

Package ‘Rduckhts’

September 16, 2026

Title 'DuckDB' High Throughput Sequencing File Formats Reader
Extension

Version 1.5.2-0.1.5

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](https://doi.org/10.1093/gigascience/giab007). VariantKey / RegionKey support follows Nicola Asuni (2018) [doi:10.1101/473744](https://doi.org/10.1101/473744).

License GPL-3

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, methods, utils

Suggests Rtinycc, tinytest

URL <https://github.com/RGenomicsETL/duckhts>,
<https://r genomicsetl.r-universe.dev/Rduckhts>

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

Config/roxygen2/version 8.0.0

Author Sounkou Mahamane Toure [aut, cre],
James K Bonfield, John Marshall,Petr Danecek ,Heng Li , Valeriu Ohan,
Andrew Whitwham,Thomas Keane , Robert M Davies [ctb] (Htslib
Authors),
Brent Pedersen [cph] (Original author of mosdepth and Somalier),
Giulio Genovese [cph] (Author of BCFTools munge,score,liftover plugins),
Nicola Asuni [cph] (Author of the VariantKey and RegionKey C API),
Devon Ryan [cph] (Author of libBigWig),
DuckDB C Extension API Authors [ctb]

Maintainer Sounkou Mahamane Toure <sounkoutoure@gmail.com>

Repository CRAN

Date/Publication 2026-09-16 12:20:19 UTC

Contents

detect_complex_types	3
duckdb_type_mappings	4
duckhts_bootstrap	5
duckhts_build	5
duckhts_load	6
extract_array_element	7
extract_map_data	7
normalize_tabix_types	8
rduckhts_bam	9
rduckhts_bam_bed_coverage	11
rduckhts_bam_bin_counts	12
rduckhts_bam_convert_parquet	13
rduckhts_bam_index	15
rduckhts_bam_multi	16
rduckhts_bcf	17
rduckhts_bcftools_norm	18
rduckhts_bcf_convert_parquet	20
rduckhts_bcf_index	21
rduckhts_bcf_multi	22
rduckhts_bcf_samples	23
rduckhts_bed	24
rduckhts_bed_multi	24
rduckhts_bgunzip	25
rduckhts_bgzip	26
rduckhts_bigwig	27
rduckhts_connect	27
rduckhts_detect_quality_encoding	29
rduckhts_fasta	29
rduckhts_fasta_index	30
rduckhts_fasta_multi	31
rduckhts_fasta_nuc	32
rduckhts_fastq	33
rduckhts_fastq_multi	34
rduckhts_functions	35
rduckhts_genome	35
rduckhts_gff	37
rduckhts_gff_convert_parquet	39
rduckhts_gff_multi	40
rduckhts_gtf	42
rduckhts_gtf_multi	43
rduckhts_haplotypes	44

rduckhts_htslib_config	47
rduckhts_htslib_info	48
rduckhts_htslib_version	48
rduckhts_hts_header	49
rduckhts_hts_index	49
rduckhts_hts_index_raw	50
rduckhts_hts_index_spans	50
rduckhts_liftover	51
rduckhts_load	52
rduckhts_mosdepth	53
rduckhts_munge	54
rduckhts_pileup	55
rduckhts_samtools_idxstats	56
rduckhts_score	57
rduckhts_simd_backend	59
rduckhts_somalier_bam_counts	60
rduckhts_somalier_charr	62
rduckhts_somalier_import_sites	64
rduckhts_somalier_matched_contamination	65
rduckhts_somalier_relatedness	67
rduckhts_somalier_sketches	68
rduckhts_somalier_vcf_counts	69
rduckhts_tabix	70
rduckhts_tabix_convert_parquet	71
rduckhts_tabix_index	73
rduckhts_tabix_multi	74
setup_hts_env	75
Index	76

detect_complex_types	<i>Detect Complex Types in DuckDB Table</i>
----------------------	---

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	<i>Bootstrap the duckhts extension sources into the R package</i>
-------------------	---

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	<i>Retired manual DuckHTS extension builder</i>
---------------	---

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	<i>Load the duckhts extension into a DuckDB connection</i>
--------------	--

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 <code>rduckhts_connect()</code> .
extension_path	Explicit path to the <code>.duckdb_extension</code> 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_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 htlib 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 UIINTEGER[] 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 htlib 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

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

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 htlib
threads	htlib indexing thread count

Value

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

rduckhts_bam_multi	<i>Read multiple BAM/SAM files into a DuckDB table</i>
--------------------	--

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	<i>Create VCF/BCF Table</i>
--------------	-----------------------------

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
)
```

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 sub-field 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 htlib 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.

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

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

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 htlib
threads	htlib indexing thread count

Value

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

rduckhts_bcf_multi	<i>Read multiple VCF/BCF files into a DuckDB table</i>
--------------------	--

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.

<code>additional_csq_column_types</code>	Optional CSQ type override string.
<code>scan_mode</code>	Optional scan mode ("auto" or "sequential").
<code>decompression_threads</code>	Integer. Number of htslib decompression worker threads per file handle. Default '0'.
<code>.params</code>	Optional data.frame with per-file parameter overrides.
<code>overwrite</code>	Logical; if TRUE, replace an existing table.
<code>samples</code>	Optional HTSlib sample-selector string, as in <code>[rduckhts_bcf()]</code> .

Value

Invisible TRUE on success.

<code>rduckhts_bcf_samples</code>	<i>Read the VCF/BCF Sample Catalog</i>
-----------------------------------	--

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

<code>con</code>	A DuckDB connection with DuckHTS loaded
<code>path</code>	Path to the VCF/BCF file
<code>samples</code>	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	<i>Create BED Table</i>
--------------	-------------------------

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  
)
```

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

Value

Invisible TRUE on success

rduckhts_bed_multi	<i>Read multiple BED files into a DuckDB table</i>
--------------------	--

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	<i>BGZF Decompress a File</i>
------------------	-------------------------------

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	<i>BGZF Compress a File</i>
----------------	-----------------------------

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 htlib 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	<i>Create a BigWig Signal Table</i>
-----------------	-------------------------------------

Description

Materializes stored zero-based, half-open BigWig intervals through the DuckHTS extension. Region filters use htlib'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	<i>Create a DuckDB connection with bundled DuckHTS loaded</i>
------------------	---

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.

Examples

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

rduckhts_detect_quality_encoding	<i>Detect FASTQ Quality Encoding</i>
----------------------------------	--------------------------------------

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	<i>Create FASTA Table</i>
----------------	---------------------------

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 htlib 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	<i>Build FASTA Index</i>
----------------------	--------------------------

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	<i>Compute FASTA Interval Nucleotide Composition</i>
--------------------	--

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	<i>Read multiple FASTQ files into a DuckDB table</i>
----------------------	--

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	<i>List DuckHTS Extension Functions</i>
--------------------	---

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_geno	<i>Read Record-Major Genotypes</i>
---------------	------------------------------------

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
)
```

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.

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()]

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_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
)
```

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

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	<i>Read multiple GFF files into a DuckDB table</i>
--------------------	--

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
)
```

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

Value

Invisible TRUE on success

rduckhts_gtf_multi	<i>Read multiple GTF files into a DuckDB table</i>
--------------------	--

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_haplotypes	<i>Replay Phased Transcript Haplotypes</i>
---------------------	--

Description

Consume a SELECT query of flat event-by-transcript-by-sample calls through the bundled native replay stream. DuckDB derives complete phase-set domains and sorts the calls. Each output row is one occupied shared path, with CDS, protein, carrier keys and all contributing events. Incomplete calls and projection/edit failures retain provenance without inventing sequence.

Usage

```
rduckhts_haplotypes(  
  con,  
  calls_query,  
  model_name,  
  phase_policy = c("strict", "vep116_compat"),  
  ...,  
  input_mode = c("alt_events", "source_records"),  
  hgvs = FALSE,  
  table_name = NULL,  
  overwrite = FALSE  
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
calls_query	One nonempty SELECT query supplying the call relation.
model_name	Name of an already loaded DuckVEP model.
phase_policy	Strict GT/PS interpretation or VEP-116 called-slot order. Decoded missing calls remain incomplete; source-record input uses the pinned raw parser and explicitly conditional missing-slot interpretation.

...	Named positive integer workspace capacities accepted by 'duckvep_haplotypes', such as 'max_active_events', 'max_active_carriers', 'max_sequence_bases', 'max_ploidy', 'max_phase_sets', and 'workspace_limit'.
input_mode	'alt_events' for decoded per-ALT calls, or 'source_records' for raw GT and complete source ALT lists under 'vep116_compat'.
hgvs	Whether to request bounded protein HGVS for supported completed paths.
table_name	Optional table to create; 'NULL' returns a data frame.
overwrite	Logical. If TRUE, overwrites existing table

Details

'nominal_length_diff' is the signed sum of projected replacement ALT-minus-REF lengths before clipping at the current CDS end. It is zero for reference-only replay and NA without a CDS. Known and conditional paths retain this value; overlapping replacements can produce a different rebuilt CDS length change.

'coding_blocks' groups physical edits that share an alternate codon or displace then restore the reading frame. Each block gives one-based reference 'cds_start', transcript-oriented 'reference'/'alternate' spans, zero-based 'alt_start0' in the rebuilt CDS, 'length_change', 'sequence_flags' and 'event_indices': one source event ID per physical edit in ascending CDS order. An ID can repeat for several differing islands, within or across blocks; the list length is the physical edit count. Join IDs to 'contributors' for raw alleles and input provenance. Spans include retained bases between edits; they are not aligned differences or HGVS normalization. An insertion has empty reference; a deletion has empty alternate. Unknown sequences have NULL blocks, not an empty known result. Blocks reuse 'max_leaf_edits' capacity and the per-call workspace limit.

'cds_differences' instead contains aligned differing runs: zero-based 'ref_start0', 'alt_start0', and 'alignment_start0', with borrowed sequence spans materialized as 'reference' and 'alternate'. An empty span denotes a gap. The reference uses replay's uppercase DNA spelling; model bytes stay immutable. Runs join only when adjacent columns have the same gap/non-gap type on both sides. Indel-bearing paths use the VEP-116 pure-Perl global-alignment score and tie order; substitution-only paths compare corresponding positions. Repeated sequence can therefore place a difference away from its contributing event. Differences are not HGVS normalization or event-provenance reassignment. Unknown sequences have NULL differences. 'max_alignment_cells' bounds the exact traceback band and 'max_leaf_differences' bounds output runs; exhaustion is an error, not approximate alignment or discarded differences.

'protein_differences' has the same span fields and alignment mode, with positions in amino acids. Its reference follows Ensembl-116 start-methionine, terminal-stop and curated single-residue peptide-edit rules. Haplosaurus then appends '*' only for an exact uppercase TAA/TAG/TGA raw-CDS suffix, including nonstandard-table and partial-CDS cases. The alternate is the displayed first-stop prefix without reference peptide edits. Unknown paths and reference CDS shorter than one codon have NULL protein differences; identical known proteins have an empty list. Both difference axes reuse the same native scratch and are separately subject to the alignment-cell and run limits.

'hgvs = TRUE' requests a VEP-116-derived protein HGVS suffix in 'hgvsp': equality, one operation, or a cis allele such as 'p.[(Gly2del;Ala4CysfsTer2)]'. Protein operations use the completed path and may combine several physical edits; source contributors and coding blocks remain unchanged. 'hgvsp_status' is 'ok' only after the complete operation set has been rendered. Other statuses retain NULL text without discarding the sequence or provenance. Conditional or unavailable sequence, ordered overlapping replacements, missing peptide data, unsupported coding

contexts, and unrepresentable protein ends remain explicit. Phase policy controls allele assignment, not HGVS nomenclature. Protein HGVS retains VEP's local-peptide and unknown-residue presentation. An 'ok' status reports a supported computation, not independent HGVS-rule certification. A path with one original ALT source uses independent-event VEP-116 HGVS, including genomic shifting and absent results. Multiple differing islands within one MNV retain that single source identity. Placement needing an unavailable genomic FASTA returns 'missing_reference'. Prepared references retain their own residues and length without changing raw CDS/frame facts or inventing source edits. Loss of only a reference stop marker supplies no alternate extension, and insertions require reference flanks. 'max_hgvs_operations' bounds the working operation stack and final operations; 'max_hgvs_bytes' bounds text bytes excluding NUL; 'max_hgvs_reference_bytes' bounds the query-local FASTA result buffer, including NUL and line-ending scratch. Sequence/edit scratch derives from 'max_sequence_bases' and 'max_leaf_edits', and allele scratch from 'max_allele_bytes' and the literal allele width. All DuckVEP-owned buffers count toward 'workspace_limit'; HTSlib handle/transport storage is separate. Exhaustion errors instead of truncating text. Disabled HGVS allocates no HGVS buffers or reference handle and returns 'not_requested'. Identifiers are joined through the model transcript ordinal; the suffix contains no accession.

'stop_in_displaced_frame' reports whether any of the first translated stop codon's three bases overlaps a frame-displaced span of the rebuilt CDS. Displacement starts at a frame-changing edit and ends after the alternate bases of the restoring edit, or continues downstream if unrestored. It is FALSE for no stop or a stop after frame restoration, and NA when sequence is unavailable. It is a sequence fact, not a combined SO consequence or a claim that a restored DNA frame rescues the protein.

Whole-haplotype SO, DNA HGVS, complete protein HGVS and structural-event composition remain unfinished. Input must contain one row per 'event_index', 'transcript_index', 'sample_index', with columns 'seq_region', 'position', 'reference', 'alternate', 'alt_index', 'alleles', 'phase_before' and nullable 'phase_set'. Event indices identify individual ALT events; retain their source-record mapping. Transcript ordinals belong to the named model. Candidate selection is explicit in this input relation.

With 'input_mode = "source_records"', the required columns are 'event_index', 'seq_region', 'position', 'reference', 'alternates' (a character list), 'transcript_index', 'sample_index', and 'gt' (original VCF text). Here 'event_index' identifies the whole source record. This mode requires 'vep116_compat': the pinned file profile consumes two slots and ignores PS. Missing calls and undefined slots can yield 'conditional' sequence with evidence bit 8; this is not known phase or biological rescue. Contributor 'alt_index' is 0 for REF, a positive source ALT ordinal, or NA for an undefined slot's full-REF deletion. Source ALT strings must be nonempty and nonmissing. Projection failures still withhold sequence. Raw spelling must be retained at ingestion; it cannot be reconstructed losslessly from decoded GT arrays.

Preparation reads committed objects on the registry's retained connection; caller-local temporary objects and uncommitted changes are not visible. One preparation may run per registry at a time. Nested or concurrent preparation returns a busy error; completed scans use independent native workspaces.

Value

A data frame, or invisible 'TRUE' when creating 'table_name'.

`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	<i>Read HTS Header Metadata</i>
---------------------	---------------------------------

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	<i>Read HTS Index Metadata</i>
--------------------	--------------------------------

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

<code>con</code>	A DuckDB connection with DuckHTS loaded
<code>path</code>	Path to input HTS file
<code>format</code>	Optional format hint
<code>index_path</code>	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

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

Value

A data frame with span-oriented index metadata.

rduckhts_liftover	<i>Lift Over Variant Coordinates Against a Query</i>
-------------------	--

Description

Applies the DuckHTS ‘duckdb_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	<i>Load DuckHTS Extension</i>
---------------	-------------------------------

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

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.

Value

TRUE if the extension was loaded successfully.

Examples

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

rduckhts_mosdepth	<i>Native mosdepth-Compatible Coverage Outputs</i>
-------------------	--

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	<i>Munge Summary Statistics Rows</i>
----------------	--------------------------------------

Description

Applies the DuckHTS 'duckdb_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 htlib’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_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

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

Value

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

<code>rduckhts_score</code>	<i>Compute Polygenic Scores</i>
-----------------------------	---------------------------------

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

<code>con</code>	A DuckDB connection with DuckHTS loaded
<code>bcf_path</code>	Path to genotype VCF/BCF file
<code>summary_path</code>	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.
<code>use</code>	Optional dosage source ("GT", "DS", "HDS", "AP", "GP", "AS")
<code>columns</code>	Optional summary preset ("PLINK", "PLINK2", "REGENIE", "SAIGE", "BOLT", "METAL", "PGS", "SSF", "GWAS-SSF")
<code>columns_file</code>	Optional two-column summary header mapping file
<code>q_score_thr</code>	Optional comma-separated p-value thresholds (e.g. "1e-8,1e-6,1e-4")
<code>summaries_list_file</code>	Optional path to a file (one summary path per line) or directory of summary files, matching upstream 'bcftools +score -summaries'.
<code>log_path</code>	Optional path for a matching/audit log with loaded, matched, allele-mismatch, and duplicate-marker counts per PRS.
<code>use_variant_id</code>	Logical; if TRUE, match variants by ID instead of CHR+BP
<code>counts</code>	Logical; if TRUE, include per-threshold matched-variant counts
<code>samples</code>	Optional comma-separated list of sample names to subset (e.g. "SAMP1,SAMP2")
<code>force_samples</code>	Logical; if TRUE, ignore missing samples instead of erroring
<code>regions</code>	Optional comma-separated region list (e.g. "1:1000-2000,2:50-90")
<code>regions_file</code>	Optional path to a regions file
<code>regions_overlap</code>	Overlap mode for regions ('0', '1', or '2'). Default 1 (trim to region).
<code>targets</code>	Optional comma-separated targets list
<code>targets_file</code>	Optional path to a targets file
<code>targets_overlap</code>	Overlap mode for targets ('0', '1', or '2'). Default 0 (record must start in region).
<code>apply_filters</code>	Optional comma-separated FILTER names to keep (e.g. "PASS,.")
<code>include</code>	Optional site expression (currently unsupported)
<code>exclude</code>	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 <code>rduckhts_load()</code> .
backend	A single backend request. "auto" selects the best available implementation independently for each logical kernel at runtime for <code>rduckhts_simd_set_backend()</code> . Backend inventory predicates such as <code>rduckhts_simd_backend_available()</code> 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 htlib 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 panel can be a committed table/view or an ordinary Parquet file. Caller-local temporary relations and uncommitted changes are not visible during panel preparation. One retained-connection preparation slot is shared by concurrent calls; nested or concurrent preparation errors and callers may retry.

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 a committed ordered panel table or view.
panel_parquet	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

<code>con</code>	A DuckDB connection with DuckHTS loaded.
<code>evidence_table</code>	Name of an ordinary evidence table or view.
<code>evidence_parquet</code>	Path to an evidence Parquet file, instead of <code>'evidence_table'</code> .
<code>panel_table</code>	Name of the required ordered panel table or view. Panel assembly and region values are each limited to 1,024 bytes.
<code>panel_parquet</code>	Path to the required ordered panel Parquet file, instead of <code>'panel_table'</code> .
<code>frequency_table</code>	Name of the required population-frequency table or view.
<code>frequency_parquet</code>	Path to the required population-frequency Parquet file, instead of <code>'frequency_table'</code> .
<code>table_name</code>	Optional output table. <code>'NULL'</code> returns a data frame.
<code>sample_ids</code>	Optional nonempty vector of distinct sample IDs to retain.
<code>min_depth</code>	Minimum measured A+B depth for CHARR homozygous-like eligibility.
<code>max_depth</code>	Maximum supported A+B depth for the binomial eligibility test, at most 1,000,000.
<code>hom_minor_rate</code>	Expected minor-read rate used to recognize homozygous-like anchors.
<code>hom_tail_alpha</code>	Binomial upper-tail threshold for homozygous-like eligibility.
<code>max_threshold_work</code>	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.
<code>max_sites</code>	Positive per-sample panel capacity, at most 100,000,000.
<code>overwrite</code>	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_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, and contamination 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’. Exact Somalier v0.3.4 X/Y aliases are excluded, matching its autosomal frequency importer.

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 $0 \leq \alpha_{\min} < \alpha_{\max} \leq 1$.
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 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,
  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.
overwrite	Whether an existing output table may be replaced.

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 $A < B$; the exact X/Y aliases excluded by Somalier v0.3.4 are rejected. 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 and contamination 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.
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	<i>Create Tabix-Indexed File Table</i>
----------------	--

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 meta-data

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(...)'. Optional character vector of column names.
header_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 htlib
threads	htlib 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

<code>con</code>	A DBI connection to DuckDB with the duckhts extension loaded.
<code>table_name</code>	Name of the DuckDB table to create.
<code>files</code>	Character vector of file paths or glob patterns.
<code>region</code>	Optional region string.
<code>index_path</code>	Optional index file path.
<code>header</code>	Logical or NULL; whether the file has a header line.
<code>header_names</code>	Character vector of column names.
<code>auto_detect</code>	Logical or NULL; enable type auto-detection.
<code>column_types</code>	Character vector of column type names.
<code>scan_mode</code>	Optional scan mode ("auto" or "sequential").
<code>.params</code>	Optional data.frame with per-file parameter overrides.
<code>overwrite</code>	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

<code>plugins_dir</code>	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, 3
duckdb_type_mappings, 4, 9
duckhts_bootstrap, 5
duckhts_build, 5
duckhts_load, 6

extract_array_element, 7
extract_map_data, 7

normalize_tabix_types, 8

rduckhts_bam, 9
rduckhts_bam_bed_coverage, 11
rduckhts_bam_bin_counts, 12
rduckhts_bam_convert_parquet, 13
rduckhts_bam_index, 15
rduckhts_bam_multi, 16
rduckhts_bcf, 17
rduckhts_bcf_convert_parquet, 20
rduckhts_bcf_index, 21
rduckhts_bcf_multi, 22
rduckhts_bcf_samples, 23
rduckhts_bcftools_norm, 18
rduckhts_bed, 24
rduckhts_bed_multi, 24
rduckhts_bgunzip, 25
rduckhts_bgzip, 26
rduckhts_bigwig, 27
rduckhts_connect, 27
rduckhts_detect_quality_encoding, 29
rduckhts_fasta, 29
rduckhts_fasta_index, 30
rduckhts_fasta_multi, 31
rduckhts_fasta_nuc, 32
rduckhts_fastq, 33
rduckhts_fastq_multi, 34
rduckhts_functions, 35
rduckhts_geno, 35
rduckhts_gff, 37
rduckhts_gff_convert_parquet, 39

rduckhts_gff_multi, 40
rduckhts_gtf, 42
rduckhts_gtf_multi, 43
rduckhts_haplotypes, 44
rduckhts_hts_header, 49
rduckhts_hts_index, 49
rduckhts_hts_index_raw, 50
rduckhts_hts_index_spans, 50
rduckhts_htslib_config, 47
rduckhts_htslib_info, 48
rduckhts_htslib_version, 48
rduckhts_liftover, 51
rduckhts_load, 52
rduckhts_mosdepth, 53
rduckhts_munge, 54
rduckhts_pileup, 55
rduckhts_samtools_idxstats, 56
rduckhts_score, 57
rduckhts_simd_backend, 59
rduckhts_simd_backend_available
 (rduckhts_simd_backend), 59
rduckhts_simd_backend_compiled
 (rduckhts_simd_backend), 59
rduckhts_simd_backend_cpu_supported
 (rduckhts_simd_backend), 59
rduckhts_simd_info
 (rduckhts_simd_backend), 59
rduckhts_simd_kernel_info
 (rduckhts_simd_backend), 59
rduckhts_simd_requested_backend
 (rduckhts_simd_backend), 59
rduckhts_simd_set_backend
 (rduckhts_simd_backend), 59
rduckhts_somalier_bam_counts, 60
rduckhts_somalier_charr, 62
rduckhts_somalier_import_sites, 64
rduckhts_somalier_matched_contamination,
 65
rduckhts_somalier_relatedness, 67

`rduckhts_somalier_sketches`, [68](#)
`rduckhts_somalier_vcf_counts`, [69](#)
`rduckhts_tabix`, [9](#), [70](#)
`rduckhts_tabix_convert_parquet`, [71](#)
`rduckhts_tabix_index`, [73](#)
`rduckhts_tabix_multi`, [74](#)

`setup_hts_env`, [75](#)