

Package ‘Rduckhts’

April 9, 2026

Title 'DuckDB' High Throughput Sequencing File Formats Reader
Extension

Version 1.1.6-0.0.2

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)>.

License GPL-3

Copyright See inst/COPYRIGHT

Encoding UTF-8

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

Depends R (>= 4.4.0)

Imports DBI, duckdb, utils

Suggests tinytest

RoxygenNote 7.3.3

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

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

NeedsCompilation no

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),
Giulio Genovese [cph] (Author of BCFTools munge,score,liftover plugins),
DuckDB C Extension API Authors [ctb]

Maintainer Sounkou Mahamane Toure <sounkoutoure@gmail.com>

Repository CRAN

Date/Publication 2026-04-09 14:50:02 UTC

Contents

detect_complex_types	2
duckdb_type_mappings	3
duckhsts_bootstrap	4
duckhsts_build	5
duckhsts_load	5
extract_array_element	6
extract_map_data	6
normalize_tabix_types	7
rduckhsts_bam	8
rduckhsts_bam_index	9
rduckhsts_bcf	10
rduckhsts_bcf_index	11
rduckhsts_bed	12
rduckhsts_bgunzip	12
rduckhsts_bgzip	13
rduckhsts_detect_quality_encoding	14
rduckhsts_fasta	15
rduckhsts_fasta_index	16
rduckhsts_fasta_nuc	16
rduckhsts_fastq	17
rduckhsts_functions	18
rduckhsts_gff	19
rduckhsts_gtf	20
rduckhsts_hts_header	21
rduckhsts_hts_index	21
rduckhsts_hts_index_raw	22
rduckhsts_hts_index_spans	22
rduckhsts_liftover	23
rduckhsts_load	24
rduckhsts_munge	25
rduckhsts_score	26
rduckhsts_tabix	27
rduckhsts_tabix_index	28
setup_hts_env	29

Index	31
--------------	-----------

detect_complex_types *Detect Complex Types in DuckDB Table*

Description

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

Usage

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

Arguments

con	A DuckDB connection
table_name	Name of the table to analyze

Value

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

Examples

```
library(DBI)
library(duckdb)

con <- dbConnect(duckdb::duckdb(config = list(allow_unsigned_extensions = "true")))
rduckhts_load(con)
bcf_path <- system.file("extdata", "vcf_file.bcf", package = "Rduckhts")
rduckhts_bcf(con, "variants", bcf_path, overwrite = TRUE)
complex_cols <- detect_complex_types(con, "variants")
print(complex_cols)
dbDisconnect(con, shutdown = TRUE)
```

duckdb_type_mappings *DuckDB to R Type Mappings*

Description

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

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

Important notes:

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

Usage

```
duckdb_type_mappings()
```

Details

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

Value

A named list with two elements:

duckdb_to_r Named character vector mapping DuckDB types to R types

r_to_duckdb Named character vector mapping R types to DuckDB types

Examples

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

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

Description

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

Usage

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

Arguments

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

Value

Invisibly returns the destination directory.

duckhts_build	<i>Build the duckhts DuckDB extension</i>
---------------	-------------------------------------------

Description

Compiles htplib and the duckhts extension from the sources bundled in the installed R package. The built `.duckdb_extension` file is placed in the extension directory.

Usage

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

Arguments

<code>build_dir</code>	Where to build. Required. Use a writable location such as <code>tempdir()</code> when the installed package directory is read-only.
<code>make</code>	Optional GNU make command to use (e.g., "gmake" or "make"). When NULL, auto-detects gmake or make. If a non-GNU make is used, htplib's configure step will fail.
<code>force</code>	Rebuild even if the extension file already exists.
<code>verbose</code>	Print build output.

Value

Path to the built `duckhts.duckdb_extension` file.

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

Description

Load the duckhts extension into a DuckDB connection

Usage

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

Arguments

<code>con</code>	An existing DuckDB connection, or NULL to create one.
<code>extension_path</code>	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 <- dbConnect(duckdb::duckdb(config = list(allow_unsigned_extensions = "true")))
rduckhts_load(con)
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 <- dbConnect(duckdb::duckdb(config = list(allow_unsigned_extensions = "true")))
rduckhts_load(con)
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,  
  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[].
overwrite	Logical. If TRUE, overwrites existing table

Value

Invisible TRUE on success

Examples

```
library(DBI)
library(duckdb)

con <- dbConnect(duckdb::duckdb(config = list(allow_unsigned_extensions = "true")))
rduckhts_load(con)
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_index *Build BAM or CRAM Index*

Description

Builds a BAM or CRAM index using the DuckHTS extension.

Usage

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

Arguments

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

Value

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

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,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
table_name	Name for the created table
path	Path to the VCF/BCF file
region	Optional genomic region (e.g., "chr1:1000-2000")
index_path	Optional explicit path to index file (.csi/.tbi)
tidy_format	Logical. If TRUE, FORMAT columns are returned in tidy format
additional_csq_column_types	Optional bcftools-style 'PATTERN TYPE' overrides for CSQ/ANN/BCSQ sub-field typing, separated by newlines or ';'.
overwrite	Logical. If TRUE, overwrites existing table

Value

Invisible TRUE on success

Examples

```
library(DBI)
library(duckdb)

con <- dbConnect(duckdb::duckdb(config = list(allow_unsigned_extensions = "true")))
rduckhts_load(con)
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_bcf_index *Build VCF or BCF Index*

Description

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

Usage

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

Arguments

con	A DuckDB connection with DuckHTS loaded
path	Path to the input VCF/BCF file
index_path	Optional explicit output path for the created index
min_shift	Optional explicit min_shift passed to htlib
threads	htlib indexing thread count

Value

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

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,
  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
overwrite	Logical. If TRUE, overwrites 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 htslib default
keep	Keep the original input file after compression
overwrite	Overwrite an existing output file

Value

A data frame describing the created BGZF file

rduckhts_detect_quality_encoding
Detect FASTQ Quality Encoding

Description

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

Usage

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

Arguments

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

Value

A data frame with the detected quality encoding summary

rduckhts_fasta	<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,
  sequence_encoding = 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)
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[[]].
overwrite	Logical. If TRUE, overwrites existing table

Value

Invisible TRUE on success

rduckhts_fasta_index *Build FASTA Index*

Description

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

Usage

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

Arguments

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

Value

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

rduckhts_fasta_nuc *Compute FASTA Interval Nucleotide Composition*

Description

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

Usage

```
rduckhts_fasta_nuc(
  con,
  path,
  bed_path = NULL,
  bin_width = NULL,
  region = NULL,
  index_path = NULL,
  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
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	<i>Create FASTQ Table</i>
----------------	---------------------------

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,
  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 htlib 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".
overwrite	Logical. If TRUE, overwrites existing table

Value

Invisible TRUE on success

rduckhts_functions *List DuckHTS Extension Functions*

Description

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

Usage

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

Arguments

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

Value

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

Examples

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

rduckhts_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,
  attributes_map = 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")
attributes_map	Logical. If TRUE, returns attributes as a MAP column
overwrite	Logical. If TRUE, overwrites 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,
  attributes_map = 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")
attributes_map	Logical. If TRUE, returns attributes as a MAP column
overwrite	Logical. If TRUE, overwrites existing table

Value

Invisible TRUE on success

rduckhts_hts_header *Read HTS Header Metadata*

Description

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

Usage

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

Arguments

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

Value

A data frame with parsed header metadata.

rduckhts_hts_index *Read HTS Index Metadata*

Description

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

Usage

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

Arguments

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

Value

A data frame with index metadata.

 rduckhts_hts_index_raw

Read Raw HTS Index Blob

Description

Returns raw index metadata blob data for a file index.

Usage

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

Arguments

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

Value

A data frame with raw index blob metadata.

 rduckhts_hts_index_spans

Read HTS Index Spans

Description

Returns index span-oriented metadata for planning range workloads.

Usage

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

Arguments

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

Value

A data frame with span-oriented index metadata.

rduckhts_liftover *Lift Over Variant Coordinates Against a Query*

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 a DuckDB connection. This must be called before using any of the HTS reader functions.

Usage

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

Arguments

con	A DuckDB connection object
extension_path	Optional path to the duckhts extension file. If NULL, will try to use the bundled extension.

Details

The DuckDB connection must be created with `allow_unsigned_extensions = "true"`.

Value

TRUE if the extension was loaded successfully

Examples

```
library(DBI)
library(duckdb)

con <- dbConnect(duckdb::duckdb(config = list(allow_unsigned_extensions = "true")))
rduckhts_load(con)
dbDisconnect(con, shutdown = TRUE)
```

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_score	<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 summary-statistics file.

Usage

```
rduckhts_score(
  con,
  bcf_path,
  summary_path,
  use = NULL,
  columns = "PLINK",
  columns_file = NULL,
  q_score_thr = 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 to summary-statistics 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>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

samples	Optional comma-separated list of sample names to subset (e.g. "SAMP1,SAMP2")
force_samples	Logical; if TRUE, ignore missing samples instead of erroring
regions	Optional comma-separated region list (e.g. "1:1000-2000,2:50-90")
regions_file	Optional path to a regions file
regions_overlap	Overlap mode for regions ('0', '1', or '2'). Default 1 (trim to region).
targets	Optional comma-separated targets list
targets_file	Optional path to a targets file
targets_overlap	Overlap mode for targets ('0', '1', or '2'). Default 0 (record must start in region).
apply_filters	Optional comma-separated FILTER names to keep (e.g. "PASS,")
include	Optional site expression (currently unsupported)
exclude	Optional site expression (currently unsupported)

Value

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

rduckhts_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,
  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")
overwrite	Logical. If TRUE, overwrites existing table

Value

Invisible TRUE on success

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'

 setup_hts_env

Setup HTSlib Environment

Description

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

Usage

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

Arguments

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

Details

Call this before querying remote URLs to allow htlib to locate its plugins.

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](#), [2](#)
[duckdb_type_mappings](#), [3](#), [8](#)
[duckhts_bootstrap](#), [4](#)
[duckhts_build](#), [5](#)
[duckhts_load](#), [5](#)

[extract_array_element](#), [6](#)
[extract_map_data](#), [6](#)

[normalize_tabix_types](#), [7](#)

[rduckhts_bam](#), [8](#)
[rduckhts_bam_index](#), [9](#)
[rduckhts_bcf](#), [10](#)
[rduckhts_bcf_index](#), [11](#)
[rduckhts_bed](#), [12](#)
[rduckhts_bgunzip](#), [12](#)
[rduckhts_bgzip](#), [13](#)
[rduckhts_detect_quality_encoding](#), [14](#)
[rduckhts_fasta](#), [15](#)
[rduckhts_fasta_index](#), [16](#)
[rduckhts_fasta_nuc](#), [16](#)
[rduckhts_fastq](#), [17](#)
[rduckhts_functions](#), [18](#)
[rduckhts_gff](#), [19](#)
[rduckhts_gtf](#), [20](#)
[rduckhts_hts_header](#), [21](#)
[rduckhts_hts_index](#), [21](#)
[rduckhts_hts_index_raw](#), [22](#)
[rduckhts_hts_index_spans](#), [22](#)
[rduckhts_liftover](#), [23](#)
[rduckhts_load](#), [24](#)
[rduckhts_munge](#), [25](#)
[rduckhts_score](#), [26](#)
[rduckhts_tabix](#), [8](#), [27](#)
[rduckhts_tabix_index](#), [28](#)

[setup_hts_env](#), [29](#)