragments to both groups
di_nuc_sizes <- rnorm(30, mean = 330, sd = 30)

# Combine into a single dataframe
example_df_size <- data.frame(
  Fragment_Size = c(mut_sizes, wt_sizes, di_nuc_sizes),
  Fragment_Status_Simple = c(
    rep('MUT', 100),
    rep('WT', 150),
    sample(c('MUT', 'WT'), 30, replace = TRUE)
  )
)

# Ensure all fragment sizes are positive
example_df_size <- example_df_size[example_df_size$Fragment_Size > 0, ]

## --- Plotting Examples ---

# 1) Default plot: grouped density with nucleosome peaks.
p1 <- plot_size_distribution(example_df_size)
print(p1)

# 2) Histogram only: add transparency so overlapping bars are visible.
p2 <- plot_size_distribution(
  df_fragments = example_df_size,
  show_histogram = TRUE,
  show_density = FALSE,
  histo_args = list(alpha = 0.6)
)
print(p2)

# 3) Combined: overlay density curves and histograms.
p3 <- plot_size_distribution(
  df_fragments = example_df_size,
  show_histogram = TRUE,
  show_density = TRUE,
  histo_args = list(alpha = 0.4)
)
print(p3)

# 4) Ungrouped + zoomed x-axis + no nucleosome peaks.
p4 <- plot_size_distribution(
  df_fragments = example_df_size,
  col_z = NULL,
  x_limits = c(50, 400),
  show_nuc_peaks = FALSE,

```

```

    title          = 'All fragments (zoomed)'
  )
  print(p4)

# 5) Custom colors using an RColorBrewer palette.
p5 <- plot_size_distribution(
  df_fragments = example_df_size,
  colors_z     = 'Set2',
  title        = 'Fragment size (Set2 palette)'
)
print(p5)

# 6) Save to file (commented for CRAN):
# out_png <- file.path(tempdir(), 'size_distribution.png')
# plot_size_distribution(
#   df_fragments = example_df_size,
#   output_path  = out_png,
#   ggsave_params = list(width = 8, height = 6, units = 'in', dpi = 300, bg = 'white'),
#   title        = 'Size distribution (saved)'
# )

```

---

run_fRagmentomics	<i>Analyze fragments</i>
-------------------	--------------------------

---

## Description

This is the main function of the package. It provides an end-to-end pipeline for analyzing the allelic state of individual DNA fragments covering specific genomic variants. It takes a list of mutations and an aligned sequencing file (BAM) as input, processes each fragment in parallel, and returns a detailed data frame of results.

## Usage

```

run_fRagmentomics(
  mut,
  bam,
  fasta,
  sample_id = NA_character_,
  neg_offset_mate_search = -1000,
  pos_offset_mate_search = 1000,
  one_based = TRUE,
  flag_bam_list = list(isPaired = TRUE, isProperPair = NA, isUnmappedQuery = FALSE,
    hasUnmappedMate = NA, isMinusStrand = NA, isMateMinusStrand = NA, isFirstMateRead =
    NA, isSecondMateRead = NA, isSecondaryAlignment = FALSE, isSupplementaryAlignment =
    FALSE, isNotPassingQualityControls = NA, isDuplicate = FALSE),
  report_tlen = FALSE,
  report_softclip = FALSE,
  report_5p_3p_bases_fragment = 5,
  remove_softclip = FALSE,
  retain_fail_qc = FALSE,
  apply_bcftools_norm = FALSE,
  tmp_folder = tempdir(),
  output_path = NA_character_,

```

```

    verbose = FALSE,
    n_cores = 8
)

```

## Arguments

mut	Path to a .vcf or .tsv file or string representation chr:pos:ref:alt of a mutation.
bam	Path to a BAM file.
fasta	Path to the FASTA file for the reference sequence used for generating the BAM file.
sample_id	Sample identifier.
neg_offset_mate_search	Integer. Use in read_bam. Represents the number of nucleotides to extend upstream (negative direction) from the position of interest when querying the BAM file with Rsamtools. This extension ensures that paired reads are retrieved, even if only one mate overlaps the queried position.
pos_offset_mate_search	Integer. Use in read_bam.
one_based	Boolean. TRUE if fasta is in one based. False if in 0 based.
flag_bam_list	A named list of logicals for filtering reads based on their SAM flag NA = Filter is ignored, TRUE = The read MUST have this flag, FALSE = The read MUST NOT have this flag.
report_tlen	Boolean. Whether to include the TLEN (template length) information in the output.
report_softclip	Boolean. Whether to include the number of soft-clipped bases at the fragment extremities in the output.
report_5p_3p_bases_fragment	Integer. Whether to include N fragment extremity bases in the output.
remove_softclip	Boolean. If set to TRUE, trim soft-clipped bases from the 5' end of Read 5p and from the 3' end of Read 3p.
retain_fail_qc	Boolean. If set to TRUE, retain fragments that failed the various quality checks in the output.
apply_bcftools_norm	Boolean. If set to TRUE, apply bcftools norm on each input variant to normalize it. Require that bcftools command is installed and available in the PATH.
tmp_folder	Character vector for the temporary folder path.
output_path	Character vector for the fragmentomics table output path.
verbose	Boolean. If set to TRUE, print all the warnings and the prints.
n_cores	Number of cores for parallel computation.

## Details

The function executes a multi-step workflow for each variant provided in the 'mut' input:

1. **Input Validation:** All parameters are rigorously checked for correctness (e.g., file existence, data types). Required file indices ('.bai', '.fai') are created automatically if missing.

2. **Variant Normalization:** The input variants are parsed and normalized into a canonical, left-aligned representation using a combination of VCF-style indel padding and the external 'bcftools norm' command.
3. **BAM Read Extraction:** For each normalized variant, the function efficiently queries the BAM file to retrieve all read pairs that cover the genomic locus.
4. **Parallel Fragment Processing:** The core analysis is performed in parallel using the 'future' framework. Each unique DNA fragment is processed by the 'extract\_fragment\_features' worker function to determine its size, quality metrics, and mutation status (e.g., 'MUT', 'WT', 'AMB', 'N/I').
5. **VAF Calculation:** After all fragments for a variant are processed, the Variant Allele Frequency (VAF) is calculated.
6. **Output Generation:** Results from all variants are aggregated into a single data frame. If a value for output\_path is provided, this data frame is also written to a tab-separated file.

## Value

A dataframe containing extracted fragment-level information.

## Examples

```
# --- 1. Locate Example Files ---
# The package includes small example files to demonstrate its functionality.
# We locate them using system.file().
mut_file <- system.file(
  'extdata', 'mutations_cfdna-test-01_chr1_27433000_27435000.tsv',
  package = 'fRagmentomics'
)
bam_file <- system.file(
  'extdata', 'cfdna-test-01_chr1_27433000_27435000.bam',
  package = 'fRagmentomics'
)
fasta_file <- system.file(
  'extdata', 'hg19_chr1_27433000_27435000.fa',
  package = 'fRagmentomics'
)

# --- 2. Run the Analysis ---
# This single call runs the full analysis pipeline on the example data.
# The output file is written to a temporary location to avoid cluttering
# the working directory. We use n_cores = 1L for examples.
results <- run_fRagmentomics(
  mut = mut_file,
  bam = bam_file,
  fasta = fasta_file,
  sample_id = 'cfdna-test-01',
  n_cores = 1L
)

# --- 3. View the Results ---
# Print the first few rows of the output data frame to see the results.
print(head(results))
```

# Index

ggplot2::geom\_col(), [3](#)  
ggplot2::geom\_density(), [8](#)  
ggplot2::geom\_errorbar(), [3](#)  
ggplot2::geom\_histogram(), [8](#)  
ggplot2::ggsave(), [3](#), [7](#), [8](#)  
ggseqlogo::ggseqlogo(), [7](#)  
  
plot\_freq\_barplot, [2](#)  
plot\_motif\_barplot, [3](#)  
plot\_qqseqlogo\_meme, [6](#)  
plot\_size\_distribution, [7](#)  
  
run\_fRagmentomics, [10](#)