Package: bambamR (via r-universe)

November 3, 2026

Title End-to-End RNA-Seq Processing from FASTQ to Publication-Ready Plots

Version 0.1.0

Description A streamlined toolkit for RNA-seq analysis covering FASTQ/BAM import, quality control, alignment, read counting, normalization, differential expression analysis, and publication-ready visualizations including onco plots, volcano plots, heatmaps, and PCA. Operates in minimal mode with CRAN-only dependencies or in full mode when Bioconductor packages are available.

Depends R (>= 4.1.0)

Imports ggplot2 (>= 3.4.0), data.table, stats, utils, grDevices, methods

Suggests BiocManager, ShortRead, Biostrings, Rsamtools, GenomicAlignments, GenomicFeatures, txdbmaker, GenomicRanges, DESeq2, edgeR, limma, SummarizedExperiment, Biobase, ComplexHeatmap, pheatmap, patchwork, ggrepel, shiny, shinydashboard, bslib, DT, testthat (>= 3.0.0), knitr, rmarkdown, withr

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

URL https://cttir.github.io/bambamR/, https://github.com/cttir/bambamR

BugReports https://github.com/cttir/bambamR/issues

Config/testthat/edition 3

Config/Needs/website CTTIR/themakR, pkgdown

Config/roxygen2/version 8.0.0

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

Date/Publication 2026-11-03 11:57:15 UTC

RemoteUrl https://github.com/CTTIR/bambamR

RemoteRef HEAD

RemoteSha 4e286818259d2dd08858b912b430172cee62d5e0

Contents

bb_align . . . 2
bb_count_bam . . . 4
bb_count_reads . . . 4
bb_deseq2 . . . 5
bb_edger . . . 6
bb_example_counts . . . 7
bb_example_de . . . 7
bb_example_mutations . . . 8
bb_export_csv . . . 9
bb_export_rds . . . 9
bb_export_tsv . . . 10
bb_heatmap . . . 11
bb_limma_voom . . . 12
bb_ma_plot . . . 12
bb_normalize . . . 13
bb_oncoplot . . . 14
bb_pca . . . 16
bb_pipeline . . . 17
bb_plot_qc . . . 19
bb_qc . . . 19
bb_qc_summary . . . 20
bb_read_bam . . . 21
bb_read_fastq . . . 22
bb_run_app . . . 23
bb_volcano . . . 23
print.bb_qc . . . 24
print.bb_result . . . 25
summary.bb_qc . . . 25

Index **26**

`bb_align` *Align Reads to Reference Genome*

Description

Wrapper for STAR, HISAT2, or minimap2 alignment tools. Checks that the selected aligner is available on the system PATH before running.

Usage

```
bb_align(
  fastq,
  genome_index,
  output_dir,
  aligner = c("STAR", "HISAT2", "minimap2"),
  threads = 4L,
  paired = FALSE,
  extra_args = NULL
)
```

Arguments

`fastq`	Character. Path(s) to FASTQ files. For paired-end, provide a character vector of length 2.
`genome_index`	Character. Path to genome index directory (STAR) or index prefix (HISAT2, minimap2).
`output_dir`	Character. New or empty output directory for BAM files.
`aligner`	Character. One of `"STAR"`, `"HISAT2"`, `"minimap2"`.
`threads`	Integer. Number of threads. Default `4`.
`paired`	Logical. Paired-end mode. If `TRUE`, `fastq` must have 2 elements. Default `FALSE`.
`extra_args`	Character or NULL. Additional arguments to pass to the aligner, one argument per element (for example `c("-x", "sr")`). HISAT2 optional arguments must not contain whitespace or shell metacharacters: its upstream wrapper reparses them through a shell. FASTQ and index paths containing these characters are handled with temporary symbolic links.

Value

A list with components:

bam Path to the output BAM file

stats A data.frame with alignment statistics

command The exact command that was executed

input_paths For HISAT2, the original FASTQ and index paths before temporary aliasing. Temporary aliases are removed after execution.

Examples

```
# Requires STAR/HISAT2/minimap2 installed
# result <- bb_align("reads.fastq", "/path/to/index", "output/")
```

bb_count_bam *Count Reads in BAM File*

Description

Returns the total number of reads in a BAM file using Rsamtools or system samtools.

Usage

```
bb_count_bam(path)
```

Arguments

path	Character. Path to a BAM file.

Value

A numeric value: total number of reads.

Examples

```
# count <- bb_count_bam("aligned.bam")
```

bb_count_reads *Count Reads per Gene/Feature*

Description

Counts reads overlapping genomic features. Uses `GenomicAlignments::summarizeOverlaps()` if available, otherwise shells out to `featureCounts` from the Subread package.

Usage

```
bb_count_reads(
  bam_paths,
  annotation,
  method = c("auto", "internal", "featureCounts"),
  threads = 4L,
  feature_type = "exon",
  attr_type = "gene_id",
  paired = FALSE
)
```

Arguments

bam_paths	Character vector. Paths to BAM files. Optional vector names supply unique sample IDs when BAM basenames are not unique.
annotation	Character. Path to GTF/GFF annotation file.
method	Character. "auto" tries GenomicAlignments first, then featureCounts; "internal" forces GenomicAlignments; "featureCounts" forces the external tool.
threads	Integer. Number of threads for featureCounts. Default 4.
feature_type	Character. Feature type to count (GTF column 3). Default "exon".
attr_type	Character. Attribute to group by. Default "gene_id".
paired	Logical. Count paired-end fragments. Default FALSE. The featureCounts backend requires Subread 2.0.2 or later in paired mode.

Value

A numeric matrix of counts (genes x samples). Rownames are gene IDs, colnames are derived from BAM file names.

Examples

```
# Requires BAM files and a GTF annotation
# counts <- bb_count_reads(c("s1.bam", "s2.bam"), "genes.gtf")
```

---

bb_deseq2 *Differential Expression with DESeq2*

Description

Runs DESeq2 differential expression analysis. Requires the DESeq2 package.

Usage

```
bb_deseq2(counts, coldata, design = ~condition, contrast = NULL, alpha = 0.05)
```

Arguments

counts	Numeric matrix. Raw (unnormalized) count matrix.
coldata	A data.frame with sample information. Rownames must match column names of counts. Must contain the variables referenced in design.
design	A formula specifying the design. Default ~ condition.
contrast	Character vector of length 3: c("variable", "numerator", "denominator"). If NULL, the last variable in the design is used.
alpha	Numeric. FDR threshold for independent filtering. Default 0.05.

Value

A data.frame with columns gene, log2fc, pvalue, padj, basemean.

Examples

```
if (requireNamespace("DESeq2", quietly = TRUE)) {
  counts <- matrix(rpois(600, 100), nrow = 100, ncol = 6,
    dimnames = list(paste0("gene", 1:100), paste0("S", 1:6)))
  coldata <- data.frame(
    condition = factor(rep(c("ctrl", "treat"), each = 3)),
    row.names = paste0("S", 1:6)
  )
  result <- bb_deseq2(counts, coldata)
  head(result)
}
```

---

bb_edger *Differential Expression with edgeR*

Description

Runs edgeR quasi-likelihood differential expression analysis. Requires the edgeR package.

Usage

```
bb_edger(counts, group, design = NULL)
```

Arguments

counts	Numeric matrix. Raw count matrix.
group	Factor or character vector. Group assignment for each sample.
design	A design matrix. If NULL, a simple group design is created.

Value

A data.frame with columns gene, log2fc, pvalue, padj.

Examples

```
if (requireNamespace("edgeR", quietly = TRUE)) {
  counts <- matrix(rpois(600, 100), nrow = 100, ncol = 6,
    dimnames = list(paste0("gene", 1:100), paste0("S", 1:6)))
  group <- factor(rep(c("ctrl", "treat"), each = 3))
  result <- bb_edger(counts, group)
  head(result)
}
```

bb_example_counts *Load Example Count Matrix*

Description

Loads a synthetic RNA-seq count matrix (200 genes x 10 samples) with sample metadata. The first 20 genes have simulated differential expression between control and treatment groups.

Usage

```
bb_example_counts()
```

Value

A list with components:

counts A 200 x 10 integer matrix of raw counts. Rownames are gene symbols, colnames are sample IDs.

metadata A data.frame with columns `condition`, `batch`, `sex`, `age`. Rownames match colnames of `counts`.

Examples

```
ex <- bb_example_counts()
dim(ex$counts)
head(ex$metadata)

# Quick pipeline from example data
cpm <- bb_normalize(ex$counts, method = "cpm")
bb_pca(cpm, ex$metadata, color_by = "condition")
```

bb_example_de *Load Example DE Results*

Description

Loads pre-computed differential expression results (500 genes) with realistic log2 fold-changes and p-values. The first 20 genes are simulated as significantly differentially expressed.

Usage

```
bb_example_de()
```

Value

A data.frame with columns `gene`, `log2fc`, `pvalue`, `padj`, `basemean`.

Examples

```
de <- bb_example_de()
head(de[order(de$padj), ])

# Volcano plot
bb_volcano(de)

# MA plot
bb_ma_plot(de)
```

`bb_example_mutations` *Load Example Mutation Data*

Description

Loads synthetic mutation data (300 mutations across 50 samples and 20 cancer-related genes) suitable for oncoplot visualization, along with clinical annotation data.

Usage

```
bb_example_mutations()
```

Value

A list with components:

mutations A data.frame with columns `sample`, `gene`, `mutation_type`.

clinical A data.frame with columns `Stage`, `Gender`, `Smoking`. Rownames are sample IDs.

Examples

```
ex <- bb_example_mutations()
head(ex$mutations)

# Basic oncoplot
bb_oncoplot(ex$mutations, n_genes = 10)

# With clinical annotations
bb_oncoplot(ex$mutations, n_genes = 10, annotation_df = ex$clinical)
```

bb_export_csv *Export DE Results to CSV*

Description

Writes a differential expression result data.frame to a CSV file.

Usage

```
bb_export_csv(de_result, path)
```

Arguments

de_result	A data.frame with DE results (columns: gene, log2fc, pvalue, padj, and optionally basemean).
path	Character. File path for the output .csv file.

Value

The path invisibly.

Examples

```
de <- data.frame(
  gene = paste0("gene", 1:5),
  log2fc = rnorm(5),
  pvalue = runif(5, 0, 0.1),
  padj = runif(5, 0, 0.1)
)
tmp <- tempfile(fileext = ".csv")
bb_export_csv(de, tmp)
```

bb_export_rds *Export Results to RDS*

Description

Saves a bambamR result object or any R object as an RDS file.

Usage

```
bb_export_rds(result, path)
```

Arguments

result	An R object to save (typically a bb_result or DE result).
path	Character. File path for the output .rds file.

Value

The path invisibly.

Examples

```
tmp <- tempfile(fileext = ".rds")
counts <- matrix(rpois(100, 50), nrow = 10,
  dimnames = list(paste0("g", 1:10), paste0("s", 1:10)))
bb_export_rds(counts, tmp)
readRDS(tmp)
```

---

bb_export_tsv *Export DE Results to TSV*

Description

Writes a differential expression result data.frame to a TSV file.

Usage

```
bb_export_tsv(de_result, path)
```

Arguments

de_result	A data.frame with DE results.
path	Character. File path for the output .tsv file.

Value

The path invisibly.

Examples

```
de <- data.frame(
  gene = paste0("gene", 1:5),
  log2fc = rnorm(5),
  pvalue = runif(5, 0, 0.1),
  padj = runif(5, 0, 0.1)
)
tmp <- tempfile(fileext = ".tsv")
bb_export_tsv(de, tmp)
```

bb_heatmap *Heatmap of Top Differentially Expressed Genes*

Description

Creates a heatmap from a normalized count matrix, optionally highlighting the top DE genes.

Usage

```
bb_heatmap(
  counts,
  de_result = NULL,
  n_genes = 50L,
  annotation_col = NULL,
  scale = c("row", "column", "none"),
  cluster_rows = TRUE,
  cluster_cols = TRUE,
  color_palette = NULL
)
```

Arguments

counts	Numeric matrix. Normalized count matrix (genes x samples).
de_result	A data.frame with DE results. If provided, the top n_genes by adjusted p-value are shown.
n_genes	Integer. Number of top genes to display. Default 50.
annotation_col	A data.frame for column (sample) annotations. Rownames must match column names of counts. Annotation tracks require the optional patchwork package.
scale	Character. Scale rows ("row"), columns ("column"), or neither ("none"). Default "row".
cluster_rows	Logical. Cluster rows. Default TRUE.
cluster_cols	Logical. Cluster columns. Default TRUE.
color_palette	Character vector. Colors for the heatmap gradient.

Value

A ggplot2::ggplot object.

Examples

```
set.seed(42)
counts <- matrix(rpois(500, 100), nrow = 50, ncol = 10,
  dimnames = list(paste0("gene", 1:50), paste0("S", 1:10)))
bb_heatmap(counts, n_genes = 20)
```

bb_limma_voom *Differential Expression with limma-voom*

Description

Runs limma-voom differential expression analysis. Requires the `limma` and `edgeR` packages.

Usage

```
bb_limma_voom(counts, design, contrast = NULL)
```

Arguments

`counts`	Numeric matrix. Raw count matrix.
`design`	A design matrix (e.g., from `stats::model.matrix()`).
`contrast`	A contrast vector or one-column matrix. If `NULL`, the last design coefficient is tested. One contrast is returned per call.

Value

A data.frame with columns `gene`, `log2fc`, `pvalue`, `padj`.

Examples

```
if (requireNamespace("limma", quietly = TRUE) &&
    requireNamespace("edgeR", quietly = TRUE)) {
  counts <- matrix(rpois(600, 100), nrow = 100, ncol = 6,
    dimnames = list(paste0("gene", 1:100), paste0("S", 1:6)))
  group <- factor(rep(c("ctrl", "treat"), each = 3))
  design <- model.matrix(~ group)
  result <- bb_limma_voom(counts, design)
  head(result)
}
```

bb_ma_plot *MA Plot*

Description

Creates an MA plot (log2 fold-change vs. mean expression) from differential expression results.

Usage

```
bb_ma_plot(
  de_result,
  p_cutoff = 0.05,
  point_size = 1,
  colors = c(sig = "#D73027", ns = "grey70")
)
```

Arguments

de_result	A data.frame with columns gene, log2fc, pvalue, padj, and basemean.
p_cutoff	Numeric. Adjusted p-value cutoff for coloring significant genes. Default 0.05.
point_size	Numeric. Size of points. Default 1.
colors	Named character vector with "sig" and "ns" colors.

Value

A ggplot2::ggplot object.

Examples

```
de <- data.frame(
  gene = paste0("gene", 1:200),
  log2fc = rnorm(200, 0, 2),
  pvalue = 10^(-runif(200, 0, 5)),
  padj = 10^(-runif(200, 0, 4)),
  basemean = 10^runif(200, 1, 4)
)
bb_ma_plot(de)
```

---

bb_normalize *Normalize Count Matrix*

Description

Supports CPM, TPM, TMM (edgeR), and RLE (DESeq2) normalization. CPM and TPM are always available. TMM requires edgeR, RLE requires DESeq2.

Usage

```
bb_normalize(
  counts,
  method = c("cpm", "tpm", "tmm", "rle"),
  gene_lengths = NULL
)
```

Arguments

counts	Numeric matrix. Raw count matrix with gene IDs as rownames and sample names as colnames.
method	Character. One of "cpm", "tpm", "tmm", "rle".
gene_lengths	Numeric vector. Gene lengths in base pairs. Required for TPM normalization. Must be named or in the same order as rownames of counts.

Details

- **CPM** (Counts Per Million): counts / library_size * 1e6. Pure base R.
- **TPM** (Transcripts Per Million): Normalizes by gene length then library size. Pure base R but requires gene_lengths.
- **TMM** (Trimmed Mean of M-values): Uses edgeR::calcNormFactors(). Requires the edgeR package.
- **RLE** (Relative Log Expression): Uses DESeq2::estimateSizeFactors(). Requires the DESeq2 package.

Value

A normalized numeric matrix with the same dimensions as counts.

Examples

```
counts <- matrix(
  rpois(600, lambda = 100),
  nrow = 100, ncol = 6,
  dimnames = list(paste0("gene", 1:100), paste0("sample", 1:6))
)
cpm <- bb_normalize(counts, method = "cpm")
head(cpm)

# TPM requires gene lengths
gene_lengths <- sample(500:5000, 100)
tpm <- bb_normalize(counts, method = "tpm", gene_lengths = gene_lengths)
```

---

bb_oncoplot *Create Publication-Ready Oncoplot*

Description

Generates a waterfall-style oncoplot showing the mutation landscape across samples. Accepts either a simple data.frame with sample, gene, mutation_type columns or a MAF-format data.frame.

Usage

```
bb_oncoplot(
  data,
  genes = NULL,
  n_genes = 20L,
  sort_by = c("frequency", "cluster"),
  annotation_df = NULL,
  mutation_colors = NULL,
  show_pct = TRUE,
  show_barplot = TRUE,
  title = NULL,
  border_color = "white"
)
```

Arguments

data	A data.frame with columns sample, gene, mutation_type. Alternatively, a MAF-like data.frame with Hugo_Symbol, Tumor_Sample_Barcode, and Variant_Classification columns.
genes	Character vector. Specific genes to display. If NULL, the top n_genes most frequently mutated genes are shown.
n_genes	Integer. Number of top mutated genes if genes is NULL. Default 20.
sort_by	Character. How to sort samples: "frequency" (by number of mutated genes shown) or "cluster" (by co-occurrence clustering). Default "frequency".
annotation_df	A data.frame for sample annotations (bottom tracks). Rownames must be sample identifiers. Each column becomes an annotation track. Annotation tracks require patchwork.
mutation_colors	Named character vector mapping mutation types to colors. If NULL, a default nature-style palette is used.
show_pct	Logical. Show mutation percentage per gene on the right margin. Default TRUE.
show_barplot	Logical. Show top (mutated genes per sample) and side (mutated samples per gene) barplots. Default TRUE.
title	Character or NULL. Plot title.
border_color	Character or NULL. Color of tile borders. Default "white".

Details

The oncoplot encodes the mutation landscape of a cohort as a tiled grid: columns = samples, rows = genes. Tiles are colored by mutation type. Multi-hit genes (multiple mutation types in one sample) are labeled "Multi_Hit".

When show_barplot = TRUE and the patchwork package is available, the plot is composed of three panels: top barplot (mutated genes per sample), main tile plot, and right barplot (mutated samples per gene). Without patchwork, the main tile plot is returned unless annotations were requested. Percentages use all samples represented in data before gene selection; samples with no recorded mutations cannot be inferred.

Value

A ggplot2::ggplot object (composed with patchwork if barplots are shown and patchwork is available).

Examples

```
set.seed(42)
mut_data <- data.frame(
  sample = sample(paste0("TCGA-", 1:30), 150, replace = TRUE),
  gene = sample(c("TP53","KRAS","PIK3CA","PTEN","APC",
                  "BRAF","EGFR","NRAS","CDKN2A","RB1"), 150, replace = TRUE),
  mutation_type = sample(c("Missense_Mutation","Nonsense_Mutation",
                           "Frame_Shift_Del","Splice_Site"), 150, replace = TRUE)
)
bb_oncoplot(mut_data, n_genes = 10)
```

---

bb_pca *PCA Plot*

Description

Creates a PCA plot from a normalized count matrix with sample metadata.

Usage

```
bb_pca(
  counts,
  metadata,
  color_by,
  shape_by = NULL,
  n_genes = 500L,
  label = FALSE,
  point_size = 3
)
```

Arguments

counts	Numeric matrix. Normalized count matrix (genes x samples).
metadata	A data.frame with sample information. Rownames must match column names of counts.
color_by	Character. Column name in metadata to use for coloring points.
shape_by	Character or NULL. Column name in metadata for point shapes.
n_genes	Integer. Number of top variable genes to use for PCA. Default 500.
label	Logical. Whether to label sample points. Default FALSE.
point_size	Numeric. Size of points. Default 3.

Value

A ggplot2::ggplot object.

Examples

```
set.seed(42)
counts <- matrix(rpois(600, 100), nrow = 100, ncol = 6,
  dimnames = list(paste0("gene", 1:100), paste0("S", 1:6)))
meta <- data.frame(
  condition = rep(c("Control", "Treatment"), each = 3),
  row.names = paste0("S", 1:6)
)
bb_pca(counts, meta, color_by = "condition")
```

---

bb_pipeline *Run Full bambamR Pipeline*

Description

Orchestrates the complete RNA-seq analysis pipeline from FASTQ/BAM files or a count matrix through alignment, counting, normalization, differential expression, and visualization.

Usage

```
bb_pipeline(
  fastq_dir = NULL,
  bam_dir = NULL,
  count_matrix = NULL,
  output_dir = "bambamR_output",
  genome_index = NULL,
  annotation = NULL,
  sample_info = NULL,
  aligner = "STAR",
  de_method = c("DESeq2", "edgeR", "limma"),
  design = ~condition,
  skip = character(0),
  threads = 4L
)
```

Arguments

fastq_dir	Character or NULL. Directory containing FASTQ files. If provided, the pipeline starts from alignment. Each file is one single-end sample, named by removing its FASTQ extension. For paired-end reads use bb_align() with explicit pairs, then bb_count_reads().
bam_dir	Character or NULL. Directory containing BAM files. If provided, the pipeline starts from read counting.

count_matrix	A numeric matrix or NULL. Pre-computed count matrix. If provided, the pipeline starts from normalization/DE.
output_dir	Character. Output directory for results. Default "bambamR_output". Existing result files are never overwritten.
genome_index	Character or NULL. Path to genome index. Required if starting from FASTQ.
annotation	Character or NULL. Path to GTF annotation. Required if starting from FASTQ or BAM.
sample_info	A data.frame with sample metadata. Must contain a condition column (or the variable in design). Rownames should match sample identifiers.
aligner	Character. Aligner for FASTQ alignment. Default "STAR".
de_method	Character. DE method: "DESeq2", "edgeR", or "limma". Default "DESeq2".
design	A formula for DE analysis. Default ~ condition.
skip	Character vector. Steps to skip: "qc", "align", "count", "de", "viz". Default character(0).
threads	Integer. Number of threads. Default 4.

Details

The pipeline allows entry at any stage:

- **From FASTQ**: requires fastq_dir, genome_index, annotation, and sample_info
- **From BAM**: requires bam_dir, annotation, and sample_info
- **From counts**: requires count_matrix and sample_info

Value

A bb_result object containing counts, metadata, DE results, and plots.

Examples

```
# Starting from a count matrix (minimal mode, no Bioconductor needed)
set.seed(42)
counts <- matrix(rpois(600, 100), nrow = 100, ncol = 6,
  dimnames = list(paste0("gene", 1:100), paste0("S", 1:6)))
sample_info <- data.frame(
  condition = factor(rep(c("ctrl", "treat"), each = 3)),
  row.names = paste0("S", 1:6)
)
# result <- bb_pipeline(count_matrix = counts, sample_info = sample_info)
```

bb_plot_qc *QC Visualization Panel*

Description

Creates a panel of QC plots from a bb_qc object, including read quality per position, GC content distribution, and read length distribution.

Usage

```
bb_plot_qc(qc_object, which = c("all", "quality", "gc", "length"))
```

Arguments

qc_object	A bb_qc object from bb_qc().
which	Character. Which plot to generate: "quality", "gc", "length", or "all". Default "all".

Value

A ggplot2::ggplot object (or a patchwork composition if which = "all" and patchwork is available).

Examples

```
tmp <- tempfile(fileext = ".fastq")
writeLines(c(
  "@r1", "ACGTACGT", "+", "IIIIIIII",
  "@r2", "GCGCGCGC", "+", "HHHHHHHH",
  "@r3", "ATATGCGC", "+", "GGGGFFFF"
), tmp)
qc <- bb_qc(fastq_path = tmp)
bb_plot_qc(qc)
```

bb_qc *Generate QC Metrics*

Description

Computes quality control metrics from FASTQ and/or BAM files.

Usage

```
bb_qc(fastq_path = NULL, bam_path = NULL, use_fastqc = TRUE)
```

Arguments

fastq_path	Character or NULL. Path(s) to FASTQ file(s).
bam_path	Character or NULL. Path(s) to BAM file(s).
use_fastqc	Logical. Try to use system FastQC if available. Default TRUE.

Value

A bb_qc object containing:

read_counts Total reads per file

quality_scores Per-position quality score summary

gc_content GC content distribution

read_lengths Read length distribution

mapping_rate Mapping rate from BAM files (if provided)

Examples

```
# Create a test FASTQ
tmp <- tempfile(fileext = ".fastq")
writeLines(c(
  "@read1", "ACGTACGT", "+", "IIIIIIII",
  "@read2", "GCGCGCGC", "+", "HHHHHHHH"
), tmp)
qc <- bb_qc(fastq_path = tmp)
qc
```

---

bb_qc_summary *QC Summary*

Description

Returns a summary data.frame of QC metrics.

Usage

```
bb_qc_summary(qc_object)
```

Arguments

qc_object	A bb_qc object from bb_qc().

Value

A data.frame with one row per file.

Examples

```
tmp <- tempfile(fileext = ".fastq")
writeLines(c("@r1", "ACGT", "+", "IIII"), tmp)
qc <- bb_qc(fastq_path = tmp)
bb_qc_summary(qc)
```

`bb_read_bam` *Read BAM Files*

Description

Reads BAM files using Rsamtools (if available) or the system `samtools` command as fallback.

Usage

```
bb_read_bam(
  path,
  index = TRUE,
  what = c("qname", "flag", "rname", "pos", "mapq", "cigar")
)
```

Arguments

`path`	Character. Path to a BAM file.
`index`	Logical. Whether to use/create a BAM index. Default `TRUE`.
`what`	Character vector. Fields to extract. Default includes basic alignment fields.

Value

A data.frame with alignment information including columns: `qname`, `flag`, `rname`, `pos`, `mapq`, `cigar`.

Examples

```
# Requires a BAM file
# bam_df <- bb_read_bam("aligned.bam")
```

bb_read_fastq *Read FASTQ Files*

Description

Reads FASTQ files using ShortRead (if available) or a minimal base-R parser as fallback. Supports `.gz` compressed files.

Usage

```
bb_read_fastq(path, n = NULL)
```

Arguments

`path`	Character. Path to a FASTQ or FASTQ.gz file.
`n`	Integer or NULL. Number of initial reads to read. `NULL` reads all. A finite limit uses the four-line base parser without reading the full file.

Value

A data.frame with columns:

id Read identifier

sequence Nucleotide sequence

quality Quality string (Phred+33 encoded)

Examples

```
# Create a temporary FASTQ file
tmp <- tempfile(fileext = ".fastq")
writeLines(c(
  "@read1", "ACGTACGT", "+", "IIIIIIII",
  "@read2", "TGCATGCA", "+", "HHHHHHHH"
), tmp)
reads <- bb_read_fastq(tmp)
reads
```

bb_run_app *Launch bambamR Shiny App*

Description

Starts the interactive bambamR Shiny application for RNA-seq analysis. Requires the `shiny` package.

Usage

```
bb_run_app(...)
```

Arguments

`...`	Additional arguments passed to `shiny::runApp()`.

Value

This function does not return a value; it launches a Shiny app.

Examples

```
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  bb_run_app()
}
```

bb_volcano *Volcano Plot*

Description

Creates a publication-ready volcano plot from differential expression results.

Usage

```
bb_volcano(
  de_result,
  fc_cutoff = 1,
  p_cutoff = 0.05,
  label_genes = NULL,
  n_label = 10L,
  point_size = 1,
  colors = c(up = "#D73027", down = "#4575B4", ns = "grey70")
)
```

Arguments

de_result	A data.frame with columns gene, log2fc, pvalue, padj. Typically the output of bb_deseq2(), bb_edger(), or bb_limma_voom().
fc_cutoff	Numeric. Absolute log2 fold-change cutoff for significance. Default 1.
p_cutoff	Numeric. Adjusted p-value cutoff for significance. Default 0.05.
label_genes	Character vector. Specific gene names to label on the plot. If NULL (default), the top n_label significant genes are labeled.
n_label	Integer. Number of top significant genes to auto-label when label_genes is NULL. Default 10.
point_size	Numeric. Size of points. Default 1.
colors	Named character vector of length 3 for up, down, and non-significant colors.

Value

A ggplot2::ggplot object.

Examples

```
de <- data.frame(
  gene = paste0("gene", 1:200),
  log2fc = rnorm(200, 0, 2),
  pvalue = 10^(-runif(200, 0, 5)),
  padj = 10^(-runif(200, 0, 4))
)
bb_volcano(de)
```

---

print.bb_qc *Print method for bb_qc*

Description

Print method for bb_qc

Usage

```
## S3 method for class 'bb_qc'
print(x, ...)
```

Arguments

x	A bb_qc object.
...	Additional arguments (ignored).

print.bb_result *Print method for bb_result*

Description

Print method for bb_result

Usage

```
## S3 method for class 'bb_result'
print(x, ...)
```

Arguments

x	A bb_result object.
...	Additional arguments (ignored).

summary.bb_qc *Summary method for bb_qc*

Description

Summary method for bb_qc

Usage

```
## S3 method for class 'bb_qc'
summary(object, ...)
```

Arguments

object	A bb_qc object.
...	Additional arguments (ignored).

Index

bb_align, 2
bb_align(), *17*
bb_count_bam, 4
bb_count_reads, 4
bb_count_reads(), *17*
bb_deseq2, 5
bb_deseq2(), *24*
bb_edger, 6
bb_edger(), *24*
bb_example_counts, 7
bb_example_de, 7
bb_example_mutations, 8
bb_export_csv, 9
bb_export_rds, 9
bb_export_tsv, 10
bb_heatmap, 11
bb_limma_voom, 12
bb_limma_voom(), *24*
bb_ma_plot, 12
bb_normalize, 13
bb_oncoplot, 14
bb_pca, 16
bb_pipeline, 17
bb_plot_qc, 19
bb_qc, 19
bb_qc(), *19*, *20*
bb_qc_summary, 20
bb_read_bam, 21
bb_read_fastq, 22
bb_run_app, 23
bb_volcano, 23

ggplot2::ggplot, *11*, *13*, *16*, *17*, *19*, *24*

print.bb_qc, 24
print.bb_result, 25

shiny::runApp(), *23*
stats::model.matrix(), *12*
summary.bb_qc, 25