Package 'Rduckhts'

October 28, 2026

Title 'DuckDB' High Throughput Sequencing File Formats Reader Extension

Version 2.0.2.0.1.1

Description Bundles the 'duckhts' 'DuckDB' extension for reading High Throughput Sequencing file formats with 'DuckDB'. The 'DuckDB' C extension API <https://duckdb.org/docs/stable/clients/c/api> and its 'htslib' dependency are compiled from vendored sources during package installation. James K Bonfield and co-authors (2021) <doi:10.1093/gigascience/giab007>. VariantKey / RegionKey support follows Nicola Asuni (2018) <doi:10.1101/473744>.

License GPL (>= 2)

Copyright See inst/COPYRIGHT

Encoding UTF-8

SystemRequirements GNU make, cmake, zlib, libbz2, liblzma, libcurl, openssl (development headers)

NeedsCompilation yes

Depends R (>= 4.4.0)

Imports DBI, duckdb (>= 1.5.0), methods, utils

Suggests Rtinycc, tinytest

URL https://github.com/RGenomicsETL/duckhts, https://rgenomicsetl.r-universe.dev/Rduckhts

BugReports https://github.com/RGenomicsETL/duckhts/issues

Config/roxygen2/version 8.0.0

Author Sounkou Mahamane Toure [aut, cre], Ryan Ward [aut] (GenBank reader and FASTA converter), Nurture Bio [cph] (GenBank reader and FASTA converter), James K Bonfield, John Marshall,Petr Danecek ,Heng Li , Valeriu Ohan, Andrew Whitwham,Thomas Keane , Robert M Davies [ctb] (Htslib Authors), DuckHTS Contributors [cph] (DuckHTS DuckDB extension (MIT)), RBCFTools Authors [cph] (Copyright notice in bcf_reader.c), Genome Research Ltd [cph] (HTSlib, htscodecs and the bcftools filter

engine (Petr Danecek)),
Broad Institute [cph] (Portions of HTSlib),
Massachusetts Institute of Technology [cph] (Portions of HTSlib),
Medical Research Council [cph] (Portions of HTSlib CRAM code),
Attractive Chaos [cph] (khash, kseq and related HTSlib and cgranges headers),
Dana-Farber Cancer Institute [cph] (cgranges),
Brent Pedersen [cph] (Original author of mosdepth and Somalier),
Giulio Genovese [cph] (Author of BCFTools munge,score,liftover plugins),
GENOMICS plc [cph] (VariantKey and RegionKey C API),
Nicola Asuni [cph] (Author of the VariantKey and RegionKey C API),
Devon Ryan [cph] (Author of libBigWig),
Stichting DuckDB Foundation [cph] (DuckDB C extension API headers)

Maintainer Sounkou Mahamane Toure <sounkoutoure@gmail.com>

Repository CRAN

Date/Publication 2026-10-28 10:20:08 UTC

Contents

detect_complex_types . . . 4
duckdb_type_mappings . . . 5
duckhts_bootstrap . . . 6
duckhts_build . . . 6
duckhts_load . . . 7
extract_array_element . . . 7
extract_map_data . . . 8
normalize_tabix_types . . . 9
rduckhts_ancestry_bam . . . 10
rduckhts_ancestry_geno . . . 11
rduckhts_ancestry_panel . . . 12
rduckhts_ancestry_proportions . . . 13
rduckhts_bam . . . 14
rduckhts_bam_bed_coverage . . . 16
rduckhts_bam_bin_counts . . . 17
rduckhts_bam_convert_parquet . . . 18
rduckhts_bam_index . . . 20
rduckhts_bam_insert_size_counts . . . 21
rduckhts_bam_mate_overlap_counts . . . 22
rduckhts_bam_mismatch_counts . . . 24
rduckhts_bam_multi . . . 25
rduckhts_bcf . . . 26
rduckhts_bcftools_norm . . . 28
rduckhts_bcf_convert_parquet . . . 30
rduckhts_bcf_index . . . 31
rduckhts_bcf_multi . . . 32
rduckhts_bcf_samples . . . 33
rduckhts_bed . . . 33

rduckhts_bed_multi . . . 34
rduckhts_bgunzip . . . 35
rduckhts_bgzip . . . 36
rduckhts_bigwig . . . 37
rduckhts_connect . . . 37
rduckhts_count_error_fit . . . 39
rduckhts_count_mixture_lowpass . . . 40
rduckhts_depth_bins . . . 42
rduckhts_depth_region_fraction . . . 43
rduckhts_depth_sex_fraction . . . 45
rduckhts_depth_state_solutions . . . 46
rduckhts_detect_quality_encoding . . . 49
rduckhts_fasta . . . 50
rduckhts_fasta_index . . . 51
rduckhts_fasta_multi . . . 51
rduckhts_fasta_nuc . . . 52
rduckhts_fastq . . . 53
rduckhts_fastq_multi . . . 54
rduckhts_functions . . . 55
rduckhts_genbank . . . 56
rduckhts_genbank_to_fasta . . . 57
rduckhts_geno . . . 58
rduckhts_geno_sites . . . 60
rduckhts_gff . . . 62
rduckhts_gff_convert_parquet . . . 63
rduckhts_gff_multi . . . 65
rduckhts_gtf . . . 66
rduckhts_gtf_multi . . . 67
rduckhts_htslib_config . . . 69
rduckhts_htslib_info . . . 70
rduckhts_htslib_version . . . 70
rduckhts_hts_header . . . 71
rduckhts_hts_index . . . 71
rduckhts_hts_index_raw . . . 72
rduckhts_hts_index_spans . . . 72
rduckhts_install_macros . . . 73
rduckhts_liftover . . . 73
rduckhts_load . . . 75
rduckhts_mosdepth . . . 76
rduckhts_munge . . . 77
rduckhts_pileup . . . 78
rduckhts_roh . . . 79
rduckhts_roh_counts . . . 82
rduckhts_samtools_idxstats . . . 85
rduckhts_score . . . 86
rduckhts_simd_backend . . . 87
rduckhts_somalier_bam_counts . . . 89
rduckhts_somalier_charr . . . 91

rduckhts_somalier_find_sites 92
rduckhts_somalier_import_sites 94
rduckhts_somalier_matched_contamination 95
rduckhts_somalier_relatedness 97
rduckhts_somalier_sex 98
rduckhts_somalier_sketches 100
rduckhts_somalier_vcf_counts 101
rduckhts_tabix 102
rduckhts_tabix_convert_parquet 103
rduckhts_tabix_index 105
rduckhts_tabix_multi 106
setup_hts_env 107

Index **108**

`detect_complex_types` *Detect Complex Types in DuckDB Table*

Description

Identifies columns in a DuckDB table that contain complex types (ARRAY or MAP) that will be returned as R lists.

Usage

```
detect_complex_types(con, table_name)
```

Arguments

`con`	A DuckDB connection
`table_name`	Name of the table to analyze

Value

A data frame with columns that have complex types, showing column_name, column_type, and a description of R type.

Examples

```
library(DBI)
library(duckdb)

con <- rduckhts_connect()
bcf_path <- system.file("extdata", "vcf_file.bcf", package = "Rduckhts")
rduckhts_bcf(con, "variants", bcf_path, overwrite = TRUE)
complex_cols <- detect_complex_types(con, "variants")
print(complex_cols)
dbDisconnect(con, shutdown = TRUE)
```

`duckdb_type_mappings` *DuckDB to R Type Mappings*

Description

The mapping covers the most common data types used in HTS file processing:

- BIGINT <-> double (not integer due to 64-bit overflow protection)
- DOUBLE <-> numeric/double
- VARCHAR <-> character/string
- BOOLEAN <-> logical
- ARRAY types (e.g., VARCHAR[], BIGINT[]) <-> list
- MAP types (e.g., MAP(VARCHAR, VARCHAR)) <-> data.frame

Important notes:

- 64-bit integers (BIGINT, UBIGINT) become double to prevent overflow
- DATE/TIME values return as Unix epoch numbers (double)
- MAP types become data frames with 'key' and 'value' columns
- ARRAY types become vectors (which are lists in R terminology)

Usage

```
duckdb_type_mappings()
```

Details

Returns a named list mapping between DuckDB and R data types. This is useful for understanding type conversions when reading HTS files or when specifying column types in tabix functions.

Value

A named list with two elements:

duckdb_to_r Named character vector mapping DuckDB types to R types

r_to_duckdb Named character vector mapping R types to DuckDB types

Examples

```
mappings <- duckdb_type_mappings()
mappings$duckdb_to_r["BIGINT"]
mappings$r_to_duckdb["integer"]
```

duckhts_bootstrap *Bootstrap the duckhts extension sources into the R package*

Description

Copies extension source files from the parent duckhts repository into `inst/duckhts_extension/` so the R package becomes self-contained. Run this before `R CMD build` to prepare the source tarball.

Usage

```
duckhts_bootstrap(repo_root = NULL)
```

Arguments

`repo_root`	Path to the duckhts repository root. Required.

Value

Invisibly returns the destination directory.

duckhts_build *Retired manual DuckHTS extension builder*

Description

The former manual builder duplicated the package configure path and could mutate an installed package tree. Build Rduckhts from a source tarball with `R CMD build` and install that tarball instead.

Usage

```
duckhts_build(build_dir = NULL, make = NULL, force = FALSE, verbose = TRUE)
```

Arguments

`build_dir`	Retained for source compatibility; ignored.
`make`	Retained for source compatibility; ignored.
`force`	Retained for source compatibility; ignored.
`verbose`	Retained for source compatibility; ignored.

Value

This function always raises an error.

duckhts_load *Load the duckhts extension into a DuckDB connection*

Description

Compatibility wrapper around `rduckhts_connect()` and `rduckhts_load()`. Prefer the `rduckhts_*` functions in new code.

Usage

```
duckhts_load(con = NULL, extension_path = NULL)
```

Arguments

`con`	An existing DuckDB connection, or `NULL` to create a package-owned connection with `rduckhts_connect()`.
`extension_path`	Explicit path to the `.duckdb_extension` file. If `NULL`, uses the default location in the installed package.

Value

The DuckDB connection (invisibly).

extract_array_element *Extract Array Elements Safely*

Description

Helper function to safely extract elements from DuckDB arrays (returned as R lists) with proper error handling.

Usage

```
extract_array_element(array_col, index = NULL, default = NA)
```

Arguments

`array_col`	A list column from DuckDB array data
`index`	Numeric index (1-based). If NULL, returns full list
`default`	Default value if index is out of bounds

Value

The array element at the specified index, or full array if index is NULL

Examples

```
library(DBI)
library(duckdb)

con <- rduckhts_connect()
bcf_path <- system.file("extdata", "vcf_file.bcf", package = "Rduckhts")
rduckhts_bcf(con, "variants", bcf_path, overwrite = TRUE)
data <- dbGetQuery(con, "SELECT ALT FROM variants LIMIT 5")
first_alt <- extract_array_element(data$ALT, 1)
all_alts <- extract_array_element(data$ALT)
dbDisconnect(con, shutdown = TRUE)
```

---

extract_map_data *Extract MAP Keys and Values*

Description

Helper function to work with DuckDB MAP data (returned as data frames). Can extract keys, values, or search for specific key-value pairs.

Usage

```
extract_map_data(map_col, operation = "keys", default = NA)
```

Arguments

map_col	A data frame column from DuckDB MAP data
operation	What to extract: "keys", "values", or a specific key name
default	Default value if key is not found (only used when operation is a key name)

Value

Extracted data based on the operation

Examples

```
library(DBI)
library(duckdb)

con <- rduckhts_connect()
gff_path <- system.file("extdata", "gff_file.gff.gz", package = "Rduckhts")
rduckhts_gff(con, "annotations", gff_path, attributes_map = TRUE, overwrite = TRUE)
data <- dbGetQuery(con, "SELECT attributes FROM annotations LIMIT 5")
keys <- extract_map_data(data$attributes, "keys")
name_values <- extract_map_data(data$attributes, "Name")
dbDisconnect(con, shutdown = TRUE)
```

`normalize_tabix_types` *Normalize R Data Types to DuckDB Types for Tabix*

Description

Normalizes R data type names to their corresponding DuckDB types for use with tabix readers. This function handles common R type name variations and maps them to appropriate DuckDB column types.

Usage

```
normalize_tabix_types(types)
```

Arguments

`types`	A character vector of R data type names to be normalized.

Details

The function performs the following normalizations:

- Integer types (integer, int, int32, int64) -> BIGINT
- Numeric types (numeric, double, float) -> DOUBLE
- Character types (character, string, chr) -> VARCHAR
- Logical types (logical, bool, boolean) -> BOOLEAN
- Other types -> Converted to uppercase as-is

If an empty vector is provided, it returns the empty vector unchanged.

Value

A character vector of normalized DuckDB type names suitable for tabix columns.

See Also

`rduckhts_tabix` for using normalized types with tabix readers, `duckdb_type_mappings` for the complete type mapping table.

Examples

```
normalize_tabix_types(c("integer", "character", "numeric"))
normalize_tabix_types(c("int", "string", "float"))
```

`rduckhts_ancestry_bam` *Project ancestry directly from an indexed BAM or CRAM*

Description

The panel must be a committed, versioned canonical panel intersected with the reference products. It has Somalier panel columns assembly, site_index, region, position, allele_a, and allele_b. Each count is extracted by the DuckHTS panel reader on the caller's connection. The count relation and frequency view are discarded on return; no private connection is created.

Usage

```
rduckhts_ancestry_bam(
  con,
  source_path,
  sample_id,
  reference_path,
  panel_table,
  reference_table,
  loadings_table,
  correction_table = NULL,
  frequency_method = c("allele_fraction", "called_genotype"),
  min_depth = 7,
  min_cor = 0.4,
  ...
)
```

Arguments

`con`	Connection with DuckHTS loaded.
`source_path`	Indexed BAM/CRAM path.
`sample_id`	Sample identifier in the output.
`reference_path`	FASTA path for reference checks and CRAM decoding.
`panel_table`	Panel relation with canonical site identity.
`reference_table`	Keyed wide or long reference relation.
`loadings_table`	PC loadings for long references; pass the correction relation here for wide references.
`correction_table`	PC correction relation for long references; NULL for wide.
`frequency_method`	‘"allele_fraction"‘ for B/(A+B) or ‘"called_genotype"‘ for the Somalier balance-rule genotype divided by two.
`min_depth`	Minimum measured A+B depth to retain a site.

min_cor	Minimum predicted-frequency correlation, default 0.4.
...	Additional arguments to 'rduckhts_somalier_bam_counts', such as index_path, map/base quality thresholds and worker_count.

Value

Proportions and matching audit with frequency_method and min_depth.

---

rduckhts_ancestry_geno *Estimate ancestry from a materialized 'read_geno' relation*

Description

Converts each diploid, biallelic call to ALT dosage / 2. A haploid, partially missing, or multiallelic call has NULL dosage and is counted in the missing-site audit. Original sample indices identify the calls unless a 'read_bcf_samples' relation is provided for original-header sample names. That relation must give every selected sample index one distinct non-null name.

Usage

```
rduckhts_ancestry_geno(
  con,
  geno_table,
  reference_table,
  loadings_table,
  correction_table,
  samples_table = NULL,
  sum_to_one = TRUE,
  min_cor = 0.4,
  non_reference_only
)
```

Arguments

con	Connection with DuckHTS loaded.
geno_table	Materialized 'read_geno' rows with CHROM, POS, REF, ALT and calls columns. Each site must contain a call for every selected sample, including homozygous-reference calls.
reference_table, loadings_table, correction_table	Reference products.
samples_table	Optional 'read_bcf_samples' relation with sample_index and sample_name.
sum_to_one, min_cor	Quality/constraint arguments for the solver.
non_reference_only	Required declaration of the 'read_geno' setting used to create 'geno_table'. Must be 'FALSE': sparse call lists omit zero dosages and cannot be identified from the materialized relation's schema.

Value

A row per sample and group with the matching audit.

`rduckhts_ancestry_panel` *Create a deterministic ancestry site panel for BAM/CRAM counts*

Description

Sites are biallelic unambiguous SNVs present in every reference group and every PC loading. Positions are positive one-based whole numbers and all group alleles must be present. With 'candidate_table', only matching candidate loci and alleles are retained. Otherwise one site is selected per chromosome/spaced genomic window. Results are sorted by chromosome and position and assigned dense zero-based panel ordinals. Keep the returned panel SHA-256 and the reference release with the materialized panel as its versioned identity.

Usage

```
rduckhts_ancestry_panel(
  con,
  reference_table,
  loadings_table,
  table_name,
  assembly,
  candidate_table = NULL,
  spacing_bp = 5000,
  max_sites = 17000,
  overwrite = FALSE
)
```

Arguments

`con`	A connection with DuckHTS loaded.
`reference_table, loadings_table`	Reference products on the connection.
`table_name`	Destination committed panel relation, visible to the BAM/CRAM panel preparation path.
`assembly`	Reference genome assembly label.
`candidate_table`	Optional site panel, e.g. a Somalier panel with region, position, allele_a and allele_b.
`spacing_bp`	Genomic window width for deterministic spacing.
`max_sites`	Maximum number of selected sites. When more sites are eligible, each contig keeps at least one site (the largest contigs, when 'max_sites' is below the contig count), the rest are shared by largest remainders in proportion to each contig's eligible sites minus that one (its remaining capacity, so no contig is offered more than it has), and each contig's sites are spread evenly along it.

`overwrite` Replace an existing destination.

Value

A one-row data frame with panel SHA-256 and selected site count.

`rduckhts_ancestry_proportions` *Estimate projected ancestry proportions from frequency or dosage relations*

Description

Input columns are sample_id, chromosome, position, allele_a, allele_b and frequency (or dosage), describing allele_b. Positions are positive one-based whole numbers. Reference data may be a keyed wide relation with consecutive PC1..PCn columns and one frequency column per group, or a long relation with group_id and frequency plus a keyed pc/loading relation. Corrections contain pc and coefficient. Groups (1..30), PCs (1..64), frequencies and loadings must be complete and finite at each contributing reference locus; group frequencies must lie in [0, 1]. Duplicate loci matching the input are rejected. Long references require complete keyed rows during pivoting. Inputs and references must share an assembly, uppercase biallelic SNV alleles and reference orientation. Chromosome names are compared with ‘duckhts_contig_key()‘: one leading ‘chr‘ is removed in any letter case, ‘M‘ and ‘MT‘ become ‘MT‘, and ‘X‘ and ‘Y‘ are uppercased. Any other name must match byte for byte, so ‘01‘ does not match ‘1‘. A reference, wide or long, whose chromosome column has an integer type compares that key as an integer, so ‘chr01‘ joins ‘1‘. Input duplicates at a sample/locus, palindromic alleles and missing frequencies are dropped; other alleles match directly, reversed, strand complemented or both. Reversed alleles use 1-frequency. Missing genotypes are not imputed. Audit counts include dropped physical input rows. The solver and correlation gates use full precision; returned proportions are rounded to seven decimals. Aligned scratch Parquet in ‘tempdir()‘ is removed on return. Correlation failures return NULL proportions and a status.

Usage

```
rduckhts_ancestry_proportions(
  con,
  input_table,
  reference_table,
  loadings_table,
  correction_table = NULL,
  input_kind = c("frequency", "dosage"),
  sum_to_one = TRUE,
  min_cor = 0.4,
  table_name = NULL,
  overwrite = FALSE,
  group_ids = NULL
)
```

Arguments

`con`	DuckDB connection with DuckHTS loaded.
`input_table, reference_table`	Caller-owned relation names.
`loadings_table`	Long-format PC loading relation; for wide references, pass the correction relation here with 'correction_table = NULL'.
`correction_table`	PC correction relation for long references.
`input_kind`	'frequency' or diploid 'dosage' (divided by two).
`sum_to_one`	Require coefficients to sum to one; otherwise at most one.
`min_cor`	Minimum predicted-frequency correlation, default 0.4 as in bigsnpr 1.12.21. A stricter gate may suit a specific panel.
`table_name`	Optional destination; NULL returns a data frame.
`overwrite`	Replace an existing destination.
`group_ids`	Optional named character vector for wide references, mapping frequency column names to output group IDs. Use distinct frequency column aliases when a group ID is also a PC column name or differs only by case.

Value

A row per sample and reference group, with status and matching audit. Data-frame results carry 'aligned_bytes', the size of temporary aligned Parquet written during the call.

`rduckhts_bam` *Create SAM/BAM/CRAM Table*

Description

Creates a DuckDB table from SAM, BAM, or CRAM files using the DuckHTS extension.

Usage

```
rduckhts_bam(
  con,
  table_name,
  path,
  region = NULL,
  index_path = NULL,
  reference = NULL,
  standard_tags = FALSE,
  auxiliary_tags = FALSE,
  sequence_encoding = NULL,
  quality_representation = NULL,
  cigar_representation = NULL,
```

```
  scan_mode = NULL,
  decompression_threads = 2,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
table_name	Name for the created table
path	Path to the SAM/BAM/CRAM file
region	Optional genomic region (e.g., "chr1:1000-2000")
index_path	Optional explicit path to index file (.bai/.csi/.crai)
reference	Optional reference file path for CRAM files
standard_tags	Logical. If TRUE, include typed standard SAMtags columns. Default FALSE.
auxiliary_tags	Logical. If TRUE, include AUXILIARY_TAGS map of non-standard tags. Default FALSE.
sequence_encoding	Character. Sequence encoding for the SEQ column: `"string"` (default) returns decoded bases as `VARCHAR`; `"nt16"` returns raw htslib nt16 4-bit codes as `UTINYINT[]`.
quality_representation	Character. Quality representation for the QUAL column: `"string"` (default) returns canonical Phred+33 text; `"phred"` returns raw Phred values as `UTINYINT[]`.
cigar_representation	Character. CIGAR representation for the CIGAR column: `"string"` (default) returns SAM text such as `"36M"`; `"binary"` returns packed BAM operations as `UINTEGER[]` where each element is `(len << 4) | op`.
scan_mode	Optional scan mode. Use `"auto"` (default extension behavior) or `"sequential"` to force full-file streaming instead of index-backed count/parallel scan paths. Sequential mode is incompatible with `region`.
decompression_threads	Integer. Number of htslib decompression worker threads per file handle. Default `2`. Use `0` to disable worker threads.
overwrite	Logical. If TRUE, overwrites existing table

Value

Invisible TRUE on success

Examples

```
library(DBI)
library(duckdb)

con <- rduckhts_connect()
bam_path <- system.file("extdata", "range.bam", package = "Rduckhts")
```

```
rduckhts_bam(con, "reads", bam_path, overwrite = TRUE)
dbGetQuery(con, "SELECT COUNT(*) FROM reads WHERE FLAG & 4 = 0")
dbDisconnect(con, shutdown = TRUE)
```

`rduckhts_bam_bed_coverage` *Native BAM/CRAM BED Regional Coverage Summary*

Description

Computes samtools coverage-like regional summaries for BAM or CRAM input over a BED target set, with DuckHTS-specific pre/post-filter and strand-aware post-filter outputs.

Usage

```
rduckhts_bam_bed_coverage(
  con,
  path,
  bed_path,
  reference = NULL,
  index_path = NULL,
  bed_index_path = NULL,
  mapq = 0,
  min_baseq = 0,
  min_read_len = 0,
  require_flags = 0,
  exclude_flags = 1796,
  min_depth = 1,
  max_depth = 1e+06,
  decompression_threads = 0,
  fragment_mode = FALSE,
  strand_outputs = TRUE,
  processing_threads = 0
)
```

Arguments

`con`	A DuckDB connection with DuckHTS loaded
`path`	Path to the input BAM or CRAM file
`bed_path`	Path to the input BED file
`reference`	Optional reference FASTA path for CRAM input when required
`index_path`	Optional explicit BAM/CRAM index path
`bed_index_path`	Optional explicit BED index path (reserved for future use)
`mapq`	Minimum mapping quality threshold for post-filter summaries

min_baseq	Minimum base quality threshold for post-filter base-level summaries
min_read_len	Minimum read length threshold for post-filter summaries
require_flags	Required SAM flag mask
exclude_flags	Excluded SAM flag mask. Defaults to samtools coverage's 'UNMAP|SECONDARY|QCFAIL|DUP' mask.
min_depth	Minimum depth threshold for covered-base and mean-depth summaries
max_depth	Maximum per-position depth cap. Set '0' to remove the cap.
decompression_threads	Integer. Number of htslib decompression worker threads to use for BAM/CRAM input. '0' disables htslib worker threads.
fragment_mode	Logical. Reserved for future fragment-level semantics.
strand_outputs	Logical. Emit forward/reverse post-filter summary columns.
processing_threads	Reserved for future parallel interval processing.

Value

A data frame with one row per BED interval and pre/post regional summaries

---

rduckhts_bam_bin_counts

Native Fixed-Width BAM/CRAM Bin Counts

Description

Count read starts into fixed-width genomic bins with optional duplicate handling and optional per-bin GC and MAPQ summary statistics.

Usage

```
rduckhts_bam_bin_counts(
  con,
  path,
  bin_width,
  chrom = NULL,
  include_unmapped = FALSE,
  reference = NULL,
  index_path = NULL,
  mapq = 0,
  require_flags = 0,
  exclude_flags = 0,
  rmdup = "none",
  stats = NULL
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
path	Path to the input BAM or CRAM file
bin_width	Positive fixed bin width in bases
chrom	Optional chromosome name filter
include_unmapped	Logical. If ‘TRUE‘, append one synthetic row for unmapped/no-coordinate records with ‘chrom = "*"‘, and ‘start‘, ‘end‘, and ‘bin_id‘ set to ‘NA‘.
reference	Optional reference FASTA path for CRAM input when required, and for reference-GC output when ‘stats‘ includes ‘"gc"‘
index_path	Optional explicit BAM/CRAM index path
mapq	Minimum mapping quality threshold applied after duplicate logic
require_flags	Required SAM flag mask
exclude_flags	Excluded SAM flag mask
rmdup	Duplicate handling mode: ‘"none"‘, ‘"flag"‘, or ‘"streaming"‘
stats	Optional comma-separated subset of ‘"gc"‘ and ‘"mq"‘

Value

A data frame with one row per fixed-width bin across the selected contig span, including zero-count bins, plus total, forward, reverse, and optional GC/MAPQ summary columns

---

rduckhts_bam_convert_parquet

Convert SAM/BAM/CRAM reader output to Parquet with DuckHTS metadata

---

Description

Thin DBI wrapper around extension macro ‘duckhts_bam_convert_parquet_sql(...)‘.

Usage

```
rduckhts_bam_convert_parquet(
  con,
  path,
  output,
  columns = NULL,
  region = NULL,
  index_path = NULL,
  reference = NULL,
  standard_tags = FALSE,
  auxiliary_tags = FALSE,
```

```
  sequence_encoding = NULL,
  quality_representation = NULL,
  cigar_representation = NULL,
  decompression_threads = 2,
  where = NULL,
  compression = "zstd",
  row_group_size = 100000L,
  partition_by = NULL,
  include_metadata = TRUE,
  header_text = NULL,
  metadata = NULL,
  metadata_json_file = NULL,
  write_format_version = "1",
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
path	Path or URI to the input SAM/BAM/CRAM file.
output	Path to the output Parquet file or partitioned directory.
columns	Optional character vector of columns to include. Defaults to all columns.
region	Optional genomic region string for indexed inputs.
index_path	Optional explicit index path.
reference	Optional reference FASTA path for CRAM.
standard_tags	Logical; include typed standard SAM tag columns.
auxiliary_tags	Logical; include non-standard auxiliary tags as a map.
sequence_encoding	Optional sequence encoding (‘"string"‘ or ‘"nt16"‘).
quality_representation	Optional quality representation (‘"string"‘ or ‘"phred"‘).
cigar_representation	Optional CIGAR representation (‘"string"‘ or ‘"binary"‘).
decompression_threads	Integer htslib decompression worker threads.
where	Optional SQL predicate applied to the reader output before conversion.
compression	Parquet compression, default ‘"zstd"‘.
row_group_size	Parquet row group size.
partition_by	Optional character vector of output partition columns.
include_metadata	Logical; include DuckHTS Parquet KV metadata.
header_text	Optional corrected header text to store instead of the source header.

`metadata` Optional named list/vector of extra metadata. This is the primary CRAN/offline-safe path for arbitrary metadata; values with the same names as DuckHTS defaults override the default values.

`metadata_json_file` Optional path to a JSON file containing a top-level object of extra metadata. This requires DuckDB's 'json' extension to be available when the conversion SQL is generated; otherwise DuckDB will report its normal missing-extension error. Use 'metadata' for offline-safe metadata.

`write_format_version` DuckHTS Parquet write-format version string.

`overwrite` Logical; replace an existing output path. The wrapper checks existence through DuckDB 'glob(...)' where possible and passes the same flag to DuckDB 'COPY' for partitioned-output overwrite handling.

Value

Invisibly returns 'output'.

`rduckhts_bam_index` *Build BAM or CRAM Index*

Description

Builds a BAM or CRAM index using the DuckHTS extension.

Usage

```
rduckhts_bam_index(con, path, index_path = NULL, min_shift = 0, threads = 4)
```

Arguments

`con`	A DuckDB connection with DuckHTS loaded
`path`	Path to the input BAM or CRAM file
`index_path`	Optional explicit output path for the created index
`min_shift`	Index format selector used by htslib
`threads`	htslib indexing thread count

Value

A data frame with 'success', 'index_path', and 'index_format'

rduckhts_bam_insert_size_counts
BAM/CRAM Insert-Size Counts

Description

Reduces eligible records into sparse orientation and absolute-TLEN counts using Fulcrum Genomics riker 0.4.1 semantics (MIT). Only non-first segments count; QC-fail records are excluded. The unsuffixed wrapper uses the native reducer; the '_sql' wrapper uses the SQL macro. The SQL arm projects binary CIGAR and flag/coordinate columns only; CRAM decoding also requests sequence and tags internally for long-CIGAR restoration. The native arm avoids these fields. For weighted metrics, materialize the counts and call 'duckhts_insert_size_metrics('counts_table')' in SQL.

Usage

```
rduckhts_bam_insert_size_counts(
  con,
  path,
  reference = NULL,
  include_duplicates = FALSE,
  threads = 1,
  decompression_threads = 0,
  scan_mode = c("auto", "indexed", "sequential"),
  index_path = NULL,
  reference_index_path = NULL
)

rduckhts_bam_insert_size_counts_sql(
  con,
  path,
  reference = NULL,
  include_duplicates = FALSE,
  threads = 1,
  decompression_threads = 0,
  scan_mode = c("auto", "sequential"),
  index_path = NULL
)
```

Arguments

con	DuckDB connection with DuckHTS loaded.
path	BAM or CRAM path.
reference	Reference FASTA for CRAM, or NULL.
include_duplicates	Include duplicate-flagged records.

`threads` Workers, from 1 to 8. The SQL wrapper temporarily sets and restores the connection's DuckDB threads setting; the native wrapper uses indexed histogram workers.

`decompression_threads` htslib decompression workers per file handle.

`scan_mode` SQL: auto or sequential. Native also accepts indexed; its sequential mode uses one histogram worker.

`index_path` Optional BAM/CRAM index path.

`reference_index_path` Optional reference FASTA index path (native only).

Value

Data frame with pair_orientation, insert_size and read_pairs. Rows are unordered; absent bins are omitted.

---

rduckhts_bam_mate_overlap_counts

Count Agreement Between Overlapping Mates

Description

Compares read 1 and read 2 of one fragment at each reference position where both mates have an aligned base, and counts the pairs of base qualities and bases. A real allele sits on the molecule and both mates show it; a sequencing error usually shows in one mate only. So the disagreement rate is about twice the instrument error rate, with no reference, mask or contamination model. Errors made before sequencing, such as PCR or damage errors, show in both mates and are not seen.

Usage

```
rduckhts_bam_mate_overlap_counts(
  con,
  path,
  region = NULL,
  index_path = NULL,
  reference = NULL,
  reference_index_path = NULL,
  min_mapq = 20,
  require_flags = 0,
  exclude_flags = 3844,
  max_pending = 1e+06
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
path	Path to the paired-end SAM, BAM or CRAM file
region	Optional comma-separated contigs or intervals. It needs an alignment index and selects alignments by overlap; a pair counts only when both mates are selected
index_path	Optional explicit alignment index path
reference	Optional indexed reference FASTA. Required for CRAM; it is otherwise used only to resolve read bases stored as '='
reference_index_path	Optional explicit FASTA index path
min_mapq	Minimum mapping quality, 0 to 255
require_flags, exclude_flags	SAM flag masks that both mates must pass. The default 'exclude_flags' leaves out unmapped, secondary, failed, duplicate and supplementary alignments
max_pending	Most mates held at once while they wait for their mate, 1 to 100000000

Details

Alignments must be in coordinate order. The first mate of a pair is held until its mate arrives; more than 'max_pending' mates held at once is an error.

Value

A data frame with 'read1_quality' and 'read2_quality' ('NA' when the read stores no qualities), 'read1_base' and 'read2_base' on the forward strand of the reference, and 'positions'. Read 1 is the mate with flag 0x40. Rows with different bases are the disagreements.

Examples

```
library(DBI)
library(duckdb)

con <- rduckhts_connect()
bam <- system.file("extdata", "empty-tids.bam", package = "Rduckhts")
counts <- rduckhts_bam_mate_overlap_counts(con, bam)
sum(counts$positions[counts$read1_base != counts$read2_base]) / sum(counts$positions)
dbDisconnect(con, shutdown = TRUE)
```

`rduckhts_bam_mismatch_counts` *Count Aligned Bases Against the Reference*

Description

Counts the aligned bases of a SAM, BAM or CRAM file against the reference by mate, cycle, base quality and substitution. With a mask of the known variants of the sample or its population, the mismatches that remain are mostly errors of the read, the library or the alignment. A variant that the mask does not hold is still counted, so their rate is an upper bound on the error rate.

Usage

```
rduckhts_bam_mismatch_counts(
  con,
  path,
  reference,
  region = NULL,
  mask = NULL,
  index_path = NULL,
  reference_index_path = NULL,
  mask_index_path = NULL,
  min_mapq = 20,
  require_flags = 0,
  exclude_flags = 3844,
  indel_flank = 5
)
```

Arguments

`con`	A DuckDB connection with DuckHTS loaded
`path`	Path to the SAM, BAM or CRAM file
`reference`	Path to the indexed reference FASTA
`region`	Optional comma-separated contigs or intervals. It needs an alignment index and selects alignments by overlap
`mask`	Optional path to an indexed BCF or a bgzip-compressed VCF with a tabix index. Its positions are left out
`index_path`, `reference_index_path`, `mask_index_path`	Optional explicit index paths
`min_mapq`	Minimum mapping quality, 0 to 255
`require_flags`, `exclude_flags`	SAM flag masks. The default ‘exclude_flags‘ leaves out unmapped, secondary, failed, duplicate and supplementary alignments
`indel_flank`	Read bases to leave out on each side of an insertion, a deletion, a reference skip or a soft clip, and reference bases on each side of a mask record with an indel or a symbolic allele; 0 to 1000

Details

Contig names of the alignment header, the reference and the mask are compared byte for byte. A contig that the reference or the mask does not know is an error.

Value

A data frame with ‘mate‘ (1 or 2 for a paired read, 0 otherwise), ‘cycle‘, ‘base_quality‘ (‘NA‘ when the read stores no qualities), ‘reference_base‘ and ‘read_base‘ on the strand of the read, and ‘bases‘. Rows with equal bases are the matches.

Examples

```
library(DBI)
library(duckdb)

con <- rduckhts_connect()
extdata <- function(name) system.file("extdata", name, package = "Rduckhts")
counts <- rduckhts_bam_mismatch_counts(
  con, extdata("bam_mismatch.bam"), extdata("bam_mismatch.fa"),
  mask = extdata("bam_mismatch.mask.bcf")
)
sum(counts$bases[counts$reference_base != counts$read_base]) / sum(counts$bases)
dbDisconnect(con, shutdown = TRUE)
```

---

rduckhts_bam_multi *Read multiple BAM/SAM files into a DuckDB table*

Description

Read and combine multiple BAM/SAM files via UNION ALL BY NAME, materialising the result as a DuckDB table. Each row includes a `filename` column identifying its source file.

Usage

```
rduckhts_bam_multi(
  con,
  table_name,
  files,
  region = NULL,
  index_path = NULL,
  reference = NULL,
  standard_tags = FALSE,
  auxiliary_tags = FALSE,
  sequence_encoding = NULL,
  quality_representation = NULL,
  cigar_representation = NULL,
  scan_mode = NULL,
```

```
  decompression_threads = 2,
  .params = NULL,
  overwrite = FALSE
)
```

Arguments

con	A DBI connection to DuckDB with the duckhts extension loaded.
table_name	Name of the DuckDB table to create.
files	Character vector of file paths or glob patterns.
region	Optional region string (e.g. "chr1:1-1000").
index_path	Optional index file path.
reference	Optional reference FASTA path (for CRAM).
standard_tags	Logical; include standard SAM tag columns.
auxiliary_tags	Logical; include auxiliary tag map column.
sequence_encoding	Optional sequence encoding (e.g. "nt16").
quality_representation	Optional quality representation.
cigar_representation	Optional CIGAR representation; use "binary" for packed BAM operations.
scan_mode	Optional scan mode ("auto" or "sequential").
decompression_threads	Integer. Number of htslib decompression worker threads per file handle. Default 2. Use 0 to disable worker threads.
.params	Optional data.frame with per-file parameter overrides. Must contain a file column; other columns override uniform parameters. NA values use the uniform default.
overwrite	Logical; if TRUE, replace an existing table.

Value

Invisible TRUE on success.

---

rduckhts_bcf *Create VCF/BCF Table*

Description

Creates a DuckDB table from a VCF or BCF file using the DuckHTS extension. This follows the RBCFTools pattern of creating a table that can be queried.

Usage

```
rduckhts_bcf(
  con,
  table_name,
  path,
  region = NULL,
  index_path = NULL,
  tidy_format = FALSE,
  additional_csq_column_types = NULL,
  scan_mode = NULL,
  decompression_threads = 0,
  decode_error_policy = "null",
  overwrite = FALSE,
  samples = NULL,
  regions_var = NULL,
  include_file_offset = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
table_name	Name for the created table
path	Path to the VCF/BCF file
region	Optional genomic region (e.g., "chr1:1000-2000")
index_path	Optional explicit path to index file (.csi/.tbi)
tidy_format	Logical. If TRUE, FORMAT columns are returned in tidy format
additional_csq_column_types	Optional bcftools-style 'PATTERN TYPE' overrides for CSQ/ANN/BCSQ subfield typing, separated by newlines or ';'
scan_mode	Optional scan mode. Use "auto" (default extension behavior) or "sequential" to force full-file streaming instead of index-backed count/parallel scan paths. Sequential mode is incompatible with 'region'.
decompression_threads	Integer. Number of htslib decompression worker threads per file handle. Default '0'. Use '0' to keep BCF/VCF reads single-threaded.
decode_error_policy	Character. VCF/BCF decode policy: "null" returns NULL for header-vs-payload type clashes or oversized numeric scalars, "warn" emits a DuckHTS warning and returns NULL, and "error" raises a DuckDB/R error. Missing elements count toward scalar cardinality; vector-end padding does not. A malformed FORMAT tag is withheld for every selected sample on that record. Physical read errors and OOM always fail.
overwrite	Logical. If TRUE, overwrites existing table
samples	Optional HTSlib sample selector: 'NULL' or '"-"' keeps all, '""' keeps none, comma-separated names include samples, and a leading '"^"' excludes them. Unknown names error; selected samples retain header order.

`regions_var` Optional name of a session variable on ‘con‘ that holds typed intervals, ‘STRUCT(chrom VARCHAR, start BIGINT, "end" BIGINT)[]‘ with 0-based half-open coordinates (a VCF position ‘p‘ is ‘[p - 1, p)‘). The name is quoted into ‘regions := getvariable('<name>')‘; create the variable with ‘SET VARIABLE‘ on the same connection. A ‘NULL‘ variable value keeps the ordinary scan, an empty list selects no records, and ‘region‘ and ‘regions_var‘ are mutually exclusive. Typed regions need an index and are incompatible with ‘scan_mode = "sequential"‘. Chromosome names are literal (not region expressions); overlapping and adjacent intervals are merged natively. Limits: 1,000,000 intervals and 128 MiB. Aggregate with ‘coalesce(list(...), []::STRUCT(...)[])‘: ‘list()‘ over no rows is ‘NULL‘, which means the ordinary full scan.

`include_file_offset` Logical. If ‘TRUE‘, add a ‘FILE_OFFSET‘ column after ‘FILTER‘: the BGZF virtual position just after each record. It increases strictly in file order and is the same for every scan mode, so ordering by it restores file order, also among records at one position. Input that is not BGZF-compressed gives ‘NA‘. The default keeps the existing schema.

Value

Invisible TRUE on success

Examples

```
library(DBI)
library(duckdb)

con <- rduckhts_connect()
bcf_path <- system.file("extdata", "vcf_file.bcf", package = "Rduckhts")
rduckhts_bcf(con, "variants", bcf_path, overwrite = TRUE)
dbGetQuery(con, "SELECT * FROM variants LIMIT 2")
dbDisconnect(con, shutdown = TRUE)
```

---

rduckhts_bcftools_norm

Normalize Variant Alleles with bcftools-style Semantics

Description

Applies the DuckHTS ‘duckhts_bcftools_norm(...)‘ table macro to rows from a SQL query or table expression. Input rows must expose chromosome, 1-based position, reference allele, and alternate allele columns. Alternate alleles may be supplied either as a comma-delimited ‘VARCHAR‘ or as a ‘VARCHAR[]‘ list, matching the common DuckDB representations used by plain tables and ‘read_bcf(...)‘.

Usage

```
rduckhts_bcftools_norm(
  con,
  query,
  fasta_ref,
  chrom_col = "chrom",
  pos_col = "pos",
  ref_col = "ref",
  alt_col = "alt",
  split_multiallelic = FALSE,
  end_pos_col = NULL,
  svlen_col = NULL,
  fasta_index_path = NULL,
  gzi_path = NULL
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
query	SQL query or table expression to normalize
fasta_ref	Path to the reference FASTA
chrom_col	Source chromosome column name
pos_col	Source 1-based position column name
ref_col	Source reference allele column name
alt_col	Source alternate allele column name (‘VARCHAR‘ or ‘VARCHAR[]‘)
split_multiallelic	If ‘TRUE‘, split multiallelic sites before normalization so ‘alt_normed‘ is emitted as ‘VARCHAR‘ plus ‘alt_index‘. If ‘FALSE‘ (default), keep sites intact and emit ‘alt_normed‘ as ‘VARCHAR[]‘.
end_pos_col	Optional source column name containing an END-like 1-based end coordinate for symbolic deletions.
svlen_col	Optional source column name containing an SVLEN-like signed length for symbolic duplications.
fasta_index_path	Optional explicit ‘.fai‘ sidecar path.
gzi_path	Optional explicit ‘.gzi‘ sidecar path for bgzipped FASTA.

Value

A data frame with the original columns plus ‘pos_normed‘, ‘end_pos_normed‘, ‘ref_normed‘, ‘alt_normed‘, ‘normed‘, and ‘norm_status‘. In split mode the result additionally includes ‘alt_index‘.

rduckhts_bcf_convert_parquet *Convert VCF/BCF reader output to Parquet with DuckHTS metadata*

Description

'rduckhts_bcf_convert_parquet()' is a thin DBI wrapper around the extension macro 'duckhts_bcf_convert_parquet_sql(...)'. The extension macro builds the 'COPY ... TO ... (FORMAT PARQUET, KV_METADATA ...)' statement, including DuckHTS write-format metadata, optional user metadata maps, selected columns, optional SQL filter text, 'tidy_format', and the VCF header under the 'vcf_header' key. The wrapper executes the generated statement.

Usage

```
rduckhts_bcf_convert_parquet(
  con,
  path,
  output,
  columns = NULL,
  region = NULL,
  index_path = NULL,
  tidy_format = FALSE,
  additional_csq_column_types = NULL,
  decompression_threads = 0,
  where = NULL,
  compression = "zstd",
  row_group_size = 100000L,
  partition_by = NULL,
  include_metadata = TRUE,
  header_text = NULL,
  metadata = NULL,
  metadata_json_file = NULL,
  write_format_version = "1",
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
path	Path or URI to the input VCF/BCF file.
output	Path to the output Parquet file or partitioned directory.
columns	Optional character vector of columns to include. Defaults to all columns.
region	Optional genomic region string for indexed inputs.
index_path	Optional explicit index path.
tidy_format	Logical; request 'read_bcf(..., tidy_format := true)'.

`additional_csq_column_types`
: Optional CSQ type override string.

`decompression_threads`
: Integer htslib decompression worker threads.

`where`
: Optional SQL predicate applied to the reader output before conversion.

`compression`
: Parquet compression, default ‘"zstd"‘.

`row_group_size`
: Parquet row group size.

`partition_by`
: Optional character vector of output partition columns.

`include_metadata`
: Logical; include DuckHTS Parquet KV metadata.

`header_text`
: Optional corrected header text to store instead of the source header.

`metadata`
: Optional named list/vector of extra metadata. This is the primary CRAN/offline-safe path for arbitrary metadata; values with the same names as DuckHTS defaults override the default values.

`metadata_json_file`
: Optional path to a JSON file containing a top-level object of extra metadata. This requires DuckDB’s ‘json‘ extension to be available when the conversion SQL is generated; otherwise DuckDB will report its normal missing-extension error. Use ‘metadata‘ for offline-safe metadata.

`write_format_version`
: DuckHTS Parquet write-format version string.

`overwrite`
: Logical; replace an existing output path. The wrapper checks existence through DuckDB ‘glob(...)‘ where possible and passes the same flag to DuckDB ‘COPY‘ for partitioned-output overwrite handling.

Value

Invisibly returns ‘output‘.

`rduckhts_bcf_index` *Build VCF or BCF Index*

Description

Builds a TBI or CSI index for a VCF/BCF file using the DuckHTS extension.

Usage

```
rduckhts_bcf_index(con, path, index_path = NULL, min_shift = NULL, threads = 4)
```

Arguments

`con`
: A DuckDB connection with DuckHTS loaded

`path`
: Path to the input VCF/BCF file

`index_path`
: Optional explicit output path for the created index

`min_shift`
: Optional explicit min_shift passed to htslib

`threads`
: htslib indexing thread count

Value

A data frame with ‘success‘, ‘index_path‘, and ‘index_format‘

`rduckhts_bcf_multi` *Read multiple VCF/BCF files into a DuckDB table*

Description

Read and combine multiple VCF/BCF files via UNION ALL BY NAME, materialising the result as a DuckDB table. Each row includes a `filename` column identifying its source file.

Usage

```
rduckhts_bcf_multi(
  con,
  table_name,
  files,
  region = NULL,
  index_path = NULL,
  tidy_format = FALSE,
  additional_csq_column_types = NULL,
  scan_mode = NULL,
  decompression_threads = 0,
  .params = NULL,
  overwrite = FALSE,
  samples = NULL
)
```

Arguments

`con`	A DBI connection to DuckDB with the duckhts extension loaded.
`table_name`	Name of the DuckDB table to create.
`files`	Character vector of file paths or glob patterns.
`region`	Optional region string.
`index_path`	Optional index file path.
`tidy_format`	Logical; use tidy FORMAT column output.
`additional_csq_column_types`	Optional CSQ type override string.
`scan_mode`	Optional scan mode (‘"auto"‘ or ‘"sequential"‘).
`decompression_threads`	Integer. Number of htslib decompression worker threads per file handle. Default ‘0‘.
`.params`	Optional data.frame with per-file parameter overrides.
`overwrite`	Logical; if `TRUE`, replace an existing table.
`samples`	Optional HTSlib sample-selector string, as in [rduckhts_bcf()].

Value

Invisible TRUE on success.

---

rduckhts_bcf_samples *Read the VCF/BCF Sample Catalog*

Description

Return one row per selected sample with its original-header zero-based ‘sample_index‘ and ‘sample_name‘. Join this relation to ‘read_geno()‘ calls from the same unchanged file; names are not duplicated into every call.

Usage

```
rduckhts_bcf_samples(con, path, samples = NULL)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
path	Path to the VCF/BCF file
samples	Optional HTSlib sample selector: ‘NULL‘ or ‘"-"‘ keeps all, ‘""‘ keeps none, comma-separated names include samples, and a leading ‘"^"‘ excludes them. Unknown names error; selected samples retain header order.

Value

A data frame with ‘sample_index‘ and ‘sample_name‘ columns.

---

rduckhts_bed *Create BED Table*

Description

Creates a DuckDB table from a BED file using the DuckHTS extension.

Usage

```
rduckhts_bed(
  con,
  table_name,
  path,
  region = NULL,
  index_path = NULL,
  scan_mode = NULL,
  overwrite = FALSE,
  error_policy = NULL
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
table_name	Name for the created table
path	Path to the BED file
region	Optional genomic region for tabix-backed BED queries
index_path	Optional explicit path to a BED tabix index
scan_mode	Optional scan mode. Use "auto" (default extension behavior) or "sequential" to force full-file streaming/counting instead of index-backed count paths. Sequential mode is incompatible with region.
overwrite	Logical. If TRUE, overwrites an existing table
error_policy	Optional short-line policy: "error" (default), "skip", or "report". Report adds diagnostic columns and requires a full-file scan rather than a region query.

Value

Invisible TRUE on success

Examples

```
con <- rduckhts_connect()

# Intervals are 0-based and half-open, like BED: keep BED starts as they are.
DBI::dbExecute(con, 'CREATE TABLE tiles AS SELECT * FROM (VALUES
  (\'CHROMOSOME_I\', 0, 10), (\'CHROMOSOME_I\', 10, 20)) t(chrom, start, "end")')
DBI::dbGetQuery(con, "SELECT * FROM duckhts_cgranges_from_table(
  'tiles_idx', 'tiles', 'chrom', 'start', 'end')")

# end is a reserved SQL word: quote the column as "end" in the select list and
# pass that column, not the string 'end', to the scalar functions.
rduckhts_bed(con, "targets", system.file("extdata", "targets.bed", package = "Rduckhts"))
DBI::dbGetQuery(con, 'SELECT chrom, start, "end",
  duckhts_cgranges_count_overlaps(\'tiles_idx\', chrom, start, "end") AS tiles
  FROM targets')

DBI::dbDisconnect(con, shutdown = TRUE)
```

---

rduckhts_bed_multi *Read multiple BED files into a DuckDB table*

Description

Read and combine multiple BED files via UNION ALL BY NAME, materialising the result as a DuckDB table. Each row includes a filename column identifying its source file.

Usage

```
rduckhts_bed_multi(
  con,
  table_name,
  files,
  region = NULL,
  index_path = NULL,
  scan_mode = NULL,
  .params = NULL,
  overwrite = FALSE
)
```

Arguments

con	A DBI connection to DuckDB with the duckhts extension loaded.
table_name	Name of the DuckDB table to create.
files	Character vector of file paths or glob patterns.
region	Optional region string.
index_path	Optional index file path.
scan_mode	Optional scan mode ("auto" or "sequential").
.params	Optional data.frame with per-file parameter overrides.
overwrite	Logical; if TRUE, replace an existing table.

Value

Invisible TRUE on success.

---

rduckhts_bgunzip *BGZF Decompress a File*

Description

Decompresses a BGZF file using the DuckHTS extension.

Usage

```
rduckhts_bgunzip(
  con,
  path,
  output_path = NULL,
  threads = 4,
  keep = TRUE,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
path	Path to the BGZF-compressed input file
output_path	Optional explicit output path
threads	BGZF worker thread count
keep	Keep the compressed input file after decompression
overwrite	Overwrite an existing output file

Value

A data frame describing the created output file

---

rduckhts_bgzip *BGZF Compress a File*

Description

Compresses a plain file to BGZF using the DuckHTS extension.

Usage

```
rduckhts_bgzip(
  con,
  path,
  output_path = NULL,
  threads = 4,
  level = -1,
  keep = TRUE,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
path	Path to the input file
output_path	Optional explicit output path
threads	BGZF worker thread count
level	Compression level, or -1 for the htslib default
keep	Keep the original input file after compression
overwrite	Overwrite an existing output file

Value

A data frame describing the created BGZF file

rduckhts_bigwig *Create a BigWig Signal Table*

Description

Materializes stored zero-based, half-open BigWig intervals through the DuckHTS extension. Region filters use htslib's one-based inclusive syntax; a character vector is combined into one multi-region request and overlapping requests emit each stored interval once.

Usage

```
rduckhts_bigwig(
  con,
  table_name,
  path,
  region = NULL,
  blocks_per_iteration = 64L,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
table_name	Name for the created table, or NULL to create the bigwig_data view.
path	Local path or URL to a BigWig file.
region	Optional character vector of genomic regions such as c("chr1:1000-2000", "chr2:1-500"). A single comma-separated string is also accepted.
blocks_per_iteration	Positive integer number of indexed BigWig data blocks decoded per iterator batch.
overwrite	Logical. If TRUE, replace an existing table.

Value

Invisibly returns TRUE.

rduckhts_connect *Create a DuckDB connection with bundled DuckHTS loaded*

Description

Creates a DuckDB connection that explicitly permits loading the package-built DuckHTS extension, disables automatic installation and loading of unrelated known DuckDB extensions, and loads the bundled DuckHTS extension.

Usage

```
rduckhts_connect(
  dbdir = ":memory:",
  read_only = FALSE,
  bigint = "numeric",
  config = list(),
  extension_path = NULL
)
```

Arguments

dbdir	Path to a DuckDB database, or ":memory:" for an in-memory database.
read_only	Logical; open a file-backed database read-only.
bigint	How DuckDB 64-bit integers are returned; passed to duckdb::duckdb().
config	Named list of additional DuckDB configuration settings.
extension_path	Optional path to a DuckHTS .duckdb_extension file. If NULL, uses the extension bundled with Rduckhts.

Details

Current versions of the **duckdb** R package can disable extension loading on Linux builds that use a C++ standard library other than libstdc++. DuckHTS is compiled locally with the same package toolchain, so this helper explicitly enables extension loading for this package-owned connection. Package-owned connections use per-session DuckDB extension/secret storage by default when the installed **duckdb** version supports that setting.

DuckDB reuses a live database instance when the same file-backed dbdir is opened again, ignoring new driver and configuration settings. This helper rejects such reuse rather than silently weakening its connection policy. Use rduckhts_load() on the existing connection if it already permits unsigned, driver-level extension loading, or close the existing database instance before calling this helper.

The settings allow_unsigned_extensions, autoinstall_known_extensions, and autoload_known_extensions are controlled by this helper and override entries with those names in config. Other named DuckDB settings are passed through unchanged.

Value

A DuckDB connection with DuckHTS loaded. File-backed connections also have connection-local TEMP macros installed without modifying the database catalog.

Examples

```
con <- rduckhts_connect()
DBI::dbGetQuery(con, "SELECT duckhts_htslib_version() AS version")
DBI::dbDisconnect(con, shutdown = TRUE)
```

rduckhts_count_error_fit *Fit Read Error and Contamination from Allele Read Counts*

Description

Fits, for each sample, the read error, the share of reads from a second genome, the allele balance at heterozygous sites, two spreads, an artefact weight and an excess of homozygosity by maximum likelihood from allele read counts at sites of known population frequency. The input is the relation of [rduckhts_roh_counts()]; the fit is meant to supply that decode's 'seq_error' and 'contamination'.

Usage

```
rduckhts_count_error_fit(
  con,
  counts_table,
  block_bases = 1e+07,
  freq_bins = 64,
  max_depth = 1000,
  min_block_sites = 1000,
  table_name = NULL,
  overwrite = FALSE,
  max_cells = 5e+06,
  max_cell_bytes = 2^30
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
counts_table	Name of a table or view with 'sample_id', 'chrom', 'pos', 'ref_count', 'alt_count' and 'af', as for [rduckhts_roh_counts()].
block_bases	Bases per block for the block refits, at least 1.
freq_bins	Frequency bins of the histogram, 1 to 1024.
max_depth	Sites with more reads are left out, 1 to 65535.
min_block_sites	Sites a block needs to be refitted, at least 1.
table_name	Optional output table. 'NULL' returns a data frame.
overwrite	Whether an existing output table may be replaced.
max_cells	Most histogram cells of one sample; an error above it.
max_cell_bytes	Most bytes the histogram buffers of all fits in the process hold at once; an error above it.

Details

The likelihood depends on a site only through its two counts and its frequency, so the fit runs on a histogram of (block, frequency bin, depth, alt count) cells that DuckDB builds and the native code holds in a bounded buffer. Each block of ‘block_bases‘ bases with at least ‘min_block_sites‘ sites is refitted for contamination with the other parameters held, and ‘contamination_sd‘ is the standard deviation of those block estimates divided by the square root of their number: an interval from the blocks, not from the likelihood, because sites are linked. Each such block is also refitted for the artefact weight alone, and ‘artefact_weight_sd‘ is formed the same way; both are NA with fewer than three blocks. [rduckhts_roh_counts()] with ‘fit_table‘ uses ‘artefact_weight_sd‘ for its ‘stable_fraction‘. The same cells are also fitted with a parent or child as the second genome (‘contamination_relative‘, ‘log_likelihood_relative‘), since the unrelated model gives a relative about half its share.

‘homozygosity_excess‘ is a nuisance parameter: it also absorbs a difference between the sample’s population and the frequencies given, so it is not inbreeding. A second genome of another population biases ‘contamination‘ upward under the receiver’s frequencies.

Value

A data frame (or invisible ‘TRUE‘ when ‘table_name‘ is given) with one row per sample: ‘sample‘, ‘sites‘, ‘reads‘, ‘mean_depth‘, ‘blocks‘, ‘seq_error‘, ‘contamination‘, ‘contamination_sd‘, ‘homozygosity_excess‘, ‘allele_balance‘, ‘spread_hom‘, ‘spread_het‘, ‘artefact_weight‘, ‘artefact_weight_sd‘, ‘log_likelihood‘, ‘contamination_relative‘, ‘log_likelihood_relative‘, ‘status‘ (‘ok‘, ‘no_sites‘ for a sample whose sites were all skipped, ‘few_sites‘, ‘few_blocks‘, ‘at_bound‘, ‘no_convergence‘ or ‘no_convergence_relative‘) and ‘method‘.

Examples

```
con <- rduckhts_connect()
DBI::dbExecute(con, paste(
  "CREATE TEMP TABLE counts AS SELECT 'S1' AS sample_id, '1' AS chrom,",
  "i * 2500 AS pos, 0.1 + 0.8 * ((i * 37) % 100) / 100.0 AS af,",
  "30 - ((i * 7) % 3) * 15 AS ref_count, ((i * 7) % 3) * 15 AS alt_count",
  "FROM range(1, 4001) AS t(i)"
))
rduckhts_count_error_fit(con, "counts")[, c("sample", "seq_error", "contamination", "status")]
DBI::dbDisconnect(con, shutdown = TRUE)
```

---

rduckhts_count_mixture_lowpass

Estimate a Conditional Second-Genome Share from Low-Pass Counts

---

Description

Uses first-read and discordant-pair moments with fixed external calibration. This estimator is separate from rduckhts_count_error_fit(). It is suitable for low-pass, cell-free DNA or low-input DNA counts only under the supplied population frequencies and calibration; the share is not an unconditional biological fraction. Matched-frequency unrelated and IBD1 hypotheses can have identical moments and are reported as unresolved.

Usage

```
rduckhts_count_mixture_lowpass(
  con,
  counts_table,
  seq_error,
  artefact_weight,
  homozygosity_excess,
  block_bases = 1e+07,
  max_depth = 8,
  min_blocks = 30,
  max_blocks = 512,
  max_candidates = 16,
  max_state_bytes = 2^28,
  table_name = NULL,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
counts_table	Table or view containing sample_id, candidate, chrom, one-based pos, ref_count, alt_count, other_count, sample_af and candidate_af. Counts and sample_af must agree across candidates; site keys must be unique within each sample and candidate. Frequencies refer to the counted alternate allele. Sites missing any candidate frequency are unavailable to every candidate of that sample.
seq_error	Required fixed sequencing error in [0, 0.5).
artefact_weight	Required fixed uniform-count artefact weight in [0, 1).
homozygosity_excess	Required fixed excess of homozygosity in [0, 1].
block_bases	Bases per chromosome-local block, at least 1.
max_depth	Maximum ref plus alt depth, 1 to 65535. Deeper sites are excluded; other_count contributes only to QC totals.
min_blocks	Informative blocks required for 95 percent limits, at least 2, default 30. Matched frequencies require pair-bearing blocks; otherwise read-bearing blocks count. Lower values support small-fixture checks, not well-calibrated inference. Below this floor the share is reported but limits are NA and status is insufficient_blocks.
max_blocks	Maximum physical blocks per sample, 1 to 4096.
max_candidates	Maximum candidates per sample, 1 to 64.
max_state_bytes	Maximum bytes of concurrently owned native moment buffers and fit copies. DuckDB owns SQL sorting, materialization and spill.
table_name	Optional output table; otherwise return a data frame.
overwrite	Whether to replace an existing output table.

Value

One row per sample, candidate and relationship model. IBD1 rows require exact `candidate_af == sample_af` at every input site, including zero-read and excluded sites. Columns include the bounded share (0 to 0.5), block-jackknife standard error and score-inverted limits, singles share, pair residual, relationship diagnostic, QC read/pair totals, calibration, score rank and status. The discordance ratio divides discordant pairs by `SUM(choose(depth, 2) * sample_af * (1 - sample_af))`. When `table_name` is supplied, returns invisible `TRUE`.

`rduckhts_depth_bins` *Correct Bin Read Counts for GC Content*

Description

Gives each bin of each sample an expected read count from the median read rate of the sample's autosomal training bins in the same GC stratum, so that 'ratio = reads / expected' is about 1 for two copies, 0.5 for one and 1.5 for three. The output feeds [rduckhts_depth_region_fraction()] and [rduckhts_depth_sex_fraction()]. Normalization uses eligible autosomal bins with positive reads; empty bins retain zero ratios when their bin and stratum rules pass.

Usage

```
rduckhts_depth_bins(
  con,
  bins_table,
  min_mappability = 0.9,
  gc_step = 0.01,
  min_stratum_bins = 10,
  outlier_quantile = 0.01,
  max_ratio = 4,
  table_name = NULL,
  overwrite = FALSE
)
```

Arguments

`con`	A DuckDB connection with DuckHTS loaded.
`bins_table`	Name of a table or view with 'sample_id', 'chrom', 'start', 'end' (zero-based, half-open), 'reads' and 'gc' (the G+C share of the A, C, G and T reference bases, from 0 to 1), and optionally 'mappability' (0 to 1).
`min_mappability`	Bins with a lower mappability are not kept.
`gc_step`	Width of a GC stratum, in \[1/2147483647, 1\], so the stratum number fits the integer output column.
`min_stratum_bins`	Training bins a stratum needs, a whole number.

`outlier_quantile`
: Training bins outside this quantile of the read rate, and its complement, set no stratum median; in \[0, 0.5).

`max_ratio`
: Bins with a higher ratio are not kept; above 1.

`table_name`
: Optional table to create instead of returning a data frame.

`overwrite`
: Whether ‘table_name‘ may replace an existing table.

Details

The GC stratum of a bin is ‘floor(gc / gc_step)‘. Training bins are autosomal bins that pass the bin rules, hold at least one read and have a read rate inside the ‘outlier_quantile‘ and ‘1 - outlier_quantile‘ quantiles of the sample. A bin is kept unless it lacks a count or a GC value, has a mappability below ‘min_mappability‘ (when the relation has a ‘mappability‘ column), falls in a stratum with fewer than ‘min_stratum_bins‘ training bins, or has a ratio above ‘max_ratio‘; ‘reason‘ names the rule. Only ‘max_ratio‘ looks at a bin’s count, so a bin with no reads stays kept. Autosomal means a contig whose ‘duckhts_contig_key()‘ is not X, Y or MT. Restrict contigs and remove problem bins in SQL first.

Value

A data frame (or invisible ‘TRUE‘ when ‘table_name‘ is given) with one row per input row: ‘sample_id‘, ‘chrom‘, ‘start‘, ‘end‘, ‘reads‘, ‘gc‘, ‘mappability‘, ‘gc_stratum‘, ‘stratum_bins‘, ‘expected‘, ‘ratio‘, ‘log2_ratio‘, ‘reason‘, ‘kept‘ and ‘method‘.

See Also

[rduckhts_bam_bin_counts()], [rduckhts_fasta_nuc()]

Examples

```
con <- rduckhts_connect()
DBI::dbExecute(con, paste(
  "CREATE TEMP TABLE bins AS SELECT 'S1' AS sample_id, CAST(c AS VARCHAR) AS chrom,",
  "b * 1000000 AS start, (b + 1) * 1000000 AS \"end\",",
  "0.35 + 0.25 * ((b * 37 + c * 11) % 100) / 100.0 AS gc,",
  "CAST(600 + 4 * ((b * 13 + c * 7) % 11) AS BIGINT) AS reads",
  "FROM range(1, 23) t(c), range(0, 100) r(b)"
))
head(rduckhts_depth_bins(con, "bins")[, c("chrom", "start", "reads", "ratio", "kept")])
DBI::dbDisconnect(con, shutdown = TRUE)
```

---

rduckhts_depth_region_fraction

Estimate the Share of Cells with a One-Copy Change of a Region

---

Description

For each sample and named region, compares the depth of the region's kept bins with the depth of the sample's other kept autosomal bins: 'ratio = (C_r / E_r) / (C_b / E_b)' with 'C' and 'E' the sums of reads and expected counts, and 'shift = 2 * (ratio - 1)', the share of cells with a one-copy gain (positive) or loss (negative) relative to two copies when the baseline is unchanged. A one-copy change in every cell gives 1 or -1.

Usage

```
rduckhts_depth_region_fraction(
  con,
  bins_table,
  regions_table,
  reference_table = NULL,
  table_name = NULL,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
bins_table	Name of a table or view of corrected bins with 'sample_id', 'chrom', 'start', 'end', 'reads', 'expected' and 'kept', such as the output of [rduckhts_depth_bins()].
regions_table	Name of a table or view with 'region', 'chrom', 'start' and 'end' (zero-based, half-open); rows with one name form one region, and a bin belongs to it when its midpoint lies in one of its intervals.
reference_table	Optional name of a table or view of corrected bins of reference samples, in the layout of 'bins_table'.
table_name	Optional table to create instead of returning a data frame.
overwrite	Whether 'table_name' may replace an existing table.

Details

Without 'reference_table', the interval treats bin counts as independent and quasi-Poisson, with the dispersion of the baseline bins, and is 'shift +- 1.959964 shift_se'. With 'reference_table', corrected bins of reference samples, the reported 'shift' is the sample's shift minus the mean shift of the references (leaving out the tested sample if it is a reference), and the interval uses Student's t with 'reference_samples - 1' degrees of freedom. Other changes in the sample move the baseline.

Value

A data frame (or invisible 'TRUE' when 'table_name' is given) with one row per sample and region: 'sample_id', 'region', 'bins', 'baseline_bins', 'ratio', 'shift', 'shift_se', 'shift_low', 'shift_high', 'dispersion', 'reference_shift', 'reference_samples', 'status' ('ok', 'no_bins', 'no_baseline' or 'few_reference_samples') and 'method'.

Examples

```
con <- rduckhts_connect()
DBI::dbExecute(con, paste(
  "CREATE TEMP TABLE bins AS SELECT 'S1' AS sample_id, CAST(c AS VARCHAR) AS chrom,",
  "b * 1000000 AS start, (b + 1) * 1000000 AS \"end\",",
  "0.35 + 0.25 * ((b * 37 + c * 11) % 100) / 100.0 AS gc,",
  "CAST((600 + 4 * ((b * 13 + c * 7) % 11)) *",
  "CASE WHEN c = 8 THEN 1.1 ELSE 1 END AS BIGINT) AS reads",
  "FROM range(1, 23) t(c), range(0, 100) r(b)"
))
rduckhts_depth_bins(con, "bins", table_name = "corrected")
DBI::dbExecute(con, paste(
  "CREATE TEMP TABLE chromosomes AS SELECT CAST(c AS VARCHAR) AS region,",
  "CAST(c AS VARCHAR) AS chrom, 0 AS start, 100000000 AS \"end\" FROM range(1, 23) t(c)"
))
fractions <- rduckhts_depth_region_fraction(con, "corrected", "chromosomes")
fractions[fractions$region == "8", c("region", "shift", "shift_low", "shift_high")]
DBI::dbDisconnect(con, shutdown = TRUE)
```

---

rduckhts_depth_sex_fraction

Estimate the Share of a Second Genome of the Other Sex from X and Y Depth

---

Description

Treats each sample as a major genome and a second genome of the other sex with a DNA share ‘f‘ below 0.5. ‘x_copies‘ and ‘y_copies‘ are twice the depth ratio of the kept X or Y bins to the kept autosomal bins. For an XX major genome ‘x_copies = 2 - f‘ and ‘y_copies = f‘; for an XY one ‘x_copies = 1 + f‘ and ‘y_copies = 1 - f‘. The major genome is XX when ‘x_copies >= 1.5‘. ‘fraction_x‘ and ‘fraction_y‘ are the two estimates, unclipped; ‘fraction‘ combines them by inverse variance and ‘agreement_z‘ compares them, since an X or Y copy change of the major genome moves them differently.

Usage

```
rduckhts_depth_sex_fraction(
  con,
  bins_table,
  reference_table = NULL,
  table_name = NULL,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
bins_table	Name of a table or view of corrected bins, as for [rduckhts_depth_region_fraction()].

`reference_table`
: Optional name of a table or view of corrected bins of reference samples with a ‘sex‘ column, ‘XX‘ or ‘XY‘.

`table_name`
: Optional table to create instead of returning a data frame.

`overwrite`
: Whether ‘table_name‘ may replace an existing table.

Details

Without ‘reference_table‘, one X copy is 1, two are 2, one Y copy is 1 and a genome without Y has no Y depth. With ‘reference_table‘, its XY samples set the depth of one X and one Y copy, and its XX samples, when present, set two X copies and the Y background; the tested sample is left out of its own reference. Remove the pseudo-autosomal regions of X, and Y bins that are not single-copy, in SQL first.

Value

A data frame (or invisible ‘TRUE‘ when ‘table_name‘ is given) with one row per sample: ‘sample_id‘, ‘autosomal_bins‘, ‘x_bins‘, ‘y_bins‘, ‘dispersion‘, ‘x_copies‘, ‘x_copies_se‘, ‘y_copies‘, ‘y_copies_se‘, ‘major_sex‘, ‘fraction_x‘, ‘fraction_x_se‘, ‘fraction_y‘, ‘fraction_y_se‘, ‘fraction‘, ‘fraction_se‘, ‘fraction_low‘, ‘fraction_high‘, ‘agreement_z‘, ‘reference_samples‘, ‘status‘ (‘ok‘, ‘no_autosomal_bins‘, ‘no_xy_reference‘, ‘bad_reference‘, ‘no_sex_bins‘, ‘no_x_bins‘, ‘no_y_bins‘ or ‘x_y_disagree‘) and ‘method‘.

Examples

```
con <- rduckhts_connect()
DBI::dbExecute(con, paste(
  "CREATE TEMP TABLE bins AS SELECT 'S1' AS sample_id, chrom,",
  "b * 1000000 AS start, (b + 1) * 1000000 AS \"end\",",
  "0.35 + 0.25 * ((b * 37 + length(chrom) * 11) % 100) / 100.0 AS gc,",
  "CAST((600 + 4 * ((b * 13) % 11)) * CASE chrom WHEN 'X' THEN 0.95 WHEN 'Y' THEN 0.05",
  "ELSE 1 END AS BIGINT) AS reads",
  "FROM (SELECT CAST(c AS VARCHAR) AS chrom FROM range(1, 23) t(c)",
  "UNION ALL SELECT 'X' UNION ALL SELECT 'Y') g, range(0, 60) r(b)"
))
rduckhts_depth_bins(con, "bins", table_name = "corrected")
rduckhts_depth_sex_fraction(con, "corrected")[, c("major_sex", "fraction", "status")]
DBI::dbDisconnect(con, shutdown = TRUE)
```

---

rduckhts_depth_state_solutions

Decode Tumour Fraction, Ploidy and Copy-Number Segments from Corrected Bins

---

Description

Fits the copy-number state model of ichorCNA (Adalsteinsson, Ha, Freeman et al. 2017; GavinHaLab/ichorCNA tag v0.6.1) to each sample's kept corrected bins: a hidden Markov model over copy numbers 1 to 'max_cn' with a Student's t emission on the log ratio, whose state means follow from the normal fraction 'n' and tumour ploidy 'phi', fitted by expectation maximisation with the upstream priors from every start of 'ploidies' x 'normal_fractions'. Each start gives one solution, ranked by its objective (data log-likelihood plus log prior). [rduckhts_depth_state_solutions()] returns the solutions; [rduckhts_depth_state_segments()] returns their Viterbi segments.

Usage

```
rduckhts_depth_state_solutions(
  con,
  bins_table,
  normal_fractions = c(0.5, 0.6, 0.7, 0.8, 0.9),
  ploidies = c(2, 3),
  max_cn = 7,
  txn_e = 0.9999999,
  txn_strength = 1e+07,
  max_iter = 20,
  min_segment_bins = 50,
  alt_fraction_threshold = 0.05,
  close_margin = 10,
  max_bins = 1e+06,
  table_name = NULL,
  overwrite = FALSE
)

rduckhts_depth_state_segments(
  con,
  bins_table,
  normal_fractions = c(0.5, 0.6, 0.7, 0.8, 0.9),
  ploidies = c(2, 3),
  max_cn = 7,
  txn_e = 0.9999999,
  txn_strength = 1e+07,
  max_iter = 20,
  min_segment_bins = 50,
  alt_fraction_threshold = 0.05,
  close_margin = 10,
  max_bins = 1e+06,
  table_name = NULL,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.

`bins_table` Name of a table or view of corrected bins with ‘sample_id‘, ‘chrom‘, ‘start‘, ‘end‘, ‘log2_ratio‘ and ‘kept‘, such as the output of [rduckhts_depth_bins()]. Restrict contigs and remove problem bins in SQL first.

`normal_fractions` Starts of the normal fraction, each in (0, 1).

`ploidies` Starts of the tumour ploidy, each in \[1, 10\]; the EM starts from their floor. A normal start of 0.95 runs only at ploidy 2, as upstream.

`max_cn` Highest copy-number state, a whole number in \[2, 20\].

`txn_e` Self-transition probability in (0, 1).

`txn_strength` Transition pseudo-counts, positive.

`max_iter` EM iteration cap, a whole number in \[2, 1000\].

`min_segment_bins`, `alt_fraction_threshold` The guard’s two thresholds.

`close_margin` Objective difference below which a lower-ranked solution is ‘close_to_best‘.

`max_bins` Kept bins one sample may hold; the model holds about ‘max_cn * 28 + 40‘ bytes a bin.

`table_name` Optional table to create instead of returning a data frame.

`overwrite` Whether ‘table_name‘ may replace an existing table.

Details

The model learns from autosomal bins (contigs whose ‘duckhts_contig_key()‘ is not X, Y or MT) and decodes every kept bin. A kept bin without a ratio takes every state with likelihood 1, as upstream carries it. Upstream’s guard sets the fraction to zero (‘status‘ ‘guard_zero‘) when the longest altered training segment has at most ‘min_segment_bins‘ bins and at most ‘alt_fraction_threshold‘ of the training bins are altered; a solution ranked below the best with an objective within ‘close_margin‘ of it is ‘close_to_best‘, the rest are ‘ok‘. A sample without copy-number change therefore reports no fraction, by the guard or by a fit whose normal fraction is near 1; in that fit the state means lie within the noise of each other and whole chromosomes can take any state, so its copy numbers are not evidence. Fraction and ploidy trade against each other, so all solutions are returned; a ploidy-3 explanation of a ploidy-2 genome usually ranks second. A genome with one altered region does not identify the fraction either: a one-copy gain in a share ‘m‘ of cells and a two-copy gain in ‘m / 2‘ give the same ratio, and the prior decides, as in upstream. The port and its kept upstream quirks are documented in the DuckHTS catalog entry and ‘design/depth_fraction_lane.md‘.

Value

A data frame (or invisible ‘TRUE‘ when ‘table_name‘ is given). Solutions have one row per sample and start: ‘sample_id‘, ‘solution‘ (rank, 1 is best), ‘start_normal‘, ‘start_ploidy‘, ‘normal_fraction‘, ‘tumour_fraction‘, ‘ploidy‘, ‘log_likelihood‘, ‘altered_fraction‘, ‘iterations‘, ‘status‘, ‘segments‘, ‘bins‘, ‘upstream‘ and ‘method‘. Segments have one row per sample, solution and segment: the solution columns through ‘status‘, then ‘chrom‘, ‘start‘, ‘end‘, ‘bins‘, ‘state‘ (the copy number of the Viterbi state), ‘copy_number‘ (after upstream’s integer correction of high-level amplifications), ‘median_log2_ratio‘, ‘upstream‘ and ‘method‘.

Examples

```
con <- rduckhts_connect()
DBI::dbExecute(con, paste(
  "CREATE TEMP TABLE bins AS SELECT 'S1' AS sample_id, CAST(c AS VARCHAR) AS chrom,",
  "b * 1000000 AS start, (b + 1) * 1000000 AS \"end\", true AS kept,",
  "log2(CASE WHEN c = 8 THEN 1.15 ELSE 1 END) +",
  "0.05 * (((b * 37 + c * 11) % 100) / 50.0 - 1) AS log2_ratio",
  "FROM range(1, 23) t(c), range(0, CASE WHEN c = 8 THEN 60 ELSE 20 END) r(b)"
))
# Chromosome 8 has 60 bins so the gain passes the 50-bin guard. One start
# keeps the example short; the default grid has ten.
best <- rduckhts_depth_state_solutions(con, "bins", normal_fractions = 0.7, ploidies = 2)
best[, c("tumour_fraction", "ploidy", "status")]
segments <- rduckhts_depth_state_segments(con, "bins", normal_fractions = 0.7, ploidies = 2)
segments[segments$chrom == "8", c("chrom", "copy_number", "bins")]
DBI::dbDisconnect(con, shutdown = TRUE)
```

---

rduckhts_detect_quality_encoding

Detect FASTQ Quality Encoding

Description

Inspects a FASTQ file's observed quality ASCII range and reports compatible legacy encodings with a heuristic guessed encoding.

Usage

```
rduckhts_detect_quality_encoding(con, path, max_records = 10000)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
path	Path to the FASTQ file
max_records	Maximum number of records to inspect

Value

A data frame with the detected quality encoding summary

rduckhts_fasta *Create FASTA Table*

Description

Creates a DuckDB table from FASTA files using the DuckHTS extension.

Usage

```
rduckhts_fasta(
  con,
  table_name,
  path,
  region = NULL,
  index_path = NULL,
  gzi_path = NULL,
  sequence_encoding = NULL,
  scan_mode = NULL,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
table_name	Name for the created table
path	Path to the FASTA file
region	Optional genomic region (e.g., "chr1:1000-2000" or "chr1:1-10,chr2:5-20")
index_path	Optional explicit path to FASTA index file (.fai)
gzi_path	Optional explicit BGZF FASTA block index path (.gzi) for bgzipped FASTA inputs when the sidecar is not colocated with the FASTA.
sequence_encoding	Character. Sequence encoding for the SEQUENCE column: `"string"` (default) returns decoded bases as `VARCHAR`; `"nt16"` returns raw htslib nt16 4-bit codes as `UTINYINT[]`.
scan_mode	Optional scan mode. Use `"auto"` (default extension behavior) or `"sequential"` to force full-file streaming/counting instead of index-backed count paths. Sequential mode is incompatible with `region`.
overwrite	Logical. If TRUE, overwrites existing table

Value

Invisible TRUE on success

rduckhts_fasta_index *Build FASTA Index*

Description

Builds a FASTA index (.fai) using the DuckHTS extension.

Usage

```
rduckhts_fasta_index(con, path, index_path = NULL)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
path	Path to the FASTA file
index_path	Optional explicit output path for FASTA index file (.fai)

Value

A data frame with columns ‘success‘ and ‘index_path‘

rduckhts_fasta_multi *Read multiple FASTA files into a DuckDB table*

Description

Read and combine multiple FASTA files via UNION ALL BY NAME, materialising the result as a DuckDB table. Each row includes a `filename` column identifying its source file.

Usage

```
rduckhts_fasta_multi(
  con,
  table_name,
  files,
  region = NULL,
  index_path = NULL,
  gzi_path = NULL,
  sequence_encoding = NULL,
  scan_mode = NULL,
  .params = NULL,
  overwrite = FALSE
)
```

Arguments

con	A DBI connection to DuckDB with the duckhts extension loaded.
table_name	Name of the DuckDB table to create.
files	Character vector of file paths or glob patterns.
region	Optional region string.
index_path	Optional index file path.
gzi_path	Optional explicit BGZF FASTA block index path (.gzi).
sequence_encoding	Optional sequence encoding.
scan_mode	Optional scan mode ("auto" or "sequential").
.params	Optional data.frame with per-file parameter overrides.
overwrite	Logical; if TRUE, replace an existing table.

Value

Invisible TRUE on success.

---

rduckhts_fasta_nuc *Compute FASTA Interval Nucleotide Composition*

Description

Computes bedtools nuc-style nucleotide composition over either a BED file or generated fixed-width bins.

Usage

```
rduckhts_fasta_nuc(
  con,
  path,
  bed_path = NULL,
  bin_width = NULL,
  region = NULL,
  index_path = NULL,
  gzi_path = NULL,
  bed_index_path = NULL,
  include_seq = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
path	Path to the FASTA file
bed_path	Optional BED path. Supply exactly one of ‘bed_path‘ or ‘bin_width‘.
bin_width	Optional fixed bin width in base pairs
region	Optional FASTA region filter
index_path	Optional explicit FASTA index path
gzi_path	Optional explicit BGZF FASTA block index path (.gzi) for bgzipped FASTA inputs when the sidecar is not colocated with the FASTA.
bed_index_path	Optional explicit BED tabix index path
include_seq	Include the fetched interval sequence

Value

A data frame with interval composition statistics

---

rduckhts_fastq *Create FASTQ Table*

Description

Creates a DuckDB table from FASTQ files using the DuckHTS extension.

Usage

```
rduckhts_fastq(
  con,
  table_name,
  path,
  mate_path = NULL,
  interleaved = FALSE,
  sequence_encoding = NULL,
  quality_representation = NULL,
  input_quality_encoding = NULL,
  scan_mode = NULL,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
table_name	Name for the created table
path	Path to the FASTQ file

`mate_path` Optional path to mate file for paired reads

`interleaved` Logical indicating if file is interleaved paired reads

`sequence_encoding`
Character. Sequence encoding for the SEQUENCE column: `"string"` (default) returns decoded bases as `VARCHAR`; `"nt16"` returns raw htslib nt16 4-bit codes as `UTINYINT[]`.

`quality_representation`
Character. Quality representation for the QUALITY column: `"string"` (default) returns canonical Phred+33 text; `"phred"` returns raw Phred values as `UTINYINT[]`.

`input_quality_encoding`
Character. Input FASTQ quality encoding: `"phred33"` (default FASTQ convention), `"auto"`, `"phred64"`, or `"solexa64"`.

`scan_mode` Optional scan mode. Use `"auto"` (default extension behavior) or `"sequential"` to force raw streaming/counting instead of index-backed count paths.

`overwrite` Logical. If TRUE, overwrites existing table

Value

Invisible TRUE on success

`rduckhts_fastq_multi` *Read multiple FASTQ files into a DuckDB table*

Description

Read and combine multiple FASTQ files via UNION ALL BY NAME, materialising the result as a DuckDB table. Each row includes a `filename` column identifying its source file.

Usage

```
rduckhts_fastq_multi(
  con,
  table_name,
  files,
  mate_path = NULL,
  interleaved = FALSE,
  sequence_encoding = NULL,
  quality_representation = NULL,
  input_quality_encoding = NULL,
  scan_mode = NULL,
  .params = NULL,
  overwrite = FALSE
)
```

Arguments

con	A DBI connection to DuckDB with the duckhts extension loaded.
table_name	Name of the DuckDB table to create.
files	Character vector of file paths or glob patterns.
mate_path	Optional mate file path (for paired-end).
interleaved	Logical; TRUE if file contains interleaved paired reads.
sequence_encoding	Optional sequence encoding.
quality_representation	Optional quality representation.
input_quality_encoding	Optional input quality encoding override.
scan_mode	Optional scan mode ("auto" or "sequential").
.params	Optional data.frame with per-file parameter overrides.
overwrite	Logical; if TRUE, replace an existing table.

Value

Invisible TRUE on success.

---

rduckhts_functions *List DuckHTS Extension Functions*

Description

Returns the package-bundled function catalog generated from the top-level functions.yaml manifest in the duckhts repository.

Usage

```
rduckhts_functions(category = NULL, kind = NULL)
```

Arguments

category	Optional function category filter.
kind	Optional function kind filter such as "scalar", "table", or "table_macro".

Value

A data frame describing the extension functions, including the DuckDB function name, kind, category, signature, return type, optional R helper wrapper, short description, and example SQL.

Examples

```
catalog <- rduckhts_functions()
subset(catalog, category == "Sequence UDFs", select = c("name", "description"))
subset(rduckhts_functions(kind = "table"), select = c("name", "r_wrapper"))
```

---

rduckhts_genbank *Create GenBank Feature Table*

Description

Creates a DuckDB table from a GenBank flat file using the DuckHTS extension. Features are emitted in `read_gff`'s column shape, so a GenBank record substitutes for a GFF without a schema change. Locations built with `join()` or `order()` give one row per segment in biological order, `complement(...)` sets strand `"-"`, and the GFF3 phase of each CDS segment is carried from `/codon_start` across segments. `Parent` links a feature to the gene sharing its `/locus_tag` wherever that gene appears in the record, repeated qualifiers become one key with comma-joined values, and valueless qualifiers read `true`. Records stream one at a time, and a record without a terminating `//` or with a malformed location is an error naming the feature and line.

Usage

```
rduckhts_genbank(
  con,
  table_name = NULL,
  path,
  attributes_map = FALSE,
  overwrite = FALSE,
  attributes = NULL
)
```

Arguments

`con`	A DuckDB connection with DuckHTS loaded
`table_name`	Name for the created table, or `NULL` to create the `genbank_data` view
`path`	Path to the GenBank flat file, optionally bgzipped
`attributes_map`	Logical. If `TRUE`, add a parsed `MAP(VARCHAR, VARCHAR)` column alongside the raw attribute string
`overwrite`	Logical. If TRUE, overwrites an existing table
`attributes`	Character vector of qualifier keys to expose as VARCHAR columns. Each equals `attributes_map[key]`, including comma-joined repeated qualifiers and `NA` for an absent key, and is computed only when the query selects it. Key matching is case-sensitive: `"ec_number"` does not match `/EC_number`

Value

Invisible TRUE on success

rduckhts_genbank_to_fasta *Write GenBank Sequence as FASTA*

Description

Writes the ORIGIN sequence of each record in a GenBank flat file to a FASTA file using the DuckHTS extension. Records are written under the same name `rduckhts_genbank` reports as `seqname`, so feature coordinates land on the contig of that name; the DEFINITION follows the name without its trailing period, as in NCBI's FASTA export. The FASTA is written to a temporary file beside `output_path` and renamed into place only after the input has been read to a clean end, so a failure never leaves a partial output and an existing file is never lost. Records without an ORIGIN block are skipped; zero written records is an error.

Usage

```
rduckhts_genbank_to_fasta(
  con,
  path,
  output_path = NULL,
  line_width = 70,
  overwrite = FALSE
)
```

Arguments

`con`	A DuckDB connection with DuckHTS loaded
`path`	Path to the GenBank flat file, optionally bgzipped
`output_path`	Optional explicit output path for the FASTA file, one non-empty string; defaults to `path` with `.fa` appended
`line_width`	Sequence characters per output line, a whole number between 1 and `.Machine$integer.max`
`overwrite`	Logical. If `TRUE`, replace an existing output file; otherwise an existing output is an error

Value

A data frame with columns `success`, `output_path` and `records_written`

rduckhts_geno *Read Record-Major Genotypes*

Description

Read typed GT/PS calls without repeating variant text per sample. Each record has a zero-based scan-local ‘record_index‘ and a list of calls with original-header sample indices, nullable allele indices, per-slot phase bits, and nullable scalar phase sets. An absent GT has NULL allele/phase lists; a missing allele still occupies a slot. Phase bits follow HTSlib decoding, including its leading-slot convention for VCF versions before 4.4. PS cardinality excludes vector-end padding retained after sample selection.

Usage

```
rduckhts_geno(
  con,
  table_name = NULL,
  path,
  region = NULL,
  index_path = NULL,
  samples = NULL,
  non_reference_only = FALSE,
  scan_mode = "auto",
  decompression_threads = 0,
  decode_error_policy = "null",
  overwrite = FALSE,
  format_fields = NULL,
  raw_gt = FALSE,
  include_filter = FALSE,
  regions_var = NULL
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
table_name	Optional table to create; ‘NULL‘ returns a data frame.
path	Path to the VCF/BCF file
region	Optional genomic region (e.g., "chr1:1000-2000")
index_path	Optional explicit path to index file (.csi/.tbi)
samples	Optional HTSlib sample selector: ‘NULL‘ or ‘"-"‘ keeps all, ‘""‘ keeps none, comma-separated names include samples, and a leading ‘"^"‘ excludes them. Unknown names error; selected samples retain header order.
non_reference_only	Omit calls without any called alternate allele. This does not infer phase or remove variant records.

`scan_mode` Optional scan mode. Use `"auto"` (default extension behavior) or `"sequential"` to force full-file streaming instead of index-backed count/parallel scan paths. Sequential mode is incompatible with ‘region‘.

`decompression_threads` Integer. Number of htslib decompression worker threads per file handle. Default ‘0‘. Use ‘0‘ to keep BCF/VCF reads single-threaded.

`decode_error_policy` Character. VCF/BCF decode policy: `"null"` returns NULL for header-vs-payload type clashes or oversized numeric scalars, `"warn"` emits a DuckHTS warning and returns NULL, and `"error"` raises a DuckDB/R error. Missing elements count toward scalar cardinality; vector-end padding does not. A malformed FORMAT tag is withheld for every selected sample on that record. Physical read errors and OOM always fail.

`overwrite` Logical. If TRUE, overwrites existing table

`format_fields` Character vector of extra FORMAT tags, for example ‘c("AD", "DP", "GQ")‘. Selected fields are typed members of each call’s ‘format‘ struct using the header’s Type and Number. Missing elements retain their positions; no allele normalization or depth inference is performed. NULL or an empty vector keeps the default GT/PS schema. Unknown, empty, missing and case-insensitively duplicate names error; GT and PS are already exposed by the typed call fields and cannot be selected again. Tag lookup uses exact header spelling: declared lowercase ‘gt‘ and ‘ps‘ are distinct extra fields. Selected names must not collide under DuckDB’s case-insensitive struct-member lookup.

`raw_gt` Retain exact original VCF genotype text in each call’s ‘raw_gt‘ member. The default ‘FALSE‘ keeps the typed-call schema unchanged. ‘TRUE‘ preserves leading phase markers, mixed separators and allele spelling; an absent GT is ‘NULL‘, while a literal missing ‘.‘ remains text. BCF input errors because its encoded genotypes do not retain the original text.

`include_filter` Add the physical record’s ‘FILTER‘ as a ‘VARCHAR[]‘ column. ‘PASS‘ is ‘c("PASS")‘; an unapplied ‘.‘ filter is ‘NULL‘; named failing filters retain header order. The default keeps the existing schema.

`regions_var` Optional name of a session variable on ‘con‘ holding typed intervals, ‘STRUCT(chrom VARCHAR, start BIGINT, "end" BIGINT)[]‘, with 0-based half-open coordinates. See [rduckhts_bcf()] for the contract; the same rules apply. [rduckhts_geno_sites()] prepares such a plan from requested sites and joins exact alleles.

Details

Full scans preserve the input stream when assigning ordinals. Indexed regions use HTSlib’s union order and start a new ordinal at zero. Use SQL ‘ORDER BY record_index‘ when order matters; the ordinal is not a persistent file locator. Empty sample selection and sparse calls preserve every selected variant row.

Value

A data frame when ‘table_name‘ is ‘NULL‘, otherwise invisible ‘TRUE‘.

See Also

[rduckhts_bcf_samples()], [rduckhts_geno_sites()]

Examples

```
con <- rduckhts_connect()
path <- system.file("extdata", "geno_calls.bcf", package = "Rduckhts")
rduckhts_geno(con, "calls", path, non_reference_only = TRUE)
DBI::dbGetQuery(con, "SELECT record_index, len(calls) AS n FROM calls ORDER BY record_index")
DBI::dbDisconnect(con, shutdown = TRUE)
```

---

rduckhts_geno_sites *Fetch Exact Alleles at Requested Sites*

Description

Fetch full-site genotypes for requested (chromosome, position, REF, ALT) alleles from an indexed VCF/BCF. Positions become 0-based half-open intervals for one native indexed scan through ‘read_geno(regions := ...)‘, then exact alleles are restored with an equality join. Everything runs on ‘con‘: the request snapshot and the plan variable are session-local, are removed on exit even after an error, and no private connection is opened.

Usage

```
rduckhts_geno_sites(
  con,
  path,
  sites,
  source_build,
  table_name = NULL,
  samples = NULL,
  format_fields = NULL,
  index_path = NULL,
  decode_error_policy = "error",
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
path	Path to the VCF/BCF file
sites	A relation name (character), a [DBI::Id()], or a [DBI::SQL()] query evaluated on ‘con‘, with columns ‘request_id‘ (unique, non-NULL), ‘build‘, ‘chrom‘, ‘pos‘ (integer type, at least 1), ‘ref‘ and ‘alt‘ (one allele, non-empty, no comma). Extra columns are preserved; the names ‘record_index‘, ‘full_alt‘, ‘alt_index‘, ‘calls‘ and ‘match_status‘ are reserved. Duplicate (chrom, pos, ref, alt) rows are separate requests.

`source_build` Single string that every ‘sites$build‘ must equal.

`table_name` Optional table to create with the result (a regular table, replaced only when ‘overwrite = TRUE‘); its name is returned invisibly. ‘NULL‘ returns a data frame.

`samples`, `index_path`, `format_fields`, `decode_error_policy`
See [rduckhts_geno()]. ‘format_fields‘ such as ‘c("GP", "DS", "HS")‘ are kept for the whole site without any allele remapping. The decode policy defaults to ‘"error"‘ here.

`overwrite` Logical. If TRUE, overwrites existing table

Details

Coordinates in ‘sites‘ are 1-based VCF positions. Requested alleles are compared verbatim: nothing is trimmed, left-aligned, strand-flipped, split or lifted over, and ‘source_build‘ is only a caller-asserted label that every request must carry (contig names cannot prove assembly identity). The ‘match_status‘ labels are position-level diagnostics, not claims about normalization equivalence.

The result holds every request column followed by:

‘record_index‘ Scan-local record ordinal (not a persistent locator).

‘full_alt‘ The record’s complete ‘ALT‘ list in file order.

‘alt_index‘ 1-based ordinal of the matching ALT slot.

‘calls‘ The full-site ‘calls‘ list of [rduckhts_geno()], including ‘format_fields‘ members such as ‘GP‘, ‘DS‘ or ‘HS‘.

‘match_status‘ ‘"matched"‘: chrom, pos, REF and ALT equal a record’s ALT slot (one row per matching record and ALT slot, so physical duplicate records and duplicated ALT slots each return a row). ‘"allele_not_at_site"‘: a record at chrom and pos has an equal REF but none of its ALTs equals ‘alt‘. ‘"ref_mismatch"‘: records exist at chrom and pos but none has an equal REF. ‘"absent"‘: no record starts at chrom and pos (an unknown contig is absent).

Unmatched requests have one row with ‘NULL‘ record and call fields; no reference genotype is fabricated. Reference and missing calls at a matched site are returned because ‘non_reference_only‘ is fixed to ‘FALSE‘. Rows carry no implicit order: use ‘ORDER BY request_id, record_index, alt_index‘.

Value

A data frame, or invisibly ‘table_name‘ when it is given.

See Also

[rduckhts_geno()]

Examples

```
con <- rduckhts_connect()
path <- system.file("extdata", "geno_sites.bcf", package = "Rduckhts")
DBI::dbWriteTable(con, "requests", data.frame(
  request_id = 1:2, build = "GRCh38", chrom = "chr1", pos = c(100L, 100L),
```

```
  ref = "A", alt = c("G", "T")))
rduckhts_geno_sites(con, path, "requests", "GRCh38", format_fields = c("GP", "DS"))
DBI::dbDisconnect(con, shutdown = TRUE)
```

`rduckhts_gff` *Create GFF3 Table*

Description

Creates a DuckDB table from GFF3 files using the DuckHTS extension.

Usage

```
rduckhts_gff(
  con,
  table_name,
  path,
  region = NULL,
  index_path = NULL,
  header = NULL,
  header_names = NULL,
  auto_detect = NULL,
  column_types = NULL,
  scan_mode = NULL,
  attributes_map = FALSE,
  attributes_list = FALSE,
  attributes_pairs = FALSE,
  strict = FALSE,
  overwrite = FALSE,
  attributes = NULL
)
```

Arguments

`con`	A DuckDB connection with DuckHTS loaded
`table_name`	Name for the created table
`path`	Path to the GFF3 file
`region`	Optional genomic region (e.g., "chr1:1000-2000")
`index_path`	Optional explicit path to index file (.tbi/.csi)
`header`	Logical. If TRUE, use first non-meta line as column names
`header_names`	Character vector to override column names
`auto_detect`	Logical. If TRUE, infer basic numeric column types
`column_types`	Character vector of column types (e.g. "BIGINT", "VARCHAR")

scan_mode	Optional scan mode. Use "auto" (default extension behavior) or "sequential" to force full-file streaming/counting instead of index-backed count paths. Sequential mode is incompatible with region.
attributes_map	Logical. If TRUE, returns raw attributes as a scalar MAP column
attributes_list	Logical. If TRUE, returns attributes as MAP(VARCHAR, VARCHAR[])
attributes_pairs	Logical. If TRUE, returns attributes as a LIST of key/value/index structs
strict	Logical. If TRUE, enforce GFF3 structural validation while scanning
overwrite	Logical. If TRUE, overwrites existing table
attributes	Character vector of attribute keys to expose as VARCHAR columns. Key matching is case-sensitive: "id" does not match ID, and an unmatched key gives a column of NA

Value

Invisible TRUE on success

---

rduckhts_gff_convert_parquet

Convert GFF3 reader output to Parquet with DuckHTS metadata

Description

Thin DBI wrapper around extension macro 'duckhts_gff_convert_parquet_sql(...)'.

Usage

```
rduckhts_gff_convert_parquet(
  con,
  path,
  output,
  columns = NULL,
  region = NULL,
  index_path = NULL,
  header = NULL,
  header_names = NULL,
  auto_detect = NULL,
  column_types = NULL,
  attributes_map = FALSE,
  attributes_list = FALSE,
  attributes_pairs = FALSE,
  strict = FALSE,
  where = NULL,
  compression = "zstd",
```

```
  row_group_size = 100000L,
  partition_by = NULL,
  include_metadata = TRUE,
  header_text = NULL,
  metadata = NULL,
  metadata_json_file = NULL,
  write_format_version = "1",
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
path	Path or URI to the input GFF3 file.
output	Path to the output Parquet file or partitioned directory.
columns	Optional character vector of columns to include. Defaults to all columns.
region	Optional genomic region string for indexed inputs.
index_path	Optional explicit index path.
header	Logical; pass 'header := true/false' to 'read_gff(...)'.
header_names	Optional character vector of column names.
auto_detect	Logical; request type auto-detection.
column_types	Optional character vector of DuckDB column types.
attributes_map	Logical; expose GFF attributes as 'MAP(VARCHAR, VARCHAR)'.
attributes_list	Logical; expose attributes as 'MAP(VARCHAR, VARCHAR[])'.
attributes_pairs	Logical; expose attributes as key/value/index structs.
strict	Logical; enable strict GFF validation while scanning.
where	Optional SQL predicate applied to the reader output before conversion.
compression	Parquet compression, default '"zstd"'.
row_group_size	Parquet row group size.
partition_by	Optional character vector of output partition columns.
include_metadata	Logical; include DuckHTS Parquet KV metadata.
header_text	Optional corrected header text to store instead of the source header.
metadata	Optional named list/vector of extra metadata. This is the primary CRAN/offline-safe path for arbitrary metadata; values with the same names as DuckHTS defaults override the default values.
metadata_json_file	Optional path to a JSON file containing a top-level object of extra metadata. This requires DuckDB's 'json' extension to be available when the conversion SQL is generated; otherwise DuckDB will report its normal missing-extension error. Use 'metadata' for offline-safe metadata.

write_format_version
: DuckHTS Parquet write-format version string.

overwrite
: Logical; replace an existing output path. The wrapper checks existence through DuckDB 'glob(...)' where possible and passes the same flag to DuckDB 'COPY' for partitioned-output overwrite handling.

Value

Invisibly returns 'output'.

`rduckhts_gff_multi` *Read multiple GFF files into a DuckDB table*

Description

Read and combine multiple GFF3 files via UNION ALL BY NAME, materialising the result as a DuckDB table. Each row includes a `filename` column identifying its source file.

Usage

```
rduckhts_gff_multi(
  con,
  table_name,
  files,
  region = NULL,
  index_path = NULL,
  header = NULL,
  header_names = NULL,
  auto_detect = NULL,
  column_types = NULL,
  scan_mode = NULL,
  attributes_map = FALSE,
  attributes_list = FALSE,
  attributes_pairs = FALSE,
  strict = FALSE,
  .params = NULL,
  overwrite = FALSE
)
```

Arguments

con
: A DBI connection to DuckDB with the duckhts extension loaded.

table_name
: Name of the DuckDB table to create.

files
: Character vector of file paths or glob patterns.

region
: Optional region string.

index_path
: Optional index file path.

header	Logical or NULL; whether the file has a header line.
header_names	Character vector of column names.
auto_detect	Logical or NULL; enable type auto-detection.
column_types	Character vector of column type names.
scan_mode	Optional scan mode ("auto" or "sequential").
attributes_map	Logical; return raw attributes as a scalar MAP.
attributes_list	Logical; return attributes as MAP(VARCHAR, VARCHAR[]).
attributes_pairs	Logical; return attributes as a LIST of key/value/index structs.
strict	Logical; enforce GFF3 structural validation while scanning.
.params	Optional data.frame with per-file parameter overrides.
overwrite	Logical; if TRUE, replace an existing table.

Value

Invisible TRUE on success.

---

rduckhts_gtf *Create GTF Table*

Description

Creates a DuckDB table from GTF files using the DuckHTS extension.

Usage

```
rduckhts_gtf(
  con,
  table_name,
  path,
  region = NULL,
  index_path = NULL,
  header = NULL,
  header_names = NULL,
  auto_detect = NULL,
  column_types = NULL,
  scan_mode = NULL,
  attributes_map = FALSE,
  attributes_list = FALSE,
  attributes_pairs = FALSE,
  overwrite = FALSE,
  attributes = NULL
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
table_name	Name for the created table
path	Path to the GTF file
region	Optional genomic region (e.g., "chr1:1000-2000")
index_path	Optional explicit path to index file (.tbi/.csi)
header	Logical. If TRUE, use first non-meta line as column names
header_names	Character vector to override column names
auto_detect	Logical. If TRUE, infer basic numeric column types
column_types	Character vector of column types (e.g. "BIGINT", "VARCHAR")
scan_mode	Optional scan mode. Use "auto" (default extension behavior) or "sequential" to force full-file streaming/counting instead of index-backed count paths. Sequential mode is incompatible with region.
attributes_map	Logical. If TRUE, returns raw attributes as a scalar MAP column
attributes_list	Logical. If TRUE, returns attributes as MAP(VARCHAR, VARCHAR[])
attributes_pairs	Logical. If TRUE, returns attributes as a LIST of key/value/index structs
overwrite	Logical. If TRUE, overwrites existing table
attributes	Character vector of attribute keys to expose as VARCHAR columns. Key matching is case-sensitive: "id" does not match ID, and an unmatched key gives a column of NA

Value

Invisible TRUE on success

---

rduckhts_gtf_multi *Read multiple GTF files into a DuckDB table*

Description

Read and combine multiple GTF files via UNION ALL BY NAME, materialising the result as a DuckDB table. Each row includes a filename column identifying its source file.

Usage

```
rduckhts_gtf_multi(
  con,
  table_name,
  files,
  region = NULL,
```

```
  index_path = NULL,
  header = NULL,
  header_names = NULL,
  auto_detect = NULL,
  column_types = NULL,
  scan_mode = NULL,
  attributes_map = FALSE,
  attributes_list = FALSE,
  attributes_pairs = FALSE,
  .params = NULL,
  overwrite = FALSE
)
```

Arguments

con	A DBI connection to DuckDB with the duckhts extension loaded.
table_name	Name of the DuckDB table to create.
files	Character vector of file paths or glob patterns.
region	Optional region string.
index_path	Optional index file path.
header	Logical or NULL; whether the file has a header line.
header_names	Character vector of column names.
auto_detect	Logical or NULL; enable type auto-detection.
column_types	Character vector of column type names.
scan_mode	Optional scan mode ("auto" or "sequential").
attributes_map	Logical; return raw attributes as a scalar MAP.
attributes_list	Logical; return attributes as MAP(VARCHAR, VARCHAR[]).
attributes_pairs	Logical; return attributes as a LIST of key/value/index structs.
.params	Optional data.frame with per-file parameter overrides.
overwrite	Logical; if TRUE, replace an existing table.

Value

Invisible TRUE on success.

rduckhts_htslib_config *Get the Installed htslib Linking Contract*

Description

Resolves headers, an exact shared or static library, linker flags, enabled features, and build identity from the installed Rduckhts package. With validation enabled, the receipt and public headers are also compared with the htslib version reported by the loaded DuckHTS extension.

Usage

```
rduckhts_htslib_config(link = NULL, validate = TRUE)
```

Arguments

link	Either ‘"shared"‘ or ‘"static"‘. When omitted, use the link mode selected when this Rduckhts package was configured.
validate	Whether to validate installed files, header identity, and the loaded htslib runtime version.

Details

The shared contract is currently available on native Unix builds. MinGW and browser-wasm builds expose the static contract unless a shared htslib was explicitly built. Static consumers must review ‘static_license_note‘.

Value

An object of class ‘rduckhts_htslib_config‘. Its ‘cppflags‘ and ‘ldflags‘ elements can be consumed by a downstream package configure script; ‘duckdb_platform‘ records the extension footer platform and the remaining fields form the versioned build receipt.

Examples

```
## Not run:
config <- rduckhts_htslib_config()
config$cppflags
config$ldflags
config$features

## End(Not run)
```

rduckhts_htslib_info *Inspect the Loaded htslib Build*

Description

Returns the version and build features reported by the htslib library that is actually loaded with DuckHTS.

Usage

```
rduckhts_htslib_info(con)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.

Value

A one-row data frame with ‘version‘, ‘feature_bits‘, and ‘feature_string‘ columns.

Examples

```
## Not run:
con <- rduckhts_connect()
rduckhts_htslib_info(con)
DBI::dbDisconnect(con, shutdown = TRUE)

## End(Not run)
```

rduckhts_htslib_version *Return the Loaded htslib Version*

Description

Return the Loaded htslib Version

Usage

```
rduckhts_htslib_version(con)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.

Value

The runtime htslib semantic version as a character scalar.

rduckhts_hts_header *Read HTS Header Metadata*

Description

Reads file header records from HTS-supported formats using the DuckHTS extension.

Usage

```
rduckhts_hts_header(con, path, format = NULL, mode = NULL)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
path	Path to input HTS file
format	Optional format hint (e.g., "auto", "vcf", "bcf", "bam", "cram", "tabix")
mode	Header output mode: "parsed" (default), "raw", or "both"

Value

A data frame with parsed header metadata.

rduckhts_hts_index *Read HTS Index Metadata*

Description

Reads index metadata from HTS-supported index files via DuckHTS.

Usage

```
rduckhts_hts_index(con, path, format = NULL, index_path = NULL)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
path	Path to input HTS file
format	Optional format hint (e.g., "auto", "vcf", "bcf", "bam", "cram", "tabix")
index_path	Optional explicit path to index file

Value

A data frame with index metadata.

rduckhts_hts_index_raw *Read Raw HTS Index Blob*

Description

Returns raw index metadata blob data for a file index.

Usage

```
rduckhts_hts_index_raw(con, path, format = NULL, index_path = NULL)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
path	Path to input HTS file
format	Optional format hint
index_path	Optional explicit path to index file

Value

A data frame with raw index blob metadata.

rduckhts_hts_index_spans *Read HTS Index Spans*

Description

Returns index span-oriented metadata for planning range workloads.

Usage

```
rduckhts_hts_index_spans(con, path, format = NULL, index_path = NULL)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
path	Path to input HTS file
format	Optional format hint
index_path	Optional explicit path to index file

Value

A data frame with span-oriented index metadata.

rduckhts_install_macros *Install DuckHTS macros on a connection*

Description

Executes the ordered definitions exported by the loaded DuckHTS extension as connection-local TEMP macros. Use this on DBI or pool connections that did not come from `rduckhts_connect()`. The operation is idempotent and shadows same-named persistent macros without deleting them. Inspect old persistent macros in `duckdb_functions()` using the file's `database_name`; drop them explicitly if no longer wanted.

Usage

```
rduckhts_install_macros(con)
```

Arguments

`con`	A DBI connection with DuckHTS loaded.

Details

This function does not begin or commit a transaction. Inside a caller transaction its DDL participates in that transaction: rollback removes newly installed TEMP macros, and the caller can install them again after rollback. Call it separately for each connection, including pool connections.

Value

Invisibly returns the definitions SHA-256 digest.

rduckhts_liftover *Lift Over Variant Coordinates Against a Query*

Description

Applies the DuckHTS 'duckhts_liftover(...)' table macro to rows from a SQL query or table expression with chromosome and position columns, plus optional reference and alternate alleles.

Usage

```
rduckhts_liftover(
  con,
  query,
  chain_path,
  dst_fasta_ref,
  chrom_col = "chrom",
```

```
  pos_col = "pos",
  ref_col = NULL,
  alt_col = NULL,
  src_fasta_ref = NULL,
  max_snp_gap = 1,
  max_indel_inc = 250,
  lift_mt = FALSE,
  end_pos_col = NULL,
  no_left_align = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
query	SQL query or table expression to lift over
chain_path	Path to a UCSC chain file
dst_fasta_ref	Path to the destination FASTA reference
chrom_col	Source chromosome column name
pos_col	Source 1-based position column name
ref_col	Optional reference allele column name
alt_col	Optional alternate allele column name
src_fasta_ref	Optional source FASTA reference
max_snp_gap	Maximum chain block merge gap
max_indel_inc	Maximum indel anchor expansion
lift_mt	If FALSE (default), mitochondrial variants with matching source/destination contig lengths are passed through with only contig rename. If TRUE, MT variants are lifted through the chain like any other contig.
end_pos_col	Optional column name containing INFO/END positions (1-based) to lift alongside the primary position. When provided, the output includes a 'dest_end' column with the lifted end position.
no_left_align	If FALSE (default), lifted indels are left-aligned against the destination reference. Set TRUE to skip left-alignment, mirroring `--no-left-align` in `bcftools +liftover`.

Value

A data frame with source columns, lifted coordinates/alleles, and warnings.

rduckhts_load *Load DuckHTS Extension*

Description

Loads the DuckHTS extension into an existing DuckDB connection. This must be called before using HTS reader functions on a connection not created by `rduckhts_connect()`.

Usage

```
rduckhts_load(con, extension_path = NULL)
```

Arguments

`con`	A DuckDB connection object.
`extension_path`	Optional path to the DuckHTS extension file. If `NULL`, uses the extension bundled with Rduckhts.

Details

DuckDB 1.5.0 or newer is required. SQL registration errors retain the underlying DuckDB diagnostic in the initialization error.

The connection must permit unsigned extension loading. With current versions of the **duckdb** R package, its driver must also permit extension loading. Prefer `rduckhts_connect()` when Rduckhts owns the connection. File-backed connections receive TEMP macros on this connection after LOAD. Use `rduckhts_install_macros()` on additional connections.

Value

`TRUE` if the extension was loaded successfully.

Examples

```
con <- rduckhts_connect()
DBI::dbDisconnect(con, shutdown = TRUE)
```

rduckhts_mosdepth *Native mosdepth-Compatible Coverage Outputs*

Description

Writes native mosdepth-compatible coverage outputs for indexed BAM or CRAM input.

Usage

```
rduckhts_mosdepth(
  con,
  prefix,
  path,
  chrom = NULL,
  by = NULL,
  fasta = NULL,
  read_groups = NULL,
  no_per_base = FALSE,
  threads = 2,
  processing_threads = 2,
  flag = 1796,
  include_flag = 0,
  fast_mode = FALSE,
  fragment_mode = FALSE,
  use_median = FALSE,
  mapq = 0,
  min_frag_len = -1,
  max_frag_len = -1,
  precision_digits = 2,
  quantize = NULL,
  thresholds = NULL,
  index_path = NULL,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
prefix	Output prefix for the mosdepth-style files
path	Path to the input BAM or CRAM file
chrom	Optional chromosome name filter
by	Optional fixed-width window size as a string or a BED file path
fasta	Optional reference FASTA path for CRAM input when required
read_groups	Optional comma-separated read-group IDs, matching mosdepth's '-R'
no_per_base	Skip writing '{prefix}.per-base.bed.gz'

threads	Number of BAM decompression threads
processing_threads	Number of parallel contig processing threads (0 = sequential)
flag	Excluded SAM flag mask, matching mosdepth's '-F'
include_flag	Required SAM flag mask, matching mosdepth's '-i'
fast_mode	Logical. If 'TRUE', use mosdepth fast mode. Defaults to 'FALSE', matching upstream mosdepth.
fragment_mode	Logical. If 'TRUE', count full fragment insert spans for proper pairs, matching mosdepth's '-a'. Cannot be combined with 'fast_mode = TRUE'.
use_median	Logical. If 'TRUE', write 'by' region values as medians instead of means, matching mosdepth's '-m'.
mapq	Minimum mapping quality threshold
min_frag_len	Minimum absolute template length to keep, matching mosdepth's '-l'
max_frag_len	Maximum absolute template length to keep, matching mosdepth's '-u'
precision_digits	Number of decimal places to write in the text outputs
quantize	Optional mosdepth-style quantize specification such as '":1:4:"'
thresholds	Optional comma-separated coverage thresholds for 'by', matching mosdepth's '-T'
index_path	Optional explicit BAM index path
overwrite	Overwrite existing output files

Value

A data frame describing the written output paths

`rduckhts_munge` *Munge Summary Statistics Rows*

Description

Applies the DuckHTS 'duckhts_munge(...)' table macro to rows from a SQL query or table expression, using either an upstream-style preset, a named column map, or a two-column mapping file. When no mapping mode is provided, the bundled 'colheaders.tsv' alias file is used by default.

Usage

```
rduckhts_munge(
  con,
  query,
  fasta_ref = NULL,
  preset = NULL,
  column_map = NULL,
```

```
  column_map_file = NULL,
  iffy_tag = "IFFY",
  mismatch_tag = "REF_MISMATCH",
  ns = NULL,
  nc = NULL,
  ne = NULL
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
query	SQL query or table expression to normalize
fasta_ref	Path to the reference FASTA. When NULL (default), operates in fai-only mode: alleles pass through as-is without reference matching or allele swapping, matching upstream '–fai'-only behavior.
preset	Optional preset such as '"PLINK"', '"PLINK2"', '"REGENIE"', '"SAIGE"', '"BOLT"', '"METAL"', '"PGS"', or '"SSF"'
column_map	Optional named character vector mapping canonical munge names such as '"CHR"', '"BP"', '"A1"', '"A2"' to source column names
column_map_file	Optional path to a two-column TSV mapping file in the upstream 'source<TAB>canonical' format
iffy_tag	FILTER tag for ambiguous reference resolution
mismatch_tag	FILTER tag for reference mismatches
ns, nc, ne	Optional global overrides for sample counts

Value

A data frame with normalized GWAS-VCF-style variant/effect columns.

---

rduckhts_pileup *Create BAM Pileup Table*

Description

Creates a DuckDB table from a region-scoped BAM pileup using the DuckHTS extension. This is a compact base/quality pileup view backed by htslib's pileup engine; it is not samtools mpileup text parity.

Usage

```
rduckhts_pileup(
  con,
  table_name,
  path,
  region,
  index_path = NULL,
  min_mapq = 0,
  flag_mask = 1796,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
table_name	Name for the created table
path	Path to the BAM file
region	Required genomic region (e.g., "chr1:1000-2000")
index_path	Optional explicit path to the BAM index (.bai/.csi)
min_mapq	Minimum mapping quality to include in the pileup
flag_mask	Bitmask of SAM flags to exclude before pileup construction. The default 1796 matches samtools depth-style filtering of unmapped, secondary, QC-fail, and duplicate reads.
overwrite	Logical. If TRUE, overwrites existing table

Value

Invisible TRUE on success

---

rduckhts_roh *Find Runs of Homozygosity*

Description

Detect runs of homozygosity in a VCF or BCF with the two-state hidden Markov model of ‘bcftools roh‘: an autozygous state and a Hardy-Weinberg state, decoded with the Viterbi algorithm. Segments, marker counts and start and end positions equal ‘bcftools roh‘; quality is the mean forward-backward phred score, equal to the one decimal ‘bcftools‘ prints. The model is a port of ‘vcfroh.c‘ and ‘HMM.c‘ (MIT, Genome Research Ltd), run by the native ‘duckhts_roh_segments()‘ kernel over one list per sample and chromosome.

Usage

```
rduckhts_roh(
  con,
  path,
  af_tag = NULL,
  af_table = NULL,
  genetic_map = NULL,
  hw_to_az = 6.7e-08,
  az_to_hw = 5e-09,
  gt_error = NULL,
  rec_rate = NULL,
  samples = NULL,
  table_name = NULL,
  overwrite = FALSE,
  reference_table = NULL,
  proportions_table = NULL,
  af_clamp = 0.001,
  max_sites = 2e+07,
  max_site_bytes = 2^32,
  site_rule = "matching",
  af_order = NULL
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
path	One VCF/BCF path or URI.
af_tag	INFO tag holding the ALT allele frequency, for example ‘"AF"‘.
af_table	Name of a table or view of site allele frequencies, instead of ‘af_tag‘ or the ancestry inputs.
genetic_map	Optional name of a genetic-map table or view.
hw_to_az, az_to_hw	Per-base-pair transition probabilities Hardy-Weinberg to autozygous (‘-a‘) and back (‘-H‘), each in ‘[0, 1]‘.
gt_error	Optional phred error for GT-only emissions (‘-G‘), at least 0.
rec_rate	Optional constant recombination rate per base pair (‘-M‘).
samples	Optional HTSlib sample selector: comma-separated inclusion, leading ‘^‘ exclusion, ‘"-"‘ for all, or ‘""‘ for none.
table_name	Optional output table. ‘NULL‘ returns a data frame.
overwrite	Whether an existing output table may be replaced.
reference_table	Long-format reference with chromosome, position, allele_a, allele_b, group_id and frequency columns.
proportions_table	Long-format sample ancestry proportions with sample_id, group_id and proportion columns. Its group set must exactly match ‘reference_table‘.

`af_clamp` Clamp ancestry-tuned frequencies to ‘[af_clamp, 1-af_clamp]‘; zero disables clamping. Must be in ‘[0, 0.5)‘.

`max_sites` Most sites one sample and chromosome may hold, from 1 to 100,000,000. A larger group is an error.

`max_site_bytes` Bound on the bytes the site buffers hold at once (16 bytes per site, 24 for read counts). A decode does not grow a buffer when the bytes held by all ROH decodes in the process would pass its own ‘max_site_bytes‘; decodes that run at the same time with different values each apply their own. Exceeding it is an error: decode fewer samples per call or raise it.

`site_rule` Which record of a position listed in ‘af_table‘ gives its site. ‘"matching"‘ (the default) uses a record whose REF and ALT match a row of ‘af_table‘ (‘duckhts_roh_af_table‘). ‘"bcftools"‘ follows ‘bcftools roh –AF-file‘ (‘duckhts_roh_af_table_bcftools‘): in file order, a record of the other indel class than the row is passed over when the next record of the file has the same POS (on any contig); a record with no non-symbolic ALT, or several, claims nothing; and the first remaining record claims the position, which is skipped unless its alleles equal the row’s (also when its first ALT is symbolic, which bcftools would decode through the second ALT). It needs one ‘af_table‘ row per position, unless ‘af_order‘ gives file order, and a sorted, BGZF-compressed VCF or BCF, and reads the file a second time without samples. Only ‘af_table‘ takes ‘"bcftools"‘.

`af_order` Optional name of a column of ‘af_table‘ that gives file order, for example a line number added by the caller, or ‘FILE_OFFSET‘ when the frequencies come from ‘read_bcf()‘. With ‘site_rule = "bcftools"‘ only. Several rows may then share a position: the row with the least value is used, as ‘bcftools roh –AF-file‘ reads the first line of a position. A ‘NULL‘ value, a value repeated within a position, or a name that is not a column of ‘af_table‘ is an error.

Details

Allele frequencies come from an INFO tag (‘af_tag‘, like ‘–AF-tag‘, which must be declared ‘Type=Float,Number=A‘) or from a relation (‘af_table‘, like ‘–AF-file‘) with columns ‘chrom‘, ‘pos‘, ‘ref‘, ‘alt‘ and ‘af‘, matched to each record by chromosome, position, REF and its ALT alleles joined by commas. Sites without a usable frequency (absent, missing or exactly 0) are skipped, as are records with more than one ALT or none. With ‘af_table‘, ‘site_rule = "bcftools"‘ chooses the record of a repeated position as ‘bcftools roh –AF-file‘ does (see the argument).

Alternatively, ‘reference_table‘ and ‘proportions_table‘ select the ancestry-tuned path. The reference is long-format with columns ‘chromosome‘, ‘position‘, ‘allele_a‘, ‘allele_b‘, ‘group_id‘ and ‘frequency‘ (frequency of ‘allele_b‘); proportions have ‘sample_id‘, ‘group_id‘ and ‘proportion‘. Their group sets must match exactly, and each VCF sample must have a proportions row for every group. Per-site AF is the sum of each group’s frequency weighted by the sample’s proportion, with the proportions divided by their sum (which must be above 0 and at most 1, so ‘sum_to_one = FALSE‘ results and rounded proportions are accepted). REF=‘allele_a‘, ALT=‘allele_b‘ uses that AF; reversed alleles use ‘1 - AF‘. Reference alleles must be on the forward strand of the VCF’s assembly, as in a FASTA-anchored panel; no strand flip is attempted, so palindromic (A/T, C/G) sites are oriented by REF like any other, and a record whose alleles match neither order is not used and does not claim a repeated position. Contig names are matched with ‘duckhts_contig_key()‘ once per distinct name (one leading ‘chr‘ removed, M/MT written as MT, X and Y uppercased), so

‘chr1‘ and ‘1‘ match, as do ‘chrX‘ and ‘X‘; accessions, patches and numeric sex chromosomes are not mapped. ‘af_clamp‘ limits nonzero-clamp frequencies to ‘[af_clamp, 1-af_clamp]‘; the default keeps zero population frequencies from being treated as impossible, and zero disables clamping. Only called sites are frequency-weighted. Supply exactly one frequency source: ‘af_tag‘, ‘af_table‘, or both ancestry relations.

Genotype evidence is FORMAT/PL unless ‘gt_error‘ is given, which uses the diploid GT calls with that phred error (‘-G‘) and needs no PL. With no recombination map, transitions are the per-base-pair probabilities ‘hw_to_az‘ and ‘az_to_hw‘ compounded over the physical distance between sites, exactly as ‘bcftools roh‘ does without ‘-m‘ or ‘-M‘. ‘rec_rate‘ is a constant rate per base pair (‘-M‘). ‘genetic_map‘ names a relation with columns ‘chrom‘, ‘pos‘ and ‘cm‘ (cumulative centimorgans, the IMPUTE2 format), interpolated as ‘-m‘ does; chromosomes it does not cover are skipped.

Value

A data frame (or invisible ‘TRUE‘ when ‘table_name‘ is given) with one row per run: ‘sample‘, ‘chrom‘, ‘start‘ and ‘end‘ (one-based, inclusive, the first and last marker), ‘length‘ (‘end - start + 1‘), ‘n_markers‘ and ‘quality‘. Rows are unordered.

Examples

```
con <- rduckhts_connect()
path <- system.file("extdata", "roh_fixture.vcf.gz", package = "Rduckhts")
roh <- rduckhts_roh(con, path, af_tag = "AF")
roh[order(roh$sample, roh$chrom, roh$start), ]
DBI::dbDisconnect(con, shutdown = TRUE)
```

---

rduckhts_roh_counts *Find Runs of Homozygosity from Read Counts*

Description

Detect runs of homozygosity from allele read counts, for example counts at panel sites from a BAM or CRAM, with the two-state model and transitions of [rduckhts_roh()]. Genotype likelihoods come from a binomial read model with a per-read sequencing error and an optional contamination fraction instead of FORMAT/PL or GT. This emission is a DuckHTS extension; ‘bcftools roh‘ has no read-count mode.

Usage

```
rduckhts_roh_counts(
  con,
  counts_table,
  seq_error = 0.001,
  contamination = 0,
  genetic_map = NULL,
  hw_to_az = 6.7e-08,
  az_to_hw = 5e-09,
```

```
  rec_rate = NULL,
  table_name = NULL,
  overwrite = FALSE,
  max_sites = 2e+07,
  max_site_bytes = 2^32,
  fit_table = NULL,
  allele_balance = 0.5,
  artefact_weight = 0
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
counts_table	Name of a table or view of read counts (see Details).
seq_error	Per-read probability of showing the other allele, in ‘(0, 0.5)‘.
contamination	Fraction of reads from a contaminating individual, in ‘[0, 1)‘.
genetic_map	Optional name of a genetic-map table or view.
hw_to_az, az_to_hw	Per-base-pair transition probabilities Hardy-Weinberg to autozygous (‘-a‘) and back (‘-H‘), each in ‘[0, 1]‘.
rec_rate	Optional constant recombination rate per base pair (‘-M‘).
table_name	Optional output table. ‘NULL‘ returns a data frame.
overwrite	Whether an existing output table may be replaced.
max_sites	Most sites one sample and chromosome may hold, from 1 to 100,000,000. A larger group is an error.
max_site_bytes	Bound on the bytes the site buffers hold at once (16 bytes per site, 24 for read counts). A decode does not grow a buffer when the bytes held by all ROH decodes in the process would pass its own ‘max_site_bytes‘; decodes that run at the same time with different values each apply their own. Exceeding it is an error: decode fewer samples per call or raise it.
fit_table	Optional table or view with one row per ‘sample‘, containing non-NULL ‘seq_error‘, ‘contamination‘, ‘allele_balance‘ and ‘artefact_weight‘, and an ‘artefact_weight_sd‘ column, such as the output of [rduckhts_count_error_fit()]. These replace the scalar arguments for each matching ‘sample_id‘. A sample of ‘counts_table‘ with no fit row, several fit rows, a NULL value or a value out of range is an error; fit rows of other samples are not checked. A NULL ‘artefact_weight_sd‘ counts as 0. Filter fit status in the caller’s SQL; the spreads are not used.
allele_balance	Share of a heterozygote’s reads that show the counted allele, in ‘[0.35, 0.65]‘. The default 0.5 assumes no mapping bias.
artefact_weight	Chance that a site’s counts are uniform on ‘0..depth‘ whatever its genotype, in ‘[0, 0.05]‘. The default 0 gives such sites no term. Values outside either range are errors, not clipped.

Details

‘counts_table‘ names a table or view with columns ‘sample_id‘, ‘chrom‘, ‘pos‘ (one-based), ‘ref_count‘, ‘alt_count‘ and ‘af‘, one row per sample and site. ‘alt_count‘ counts reads that show the allele whose population frequency is ‘af‘; ‘ref_count‘ counts reads that show the other allele. The frequency can come from any source joined in SQL beforehand: an INFO tag, a frequency table or ancestry-weighted frequencies. Somalier site counts map as ‘ref_count = a‘, ‘alt_count = b‘, with ‘af‘ the frequency of ‘allele_b‘.

Each read shows the counted allele with probability ‘(1 - contamination) * q + contamination * c‘, where ‘q‘ is ‘seq_error‘, ‘allele_balance‘ or ‘1 - seq_error‘ for zero, one or two copies, and ‘c‘ is the mean of ‘q‘ over the genotype of a contaminating individual of the same population (Hardy-Weinberg at ‘af‘). With chance ‘artefact_weight‘ a site’s counted reads are instead uniform on ‘0..depth‘, whatever its genotype. The four numbers mean what [rduckhts_count_error_fit()] reports under these names. Without the contamination term, contaminant reads at homozygous sites look like heterozygous evidence, which shortens runs and lowers FROH.

Sites with no reads, a NULL count, or an ‘af‘ that is NULL, NaN or 0 are skipped; a repeated position keeps one row. Negative counts are an error. The model assumes biallelic sites, independent reads (no duplicate or strand modelling) and one error rate for all reads. Reference-biased mapping makes the other allele of a heterozygote rarer; ‘allele_balance‘ takes one value per sample, while the bias is stronger where flanking heterozygosity is high, so such heterozygotes can still look homozygous and lengthen runs in divergent regions. The contamination fraction is supplied by the caller, for example from [rduckhts_count_error_fit()]. A contaminant from a different population than ‘af‘ describes is approximated by ‘af‘.

‘seq_error‘ is an error of one read, and the reads of a site multiply. With ‘artefact_weight = 0‘, a site with balanced reads excludes both homozygous genotypes whatever ‘seq_error‘ is: the decode then corresponds to the genotype decode with no tolerated genotype error, and reports fewer bases in runs than ‘rduckhts_roh(gt_error = 30)‘ on the same sample. With ‘artefact_weight = w > 0‘, one site at depth ‘d‘ changes the odds between the two states by a factor of at most ‘1 + (d + 1) * (1 - w) / w‘, as ‘gt_error‘ caps one called genotype at ‘10^(gt_error / 10)‘.

The runs depend on ‘artefact_weight‘ more than on the other fitted numbers. With ‘fit_table‘, each sample is therefore decoded three times from one collection of its sites: at its ‘artefact_weight‘ ‘w‘, and at ‘w - artefact_weight_sd‘ and ‘w + artefact_weight_sd‘, each kept in ‘[0, 0.05]‘. Each run of the decode at ‘w‘ gets ‘stable_fraction‘: the share of its bases that are also in a run of both other decodes. A run with ‘stable_fraction‘ 1 does not depend on where ‘w‘ lies within one standard error. Filter on it to keep the runs, or the run bases, that the interval of the fit does not move; a low value marks a run whose call hinges on the artefact weight.

‘gained_bases‘ guards the other direction: the bases that the decode at ‘w + artefact_weight_sd‘ calls next to the run but outside every run at ‘w‘. A run at ‘w + sd‘ meets a run at ‘w‘ when they share a base or abut. A base of such a run before the first run at ‘w‘ it meets counts for that run; any other counts for the run at ‘w‘ before it, so a gap that the higher weight closes between two runs counts once, for the run before the gap. A run at ‘w + sd‘ that meets no run at ‘w‘ has no row, so its bases are not reported. A high ‘gained_bases‘ marks a run that a higher weight would extend or join to its neighbour.

A NULL ‘artefact_weight_sd‘ (fewer than three blocks in the fit) counts as 0 and gives ‘stable_fraction‘ 1 and ‘gained_bases‘ 0, so check the fit’s ‘blocks‘ too. The decode then takes about three times as long.

Value

A data frame (or invisible 'TRUE' when 'table_name' is given) with the columns of [rduckhts_roh()]: one row per run, unordered. With 'fit_table' it also has 'stable_fraction' and 'gained_bases' (see Details).

Examples

```
con <- rduckhts_connect()
DBI::dbExecute(con, paste(
  "CREATE TEMP TABLE counts AS SELECT 'S1' AS sample_id, '1' AS chrom,",
  "i * 25000 AS pos, 0.5 AS af,",
  "CASE WHEN i BETWEEN 40 AND 120 OR i % 2 = 0 THEN 30 ELSE 15 END AS ref_count,",
  "CASE WHEN i BETWEEN 40 AND 120 OR i % 2 = 0 THEN 0 ELSE 15 END AS alt_count",
  "FROM range(1, 161) AS t(i)"
))
rduckhts_roh_counts(con, "counts")
DBI::dbDisconnect(con, shutdown = TRUE)
```

---

rduckhts_samtools_idxstats

samtools idxstats-Compatible Alignment Summary

Description

Writes samtools idxstats-compatible alignment summary output for BAM, CRAM, or SAM input.

Usage

```
rduckhts_samtools_idxstats(
  con,
  path,
  output = NULL,
  index_path = NULL,
  threads = 0,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
path	Path to the input alignment file
output	Optional output path for the written idxstats text file
index_path	Optional explicit BAM/CRAM index path
threads	htslib decompression thread count for scan fallback
overwrite	Overwrite an existing output file

Value

A data frame with ‘success‘, ‘path‘, ‘output_path‘, ‘used_index_fast_path‘, and ‘error_message‘

`rduckhts_score` *Compute Polygenic Scores*

Description

Calls the DuckHTS ‘bcftools_score(...)‘ table function to compute sample-level polygenic scores from one genotype VCF/BCF file and one or more summary-statistics files.

Usage

```
rduckhts_score(
  con,
  bcf_path,
  summary_path = NULL,
  use = NULL,
  columns = "PLINK",
  columns_file = NULL,
  q_score_thr = NULL,
  summaries_list_file = NULL,
  log_path = NULL,
  use_variant_id = FALSE,
  counts = FALSE,
  samples = NULL,
  force_samples = FALSE,
  regions = NULL,
  regions_file = NULL,
  regions_overlap = 1,
  targets = NULL,
  targets_file = NULL,
  targets_overlap = 0,
  apply_filters = NULL,
  include = NULL,
  exclude = NULL
)
```

Arguments

`con`	A DuckDB connection with DuckHTS loaded
`bcf_path`	Path to genotype VCF/BCF file
`summary_path`	Path(s) to summary-statistics file(s). A character vector computes multiple TSV/SSF PRS columns in one genotype scan. Use ‘NULL‘ with ‘summaries_list_file‘ to read paths from a file.
`use`	Optional dosage source (‘"GT"‘, ‘"DS"‘, ‘"HDS"‘, ‘"AP"‘, ‘"GP"‘, ‘"AS"‘)

columns	Optional summary preset (‘"PLINK"‘, ‘"PLINK2"‘, ‘"REGENIE"‘, ‘"SAIGE"‘, ‘"BOLT"‘, ‘"METAL"‘, ‘"PGS"‘, ‘"SSF"‘, ‘"GWAS-SSF"‘)
columns_file	Optional two-column summary header mapping file
q_score_thr	Optional comma-separated p-value thresholds (e.g. ‘"1e-8,1e-6,1e-4"‘)
summaries_list_file	Optional path to a file (one summary path per line) or directory of summary files, matching upstream ‘bcftools +score –summaries‘.
log_path	Optional path for a matching/audit log with loaded, matched, allele-mismatch, and duplicate-marker counts per PRS.
use_variant_id	Logical; if TRUE, match variants by ID instead of CHR+BP
counts	Logical; if TRUE, include per-threshold matched-variant counts
samples	Optional comma-separated list of sample names to subset (e.g. ‘"SAMP1,SAMP2"‘)
force_samples	Logical; if TRUE, ignore missing samples instead of erroring
regions	Optional comma-separated region list (e.g. ‘"1:1000-2000,2:50-90"‘)
regions_file	Optional path to a regions file
regions_overlap	Overlap mode for regions (‘0‘, ‘1‘, or ‘2‘). Default 1 (trim to region).
targets	Optional comma-separated targets list
targets_file	Optional path to a targets file
targets_overlap	Overlap mode for targets (‘0‘, ‘1‘, or ‘2‘). Default 0 (record must start in region).
apply_filters	Optional comma-separated FILTER names to keep (e.g. ‘"PASS,."‘)
include	Optional site expression (currently unsupported)
exclude	Optional site expression (currently unsupported)

Value

A data frame with one row per sample and score/count columns.

---

rduckhts_simd_backend *DuckHTS SIMD backend diagnostics*

Description

Inspect or explicitly select the SIMD dispatch policy used by bundled DuckHTS byte-oriented helper kernels such as `seq_gc_content(...)`.

Usage

```
rduckhts_simd_backend(con)

rduckhts_simd_requested_backend(con)

rduckhts_simd_backend_compiled(con, backend)

rduckhts_simd_backend_cpu_supported(con, backend)

rduckhts_simd_backend_available(con, backend)

rduckhts_simd_info(con)

rduckhts_simd_kernel_info(con)

rduckhts_simd_set_backend(con, backend = "auto")
```

Arguments

con	A DuckDB connection with DuckHTS loaded via `rduckhts_load()`.
backend	A single backend request. `"auto"` selects the best available implementation independently for each logical kernel at runtime for `rduckhts_simd_set_backend()`. Backend inventory predicates such as `rduckhts_simd_backend_available()` refer to concrete backends such as `"scalar"`, `"sse2"`, `"sse41"`, `"avx2"`, `"avx512"`, `"neon"`, and `"wasm_simd128"`.

Value

`rduckhts_simd_backend()`, `rduckhts_simd_requested_backend()`, and `rduckhts_simd_set_backend()` return a character scalar. `rduckhts_simd_backend_compiled()`, `rduckhts_simd_backend_cpu_supported()`, and `rduckhts_simd_backend_available()` return logical scalars. `rduckhts_simd_info()` returns the extension-owned backend inventory table with one row per known backend; availability means compiled and CPU/runtime supported; SQL-level backend changes are process-wide and use the one-row `duckhts_simd_set_backend(...)` table function; the `selectable` column reports whether the backend has a selectable implementation path. Explicit selection still requires `available = TRUE`. `rduckhts_simd_kernel_info()` returns one row per logical SIMD kernel and is the authoritative diagnostic for mixed per-kernel auto-dispatch.

Examples

```
## Not run:
con <- rduckhts_connect()
rduckhts_simd_info(con)
rduckhts_simd_kernel_info(con)
rduckhts_simd_backend_available(con, "scalar")
rduckhts_simd_set_backend(con, "scalar")
rduckhts_simd_set_backend(con, "auto")
DBI::dbDisconnect(con, shutdown = TRUE)
```

```
## End(Not run)
```

---

rduckhts_somalier_bam_counts *Extract Panel-Aligned Counts from BAM or CRAM*

Description

Count observed A, B, and other query bases at every site in a typed panel. The panel is prepared once, then each scan worker owns one indexed multi-region scan and independent BAM/CRAM, index, reference, pileup, and overlap state. ‘worker_count‘ bounds the number of DuckDB-scheduled panel shards; the connection’s thread setting bounds how many can run concurrently. ‘decompression_threads‘ separately controls htslib decompression workers per source handle. Valid uncovered sites are measured zero depth; reference or alignment-header mismatches remain rows with NULL counts and a named status. The native function reads the panel from a local Parquet file on a private in-memory instance. A panel table or view is therefore written to a scratch Parquet file for the call, which also makes caller-local temporary relations and uncommitted rows visible.

Usage

```
rduckhts_somalier_bam_counts(
  con,
  source_path,
  sample_id,
  reference_path,
  panel_table = NULL,
  panel_parquet = NULL,
  index_path = NULL,
  reference_index_path = NULL,
  min_mapq = 1,
  min_baseq = 0,
  require_flags = 0,
  exclude_flags = 1796,
  overlap_policy = c("hileup_v0.1.0", "none"),
  decompression_threads = 0,
  worker_count = 1,
  max_depth = 1e+05,
  max_overlap_qnames = 1e+05,
  max_sites = 1e+06,
  max_region_bytes = 67108864,
  remote_block_bytes = 1048576,
  remote_cache_bytes = 67108864,
  reference_cache_bytes = 67108864,
  table_name = NULL,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
source_path	One indexed BAM/CRAM path or URI.
sample_id	Nonempty sample identity assigned to the extracted rows.
reference_path	Reference FASTA matching the panel and alignments.
panel_table	Name of an ordered panel table or view visible to ‘con‘.
panel_parquet	Local ordinary panel Parquet path, instead of ‘panel_table‘.
index_path	Optional explicit BAM/CRAM index path.
reference_index_path	Optional explicit FASTA index path.
min_mapq	Minimum alignment mapping quality.
min_baseq	Minimum observed-base quality. Missing qualities pass only when this is zero.
require_flags	SAM flag bits that every retained alignment must have.
exclude_flags	SAM flag bits that exclude an alignment.
overlap_policy	Either ‘"hileup_v0.1.0"‘ encounter-order mate suppression or ‘"none"‘.
decompression_threads	Number of htslib decompression worker threads.
worker_count	Number of independently schedulable panel shards, from 1 through 64. Each nonempty shard opens its own reader/reference state.
max_depth	Maximum admitted pileup depth before an explicit error.
max_overlap_qnames	Per-site, per-job capacity for overlap-suppression names.
max_sites	Maximum panel cardinality.
max_region_bytes	Per-job capacity for the indexed multi-region request.
remote_block_bytes	Remote alignment block size per worker handle.
remote_cache_bytes	Remote alignment cache size per worker handle.
reference_cache_bytes	Remote reference cache size per worker handle.
table_name	Optional output table. ‘NULL‘ returns a data frame.
overwrite	Whether an existing output table may be replaced.

Value

A data frame ordered by ‘site_index‘ if ‘table_name‘ is ‘NULL‘; otherwise invisible ‘TRUE‘.

rduckhts_somalier_charr *Estimate Per-Sample Contamination with CHARR*

Description

Apply the Somalier-derived CHARR estimator to measured A/B/other count evidence aligned to an ordered panel and population-B allele frequencies. 'frequency_table' contains the panel's six identity columns plus 'population_b_af'; it must cover every panel site exactly once. Evidence and frequency identities, coordinates, and A/B orientation are checked against the panel before their digests are derived. A result with no usable homozygous-like evidence has status 'no_evidence' and a NULL estimate.

Usage

```
rduckhts_somalier_charr(
  con,
  evidence_table = NULL,
  evidence_parquet = NULL,
  panel_table = NULL,
  panel_parquet = NULL,
  frequency_table = NULL,
  frequency_parquet = NULL,
  table_name = NULL,
  sample_ids = NULL,
  min_depth = 15,
  max_depth = 1e+06,
  hom_minor_rate = 0.12,
  hom_tail_alpha = 0.002,
  max_threshold_work = 1.6e+07,
  max_sites = 1e+06,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
evidence_table	Name of an ordinary evidence table or view.
evidence_parquet	Path to an evidence Parquet file, instead of 'evidence_table'.
panel_table	Name of the required ordered panel table or view. Panel assembly and region values are each limited to 1,024 bytes.
panel_parquet	Path to the required ordered panel Parquet file, instead of 'panel_table'.
frequency_table	Name of the required population-frequency table or view.

`frequency_parquet`
: Path to the required population-frequency Parquet file, instead of 'frequency_table'.

`table_name`
: Optional output table. 'NULL' returns a data frame.

`sample_ids`
: Optional nonempty vector of distinct sample IDs to retain.

`min_depth`
: Minimum measured A+B depth for CHARR homozygous-like eligibility.

`max_depth`
: Maximum supported A+B depth for the binomial eligibility test, at most 1,000,000.

`hom_minor_rate`
: Expected minor-read rate used to recognize homozygous-like anchors.

`hom_tail_alpha`
: Binomial upper-tail threshold for homozygous-like eligibility.

`max_threshold_work`
: Positive cumulative limit on exact binomial certification steps, at most 100,000,000. Distinct observed depths are certified once per call and shared across samples.

`max_sites`
: Positive per-sample panel capacity, at most 100,000,000.

`overwrite`
: Whether an existing output table may be replaced.

Details

Each of the evidence, panel, and frequency inputs is supplied as exactly one named table/view or ordinary Parquet path. Optional sample selection is exact: every requested ID must occur. Results contain the panel and frequency digests, usable-site denominators, numerical status, and all filter settings.

Value

A data frame if 'table_name' is 'NULL'; otherwise invisible 'TRUE'.

`rduckhts_somalier_find_sites` *Select a Somalier Panel from Population Variants*

Description

Select typed population variants on the caller's DuckDB connection. A table or view may be populated by 'read_bcf()'; Parquet and VCF/BCF paths are also supported. Input columns follow 'read_bcf()' ('CHROM', 'POS', 'REF', 'ALT', 'FILTER', and declared 'INFO_*' fields). Interval relations have 'chrom', 'start', 'end' in zero-based half-open BED coordinates, as returned by 'read_bed()'. A gnotate exclusion relation has 'chrom', 'pos', 'ref', 'alt' in one-based VCF coordinates.

Usage

```
rduckhts_somalier_find_sites(
  con,
  source_table = NULL,
  source_parquet = NULL,
  source_vcf = NULL,
```

```
  assembly,
  min_af = 0.15,
  min_an = 115000,
  af_field = "AF",
  an_field = "AN",
  snp_dist = 10000,
  target_af = 0.48,
  include_table = NULL,
  exclude_table = NULL,
  gnotate_exclude_table = NULL,
  compatibility_mode = c("somalier_v0.3.4", "generic"),
  max_autosomal = 65535,
  max_x = 10001,
  max_y = 5001,
  table_name = NULL,
  overwrite = FALSE,
  sex_spacing = c("upstream", "enforced"),
  tie_order = c("input", "lexical")
)
```

Arguments

`con`	A DuckDB connection with DuckHTS loaded.
`source_table`	A typed source table or view, including a view over read_bcf().
`source_parquet`	Parquet source path instead of source_table.
`source_vcf`	VCF/BCF path instead of source_table.
`assembly`	Nonempty assembly identifier.
`min_af`, `min_an`, `af_field`, `an_field`, `snp_dist`, `target_af`	Somalier find-sites thresholds and INFO field names; defaults match Somalier v0.3.4.
`include_table`, `exclude_table`	BED interval relations; NULL omits the gate.
`gnotate_exclude_table`	Optional allele-specific exclusion relation in place of Somalier's gnotate zip.
`compatibility_mode`	Named contig and REF=C policy.
`max_autosomal`, `max_x`, `max_y`	Site caps after greedy selection.
`table_name`	Optional output table, otherwise return a data frame.
`overwrite`	Whether to replace an existing output table.
`sex_spacing`	Whether to reproduce upstream's unspaced X/Y selection or enforce X/Y distances.
`tie_order`	Retain input scan order or use lexical position and allele order for equal AF scores.

Details

'somalier_v0.3.4' names the pinned human-contig rules. Upstream skips autosomal REF=C and restricts X to the inclusive zero-based range 2781479..154931044. By default X/Y selections do not enter the spacing state, as in upstream v0.3.4. 'sex_spacing = "enforced"' applies the X/Y distances to every selection. Upstream's stable AF-score sort retains input order on ties; this selector uses scan order by default, or lexical position and allele order with 'tie_order = "lexical"'. For a table or view, scan order is only reliable when its query explicitly orders the records. Only uppercase A/C/G/T biallelic SNVs enter the canonical panel. 'generic' does not interpret contig names as human sex chromosomes. The result can include X/Y, which 'duckhts_somalier_import_sites()' keeps as sex-chromosome sites for 'rduckhts_somalier_sex()'. Gnotate zip input is not supported: use an allele-specific exclusion relation. 'assembly' labels the input coordinates; this function does not run the liftover steps in Somalier's 'scripts/find_sites.sh'.

Value

A canonical panel relation with selection score, source AN and spacing-distance diagnostics.

---

rduckhts_somalier_import_sites

Import an Already Selected Somalier Sites VCF or BCF

Description

Convert an existing Somalier-compatible sites file into the canonical typed panel and population-frequency relation used by DuckHTS extraction, relatedness, contamination, and sex functions. REF and ALT are oriented into lexical A/B order and alternate-allele frequency is flipped with the alleles, so 'population_b_af' always describes 'allele_b'. Records on the exact Somalier v0.3.4 X/Y aliases are kept as sex-chromosome sites, numbered after every autosomal site; relatedness and contamination functions ignore them and 'rduckhts_somalier_sex()' reads them. No pseudo-autosomal filtering happens on import or extraction, as in Somalier: select X sites outside the PAR when choosing the sites file. At least one autosomal record is required.

Usage

```
rduckhts_somalier_import_sites(
  con,
  path,
  assembly,
  max_sites = 1e+06,
  table_name = NULL,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
path	One already selected sites VCF/BCF path or URI. 'INFO/AF' must be declared 'Number=A,Type=Float' and every retained record must be one canonical biallelic SNV with one finite AF value.
assembly	Nonempty assembly identifier attached to every site.
max_sites	Positive input-site limit, at most 100,000,000.
table_name	Optional output table. 'NULL' returns a data frame.
overwrite	Whether an existing output table may be replaced.

Details

This function does not select sites from a population VCF. Somalier's 'find-sites' algorithm has separate AF/AN, QC, interval-exclusion, and spacing semantics and is not implied by importing its output.

Value

A data frame if 'table_name' is 'NULL'; otherwise invisible 'TRUE'.

---

rduckhts_somalier_matched_contamination

Estimate Directional Contamination Against Matched Anchors

Description

Estimate one contamination fraction for each explicitly ordered receiver/anchor pair. The pair relation must contain distinct nonempty 'receiver_id' and 'anchor_id' columns. Every listed sample must have one count tuple for every ordered panel site. Both samples and population-B frequencies use the panel's A/B orientation; the anchor supplies the expected uncontaminated receiver genotype and is not interpreted as the contaminating donor. Reversing a pair is therefore a different analysis.

Usage

```
rduckhts_somalier_matched_contamination(
  con,
  evidence_table = NULL,
  evidence_parquet = NULL,
  panel_table = NULL,
  panel_parquet = NULL,
  frequency_table = NULL,
  frequency_parquet = NULL,
  pairs_table = NULL,
  pairs_parquet = NULL,
  table_name = NULL,
```

```
  min_depth = 15,
  max_depth = 1e+06,
  hom_minor_rate = 0.05,
  hom_tail_alpha = 0.001,
  error_rate = 0.002,
  min_probability = 1e-10,
  min_prior_frequency = 1e-06,
  alpha_min = 0,
  alpha_max = 1,
  grid_step = 0.01,
  refine_tolerance = 1e-10,
  max_evaluations = 4096,
  max_threshold_work = 1.6e+07,
  max_sites = 1e+06,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
evidence_table	Name of an ordinary evidence table or view.
evidence_parquet	Path to an evidence Parquet file, instead of 'evidence_table'.
panel_table	Name of the required ordered panel table or view. Panel assembly and region values are each limited to 1,024 bytes.
panel_parquet	Path to the required ordered panel Parquet file, instead of 'panel_table'.
frequency_table	Name of the required population-frequency table or view.
frequency_parquet	Path to the required population-frequency Parquet file, instead of 'frequency_table'.
pairs_table	Name of an ordinary ordered-pair table or view.
pairs_parquet	Path to an ordered-pair Parquet file, instead of 'pairs_table'.
table_name	Optional output table. 'NULL' returns a data frame.
min_depth	Minimum measured receiver A+B depth for matched-contamination eligibility.
max_depth	Maximum supported A+B depth for the binomial eligibility test, at most 1,000,000.
hom_minor_rate	Expected minor-read rate used to recognize homozygous-like anchors.
hom_tail_alpha	Binomial upper-tail threshold for homozygous-like eligibility.
error_rate	Count error probability used by the fitted likelihood.
min_probability	Positive probability floor used in logarithms.
min_prior_frequency	Lower population-frequency clamp.
alpha_min, alpha_max	Closed contamination search range, satisfying '0 <= alpha_min < alpha_max <= 1'.

`grid_step` Positive initial grid spacing, at most one.

`refine_tolerance`
Positive local-refinement tolerance.

`max_evaluations`
Positive bound on likelihood evaluations.

`max_threshold_work`
Positive cumulative limit on exact binomial certification steps, at most 100,000,000. Distinct observed depths are certified once per call and shared across samples.

`max_sites` Positive per-sample panel capacity, at most 100,000,000.

`overwrite` Whether an existing output table may be replaced.

Details

Sample identifiers and assembly strings are limited to 1,024 bytes, including when prepared profiles are persisted and supplied directly.

The returned score omits alpha-independent binomial coefficients. It compares alpha candidates for the same observed receiver/anchor pair and is not an absolute likelihood comparable between pairs. No usable evidence yields status ‘no_evidence‘ with NULL alpha and relative log-likelihood.

Value

A data frame if ‘table_name‘ is ‘NULL‘; otherwise invisible ‘TRUE‘.

`rduckhts_somalier_relatedness`
Compare Somalier-Derived Sample Sketches

Description

Compute the named relatedness and concordance statistics for either every distinct pair in one sketch relation or the ordered pairs in ‘pairs_table‘. The sketch relation must contain one non-NULL ‘sketch‘ struct per sample. Selected pairs are an ordinary relation with ‘sample_a‘ and ‘sample_b‘ columns. Missing or duplicate sample and pair identities error instead of silently dropping or multiplying requested comparisons. The native kernel checks assembly, ordered-panel digest, classification settings, mask shape, and mask contents. For all pairs it checks the contents of each sketch once and then compares the checked sketches; for selected pairs it checks both sketches of each comparison. No SQL row-order guarantee is implied.

Usage

```
rduckhts_somalier_relatedness(
  con,
  sketches_table = NULL,
  sketches_parquet = NULL,
  pairs_table = NULL,
  table_name = NULL,
```

```
  max_sites = 1e+06,
  max_pairs = 5e+07,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
sketches_table	Name of a prepared sketch table or view.
sketches_parquet	Path to ordinary Parquet-persisted sketches instead of ‘sketches_table‘.
pairs_table	Optional name of an ordered-pair table or view; ‘NULL‘ requests all distinct unordered sample pairs.
table_name	Optional output table. ‘NULL‘ returns a data frame, which should be used only for a result small enough to fit in R memory.
max_sites	Positive per-pair panel capacity, at most 100,000,000.
max_pairs	Positive cap on the number of pairs when ‘pairs_table‘ is ‘NULL‘. A sketch relation of N samples has N(N-1)/2 pairs; more pairs than ‘max_pairs‘ is an error before any pair is compared.
overwrite	Whether an existing output table may be replaced.

Value

A data frame if ‘table_name‘ is ‘NULL‘; otherwise invisible ‘TRUE‘.

---

rduckhts_somalier_sex *Infer X/Y Dosage Sex Evidence*

Description

Report, per sample, the X and Y depth relative to autosomal depth, the heterozygous and homozygous-alternate counts of usable X sites, and a call of ‘"XX"‘, ‘"XY"‘ or ‘"ambiguous"‘ following Somalier v0.3.4 (‘relate.nim‘). The counts relation comes from ‘rduckhts_somalier_vcf_counts()‘ or ‘rduckhts_somalier_bam_counts()‘ over a panel that includes X/Y sites (see ‘rduckhts_somalier_import_sites()‘); every selected sample needs one row for every panel site.

Usage

```
rduckhts_somalier_sex(
  con,
  counts_table = NULL,
  counts_parquet = NULL,
  panel_table = NULL,
  panel_parquet = NULL,
  table_name = NULL,
```

```
  sample_ids = NULL,
  min_depth = 7,
  min_sex_site_depth = 7,
  min_het_balance = 0.3,
  hom_balance_cutoff = 0.01,
  min_usable_x_sites = 11,
  xy_max_het_ratio = 0.05,
  xx_min_het_ratio = 0.4,
  y_signal_min = 0.4,
  y_gate = c("sample", "cohort"),
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
counts_table	Name of a count relation table or view.
counts_parquet	Path to a count relation Parquet file, instead of ‘counts_table‘.
panel_table	Name of the required ordered panel table or view. Panel assembly and region values are each limited to 1,024 bytes.
panel_parquet	Path to the required ordered panel Parquet file, instead of ‘panel_table‘.
table_name	Optional output table. ‘NULL‘ returns a data frame.
sample_ids	Optional nonempty vector of distinct sample IDs to retain.
min_depth	Minimum A+B depth of an autosomal site in the normaliser.
min_sex_site_depth	Minimum A+B depth of a usable X or Y site.
min_het_balance	Lower B balance accepted as heterozygous.
hom_balance_cutoff	B balance below which a site is homozygous A.
min_usable_x_sites	Usable X sites required for an XX or XY call.
xy_max_het_ratio, xx_min_het_ratio	X het/hom-alt ratio limits for XY (below) and XX (above), with ‘xy_max_het_ratio <= xx_min_het_ratio‘.
y_signal_min	Y depth ratio above which the Y signal is present.
y_gate	Either ‘"sample"‘ or ‘"cohort"‘.
overwrite	Whether an existing output table may be replaced.

Details

Depth ratios are ‘2 * mean A+B depth over usable sites / autosomal mean depth‘, with the autosomal mean taken over sites passing the relatedness allele-balance gate. The call is ‘"XY"‘ when at least ‘min_usable_x_sites‘ X sites are usable and their het/hom-alt ratio is below ‘xy_max_het_ratio‘, ‘"XX"‘ when it exceeds ‘xx_min_het_ratio‘, and otherwise ‘"ambiguous"‘; an XX dosage with a

present Y signal is also ‘"ambiguous"‘. Metrics are ‘NA‘ without usable sites or autosomal depth, with a ‘status‘ naming why. The call is evidence for review, not a diagnosis or determination of sex.

Somalier applies its Y check only when the cohort has Y depth, which makes a sample's call depend on the rest of the batch. ‘y_gate = "sample"‘ (the default) applies it whenever the sample has a Y signal; ‘"cohort"‘ reproduces Somalier's gate, computed among the selected samples, and ‘cohort_has_y‘ reports it.

Value

A data frame if ‘table_name‘ is ‘NULL‘; otherwise invisible ‘TRUE‘.

`rduckhts_somalier_sketches` *Prepare Somalier-Derived Sample Sketches*

Description

Build packed, panel-verified relatedness sketches from measured A/B/other count evidence. The panel must contain ‘assembly‘, zero-based ‘site_index‘, ‘region‘, one-based ‘position‘, and uppercase single-base ‘allele_a‘ and ‘allele_b‘. Evidence must contain the same site identity columns plus ‘sample_id‘ and nullable count columns ‘a‘, ‘b‘, and ‘other‘. All three counts are NULL for unavailable evidence; three measured zeros are not unavailable. The native SQL preparation checks every evidence site's geometry and A/B orientation against the ordered panel before computing its digest. Panel alleles must be distinct uppercase single-base A/C/G/T with lexical A < B. Rows on the exact Somalier v0.3.4 X/Y aliases are sex-chromosome sites that must follow every autosomal site in ‘site_index‘; they and their evidence are ignored here, so results equal those on the autosomal-only panel. Other contig aliases cannot be classified biologically from the region string. The three-state calculation assumes diploid sites; count evidence alone does not prove sample ploidy.

Usage

```
rduckhts_somalier_sketches(
  con,
  evidence_table = NULL,
  evidence_parquet = NULL,
  panel_table = NULL,
  panel_parquet = NULL,
  table_name = NULL,
  sample_ids = NULL,
  min_depth = 7,
  min_het_balance = 0.3,
  hom_balance_cutoff = 0.01,
  max_sites = 1e+06,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
evidence_table	Name of an ordinary evidence table or view.
evidence_parquet	Path to an evidence Parquet file, instead of 'evidence_table'.
panel_table	Name of the required ordered panel table or view. Panel assembly and region values are each limited to 1,024 bytes.
panel_parquet	Path to the required ordered panel Parquet file, instead of 'panel_table'.
table_name	Optional output table. 'NULL' returns a data frame.
sample_ids	Optional nonempty vector of distinct sample IDs to retain.
min_depth	Minimum A+B count depth for a relatedness genotype call.
min_het_balance	Lower B/(A+B) balance accepted as heterozygous.
hom_balance_cutoff	B/(A+B) balance below which a site is homozygous A; its upper symmetric limit determines homozygous B.
max_sites	Positive per-sample panel capacity, at most 100,000,000.
overwrite	Whether an existing output table may be replaced.

Details

Supply each source as either a table/view name or an ordinary Parquet path. Parquet inputs are exposed through query-scoped temporary views; no private sketch format or user-supplied panel digest is involved. The result has one 'sketch' struct per selected sample. Its packed words can be persisted with DuckDB's usual Parquet 'COPY' statement.

Value

A data frame if 'table_name' is 'NULL'; otherwise invisible 'TRUE'.

---

rduckhts_somalier_vcf_counts

Extract Panel-Aligned Counts from VCF or BCF

Description

Produce the complete sample-by-panel count relation consumed by the Somalier-derived relatedness, contamination and sex functions. 'FORMAT/AD' must declare 'Number=R,Type=Integer'; A and B slots are matched by exact REF/ALT identity, and 'other' sums only the remaining declared-allele slots. Missing sites and unavailable AD remain rows with three NULL counts, distinct from measured zero depth. The panel can be any typed table/view or ordinary Parquet file with the canonical six panel identity columns.

Usage

```
rduckhts_somalier_vcf_counts(
  con,
  path,
  panel_table = NULL,
  panel_parquet = NULL,
  samples = NULL,
  filter_policy = c("pass_or_unapplied", "include_all", "error"),
  table_name = NULL,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
path	One VCF/BCF path or URI.
panel_table	Name of the ordered panel table or view.
panel_parquet	Ordinary panel Parquet path, instead of ‘panel_table‘.
samples	Optional HTSlib sample selector: comma-separated inclusion, leading ‘^‘ exclusion, ‘"-"‘ for all, or ‘""‘ for none.
filter_policy	Record FILTER policy: ‘"pass_or_unapplied"‘ makes named failures unavailable, ‘"include_all"‘ uses their AD, and ‘"error"‘ rejects a selected panel record with a named failure. X/Y sites ignore the policy and always use their AD, as Somalier extraction does.
table_name	Optional output table. ‘NULL‘ returns a data frame.
overwrite	Whether an existing output table may be replaced.

Value

A data frame if ‘table_name‘ is ‘NULL‘; otherwise invisible ‘TRUE‘.

---

rduckhts_tabix *Create Tabix-Indexed File Table*

Description

Creates a DuckDB table from any tabix-indexed file using the DuckHTS extension.

Usage

```
rduckhts_tabix(
  con,
  table_name,
  path,
  region = NULL,
```

```
  index_path = NULL,
  header = NULL,
  header_names = NULL,
  auto_detect = NULL,
  column_types = NULL,
  scan_mode = NULL,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
table_name	Name for the created table
path	Path to the tabix-indexed file
region	Optional genomic region (e.g., "chr1:1000-2000")
index_path	Optional explicit path to index file (.tbi/.csi)
header	Logical. If TRUE, use first non-meta line as column names
header_names	Character vector to override column names
auto_detect	Logical. If TRUE, infer basic numeric column types
column_types	Character vector of column types (e.g. "BIGINT", "VARCHAR")
scan_mode	Optional scan mode. Use `"auto"` (default extension behavior) or `"sequential"` to force full-file streaming/counting instead of index-backed count paths. Sequential mode is incompatible with `region`.
overwrite	Logical. If TRUE, overwrites existing table

Value

Invisible TRUE on success

`rduckhts_tabix_convert_parquet` *Convert generic tabix reader output to Parquet with DuckHTS metadata*

Description

Thin DBI wrapper around extension macro ‘duckhts_tabix_convert_parquet_sql(...)‘.

Usage

```
rduckhts_tabix_convert_parquet(
  con,
  path,
  output,
  columns = NULL,
  region = NULL,
  index_path = NULL,
  header = NULL,
  header_names = NULL,
  auto_detect = NULL,
  column_types = NULL,
  where = NULL,
  compression = "zstd",
  row_group_size = 100000L,
  partition_by = NULL,
  include_metadata = TRUE,
  header_text = NULL,
  metadata = NULL,
  metadata_json_file = NULL,
  write_format_version = "1",
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded.
path	Path or URI to the input tabix-indexed text file.
output	Path to the output Parquet file or partitioned directory.
columns	Optional character vector of columns to include. Defaults to all columns.
region	Optional genomic region string for indexed inputs.
index_path	Optional explicit index path.
header	Logical; pass ‘header := true/false‘ to ‘read_tabix(...)‘.
header_names	Optional character vector of column names.
auto_detect	Logical; request type auto-detection.
column_types	Optional character vector of DuckDB column types.
where	Optional SQL predicate applied to the reader output before conversion.
compression	Parquet compression, default ‘"zstd"‘.
row_group_size	Parquet row group size.
partition_by	Optional character vector of output partition columns.
include_metadata	Logical; include DuckHTS Parquet KV metadata.
header_text	Optional corrected header text to store instead of the source header.

`metadata` Optional named list/vector of extra metadata. This is the primary CRAN/offline-safe path for arbitrary metadata; values with the same names as DuckHTS defaults override the default values.

`metadata_json_file` Optional path to a JSON file containing a top-level object of extra metadata. This requires DuckDB's 'json' extension to be available when the conversion SQL is generated; otherwise DuckDB will report its normal missing-extension error. Use 'metadata' for offline-safe metadata.

`write_format_version` DuckHTS Parquet write-format version string.

`overwrite` Logical; replace an existing output path. The wrapper checks existence through DuckDB 'glob(...)' where possible and passes the same flag to DuckDB 'COPY' for partitioned-output overwrite handling.

Value

Invisibly returns 'output'.

`rduckhts_tabix_index` *Build Tabix Index*

Description

Builds a tabix index for a BGZF-compressed text file using the DuckHTS extension.

Usage

```
rduckhts_tabix_index(
  con,
  path,
  preset = "vcf",
  index_path = NULL,
  min_shift = 0,
  threads = 4,
  seq_col = NULL,
  start_col = NULL,
  end_col = NULL,
  comment_char = NULL,
  skip_lines = NULL
)
```

Arguments

`con` A DuckDB connection with DuckHTS loaded

`path` Path to the BGZF-compressed input file

`preset` Optional preset such as '"vcf"', '"bed"', '"gff"', or '"sam"'

index_path	Optional explicit output path for the created index
min_shift	Index format selector used by htslib
threads	htslib indexing thread count
seq_col, start_col, end_col	Optional explicit tabix coordinate columns
comment_char	Optional tabix comment/header prefix
skip_lines	Optional fixed number of header lines to skip

Value

A data frame with ‘success‘, ‘index_path‘, and ‘index_format‘

---

rduckhts_tabix_multi *Read multiple tabix-indexed files into a DuckDB table*

Description

Read and combine multiple tabix-indexed files via UNION ALL BY NAME, materialising the result as a DuckDB table. Each row includes a `filename` column identifying its source file.

Usage

```
rduckhts_tabix_multi(
  con,
  table_name,
  files,
  region = NULL,
  index_path = NULL,
  header = NULL,
  header_names = NULL,
  auto_detect = NULL,
  column_types = NULL,
  scan_mode = NULL,
  .params = NULL,
  overwrite = FALSE
)
```

Arguments

con	A DBI connection to DuckDB with the duckhts extension loaded.
table_name	Name of the DuckDB table to create.
files	Character vector of file paths or glob patterns.
region	Optional region string.
index_path	Optional index file path.

header	Logical or NULL; whether the file has a header line.
header_names	Character vector of column names.
auto_detect	Logical or NULL; enable type auto-detection.
column_types	Character vector of column type names.
scan_mode	Optional scan mode ("auto" or "sequential").
.params	Optional data.frame with per-file parameter overrides.
overwrite	Logical; if TRUE, replace an existing table.

Value

Invisible TRUE on success.

---

setup_hts_env *Setup HTSlib Environment*

Description

Sets the 'HTS_PATH' environment variable to point to the bundled htslib plugins directory. This enables remote file access via libcurl plugins (e.g., s3://, gs://, http://) when plugins are available.

Usage

```
setup_hts_env(plugins_dir = NULL)
```

Arguments

plugins_dir	Optional path to the htslib plugins directory. When NULL, uses the bundled plugins directory if available.

Details

Call this before the process opens its first HTS file. htslib discovers dynamic plugins on first file access and does not rescan 'HTS_PATH' later.

Value

Invisibly returns the previous value of 'HTS_PATH' (or 'NA' if unset).

Examples

```
## Not run:
setup_hts_env()

plugins_path <- tempfile("hts_plugins_")
dir.create(plugins_path)
setup_hts_env(plugins_dir = plugins_path)
unlink(plugins_path, recursive = TRUE)

## End(Not run)
```

Index

detect_complex_types, 4
duckdb_type_mappings, 5, *9*
duckhts_bootstrap, 6
duckhts_build, 6
duckhts_load, 7

extract_array_element, 7
extract_map_data, 8

normalize_tabix_types, 9

rduckhts_ancestry_bam, 10
rduckhts_ancestry_geno, 11
rduckhts_ancestry_panel, 12
rduckhts_ancestry_proportions, 13
rduckhts_bam, 14
rduckhts_bam_bed_coverage, 16
rduckhts_bam_bin_counts, 17
rduckhts_bam_convert_parquet, 18
rduckhts_bam_index, 20
rduckhts_bam_insert_size_counts, 21
rduckhts_bam_insert_size_counts_sql
*(*rduckhts_bam_insert_size_counts*)*, 21
rduckhts_bam_mate_overlap_counts, 22
rduckhts_bam_mismatch_counts, 24
rduckhts_bam_multi, 25
rduckhts_bcf, 26
rduckhts_bcf_convert_parquet, 30
rduckhts_bcf_index, 31
rduckhts_bcf_multi, 32
rduckhts_bcf_samples, 33
rduckhts_bcftools_norm, 28
rduckhts_bed, 33
rduckhts_bed_multi, 34
rduckhts_bgunzip, 35
rduckhts_bgzip, 36
rduckhts_bigwig, 37
rduckhts_connect, 37
rduckhts_count_error_fit, 39
rduckhts_count_error_fit(), *40*
rduckhts_count_mixture_lowpass, 40
rduckhts_depth_bins, 42
rduckhts_depth_region_fraction, 43
rduckhts_depth_sex_fraction, 45
rduckhts_depth_state_segments
*(*rduckhts_depth_state_solutions*)*, 46
rduckhts_depth_state_solutions, 46
rduckhts_detect_quality_encoding, 49
rduckhts_fasta, 50
rduckhts_fasta_index, 51
rduckhts_fasta_multi, 51
rduckhts_fasta_nuc, 52
rduckhts_fastq, 53
rduckhts_fastq_multi, 54
rduckhts_functions, 55
rduckhts_genbank, 56
rduckhts_genbank_to_fasta, 57
rduckhts_geno, 58
rduckhts_geno_sites, 60
rduckhts_gff, 62
rduckhts_gff_convert_parquet, 63
rduckhts_gff_multi, 65
rduckhts_gtf, 66
rduckhts_gtf_multi, 67
rduckhts_hts_header, 71
rduckhts_hts_index, 71
rduckhts_hts_index_raw, 72
rduckhts_hts_index_spans, 72
rduckhts_htslib_config, 69
rduckhts_htslib_info, 70
rduckhts_htslib_version, 70
rduckhts_install_macros, 73
rduckhts_liftover, 73
rduckhts_load, 75
rduckhts_mosdepth, 76
rduckhts_munge, 77
rduckhts_pileup, 78

rduckhts_roh, 79
rduckhts_roh_counts, 82
rduckhts_samtools_idxstats, 85
rduckhts_score, 86
rduckhts_simd_backend, 87
rduckhts_simd_backend_available
        (rduckhts_simd_backend), 87
rduckhts_simd_backend_compiled
        (rduckhts_simd_backend), 87
rduckhts_simd_backend_cpu_supported
        (rduckhts_simd_backend), 87
rduckhts_simd_info
        (rduckhts_simd_backend), 87
rduckhts_simd_kernel_info
        (rduckhts_simd_backend), 87
rduckhts_simd_requested_backend
        (rduckhts_simd_backend), 87
rduckhts_simd_set_backend
        (rduckhts_simd_backend), 87
rduckhts_somalier_bam_counts, 89
rduckhts_somalier_charr, 91
rduckhts_somalier_find_sites, 92
rduckhts_somalier_import_sites, 94
rduckhts_somalier_matched_contamination,
        95
rduckhts_somalier_relatedness, 97
rduckhts_somalier_sex, 98
rduckhts_somalier_sketches, 100
rduckhts_somalier_vcf_counts, 101
rduckhts_tabix, 9, 102
rduckhts_tabix_convert_parquet, 103
rduckhts_tabix_index, 105
rduckhts_tabix_multi, 106

setup_hts_env, 107