Package 'Rduckhts'

May 29, 2026

Title 'DuckDB' High Throughput Sequencing File Formats Reader Extension

Version 1.3.0-0.1.0

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

License GPL-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://rgenomicsetl.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), Brent Pedersen [cph] (Mosdepth Original Author), Giulio Genovese [cph] (Author of BCFTools munge,score,liftover plugins), Nicola Asuni [cph] (Author of the VariantKey and RegionKey C API), DuckDB C Extension API Authors [ctb]

Maintainer Sounkou Mahamane Toure <sounkoutoure@gmail.com>

Repository CRAN

Date/Publication 2026-05-29 20:10:02 UTC

Contents

`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 *Build the duckhts DuckDB extension*

Description

Compiles htslib 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

`build_dir`	Where to build. Required. Use a writable location such as `tempdir()` when the installed package directory is read-only.
`make`	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, htslib's configure step will fail.
`force`	Rebuild even if the extension file already exists.
`verbose`	Print build output.

Value

Path to the built `duckhts.duckdb_extension` file.

duckhts_load *Load the duckhts extension into a DuckDB connection*

Description

Load the duckhts extension into a DuckDB connection

Usage

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

Arguments

con	An existing DuckDB connection, or NULL to create one.
extension_path	Explicit path to the .duckdb_extension file. If NULL, uses the default location in the installed package.

Value

The DuckDB connection (invisibly).

extract_array_element *Extract Array Elements Safely*

Description

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

Usage

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

Arguments

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

Value

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

Examples

```
library(DBI)
library(duckdb)

con <- 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,
  cigar_representation = NULL,
  decompression_threads = 2,
  overwrite = FALSE
)
```

Arguments

con	A DuckDB connection with DuckHTS loaded
table_name	Name for the created table
path	Path to the SAM/BAM/CRAM file
region	Optional genomic region (e.g., "chr1:1000-2000")
index_path	Optional explicit path to index file (.bai/.csi/.crai)
reference	Optional reference file path for CRAM files
standard_tags	Logical. If TRUE, include typed standard SAMtags columns. Default FALSE.
auxiliary_tags	Logical. If TRUE, include AUXILIARY_TAGS map of non-standard tags. Default FALSE.
sequence_encoding	Character. Sequence encoding for the SEQ column: `"string"` (default) returns decoded bases as `VARCHAR`; `"nt16"` returns raw htslib nt16 4-bit codes as `UTINYINT[]`.
quality_representation	Character. Quality representation for the QUAL column: `"string"` (default) returns canonical Phred+33 text; `"phred"` returns raw Phred values as `UTINYINT[]`.

`cigar_representation`
: Character. CIGAR representation for the CIGAR column: `"string"` (default) returns SAM text such as `"36M"`; `"binary"` returns packed BAM operations as `UINTEGER[]` where each element is `(len << 4) | op`.

`decompression_threads`
: Integer. Number of htslib decompression worker threads per file handle. Default `2`. Use `0` to disable worker threads.

`overwrite`
: Logical. If TRUE, overwrites existing table

Value

Invisible TRUE on success

Examples

```
library(DBI)
library(duckdb)

con <- 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_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_index` *Build BAM or CRAM Index*

Description

Builds a BAM or CRAM index using the DuckHTS extension.

Usage

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

Arguments

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

Value

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

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

Description

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

Usage

```
rduckhts_bam_multi(
  con,
  table_name,
  files,
  region = NULL,
  index_path = NULL,
  reference = NULL,
  standard_tags = FALSE,
  auxiliary_tags = FALSE,
  sequence_encoding = NULL,
  quality_representation = NULL,
  cigar_representation = NULL,
  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.
decompression_threads	Integer. Number of htslib decompression worker threads per file handle. Default 2. Use 0 to disable worker threads.
.params	Optional data.frame with per-file parameter overrides. Must contain a file column; other columns override uniform parameters. NA values use the uniform default.
overwrite	Logical; if TRUE, replace an existing table.

Value

Invisible TRUE on success.

rduckhts_bcf *Create VCF/BCF Table*

Description

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

Usage

```
rduckhts_bcf(
  con,
  table_name,
  path,
  region = NULL,
  index_path = NULL,
  tidy_format = FALSE,
  additional_csq_column_types = NULL,
  decompression_threads = 0,
  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 subfield typing, separated by newlines or ';'
decompression_threads	Integer. Number of htslib decompression worker threads per file handle. Default '0'. Use '0' to keep BCF/VCF reads single-threaded.
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_bcftools_norm

Normalize Variant Alleles with bcftools-style Semantics

Description

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

Usage

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

Arguments

con	A DuckDB connection with DuckHTS loaded
query	SQL query or table expression to normalize
fasta_ref	Path to the reference FASTA
chrom_col	Source chromosome column name

pos_col	Source 1-based position column name
ref_col	Source reference allele column name
alt_col	Source alternate allele column name (‘VARCHAR‘ or ‘VARCHAR[]‘)
split_multiallelic	If ‘TRUE‘, split multiallelic sites before normalization so ‘alt_normed‘ is emitted as ‘VARCHAR‘ plus ‘alt_index‘. If ‘FALSE‘ (default), keep sites intact and emit ‘alt_normed‘ as ‘VARCHAR[]‘.
end_pos_col	Optional source column name containing an END-like 1-based end coordinate for symbolic deletions.
svlen_col	Optional source column name containing an SVLEN-like signed length for symbolic duplications.
fasta_index_path	Optional explicit ‘.fai‘ sidecar path.
gzi_path	Optional explicit ‘.gzi‘ sidecar path for bgzipped FASTA.

Value

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

---

rduckhts_bcf_index *Build VCF or BCF Index*

Description

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

Usage

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

Arguments

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

Value

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

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

Description

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

Usage

```
rduckhts_bcf_multi(
  con,
  table_name,
  files,
  region = NULL,
  index_path = NULL,
  tidy_format = FALSE,
  additional_csq_column_types = NULL,
  decompression_threads = 0,
  .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.
`tidy_format`	Logical; use tidy FORMAT column output.
`additional_csq_column_types`	Optional CSQ type override string.
`decompression_threads`	Integer. Number of htslib decompression worker threads per file handle. Default '0'.
`.params`	Optional data.frame with per-file parameter overrides.
`overwrite`	Logical; if `TRUE`, replace an existing table.

Value

Invisible `TRUE` on success.

rduckhts_bed *Create BED Table*

Description

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

Usage

```
rduckhts_bed(
  con,
  table_name,
  path,
  region = NULL,
  index_path = NULL,
  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_bed_multi *Read multiple BED files into a DuckDB table*

Description

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

Usage

```
rduckhts_bed_multi(
  con,
  table_name,
  files,
  region = NULL,
  index_path = NULL,
  .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.
.params	Optional data.frame with per-file parameter overrides.
overwrite	Logical; if TRUE, replace an existing table.

Value

Invisible TRUE on success.

---

rduckhts_bgunzip *BGZF Decompress a File*

Description

Decompresses a BGZF file using the DuckHTS extension.

Usage

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

Arguments

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

Value

A data frame describing the created output file

---

rduckhts_bgzip *BGZF Compress a File*

Description

Compresses a plain file to BGZF using the DuckHTS extension.

Usage

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

Arguments

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

Value

A data frame describing the created BGZF file

rduckhts_detect_quality_encoding *Detect FASTQ Quality Encoding*

Description

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

Usage

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

Arguments

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

Value

A data frame with the detected quality encoding summary

rduckhts_fasta *Create FASTA Table*

Description

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

Usage

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

Arguments

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

Value

Invisible TRUE on success

---

rduckhts_fasta_index *Build FASTA Index*

Description

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

Usage

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

Arguments

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

Value

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

`rduckhts_fasta_multi` *Read multiple FASTA files into a DuckDB table*

Description

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

Usage

```
rduckhts_fasta_multi(
  con,
  table_name,
  files,
  region = NULL,
  index_path = NULL,
  gzi_path = NULL,
  sequence_encoding = NULL,
  .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.
`.params`	Optional data.frame with per-file parameter overrides.
`overwrite`	Logical; if `TRUE`, replace an existing table.

Value

Invisible `TRUE` on success.

rduckhts_fasta_nuc *Compute FASTA Interval Nucleotide Composition*

Description

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

Usage

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

Arguments

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

Value

A data frame with interval composition statistics

rduckhts_fastq *Create FASTQ Table*

Description

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

Usage

```
rduckhts_fastq(
  con,
  table_name,
  path,
  mate_path = NULL,
  interleaved = FALSE,
  sequence_encoding = NULL,
  quality_representation = NULL,
  input_quality_encoding = NULL,
  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".
overwrite	Logical. If TRUE, overwrites existing table

Value

Invisible TRUE on success

rduckhts_fastq_multi *Read multiple FASTQ files into a DuckDB table*

Description

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

Usage

```
rduckhts_fastq_multi(
  con,
  table_name,
  files,
  mate_path = NULL,
  interleaved = FALSE,
  sequence_encoding = NULL,
  quality_representation = NULL,
  input_quality_encoding = NULL,
  .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.
`.params`	Optional data.frame with per-file parameter overrides.
`overwrite`	Logical; if `TRUE`, replace an existing table.

Value

Invisible `TRUE` on success.

rduckhts_functions *List DuckHTS Extension Functions*

Description

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

Usage

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

Arguments

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

Value

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

Examples

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

rduckhts_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,
  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")
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_multi *Read multiple GFF files into a DuckDB table*

Description

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

Usage

```
rduckhts_gff_multi(
  con,
  table_name,
  files,
  region = NULL,
  index_path = NULL,
  header = NULL,
  header_names = NULL,
  auto_detect = NULL,
  column_types = NULL,
  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.
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,
  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")
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` *Read multiple GTF files into a DuckDB table*

Description

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

Usage

```
rduckhts_gtf_multi(
  con,
  table_name,
  files,
  region = NULL,
  index_path = NULL,
  header = NULL,
  header_names = NULL,
  auto_detect = NULL,
  column_types = NULL,
  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.
`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_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` *Load DuckHTS Extension*

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_mosdepth *Native mosdepth-Compatible Coverage Outputs*

Description

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

Usage

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

Arguments

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

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

Value

A data frame describing the written output paths

`rduckhts_munge` *Munge Summary Statistics Rows*

Description

Applies the DuckHTS '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 htslib's pileup engine; it is not samtools mpileup text parity.

Usage

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

Arguments

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

Value

Invisible TRUE on success

---

rduckhts_samtools_idxstats

samtools idxstats-Compatible Alignment Summary

Description

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

Usage

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

Arguments

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

Value

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

---

rduckhts_score *Compute Polygenic Scores*

Description

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

Usage

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

Arguments

con	A DuckDB connection with DuckHTS loaded
bcf_path	Path to genotype VCF/BCF file
summary_path	Path(s) to summary-statistics file(s). A character vector computes multiple TSV/SSF PRS columns in one genotype scan. Use ‘NULL‘ with ‘summaries_list_file‘ to read paths from a file.
use	Optional dosage source (‘"GT"‘, ‘"DS"‘, ‘"HDS"‘, ‘"AP"‘, ‘"GP"‘, ‘"AS"‘)
columns	Optional summary preset (‘"PLINK"‘, ‘"PLINK2"‘, ‘"REGENIE"‘, ‘"SAIGE"‘, ‘"BOLT"‘, ‘"METAL"‘, ‘"PGS"‘, ‘"SSF"‘, ‘"GWAS-SSF"‘)
columns_file	Optional two-column summary header mapping file
q_score_thr	Optional comma-separated p-value thresholds (e.g. ‘"1e-8,1e-6,1e-4"‘)
summaries_list_file	Optional path to a file (one summary path per line) or directory of summary files, matching upstream ‘bcftools +score –summaries‘.
log_path	Optional path for a matching/audit log with loaded, matched, allele-mismatch, and duplicate-marker counts per PRS.
use_variant_id	Logical; if TRUE, match variants by ID instead of CHR+BP
counts	Logical; if TRUE, include per-threshold matched-variant counts
samples	Optional comma-separated list of sample names to subset (e.g. ‘"SAMP1,SAMP2"‘)
force_samples	Logical; if TRUE, ignore missing samples instead of erroring
regions	Optional comma-separated region list (e.g. ‘"1:1000-2000,2:50-90"‘)
regions_file	Optional path to a regions file
regions_overlap	Overlap mode for regions (‘0‘, ‘1‘, or ‘2‘). Default 1 (trim to region).
targets	Optional comma-separated targets list
targets_file	Optional path to a targets file
targets_overlap	Overlap mode for targets (‘0‘, ‘1‘, or ‘2‘). Default 0 (record must start in region).
apply_filters	Optional comma-separated FILTER names to keep (e.g. ‘"PASS,."‘)
include	Optional site expression (currently unsupported)
exclude	Optional site expression (currently unsupported)

Value

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

rduckhts_simd_backend *DuckHTS SIMD backend diagnostics*

Description

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

Usage

```
rduckhts_simd_backend(con)

rduckhts_simd_requested_backend(con)

rduckhts_simd_backend_compiled(con, backend)

rduckhts_simd_backend_cpu_supported(con, backend)

rduckhts_simd_backend_available(con, backend)

rduckhts_simd_info(con)

rduckhts_simd_kernel_info(con)

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

Arguments

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

Value

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

Examples

```
## Not run:
con <- DBI::dbConnect(duckdb::duckdb(config = list(allow_unsigned_extensions = "true")))
rduckhts_load(con)
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_tabix *Create Tabix-Indexed File Table*

Description

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

Usage

```
rduckhts_tabix(
  con,
  table_name,
  path,
  region = NULL,
  index_path = NULL,
  header = NULL,
  header_names = NULL,
  auto_detect = NULL,
  column_types = NULL,
  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 htslib
`threads`	htslib indexing thread count
`seq_col`, `start_col`, `end_col`	Optional explicit tabix coordinate columns
`comment_char`	Optional tabix comment/header prefix
`skip_lines`	Optional fixed number of header lines to skip

Value

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

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

Description

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

Usage

```
rduckhts_tabix_multi(
  con,
  table_name,
  files,
  region = NULL,
  index_path = NULL,
  header = NULL,
  header_names = NULL,
  auto_detect = NULL,
  column_types = NULL,
  .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.
`.params`	Optional data.frame with per-file parameter overrides.
`overwrite`	Logical; if `TRUE`, replace an existing table.

Value

Invisible `TRUE` on success.

setup_hts_env *Setup HTSlib Environment*

Description

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

Usage

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

Arguments

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

Details

Call this before querying remote URLs to allow htslib 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