Package: duckh3 (via r-universe)

June 1, 2026

Type Package

Title H3 Extension of 'DuckDB'

Version 0.1.0

Description Fast & memory-efficient functions to analyze and manipulate large data sets. It leverages the fast analytical capabilities of 'DuckDB' and its spatial extension (see <https://duckdb.org/community_extensions/extensions/h3>) while maintaining compatibility with R's spatial data ecosystem to work with spatial vector data.

URL https://cidree.github.io/duckh3/, https://github.com/Cidree/duckh3

BugReports https://github.com/Cidree/duckh3/issues

Encoding UTF-8

LazyData true

License GPL (>= 3)

Imports cli, DBI, dbplyr, dplyr, duckdb, duckspatial (>= 1.0.0), glue

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Suggests sf, testthat (>= 3.0.0)

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libicu-dev libzstd-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev xz-utils

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

Date/Publication 2026-04-25 09:42:28 UTC

RemoteUrl https://github.com/cidree/duckh3

RemoteRef HEAD

RemoteSha 59de57c82f3ba9ab5b0df53ecdbcd3b520d9045a

Contents

ddbh3_create_conn 2
ddbh3_default_conn 3
ddbh3_get_child_pos 4
ddbh3_get_hierarchy 5
ddbh3_get_icosahedron_faces 9
ddbh3_get_resolution 11
ddbh3_h3_to 13
ddbh3_is 17
ddbh3_lonlat_to 20
ddbh3_points_to 23
ddbh3_vertex 26

Index **30**

`ddbh3_create_conn` *Create a DuckDB connection with spatial and h3 extensions*

Description

It creates a DuckDB connection, and then it installs and loads the spatial and h3 extensions

Usage

```
ddbh3_create_conn(
  dbdir = "memory",
  threads = NULL,
  memory_limit_gb = NULL,
  bigint = "integer64",
  ...
)
```

Arguments

`dbdir`	String. Either "tempdir", "memory", or file path with .duckdb or .db extension. Defaults to "memory".
`threads`	Integer. Number of threads to use. If `NULL` (default), the setting is not changed, and DuckDB engine will use all available cores it detects (warning, on some shared HPC nodes the detected number of cores might be total number of cores on the node, not the per-job allocation).
`memory_limit_gb`	Numeric. Memory limit in GB. If `NULL` (default), the setting is not changed, and DuckDB engine will use 80% of available operating system memory it detects (warning, on some shared HPC nodes the detected memory might be the full node memory, not the per-job allocation).
`bigint`	String. How to handle 64-bit integers. One of "integer64" or "numeric"
`...`	Other parameters passed to `DBI::dbConnect()`

Value

A duckdb_connection

Examples

```
# load packages
library(duckspatial)
library(duckh3)

# create a duckdb database in memory
conn <- ddbh3_create_conn(dbdir = "memory", threads = 1)

# create an in-memory connection with 1 thread and 2GB memory limit
conn <- ddbh3_create_conn(threads = 1, memory_limit_gb = 2)

# Create a persistent database in disk
# conn <- ddbh3_create_conn(dbdir = "my_database.duckdb")

ddbs_stop_conn(conn)
```

---

ddbh3_default_conn *Get or create default DuckDB connection*

Description

Setup the default connection with h3 and spatial extensions installed and loaded. It will be used internally by the package functions if no other connection is provided

Usage

```
ddbh3_default_conn(create = TRUE, upgrade_h3 = FALSE, ...)
```

Arguments

create	Logical. If TRUE and no connection exists, create one. Default is TRUE.
upgrade_h3	Logical. If TRUE, will attempt to upgrade the h3 extension
...	Additional parameters to pass to duckspatial::ddbs_create_conn()

Value

Invisibly, the default duckdb_connection

Examples

```
# Get or create default connection
```

ddbh3_get_child_pos *Get the position of an H3 cell within its parent*

Description

Get the position of H3 cell indexes stored as strings or unsigned 64-bit integers (UBIGINT) within their parent cell at a specified resolution. The position is a zero-based index among all children of the parent cell.

Usage

```
ddbh3_get_child_pos(
  x,
  resolution = 8,
  h3 = "h3string",
  new_column = "h3child_pos",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

x	Input data. One of: duckspatial_df A lazy spatial data frame via dbplyr. sf A spatial data frame. tbl_lazy A lazy data frame from dbplyr. data.frame A standard R data frame. **character string** A table or view name in conn. **character vector** A vector of values to operate on in vectorized mode (requires conn = NULL).
resolution	A number specifying the resolution level of the H3 string (between 0 and 15)
h3	The name of a column in x containing the H3 strings or H3 unsigned 64-bit integers (UBIGINT)
new_column	Name of the new column to create on the input data. If NULL, the function will return a vector with the result
conn	A connection object to a DuckDB database. If NULL, the function runs on a temporary DuckDB database.
name	A character string of length one specifying the name of the table, or a character string of length two specifying the schema and table names. If NULL (the default), the function returns the result as an sf object
overwrite	Boolean. whether to overwrite the existing table if it exists. Defaults to FALSE. This argument is ignored when name is NULL.
quiet	A logical value. If TRUE, suppresses any informational messages. Defaults to FALSE.

Value

One of the following, depending on the inputs:

`tbl_lazy` If `x` is not spatial.

`duckspatial_df` If `x` is spatial (e.g. an `sf` or `duckspatial_df` object).

`TRUE` **(invisibly)** If `name` is provided, a table is created in the connection and `TRUE` is returned invisibly.

vector If `x` is a character vector and `conn = NULL`, the function operates in vectorized mode, returning a vector of the same length as `x`.

Examples

```
## Load needed packages
library(duckh3)
library(dplyr)

## Setup the default connection with h3 and spatial extensions
## This is a mandatory step to use duckh3 functions
ddbh3_default_conn(threads = 1)

## Load example data
points_tbl <- read.csv(
  system.file("extdata/example_pts.csv", package = "duckh3")
)

## Add H3 string column
points_tbl <- ddbh3_lonlat_to_h3(points_tbl, resolution = 6)

## Get position relative to resolution 4
ddbh3_get_child_pos(points_tbl, resolution = 4)
```

---

ddbh3_get_hierarchy *Get parent and children H3 cells*

Description

Get the parent or children H3 cells of H3 cell indexes stored as strings or unsigned 64-bit integers (`UBIGINT`) at a specified resolution: `ddbh3_get_parent()`, `ddbh3_get_center_child()`, `ddbh3_get_children()`, and `ddbh3_get_n_children()`.

Usage

```
ddbh3_get_parent(
  x,
  resolution = 8,
  h3 = "h3string",
  new_column = "h3parent",
```

```
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)

ddbh3_get_children(
  x,
  resolution = 8,
  h3 = "h3string",
  new_column = "h3children",
  conn = NULL,
  name = NULL,
  nested = FALSE,
  overwrite = FALSE,
  quiet = FALSE
)

ddbh3_get_n_children(
  x,
  resolution = 8,
  h3 = "h3string",
  new_column = "h3n_children",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)

ddbh3_get_center_child(
  x,
  resolution = 8,
  h3 = "h3string",
  new_column = "h3center_child",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

x	Input data. One of: `duckspatial_df` A lazy spatial data frame via dbplyr. `sf` A spatial data frame. `tbl_lazy` A lazy data frame from dbplyr. `data.frame` A standard R data frame. **character string** A table or view name in `conn`.

character vector A vector of values to operate on in vectorized mode (requires `conn = NULL`).

`resolution` A number specifying the resolution level of the H3 string (between 0 and 15)

`h3` The name of a column in `x` containing the H3 strings or H3 unsigned 64-bit integers (`UBIGINT`)

`new_column` Name of the new column to create on the input data. If NULL, the function will return a vector with the result

`conn` A connection object to a DuckDB database. If `NULL`, the function runs on a temporary DuckDB database.

`name` A character string of length one specifying the name of the table, or a character string of length two specifying the schema and table names. If `NULL` (the default), the function returns the result as an `sf` object

`overwrite` Boolean. whether to overwrite the existing table if it exists. Defaults to `FALSE`. This argument is ignored when `name` is `NULL`.

`quiet` A logical value. If `TRUE`, suppresses any informational messages. Defaults to `FALSE`.

`nested` Logical. If `TRUE`, children are returned as a nested list column (one row per parent cell). If `FALSE` (default), the result is unnested so each child cell occupies its own row.

Details

The four functions differ in the type of related cell they retrieve:

- `ddbh3_get_parent()` returns the parent cell at a coarser resolution
- `ddbh3_get_center_child()` returns the center child cell at a finer resolution
- `ddbh3_get_children()` returns all children cells at a finer resolution
- `ddbh3_get_n_children()` returns the number of children cells at a finer resolution, without computing them

Value

One of the following, depending on the inputs:

`tbl_lazy` If `x` is not spatial.

`duckspatial_df` If `x` is spatial (e.g. an `sf` or `duckspatial_df` object).

`TRUE` **(invisibly)** If `name` is provided, a table is created in the connection and `TRUE` is returned invisibly.

vector If `x` is a character vector and `conn = NULL`, the function operates in vectorized mode, returning a vector of the same length as `x`.

Examples

```
## Load needed packages
library(duckh3)
library(dplyr)

## Setup the default connection with h3 and spatial extensions
## This is a mandatory step to use duckh3 functions
ddbh3_default_conn(threads = 1)

## Load example data
points_tbl <- read.csv(
  system.file("extdata/example_pts.csv", package = "duckh3")
)

## Add h3 strings
points_tbl <- ddbh3_lonlat_to_h3(points_tbl, resolution = 8)

## GET PARENTS ----------

## Get resolution-7 parent
points_parent_tbl <- ddbh3_get_parent(points_tbl, resolution = 7)

## Check the resolution
ddbh3_get_resolution(
  points_parent_tbl,
  h3 = "h3parent"
)

## Add with mutate
points_tbl |>
  mutate(parent4 = ddbh3_get_parent(h3string, 4))

## GET CHILDREN ----------

## Get level 9 children
children_9_tbl <- ddbh3_get_children(points_tbl, resolution = 9)

## Get level 9 children in a nested list
children_9_nested_tbl <- ddbh3_get_children(points_tbl, resolution = 9, nested = TRUE)

## Add with mutate (nested)
points_tbl |>
  mutate(children9 = ddbh3_get_children(h3string, 9))

## Add with mutate (unnested)
points_tbl |>
  mutate(children9 = ddbh3_get_children(h3string, 9)) |>
  mutate(children9 = unnest(children9))

## GET CENTER CHILD ------

## Get the center child of res 10 (1 child per row)
```

```
center_child_10_tbl <- ddbh3_get_center_child(points_tbl, resolution = 10)

## Add with mutate
points_tbl |>
  mutate(center = ddbh3_get_center_child(h3string, 9))

## NUMBER OF CHILDREN -----

## How many children of level 10 does each level 8 have?
n_children_tbl <- ddbh3_get_n_children(points_tbl, resolution = 10)

## Add with mutate
points_tbl |>
  mutate(n_children = ddbh3_get_n_children(h3string, 15))
```

`ddbh3_get_icosahedron_faces` *Get the icosahedron faces of H3 cell indexes*

Description

Get the icosahedron faces intersected by H3 cell indexes stored as strings or unsigned 64-bit integers (`UBIGINT`). Each H3 cell maps onto one or more of the 20 faces of the underlying icosahedron used to construct the H3 grid.

Usage

```
ddbh3_get_icosahedron_faces(
  x,
  h3 = "h3string",
  new_column = "h3faces",
  conn = NULL,
  name = NULL,
  nested = FALSE,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

`x`	Input data. One of: `duckspatial_df` A lazy spatial data frame via dbplyr. `sf` A spatial data frame. `tbl_lazy` A lazy data frame from dbplyr. `data.frame` A standard R data frame. **character string** A table or view name in `conn`.

character vector A vector of values to operate on in vectorized mode (requires `conn = NULL`).

`h3` The name of a column in `x` containing the H3 strings or H3 unsigned 64-bit integers (`UBIGINT`)

`new_column` Name of the new column to create on the input data. If NULL, the function will return a vector with the result

`conn` A connection object to a DuckDB database. If `NULL`, the function runs on a temporary DuckDB database.

`name` A character string of length one specifying the name of the table, or a character string of length two specifying the schema and table names. If `NULL` (the default), the function returns the result as an `sf` object

`nested` Logical. If `TRUE`, children are returned as a nested list column (one row per parent cell). If `FALSE` (default), the result is unnested so each child cell occupies its own row.

`overwrite` Boolean. whether to overwrite the existing table if it exists. Defaults to `FALSE`. This argument is ignored when `name` is `NULL`.

`quiet` A logical value. If `TRUE`, suppresses any informational messages. Defaults to `FALSE`.

Value

One of the following, depending on the inputs:

`tbl_lazy` If `x` is not spatial.

`duckspatial_df` If `x` is spatial (e.g. an `sf` or `duckspatial_df` object).

`TRUE` **(invisibly)** If `name` is provided, a table is created in the connection and `TRUE` is returned invisibly.

vector If `x` is a character vector and `conn = NULL`, the function operates in vectorized mode, returning a vector of the same length as `x`.

Examples

```
## Load needed packages
library(duckh3)
library(dplyr)

## Setup the default connection with h3 and spatial extensions
## This is a mandatory step to use duckh3 functions
ddbh3_default_conn(threads = 1)

## Load example data
points_tbl <- read.csv(
  system.file("extdata/example_pts.csv", package = "duckh3")
)

## Add H3 string column
points_tbl <- ddbh3_lonlat_to_h3(points_tbl, resolution = 6)
```

```
## Get faces (unnested)
faces_tbl <- ddbh3_get_icosahedron_faces(points_tbl)

## Get faces (nested)
faces_nested_tbl <- ddbh3_get_icosahedron_faces(points_tbl, nested = TRUE)

## Add using mutate (nested)
points_tbl |>
  mutate(faces = ddbh3_get_icosahedron_faces(h3string))

## Add using mutate (unnested)
points_tbl |>
  mutate(faces = ddbh3_get_icosahedron_faces(h3string)) |>
  mutate(faces_unnested = unnest(faces))
```

`ddbh3_get_resolution` *Get the resolution of H3 cell indexes*

Description

Get the resolution of H3 cell indexes stored as strings or unsigned 64-bit integers (`UBIGINT`) from an existing column.

Usage

```
ddbh3_get_resolution(
  x,
  h3 = "h3string",
  new_column = "h3resolution",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

`x`	Input data. One of: `duckspatial_df` A lazy spatial data frame via dbplyr. `sf` A spatial data frame. `tbl_lazy` A lazy data frame from dbplyr. `data.frame` A standard R data frame. **character string** A table or view name in `conn`. **character vector** A vector of values to operate on in vectorized mode (requires `conn = NULL`).

h3	The name of a column in x containing the H3 strings or H3 unsigned 64-bit integers (UBIGINT)
new_column	Name of the new column to create on the input data. If NULL, the function will return a vector with the result
conn	A connection object to a DuckDB database. If NULL, the function runs on a temporary DuckDB database.
name	A character string of length one specifying the name of the table, or a character string of length two specifying the schema and table names. If NULL (the default), the function returns the result as an sf object
overwrite	Boolean. whether to overwrite the existing table if it exists. Defaults to FALSE. This argument is ignored when name is NULL.
quiet	A logical value. If TRUE, suppresses any informational messages. Defaults to FALSE.

Value

One of the following, depending on the inputs:

tbl_lazy If x is not spatial.

duckspatial_df If x is spatial (e.g. an sf or duckspatial_df object).

TRUE **(invisibly)** If name is provided, a table is created in the connection and TRUE is returned invisibly.

vector If x is a character vector and conn = NULL, the function operates in vectorized mode, returning a vector of the same length as x.

Examples

```
## Load needed packages
library(duckh3)
library(duckspatial)
library(dplyr)

## Setup the default connection with h3 and spatial extensions
## This is a mandatory step to use duckh3 functions
ddbh3_default_conn(threads = 1)

## Load example data
points_tbl <- read.csv(
  system.file("extdata/example_pts.csv", package = "duckh3")
)

## Add h3 strings
points_tbl <- ddbh3_lonlat_to_h3(points_tbl, resolution = 10)

## Convert to duckspatial_df
points_ddbs <- ddbs_as_points(points_tbl)

## Get resolution of the h3 strings
ddbh3_get_resolution(points_tbl)
```

```
ddbh3_get_resolution(points_ddbs, new_column = "res")

## Add using mutate
points_tbl |>
  mutate(res = ddbh3_get_resolution(h3string))
```

---

ddbh3_h3_to *Convert H3 string or UBIGINT indexes to other representations*

Description

Convert H3 cell indexes stored as strings or UBIGINT into other representations (e.g. `lon`, `lat`, `spatial`)

Usage

```
ddbh3_h3_to_lon(
  x,
  h3 = "h3string",
  new_column = "lon",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)

ddbh3_h3_to_lat(
  x,
  h3 = "h3string",
  new_column = "lat",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)

ddbh3_h3_to_spatial(
  x,
  h3 = "h3string",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)

ddbh3_strings_to_bigint(
  x,
```

```
  h3 = "h3string",
  new_column = "h3bigint",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)

ddbh3_bigint_to_strings(
  x,
  h3 = "h3bigint",
  new_column = "h3string",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)

ddbh3_h3_to_points(
  x,
  h3 = "h3string",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

x	Input data. One of: duckspatial_df A lazy spatial data frame via dbplyr. sf A spatial data frame. tbl_lazy A lazy data frame from dbplyr. data.frame A standard R data frame. **character string** A table or view name in conn. **character vector** A vector of values to operate on in vectorized mode (requires conn = NULL).
h3	The name of a column in x containing the H3 strings or H3 unsigned 64-bit integers (UBIGINT)
new_column	Name of the new column to create on the input data. If NULL, the function will return a vector with the result
conn	A connection object to a DuckDB database. If NULL, the function runs on a temporary DuckDB database.
name	A character string of length one specifying the name of the table, or a character string of length two specifying the schema and table names. If NULL (the default), the function returns the result as an sf object

overwrite	Boolean. whether to overwrite the existing table if it exists. Defaults to FALSE. This argument is ignored when name is NULL.
quiet	A logical value. If TRUE, suppresses any informational messages. Defaults to FALSE.

Details

The four functions differ only in the output format:

- ddbh3_h3_to_spatial() converts H3 indexes to spatial hexagon polygons
- ddbh3_h3_to_lon() extracts the longitude of the H3 cell centroid
- ddbh3_h3_to_lat() extracts the latitude of the H3 cell centroid
- ddbh3_strings_to_bigint() converts H3 indexes to unsigned 64-bit integers (UBIGINT)
- ddbh3_bigint_to_strings() converts H3 indexes to strings (e.g. "8928308280fffff")
- ddbh3_h3_to_points() converts H3 indexes to spatial points located at the centroid of the H3 cell

Value

One of the following, depending on the inputs:

tbl_lazy If x is not spatial.

duckspatial_df If x is spatial (e.g. an sf or duckspatial_df object).

TRUE **(invisibly)** If name is provided, a table is created in the connection and TRUE is returned invisibly.

vector If x is a character vector and conn = NULL, the function operates in vectorized mode, returning a vector of the same length as x.

Examples

```
## Load needed packages
library(duckh3)
library(duckspatial)
library(dplyr)

## Setup the default connection with h3 and spatial extensions
## This is a mandatory step to use duckh3 functions
ddbh3_default_conn(threads = 1)

## Load example data
points_tbl <- read.csv(
  system.file("extdata/example_pts.csv", package = "duckh3")
)

## Add H3 string column
points_tbl <- ddbh3_lonlat_to_h3(points_tbl, resolution = 6) |>
  select(-lon, -lat)

## TO LON/LAT ------------
```

```
## Add longitude and latitude of the H3 string
points_coords_tbl <- points_tbl |>
  ddbh3_h3_to_lat() |>
  ddbh3_h3_to_lon()

## Add lon/lat with other names
points_coords_2_tbl <- points_tbl |>
  ddbh3_h3_to_lat(new_column = "latitude") |>
  ddbh3_h3_to_lon(new_column = "longitude")

## Add using mutate
points_tbl |>
  mutate(
    lon = ddbh3_h3_to_lon(h3string),
    lat = ddbh3_h3_to_lat(h3string)
  )

## TO SPATIAL ------------------

## Convert h3 strings to spatial polygons
points_ddbs <- ddbh3_h3_to_spatial(points_tbl)

## Collect as sf
points_sf <- ddbs_collect(points_ddbs)

## FROM STRING TO UBIGINT -----

## Add ubigint, and remove strings
points_bigint_tbl <- ddbh3_strings_to_bigint(
  points_tbl,
  new_column = "h3_integers"
) |>
  select(-h3string)

## Add using mutate
points_tbl |>
  mutate(h3int = ddbh3_strings_to_bigint(h3string))

## FROM UBIGINT TO STRING -----

## Add column with strings
points_strings_tbl <- ddbh3_bigint_to_strings(
  points_bigint_tbl,
  h3 = "h3_integers"
)

## Add using mutate
points_bigint_tbl |>
  mutate(h3string = ddbh3_bigint_to_strings(h3_integers))
```

ddbh3_is *Check properties of H3 cell indexes*

Description

Check properties of H3 cell indexes stored as strings or unsigned 64-bit integers

Usage

```
ddbh3_is_pentagon(
  x,
  h3 = "h3string",
  new_column = "ispentagon",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)

ddbh3_is_h3(
  x,
  h3 = "h3string",
  new_column = "ish3",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)

ddbh3_is_res_class_iii(
  x,
  h3 = "h3string",
  new_column = "isclassiii",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)

ddbh3_is_vertex(
  x,
  h3vertex = "h3vertex",
  new_column = "isvertex",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
```

)

Arguments

x	Input data. One of: duckspatial_df A lazy spatial data frame via dbplyr. sf A spatial data frame. tbl_lazy A lazy data frame from dbplyr. data.frame A standard R data frame. **character string** A table or view name in conn. **character vector** A vector of values to operate on in vectorized mode (requires conn = NULL).
h3	The name of a column in x containing the H3 strings or H3 unsigned 64-bit integers (UBIGINT)
new_column	Name of the new column to create on the input data. If NULL, the function will return a vector with the result
conn	A connection object to a DuckDB database. If NULL, the function runs on a temporary DuckDB database.
name	A character string of length one specifying the name of the table, or a character string of length two specifying the schema and table names. If NULL (the default), the function returns the result as an sf object
overwrite	Boolean. whether to overwrite the existing table if it exists. Defaults to FALSE. This argument is ignored when name is NULL.
quiet	A logical value. If TRUE, suppresses any informational messages. Defaults to FALSE.
h3vertex	Name of the column containing H3 vertex strings. Defaults to "h3vertex"

Details

The functions check different properties of H3 cell indexes, all returning a logical column:

- ddbh3_is_pentagon(): returns TRUE if the H3 cell is one of the 12 pentagonal cells that exist at each H3 resolution
- ddbh3_is_h3(): returns TRUE if the H3 cell index is a valid H3 cell
- ddbh3_is_res_class_iii(): returns TRUE if the H3 cell belongs to a Class III resolution (odd resolutions: 1, 3, 5, 7, 9, 11, 13, 15)
- ddbh3_is_vertex(): returns TRUE if the index is a valid H3 vertex

Value

One of the following, depending on the inputs:

tbl_lazy If x is not spatial.

duckspatial_df If x is spatial (e.g. an sf or duckspatial_df object).

TRUE **(invisibly)** If name is provided, a table is created in the connection and TRUE is returned invisibly.

vector If x is a character vector and conn = NULL, the function operates in vectorized mode, returning a vector of the same length as x.

Examples

```
## Load needed packages
library(duckh3)
library(dplyr)

## Setup the default connection with h3 and spatial extensions
## This is a mandatory step to use duckh3 functions
ddbh3_default_conn(threads = 1)

## Load example data
points_tbl <- read.csv(
  system.file("extdata/example_pts.csv", package = "duckh3")
)

## Add h3 strings
points_tbl <- ddbh3_lonlat_to_h3(points_tbl, resolution = 8)

## IS VALID H3 -------------

## Check if h3 indexes are valid
ddbh3_is_h3(points_tbl)

## Check in mutate
points_tbl |>
  mutate(valid = ddbh3_is_h3(h3string))

## IS PENTAGON -------------

## Check if h3 indexes are pentagons
ddbh3_is_pentagon(points_tbl)

## Check in mutate
points_tbl |>
  mutate(is_pent = ddbh3_is_pentagon(h3string))

## IS CLASS III ------------

## Check if h3 indexes belong to a Class III resolution
ddbh3_is_res_class_iii(points_tbl)

## Check across multiple resolutions
ddbh3_lonlat_to_h3(points_tbl, resolution = 7) |>
  ddbh3_is_res_class_iii()

## IS VERTEX ---------------
```

```
## Get vertexes first
vertex_tbl <- ddbh3_h3_to_vertex(points_tbl, n = 1)

## Check if indexes are valid vertexes
ddbh3_is_vertex(vertex_tbl, h3 = "h3vertex")

## Check in mutate (mix of h3 cells and vertexes)
vertex_tbl |>
  mutate(
    cell_valid  = ddbh3_is_h3(h3string),
    vertex_valid = ddbh3_is_vertex(h3vertex)
  )
```

---

ddbh3_lonlat_to *Convert longitude and latitude to H3 cell representations*

Description

Convert geographic coordinates (longitude and latitude) into H3 cell representations at a specified resolution, with different output formats

Usage

```
ddbh3_lonlat_to_spatial(
  x,
  lon = "lon",
  lat = "lat",
  resolution = 8,
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)

ddbh3_lonlat_to_h3(
  x,
  lon = "lon",
  lat = "lat",
  resolution = 8,
  new_column = "h3string",
  h3_format = "string",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

x	Input data. One of: duckspatial_df A lazy spatial data frame via dbplyr. sf A spatial data frame. tbl_lazy A lazy data frame from dbplyr. data.frame A standard R data frame. **character string** A table or view name in conn. **character vector** A vector of values to operate on in vectorized mode (requires conn = NULL).
lon	The name of a column in x containing the longitude
lat	The name of a column in x containing the latitude
resolution	A number specifying the resolution level of the H3 string (between 0 and 15)
conn	A connection object to a DuckDB database. If NULL, the function runs on a temporary DuckDB database.
name	A character string of length one specifying the name of the table, or a character string of length two specifying the schema and table names. If NULL (the default), the function returns the result as an sf object
overwrite	Boolean. whether to overwrite the existing table if it exists. Defaults to FALSE. This argument is ignored when name is NULL.
quiet	A logical value. If TRUE, suppresses any informational messages. Defaults to FALSE.
new_column	Name of the new column to create on the input data. If NULL, the function will return a vector with the result
h3_format	Character. The format of the H3 cell index: string or bigint

Details

The three functions differ only in the output format of the H3 cell index:

- ddbh3_lonlat_to_h3() returns H3 cell indexes as strings (e.g. "8928308280fffff") or as unsigned 64-bit integers (UBIGINT)
- ddbh3_lonlat_to_spatial() returns H3 cells as spatial hexagon polygons

Value

One of the following, depending on the inputs:

tbl_lazy If x is not spatial.

duckspatial_df If x is spatial (e.g. an sf or duckspatial_df object).

TRUE **(invisibly)** If name is provided, a table is created in the connection and TRUE is returned invisibly.

vector If x is a character vector and conn = NULL, the function operates in vectorized mode, returning a vector of the same length as x.

Examples

```
## Load needed packages
library(duckdb)
library(duckh3)
library(duckspatial)
library(dplyr)

## Setup the default connection with h3 and spatial extensions
## This is a mandatory step to use duckh3 functions
ddbh3_default_conn(threads = 1)

## Load example data
points_tbl <- read.csv(
  system.file("extdata/example_pts.csv", package = "duckh3")
)

## Create a connection with spatial and h3 extensions
conn <- ddbh3_create_conn(threads = 1)

## TO H3 ------------

## Add h3 strings as a new column (res 5)
points_strings_5_tbl <- ddbh3_lonlat_to_h3(
  points_tbl,
  resolution = 5
)

## Add h3 UBIGINT as a new column (res 8 by default)
points_bigint_8_tbl <- ddbh3_lonlat_to_h3(
  points_tbl,
  new_column = "h3bigint",
  h3_format  = "bigint"
)

## If column names are different from lon/lat:
points_renamed <- rename(points_tbl, long = lon, lati = lat)

ddbh3_lonlat_to_h3(
  points_renamed,
  lon = "long",
  lat = "lati",
  resolution = 10
)


## Create a new table in the connection
ddbh3_lonlat_to_h3(
  points_tbl,
  conn = conn,
  name = "points_strings_8"
)
```

```
## Open the created table lazily
points_lazy <- dplyr::tbl(conn, "points_strings_8")

## Read it in memory
points_eager <- dbReadTable(conn, "points_strings_8")


## TO SPATIAL -----------

## Add h3 strings as a new column (res 5)
points_5_ddbs <- ddbh3_lonlat_to_spatial(
  points_tbl,
  resolution = 5
)

## Create a new table in the connection
ddbh3_lonlat_to_spatial(
  points_tbl,
  conn = conn,
  name = "points_strings_spatial"
)

## Open the created table lazily
as_duckspatial_df("points_strings_spatial", conn)

## Read it in memory as an sf object
ddbs_read_table(conn, "points_strings_spatial")
```

---

ddbh3_points_to *Convert spatial points to H3 cell representations*

Description

Convert spatial point geometries into H3 cell representations at a specified resolution. Input must be an duckspatial_df, sf, or table with point geometries

Usage

```
ddbh3_points_to_spatial(
  x,
  resolution,
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)

ddbh3_points_to_h3(
```

```
  x,
  resolution,
  new_column = "h3string",
  h3_format = "string",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

x	Input data. One of: duckspatial_df A lazy spatial data frame via dbplyr. sf A spatial data frame. tbl_lazy A lazy data frame from dbplyr. data.frame A standard R data frame. **character string** A table or view name in conn. **character vector** A vector of values to operate on in vectorized mode (requires conn = NULL).
resolution	A number specifying the resolution level of the H3 string (between 0 and 15)
conn	A connection object to a DuckDB database. If NULL, the function runs on a temporary DuckDB database.
name	A character string of length one specifying the name of the table, or a character string of length two specifying the schema and table names. If NULL (the default), the function returns the result as an sf object
overwrite	Boolean. whether to overwrite the existing table if it exists. Defaults to FALSE. This argument is ignored when name is NULL.
quiet	A logical value. If TRUE, suppresses any informational messages. Defaults to FALSE.
new_column	Name of the new column to create on the input data. If NULL, the function will return a vector with the result
h3_format	Character. Output format for the H3 cell index. Either "string" (default) or "bigint". Only used in ddbh3_points_to_h3().

Details

The two functions differ in the output format:

- ddbh3_points_to_h3() returns the H3 cell index containing each point, either as a string or UBIGINT depending on h3_format
- ddbh3_points_to_spatial() returns the H3 cell hexagon polygon containing each point as a spatial geometry

Value

One of the following, depending on the inputs:

`tbl_lazy` If x is not spatial.

`duckspatial_df` If x is spatial (e.g. an `sf` or `duckspatial_df` object).

`TRUE` **(invisibly)** If `name` is provided, a table is created in the connection and `TRUE` is returned invisibly.

vector If x is a character vector and `conn = NULL`, the function operates in vectorized mode, returning a vector of the same length as x.

Examples

```
## Load needed packages
library(duckh3)
library(duckspatial)

## Setup the default connection with h3 and spatial extensions
## This is a mandatory step to use duckh3 functions
ddbh3_default_conn(threads = 1)

## Load example data
points_tbl <- read.csv(
  system.file("extdata/example_pts.csv", package = "duckh3")
)

## Convert to duckspatial_df
points_ddbs <- ddbs_as_points(points_tbl)

## TO H3 strings/ubigint ------------

## Add column with h3 strings at resolution 8
points_strings_ddbs <- ddbh3_points_to_h3(points_ddbs, 8)

## Add column with h3 ubigint at resolution 10
points_bigint_ddbs <- ddbh3_points_to_h3(
  points_ddbs,
  resolution = 8,
  new_column = "h3bigint",
  h3_format = "bigint"
)

## TO SPATIAL -----------------

## Convert from POINTS to H3 POLYGONS of res 8
polygons_8_ddbs <- ddbh3_points_to_spatial(points_ddbs, 8)

## Collect as sf
polygons_8_sf <- ddbs_collect(polygons_8_ddbs)
```

ddbh3_vertex *Convert H3 cell indexes to vertex representations*

Description

Convert H3 cell indexes stored as strings or unsigned 64-bit integers (UBIGINT) to their vertex representations

Usage

```
ddbh3_h3_to_vertex(
  x,
  n = 0,
  h3 = "h3string",
  new_column = "h3vertex",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)

ddbh3_vertex_to_lon(
  x,
  h3vertex = "h3vertex",
  new_column = "lon_vertex",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)

ddbh3_vertex_to_lat(
  x,
  h3vertex = "h3vertex",
  new_column = "lat_vertex",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)

ddbh3_h3_to_vertexes(
  x,
  h3 = "h3string",
  new_column = "h3vertex",
  conn = NULL,
  name = NULL,
```

```
  nested = FALSE,
  overwrite = FALSE,
  quiet = FALSE
)

ddbh3_vertex_to_spatial(
  x,
  h3vertex = "h3vertex",
  conn = NULL,
  name = NULL,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

x	Input data. One of: duckspatial_df A lazy spatial data frame via dbplyr. sf A spatial data frame. tbl_lazy A lazy data frame from dbplyr. data.frame A standard R data frame. **character string** A table or view name in conn. **character vector** A vector of values to operate on in vectorized mode (requires conn = NULL).
n	Integer. Vertex number to retrieve. Must be in the range 0–5 for hexagons and 0–4 for pentagons. Only used in ddbh3_h3_to_vertex().
h3	The name of a column in x containing the H3 strings or H3 unsigned 64-bit integers (UBIGINT)
new_column	Name of the new column to create on the input data. If NULL, the function will return a vector with the result
conn	A connection object to a DuckDB database. If NULL, the function runs on a temporary DuckDB database.
name	A character string of length one specifying the name of the table, or a character string of length two specifying the schema and table names. If NULL (the default), the function returns the result as an sf object
overwrite	Boolean. whether to overwrite the existing table if it exists. Defaults to FALSE. This argument is ignored when name is NULL.
quiet	A logical value. If TRUE, suppresses any informational messages. Defaults to FALSE.
h3vertex	Name of the column containing H3 vertex strings. Defaults to "h3vertex"
nested	Logical. If TRUE, children are returned as a nested list column (one row per parent cell). If FALSE (default), the result is unnested so each child cell occupies its own row.

Details

The functions cover the full vertex workflow:

- `ddbh3_h3_to_vertex()` returns a single vertex of an H3 cell, identified by its vertex number `n` (0–5 for hexagons, 0–4 for pentagons), as an H3 vertex string
- `ddbh3_h3_to_vertexes()` returns all vertices of an H3 cell as H3 vertex strings — either nested (one row per cell) or unnested (one row per vertex) depending on `nested`
- `ddbh3_vertex_to_lat()` returns the latitude of an H3 vertex string
- `ddbh3_vertex_to_lon()` returns the longitude of an H3 vertex string
- `ddbh3_vertex_to_spatial()` converts H3 vertex strings to spatial point geometries. If the input column is nested, vertices are automatically unnested and aggregated into a `MULTIPOINT` geometry per cell

Value

One of the following, depending on the inputs:

`tbl_lazy` If `x` is not spatial.

`duckspatial_df` If `x` is spatial (e.g. an `sf` or `duckspatial_df` object).

`TRUE` **(invisibly)** If `name` is provided, a table is created in the connection and `TRUE` is returned invisibly.

vector If `x` is a character vector and `conn = NULL`, the function operates in vectorized mode, returning a vector of the same length as `x`.

Examples

```
## Load needed packages
library(duckh3)
library(duckspatial)
library(dplyr)

## Setup the default connection with h3 and spatial extensions
## This is a mandatory step to use duckh3 functions
ddbh3_default_conn(threads = 1)

## Load example data
points_tbl <- read.csv(
  system.file("extdata/example_pts.csv", package = "duckh3")
)

## Add h3 strings
points_tbl <- ddbh3_lonlat_to_h3(points_tbl, resolution = 8)

## TO VERTEX ---------------

## Add second vertex
vertex_2_tbl <- ddbh3_h3_to_vertex(points_tbl, n = 2)

## Add add vertexes (unnested)
```

```
vertexes_tbl <- ddbh3_h3_to_vertexes(points_tbl)

## Add add vertexes (nested)
vertexes_nested_tbl <- ddbh3_h3_to_vertexes(points_tbl, nested = TRUE)

## Add some vertexes with with mutate
points_tbl |>
  mutate(
    v1 = ddbh3_h3_to_vertex(h3string, 1),
    v3 = ddbh3_h3_to_vertex(h3string, 3)
  )

## VERTEX TO LON/LAT ------

## Add coords
coords_vertex_tbl <- vertex_2_tbl |>
  ddbh3_vertex_to_lon(new_column = "lon_v2") |>
  ddbh3_vertex_to_lat(new_column = "lat_v2")

## Add coords in mutate
vertex_2_tbl |>
  mutate(
    lon_v2 = ddbh3_vertex_to_lon(h3vertex),
    lat_v2 = ddbh3_vertex_to_lat(h3vertex)
  )

## VERTEX TO SPATIAL ------

## Convert unnested vertexes (returns POINTS)
ddbh3_h3_to_vertexes(points_tbl) |>
  ddbh3_vertex_to_spatial()


## Convert nested vertexes (returns MULTIPOINTS)
ddbh3_h3_to_vertexes(points_tbl, nested = TRUE) |>
  ddbh3_vertex_to_spatial()
```

Index

ddbh3_bigint_to_strings *(*ddbh3_h3_to*)*, 13
ddbh3_create_conn, 2
ddbh3_default_conn, 3
ddbh3_get_center_child *(*ddbh3_get_hierarchy*)*, 5
ddbh3_get_child_pos, 4
ddbh3_get_children *(*ddbh3_get_hierarchy*)*, 5
ddbh3_get_hierarchy, 5
ddbh3_get_icosahedron_faces, 9
ddbh3_get_n_children *(*ddbh3_get_hierarchy*)*, 5
ddbh3_get_parent *(*ddbh3_get_hierarchy*)*, 5
ddbh3_get_resolution, 11
ddbh3_h3_to, 13
ddbh3_h3_to_lat *(*ddbh3_h3_to*)*, 13
ddbh3_h3_to_lon *(*ddbh3_h3_to*)*, 13
ddbh3_h3_to_points *(*ddbh3_h3_to*)*, 13
ddbh3_h3_to_spatial *(*ddbh3_h3_to*)*, 13
ddbh3_h3_to_vertex *(*ddbh3_vertex*)*, 26
ddbh3_h3_to_vertexes *(*ddbh3_vertex*)*, 26
ddbh3_is, 17
ddbh3_is_h3 *(*ddbh3_is*)*, 17
ddbh3_is_pentagon *(*ddbh3_is*)*, 17
ddbh3_is_res_class_iii *(*ddbh3_is*)*, 17
ddbh3_is_valid *(*ddbh3_is*)*, 17
ddbh3_is_vertex *(*ddbh3_is*)*, 17
ddbh3_lonlat_to, 20
ddbh3_lonlat_to_h3 *(*ddbh3_lonlat_to*)*, 20
ddbh3_lonlat_to_spatial *(*ddbh3_lonlat_to*)*, 20
ddbh3_points_to, 23
ddbh3_points_to_h3 *(*ddbh3_points_to*)*, 23
ddbh3_points_to_spatial *(*ddbh3_points_to*)*, 23
ddbh3_strings_to_bigint *(*ddbh3_h3_to*)*, 13
ddbh3_vertex, 26
ddbh3_vertex_to_lat *(*ddbh3_vertex*)*, 26
ddbh3_vertex_to_lon *(*ddbh3_vertex*)*, 26
ddbh3_vertex_to_spatial *(*ddbh3_vertex*)*, 26
duckspatial::ddbs_create_conn(), *3*