Package 'h3jsr'

January 21, 2023

Type Package

Title Access Uber's H3 Library

Version 1.3.1

Date 2023-01-21

Description
Provides access to Uber's H3 library for geospatial indexing via its JavaScript transpile 'h3-js' <`https://github.com/uber/h3-js`> and 'V8' <`https://github.com/jeroen/v8`>.

License Apache License (>= 2)

Encoding UTF-8

LazyData true

Depends R (>= 2.10)

SystemRequirements V8 engine version 6+ is needed for modern JS and
WASM support. On Debian / Ubuntu install either libv8-dev or
libnode-dev, on Fedora use v8-devel.

Imports geojsonsf, methods, sf, tidyr, utils, V8 (>= 4.0.0)

Suggests covr, dplyr, ggplot2, knitr, rmarkdown, testthat

RoxygenNote 7.2.3

VignetteBuilder knitr

URL `https://obrl-soil.github.io/h3jsr/`

NeedsCompilation no

Author Lauren O'Brien [aut, cre] (<`https://orcid.org/0000-0002-7336-2171`>)

Maintainer Lauren O'Brien <`obrlsoilau@gmail.com`>

Repository CRAN

Date/Publication 2023-01-21 09:20:10 UTC

R topics documented:

`h3jsr-package` *h3jsr: Access Uber's H3 library*

Description

This package uses package V8 to access the javascript bindings for Uber's H3 library

Author(s)

Maintainer: Lauren O'Brien <`obrlsoilau@gmail.com`> (ORCID)

See Also

Useful links:

- `https://obrl-soil.github.io/h3jsr/`

`are_neighbours` *check if H3 cells are neighbours*

Description

This function checks whether two H3 cells share an edge.

Usage

```
are_neighbours(origin = NULL, destination = NULL, simple = TRUE)
```

Arguments

`origin`	Character; 15-character cell index generated by H3. A vector of indexes can also be supplied.
`destination`	Character; 15-character cell index generated by H3. A vector of indexes can also be supplied.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

Logical; `TRUE` if neighbours.

Note

- The number of indexes supplied to origin and destination must be equal.
- This function will always return false if the indexes are of different resolutions.

Examples

```
# Are the following cells neighbours?
are_neighbours(origin = '86be8d12fffffff', destination = '86be8d127ffffff')
```

---

cells_to_multipolygon *Get geometry for a set of H3 cells*

Description

This function returns geometry associated with a set of H3 cells, as a single `sfc_MULTIPOLYGON`.

Usage

```
cells_to_multipolygon(h3_addresses = NULL, simple = TRUE)
```

Arguments

`h3_addresses`	Character vector or list of 15-character cell indices generated by H3.
`simple`	Logical; whether to return an `sfc_MULTIPOLYGON` or an `sf` object including the input cells.

Value

By default, object of type `sfc_MULTIPOLYGON` of length 1.

Note

The geometry returned by this function will not be valid where the addresses supplied overlap at the same resolution. The main use case for this function appears to be visualising the outputs of `polygon_to_cells` and `compact`.

Examples

```
## Not run:
# Give me the outline of the cells around Brisbane Town Hall at
# resolution 10 (not run as slow-ish)
bth <- sf::st_sfc(sf::st_point(c(153.023503, -27.468920)), crs = 4326)
bth_10 <- point_to_h3(bth, res = 10)
bth_patch <- get_disk(h3_address = bth_10, ring_size = 2)
bth_patch_sf <- cells_to_multipolygon(bth_patch)

## End(Not run)
```

`cell_area` *Get exact cell area*

Description

This function calculates the exact area of an H3 cell.

Usage

```
cell_area(h3_address = NULL, units = c("m2", "km2", "rads2"), simple = TRUE)
```

Arguments

`h3_address`	Character; 15-character index generated by H3.
`units`	Length unit to report in. Options are square meters, square kilometers, or steradians.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, a numeric vector of length(h3_address).

Examples

```
cell_area(h3_address = '8abe8d12acaffff', 'm2')
```

`cell_to_childpos` *Cell to Child position*

Description

Get the position of the cell within an ordered list of all children of the cell's parent at the specified resolution.

Usage

```
cell_to_childpos(h3_address = NULL, parent_res = NULL, simple = TRUE)
```

Arguments

`h3_address`	Character; 15-character index generated by H3.
`parent_res`	numeric; resolution of reference parent cell.
`simple`	Logical; whether to return a vector or a data frame containing both inputs and outputs.

Value

Numeric, Position of child within parent at ‘parent_res‘.

Note

Function will return 0 if ‘parent_res‘ is the same as the resolution of the supplied cell.

Examples

```
# example address has resolution 7
cell_to_childpos('872830b82ffffff', c(3,4,5,6), simple = FALSE)
```

---

cell_to_children_size *Cell to children size*

Description

Get the number of children for a cell at a given resolution.

Usage

```
cell_to_children_size(h3_address = NULL, child_res = NULL, simple = TRUE)
```

Arguments

h3_address	Character; 15-character index generated by H3.
child_res	numeric; child resolution to report on.
simple	Logical; whether to return a vector or a data frame containing both inputs and outputs.

Value

numeric; number of children at the requested resolution

Examples

```
# example address has resolution 7:
cell_to_children_size('872830b82ffffff', c(8,9,10,11), simple = FALSE)
```

cell_to_line *Convert H3 cell indexes to a line*

Description

Return line geometry for a sequence of H3 cell indexes in WGS84 coordinates.

Usage

```
cell_to_line(input = NULL, simple = TRUE)

## S3 method for class 'data.frame'
cell_to_line(input = NULL, simple = TRUE)

## S3 method for class 'list'
cell_to_line(input = NULL, simple = TRUE)

## S3 method for class 'character'
cell_to_line(input = NULL, simple = TRUE)
```

Arguments

`input`	Character vector of 15-character indexes generated by H3, a list of such, or a data frame where the last column is a list-column of H3 cell indexes (usually the output of `h3jsr::grid_path()`.
`simple`	Logical; whether to return an `sfc_LINESTRING` object or an `sf` data frame containing both inputs and outputs.

Value

An `sfc_LINESTRING` object containing a line for each vector of H3 cell indexes supplied. If `simple = FALSE`, an `sf` object including the input data.

Note

This function can accept any arbitrary vector of cell indexes (including cells at multiple resolutions) but results may be unexpected. It is assumed that indexes are supplied in a pre-ordered fashion.

Examples

```
# What is the cell index over the Brisbane Town Hall at resolution 10?
brisbane_hex_10 <- cell_to_polygon(input = '8abe8d12acaffff')

# Give me a some nearby cells
hex_sample <- get_disk_list('8abe8d12acaffff', 4)[[1]][[4]][seq(1,18,3)]
hex_sample_polys <- cell_to_polygon(hex_sample)

# find connecting paths
```

```
paths <- grid_path(rep('8abe8d12acaffff', 6), hex_sample)

# make lines
lines <- cell_to_line(paths)

## Not run:
plot(hex_sample_polys, reset = FALSE)
plot(brisbane_hex_10, add = TRUE)
plot(lines, col = 'red', add = TRUE)

## End(Not run)
```

`cell_to_point` *Convert H3 cell index to point location*

Description

This function takes a H3 cell index and returns its center coordinates in WGS84.

Usage

```
cell_to_point(h3_address = NULL, simple = TRUE)
```

Arguments

`h3_address`	Character; 15-character index generated by H3.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, an `sfc_POINT` object of `length(h3_address)`. EPSG:WGS84.

Examples

```
# Where is the center of the hexagon over the Brisbane Town Hall at resolution 10?
brisbane_10 <- cell_to_point(h3_address = '8abe8d12acaffff')
```

`cell_to_polygon` *Get the boundary of an H3 cell index*

Description

This function takes an H3 cell index and returns its bounding shape (usually a hexagon) in WGS84.

Usage

```
cell_to_polygon(input = NULL, simple = TRUE)
```

Arguments

`input`	Character; 15-character index generated by H3, or a vector or list of same, or a data frame where the first column contains H3 addresses.
`simple`	Logical; whether to return an `sfc_POLYGON` object or an `sf` data frame containing both inputs and outputs.

Value

By default, an `sfc_POLYGON` object of `length(input)`. If an appropriately formatted data frame is supplied, an `sf` data frame containing input attributes and geometry.

Examples

```
# What is the hexagon over the Brisbane Town Hall at resolution 10?
brisbane_hex_10 <- cell_to_polygon(input = '8abe8d12acaffff')

# Give me some of the cells over Brisbane Town Hall as an sf object
bth <- sf::st_sfc(sf::st_point(c(153.023503, -27.468920)), crs = 4326)
bth_addys <- unlist(point_to_cell(bth, res = seq(10, 15)), use.names = FALSE)
bth_hexes <- cell_to_polygon(input = bth_addys)
plot(bth_hexes, axes = TRUE)
```

`cell_to_splitlong` *H3 cell to split long*

Description

Convert an H3 cell (64-bit hexidecimal string) into a "split long" - a pair of 32-bit integers.

Usage

```
cell_to_splitlong(h3_address, simple = TRUE)
```

Arguments

`h3_address`	Character; 15-character index generated by H3.
`simple`	Logical; whether to return a vector or a data frame containing both inputs and outputs.

Value

list of integer pairs, one for each address supplied.

Examples

```
cell_to_splitlong(h3_address = '8abe8d12acaffff')
```

`childpos_to_cell` *Child position to cell*

Description

Get the child cell at a given position within an ordered list of all children at the specified resolution.

Usage

```
childpos_to_cell(
  child_pos = NULL,
  h3_address = NULL,
  child_res = NULL,
  simple = TRUE
)
```

Arguments

`child_pos`	numeric; position of the child cell to get.
`h3_address`	Character; 15-character index generated by H3.
`child_res`	numeric; resolution of the child cell to return.
`simple`	Logical; whether to return a vector or a data frame containing both inputs and outputs.

Value

Character, H3 address of child

Note

‘child_pos‘ is 0-indexed and capped at the maximum number of hexagons within the parent cell at the supplied resolution. This figure can be determined using `cell_to_children_size`.

Examples

```
# example address has resolution 7:
childpos_to_cell(0, '872830b82ffffff', 9, simple = FALSE)
```

`compact` *Compact H3 cells*

Description

This function compacts a set of cells of the same resolution into a set of cells across multiple resolutions that represents the same area.

Usage

```
compact(h3_addresses = NULL, simple = TRUE)
```

Arguments

`h3_addresses`	Character vector or list of 15-character indices generated by H3 at a single resolution, generally the output of `polygon_to_cells`.
`simple`	Logical; whether to return a vector of outputs or a list object containing both inputs and outputs.

Value

A list of H3 cells with multiple resolutions. The minimum resolution of the output list matches the resolution of the input list.

Examples

```
## Not run:
# Give me a compacted representation of County Ashe, NC
nc <- sf::st_read(system.file("shape/nc.shp", package="sf"), quiet = TRUE)
nc1 <- nc[1, ]
nc1 <- sf::st_cast(nc1, 'POLYGON')
fillers <- polygon_to_cells(geometry = nc1, res = 6)
compacted <- compact(fillers)

## End(Not run)
```

degs_to_rads *Convert degrees to radians*

Description

Convert degrees to radians.

Usage

```
degs_to_rads(degree = NULL, lang = c("r", "h3"), simple = TRUE)
```

Arguments

`degree`	Numeric, value in degrees
`lang`	Character; whether to perform the conversion using base R or the H3 library. Defaults to R for speed.
`simple`	Logical; whether to return a vector or a data frame containing both inputs and outputs.

Value

Numeric, value in radians

Examples

```
degs_to_rads(120)
```

edge_length *Get exact cell edge length*

Description

This function calculates the exact length of an H3 cell edge.

Usage

```
edge_length(h3_edge = NULL, units = c("m", "km", "rads"), simple = TRUE)
```

Arguments

`h3_edge`	Character; address of unidirectional edge.
`units`	Length unit to report in. Options are meters, kilometers, or radians.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, a numeric vector of length(h3_address).

Examples

```
edge_length(h3_edge = '166be8d12fffffff', 'm')
```

`get_base_cell` *get the base cell of an H3 cell index*

Description

This function returns the number of the base (Level 1) cell for an H3 cell idnex.

Usage

```
get_base_cell(h3_address = NULL, simple = TRUE)
```

Arguments

`h3_address`	Character; 15-character index generated by H3.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, an integer vector of `length(h3_address)`, ranging from 0 to 121.

Examples

```
# What is Brisbane Town Hall's base cell number?
get_base_cell(h3_address = '8abe8d12acaffff')
```

`get_cell_vertex` *Get a vertex index*

Description

This function returns the vertex index for a supplied H3 cell and vertex number.

Usage

```
get_cell_vertex(h3_address = NULL, v_num = 0, simple = TRUE)
```

Arguments

`h3_address`	Character; 15-character cell index generated by H3. A vector of indexes can also be supplied.
`v_num`	Numeric; the vertex number required. Options are 0-5 inclusive.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, a list of `length(h3_address)`.

Examples

```
# Get vertex 3 for this cell
get_cell_vertex(h3_address = '86be8d12fffffff', 3)
```

`get_cell_vertexes` *Get all vertex indexes*

Description

This function returns all 6 vertex indices for a supplied H3 cell.

Usage

```
get_cell_vertexes(h3_address = NULL, simple = TRUE)
```

Arguments

`h3_address`	Character; 15-character cell index generated by H3. A vector of indexes can also be supplied.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, a list of `length(h3_address)`.

Examples

```
# Get vertexes for this cell
get_cell_vertexes(h3_address = '86be8d12fffffff')
```

get_centerchild *get central child H3 cell index*

Description

This function returns the central child of a particular H3 cell index at the requested resolution.

Usage

```
get_centerchild(h3_address = NULL, res = NULL, simple = TRUE)
```

Arguments

h3_address	Character; 15-character index generated by H3.
res	Integer; Desired H3 resolution. See https://h3geo.org/docs/core-library/restable/ for allowable values and related dimensions.
simple	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, a list of `length(h3_address)`. Each list element contains a vector of H3 cells.

Examples

```
# What is the central child of this resolution 6 index at resolution 8?
get_centerchild(h3_address = '86be8d12fffffff', res = 8)
```

get_children *get child H3 cell indices*

Description

This function returns the children of a particular H3 cell at the requested resolution.

Usage

```
get_children(h3_address = NULL, res = NULL, simple = TRUE)
```

Arguments

h3_address	Character; 15-character index generated by H3.
res	Integer; Desired H3 resolution. See https://h3geo.org/docs/core-library/restable/ for allowable values and related dimensions.
simple	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, a list of `length(h3_address)`. Each list element contains a vector of H3 cell indexes.

Note

The number of cells returned for each request is `7 ^ (parent_res - child_res)`, so jumping three levels will return 343 indexes per request. This can cause memory issues with larger requests.

Examples

```
# What are the children of this resolution 6 cell index at resolution 8?
get_children(h3_address = '86be8d12fffffff', res = 8)
```

`get_disk` *Get nearby H3 cell indices*

Description

This function returns all the H3 cell indices within a specified number of steps from the index supplied.

Usage

```
get_disk(h3_address = NULL, ring_size = 1, simple = TRUE)
```

Arguments

`h3_address`	Character; 15-character cell index generated by H3.
`ring_size`	Character; number of steps away from the central cell. Defaults to 1.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, a list of `length(h3_address)`. Each list element contains a character vector of H3 cells.

Note

The number of cells returned for each input index conforms to the centered hexagonal number sequence, so at `ring_size` = 5, 91 addresses are returned. The first address returned is the input address, the rest follow in a spiral anticlockwise order.

Examples

```
# What are all the neighbours of this cell within two steps?
get_disk(h3_address = '86be8d12fffffff', ring_size = 2)
```

`get_disk_list` *Get nearby H3 cell indexes separated by distance*

Description

This function returns all the H3 cell indexes within a specified number of steps from the address supplied, grouped by step.

Usage

```
get_disk_list(h3_address = NULL, ring_size = 1, simple = TRUE)
```

Arguments

`h3_address`	Character; 15-character cell index generated by H3.
`ring_size`	Character; number of steps away from the central cell. Defaults to 1.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, a list of `length(h3_address)`. Each list element contains a list of `length(ring_size + 1)`. Each of those lists contains a character vector of H3 cell indices belonging to that step away from the input cell.

Note

In total, the number of indices returned for each input cell conforms to the centered hexagonal number sequence, so at `ring_size = 5`, 91 cells are returned. Cells are returned in separate lists, one for each step.

Examples

```
# What are the nested neighbours of this cell within two steps?
get_disk_list(h3_address = '86be8d12fffffff', ring_size = 2)
```

`get_faces` *get the icosahedron faces of an H3 cell index*

Description

This function returns the indices of all icosahedron faces intersected by a given H3 cell index.

Usage

```
get_faces(h3_address = NULL, simple = TRUE)
```

Arguments

`h3_address`	Character; 15-character index generated by H3.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, an integer vector of `length(h3_address)`, ranging from 1 to 20. If `simple = FALSE`, a data.frame with a column of H3 cell indexes and a list-column of faces.

Examples

```
# Which faces does this h3 cell index intersect?
get_faces(h3_address = '8abe8d12acaffff')
```

`get_gcdist` *Great circle distance*

Description

Get the great circle distance between WGS84 lat/long points

Usage

```
get_gcdist(pt1 = NULL, pt2 = NULL, units = c("m", "km", "rads"), simple = TRUE)
```

Arguments

`pt1`	‘sf‘ object with point geometry, ‘sfc_POINT‘ object, ‘sfg‘ point, data frame or matrix.
`pt2`	‘sf‘ object with point geometry, ‘sfc_POINT‘ object, ‘sfg‘ point, data frame or matrix.
`units`	whether to return the great circle distance in meters, kilometers, or radians.
`simple`	whether to return a numeric vector of distances or a ‘data.frame‘ containing start and end coordinates as well as distance.

Value

Numeric vector of point to point distances, or data frame of origin and destination coordinates accompanied by their distances.

Note

This functionality also exists in R packages `sp`, `sf`, `geosphere` and `fields`. H3’s version appears to return slightly shorter distances than most other implementations, but is included here for completeness.

Examples

```
# distance between Brisbane and Melbourne
bne <- c(153.028, -27.468)
mlb <- c(144.963, -37.814)
get_gcdist(bne, mlb, 'km')
```

`get_local_cell` *Get H3 cell from local i, j coordinates*

Description

This function returns H3 destination cells for local i, j coordinate pairs anchored by an H3 origin cell.

Usage

```
get_local_cell(origin = NULL, i = NULL, j = NULL, simple = TRUE)
```

Arguments

`origin`	Character; 15-character cell index generated by H3. A vector of indexes can also be supplied.
`i`	a single i coordinate or vector of same, generated by `get_local_ij`
`j`	a single j coordinate or vector of same, generated by `get_local_ij`
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

If `simple` = `TRUE`, a character vector of destination H3 cells. If not, a data frame containing columns origin, i, j, destination.

Note

- The coordinate space used by this function may have deleted regions or warping due to pentagonal distortion.
- Coordinates are only comparable if they come from the same origin cell.
- Failure may occur if the destination is too far away from the origin or if the destination is on the other side of a pentagon.
- This function is experimental, and its output is not guaranteed to be compatible across different versions of H3.

Examples

```
# Get local coordinates for a nearby cell
local <- get_local_ij(origin = '86be8d12fffffff', destination = '86be8d127ffffff')

# Convert back to destination cell
get_local_cell(origin = '86be8d12fffffff', i = local[, 1], j = local[, 2])
```

get_local_ij *Get local i, j coordinates*

Description

This function defines local i, j coordinates for an H3 destination cell relative to an H3 origin cell.

Usage

```
get_local_ij(origin = NULL, destination = NULL, simple = TRUE)
```

Arguments

`origin`	Character; 15-character cell index generated by H3. A vector of indexes can also be supplied.
`destination`	Character; 15-character cell index generated by H3. A vector of indexes can also be supplied.
`simple`	Logical; whether to include an unprojected sfc_POINT geometry column in the output object.

Value

If `simple` = `TRUE`, a matrix where each row contains the local i, j coordinates for the supplied destination indexes. If not, an `sf` object with origin and destination attributes, point geometry of the destination cell centers, and an undefined coordinate reference system.

Note

- The number of indexes supplied to origin and destination must be equal.
- The coordinate space used by this function may have deleted regions or warping due to pentagonal distortion.
- Coordinates are only comparable if they come from the same origin index.
- Failure may occur if the index is too far away from the origin or if the index is on the other side of a pentagon.
- This function is experimental, and its output is not guaranteed to be compatible across different versions of H3.

Examples

```
# Get local coordinates for a nearby cell
get_local_ij(origin = '86be8d12fffffff', destination = '86be8d127ffffff')

# Get local coordinates for a donut of nearby cells
destinations <- get_ring(h3_address = '86be8d12fffffff', ring_size = 2)
local_coords <- get_local_ij(origin = rep('86be8d12fffffff', length(destinations[[1]])),
                             destination = destinations[[1]],
                             simple = FALSE)

plot(local_coords['destination'], pch = 19) # note origin is (0,0)
```

`get_parent` *get parent H3 cell index*

Description

This function returns the parent of a particular H3 cell index at the requested resolution.

Usage

```
get_parent(h3_address = NULL, res = NULL, simple = TRUE)
```

Arguments

`h3_address`	Character; 15-character index generated by H3.
`res`	Integer; Desired H3 resolution. See `https://h3geo.org/docs/core-library/restable/` for allowable values and related dimensions.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, a logical vector of `length(h3_address)`.

Examples

```
# What is the parent of this cell at resolution 6?
get_parent(h3_address = '8abe8d12acaffff', res = 6)
```

`get_pentagons` *get the pentagon indices for an H3 resolution*

Description

This function returns the indices of all pentagons occurring at a given H3 resolution.

Usage

```
get_pentagons(res = NULL, simple = TRUE)
```

Arguments

`res`	Integer; Desired H3 resolution. See `https://h3geo.org/docs/core-library/restable/` for allowable values and related dimensions.
`simple`	Logical; whether to return outputs as list of outputs (TRUE) or data frame with both inputs and outputs.

Value

By default, a list of `length(h3_address)`. Each list element contains a vector of twelve H3 addresses. If `simple = FALSE`, a data frame with a column of input resolutions and a list-column of pentagon indexes for each.

Examples

```
# Which indexes are pentagons at resolution 7?
get_pentagons(res = 7)
```

`get_res` *get the resolution of an H3 cell index*

Description

This function returns an H3 cell index's resolution level.

Usage

```
get_res(h3_address = NULL, simple = TRUE)
```

Arguments

`h3_address`	Character; 15-character index generated by H3.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, an integer vector of `length(h3_address)`, ranging from 1 to 15.

Examples

```
# What is the resolution of this H3 cell index?
get_res(h3_address = '8abe8d12acaffff')
```

`get_res0` *Get resolution 0 indexes*

Description

Get all H3 cell indexes at resolution 0.

Usage

```
get_res0()
```

Value

length 122 character vector of top-level H3 cell indices.

Note

As every index at every resolution > 0 is the descendant of a res 0 index, this can be used with `get_children` to iterate over H3 indexes at any resolution.

Examples

```
res0 <- get_res0()
cell_area(res0[1], 'km2')
```

`get_ring` *Get a ring of H3 cell indexes*

Description

This function returns all the H3 cell indexes at the specified step from the address supplied.

Usage

```
get_ring(h3_address = NULL, ring_size = 1, simple = TRUE)
```

Arguments

`h3_address`	Character; 15-character cell index generated by H3.
`ring_size`	Character; number of steps away from the central cell. Defaults to 1.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, a list of `length(h3_address)`. Each list element contains a character vector of H3 cells belonging to that step away from the input address.

Note

In total, the number of cells returned for each input index is `ring_size * 6`. This function will throw an error if there is a pentagon anywhere in the ring.

Examples

```
# What are the neighbours of this cell at step 2?
get_ring(h3_address = '86be8d12fffffff', ring_size = 2)
```

`get_uddest` *Get destination cell from directed edge*

Description

Get an H3 index representing the destination of a directed edge.

Usage

```
get_uddest(h3_edge = NULL, simple = TRUE)
```

Arguments

`h3_edge`	Character; address of unidirectional edge.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, character vector of h3 cell indexes.

Examples

```
# Get the destination cell index of this directed edge index
get_uddest(h3_edge = '166be8d12fffffff')
```

`get_udedge` *Get a unidirectional edge index*

Description

Returns an H3 index representing a unidirectional edge for a given origin and destination cell pair.

Usage

```
get_udedge(origin = NULL, destination = NULL, simple = TRUE)
```

Arguments

`origin`	Character; 15-character cell index generated by H3. A vector of indexes can also be supplied.
`destination`	Character; 15-character cell index generated by H3. A vector of indexes can also be supplied.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, character vector of unidirectional edge indexes.

Note

The number of cell indexes supplied to origin and destination must be equal.

Examples

```
# Return the unidirectional edge representing the transition between these two cells:
get_udedge(origin = '86be8d12fffffff', destination = '86be8d127ffffff')
```

`get_udedges` *Get all directed edge indexes for a given H3 cell*

Description

Get all directed edge indexes for a given H3 cell index.

Usage

```
get_udedges(h3_address = NULL, simple = TRUE)
```

Arguments

`h3_address`	Character; 15-character index generated by H3.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, list of `length(h3_address)`. Each list contains a character vector of H3 edge indexes.

Examples

```
# Get all the edge indexes for this cell
get_udedges(h3_address = '86be8d12fffffff')
```

`get_udends` *Get origin and destination indexes of directed edge*

Description

Get H3 cell indexes representing the origin and destination of a directed edge index.

Usage

```
get_udends(h3_edge = NULL, simple = TRUE)
```

Arguments

`h3_edge`	Character; address of unidirectional edge.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, character matrix of h3 cell indexes.

Examples

```
# Get the origin and destination of this directed edge
get_udends(h3_edge = '166be8d12fffffff')
```

get_udorigin — *Get origin cell index from directed edge*

Description

Get an H3 cell index representing the origin of a directed edge.

Usage

```
get_udorigin(h3_edge = NULL, simple = TRUE)
```

Arguments

`h3_edge`	Character; address of unidirectional edge.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, character vector of H3 indexes.

Examples

```
# Get the origin cell of this directed edge
get_udorigin(h3_edge = '166be8d12fffffff')
```

grid_distance — *Grid distance between H3 cells*

Description

This function gets the grid distance between two H3 cell indices.

Usage

```
grid_distance(origin = NULL, destination = NULL, simple = TRUE)
```

Arguments

`origin`	Character vector or list of 15-character indices generated by H3.
`destination`	Character vector or list of 15-character indices generated by H3.
`simple`	Logical; whether to return a vector of outputs or a list object containing both inputs and outputs.

Value

The distance between two H3 cells, expressed as the minimum number of hexagon 'steps' required to get from the origin to the destination. Thus, a neighbour cell is one step away, and two cells with one hexagon between them are two steps apart.

Note

Input H3 indices must be of the same resolution or results cannot be computed. This function may fail to find the distance between two indices if they are very far apart or on opposite sides of a pentagon.

Examples

```
## Not run:
nc <- sf::st_read(system.file("shape/nc.shp", package="sf"), quiet = TRUE)
nc_pts <- sf::st_centroid(nc[c(1, 2), ])
nc_6 <- point_to_cell(nc_pts, res = 6)
# how far apart are these two addresses?
grid_distance(nc_6[1], nc_6[2])

## End(Not run)
```

---

grid_path *Path between H3 cells*

Description

This function returns a path of H3 cells between a start and end cell (inclusive).

Usage

```
grid_path(origin = NULL, destination = NULL, simple = TRUE)
```

Arguments

`origin`	Character vector or list of 15-character indices generated by H3.
`destination`	Character vector or list of 15-character indices generated by H3.
`simple`	Logical; whether to return a vector of outputs or a list object containing both inputs and outputs.

Value

A vector of h3 cells of form c(origin, c(path), destination).

Note

- Input H3 cells must be of the same resolution or results cannot be computed. This function may fail to find the distance between two indexes if they are very far apart or on opposite sides of a pentagon.
- The specific output of this function should not be considered stable across library versions. The only guarantees the library provides are that the line length will be `h3_distance(start, end) + 1` and that every index in the line will be a neighbor of the preceding index.
- Lines are drawn in grid space, and may not correspond exactly to either Cartesian lines or great arcs

Examples

```
## Not run:
nc <- sf::st_read(system.file("shape/nc.shp", package="sf"), quiet = TRUE)
nc_pts <- sf::st_centroid(nc[c(1, 2), ])
nc_6 <- point_to_cell(nc_pts, res = 6)
# find a path between these two addresses:
grid_path(nc_6[1], nc_6[2], simple = TRUE)


## End(Not run)
```

`h3_info_table` *H3 index utility information table*

Description

A dataset containing information about h3 cell indexes at each resolution, calculated using H3's built-in functions.

Usage

```
h3_info_table
```

Format

A data frame with 16 rows and 6 variables:

h3_resolution H3 resolution index number

avg_area_sqm Average area of an H3 cell index at the given resolution, in square meters.

avg_area_sqkm Average area of an H3 cell index at the given resolution, in square kilometers.

avg_edge_m Average edge length of an H3 cell index at the given resolution, in meters.

avg_edge_km Average edge length of an H3 cell index at the given resolution, in kilometers.

avg_cendist_m Average distance between cell centers at the given resolution, in meters.

avg_cendist_km Average distance between cellcenters at the given resolution, in kilometers.

total_unique_indexes Total number of H3 cells at the given resolution.

Source

See also https://h3geo.org/docs/core-library/restable/

`is_pentagon` *check if H3 cell index is a pentagon*

Description

This function checks whether a H3 cell index refers to one of the pentagons that occur at icosahedron corners.

Usage

```
is_pentagon(h3_address = NULL, simple = TRUE)
```

Arguments

`h3_address`	Character; 15-character index generated by H3.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, a logical vector of `length(h3_address)`.

Examples

```
# is the following cell index a pentagon?
is_pentagon(h3_address = '8abe8d12acaffff')
```

`is_rc3` *check if H3 cell index is in a Class III resolution*

Description

This function checks whether a H3 cell index is in a Class III resolution (rotated versus the icosahedron and subject to shape distortion adding extra points on icosahedron edges).

Usage

```
is_rc3(h3_address = NULL, simple = TRUE)
```

Arguments

`h3_address`	Character; 15-character index generated by H3.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, a logical vector of `length(h3_address)`.

Examples

```
# is the following cell index Class III?
is_rc3(h3_address = '8abe8d12acaffff')
```

`is_valid` *check H3 cell index*

Description

This function checks whether an H3 cell index is valid.

Usage

```
is_valid(h3_address = NULL, simple = TRUE)
```

Arguments

`h3_address`	Character; 15-character index generated by H3.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, a logical vector of `length(h3_address)`.

Examples

```
# is the following cell index valid?
is_valid(h3_address = '8abe8d12acaffff')
```

`is_valid_edge` *Check H3 unidirectional edge index*

Description

This function checks whether an H3 unidirectional edge index is valid.

Usage

```
is_valid_edge(h3_edge = NULL, simple = TRUE)
```

Arguments

`h3_edge`	Character; address of unidirectional edge.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, a logical vector of `length(h3_edge)`.

Examples

```
# is the following unidirectional edge index valid?
is_valid_edge(h3_edge = '166be8d12fffffff')
```

`is_valid_vertex` *check H3 cell index*

Description

This function checks whether an H3 cell index is valid.

Usage

```
is_valid_vertex(h3_vertex = NULL, simple = TRUE)
```

Arguments

`h3_vertex`	Character; 15-character index generated by H3.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, a logical vector of `length(h3_vertex)`.

Examples

```
# is the following cell index valid?
is_valid_vertex(h3_vertex = '25abe8d12ac87fff')
```

`num_cells` *Get total H3 cells*

Description

This function returns total number of H3 cells at a given resolution.

Usage

```
num_cells(res = NULL, fast = TRUE)
```

Arguments

`res`	Integer; Desired H3 resolution. See https://h3geo.org/docs/core-library/restable/ for allowable values and related dimensions.
`fast`	Logical; whether to retrieve values from a locally stored table or recalculate from source.

Value

Numeric; H3 cell count.

Note

Above resolution 8 the exact count cannot be represented in a JavaScript 32-bit number, so consumers should use caution when applying further operations to the output.

Examples

```
# Return cell count for resolution 8
num_cells(res = 8)
```

`point_to_cell` *Convert point location to H3 cell index*

Description

This function takes point location data and returns a H3 cell index for each point at the chosen resolution(s).

Usage

```
point_to_cell(input = NULL, res = NULL, simple = TRUE)
```

Arguments

`input`	`sf` object with point geometry, `sfc_POINT` object, `sfg` point, data frame or matrix.
`res`	Integer; Desired H3 resolution. See https://h3geo.org/docs/core-library/restable/ for allowable values and related dimensions.
`simple`	Logical; whether to return outputs as character vector where possible.

Value

- if `simple` = `TRUE` and one resolution is requested, a character vector of H3 addresses.
- if `simple` = `TRUE` and multiple resolutions are requested, a data frame of H3 addresses.
- if `simple` = `FALSE` and a matrix, sfc or sfg object is supplied, a data frame of H3 addresses.
- if `simple` = `FALSE` and a data frame or sf object with other attributes is supplied, a data frame of non-spatial attributes with new columns containing addresses for one or more H3 resolutions.

Note

While multiple resolutions can be requested for multiple points, be aware of the memory demand on large datasets.

Examples

```
# where is the Brisbane Town Hall at resolution 15?
bth <- sf::st_sfc(sf::st_point(c(153.023503, -27.468920)), crs = 4326)
bth_15 <- point_to_cell(bth, res = 15)

# where is it at several resolutions?
bth_many <- point_to_cell(bth, res = seq(10, 15), simple = FALSE)
```

`polygon_to_cells` *Get H3 cell index within a polygon*

Description

This function returns all the H3 cell indexes within the supplied polygon geometry.

Usage

```
polygon_to_cells(geometry = NULL, res = NULL, simple = TRUE)
```

Arguments

`geometry`	sf object of type `POLYGON` or `MULTIPOLYGON`.
`res`	Integer; Desired H3 resolution. See https://h3geo.org/docs/core-library/restable/ for allowable values and related dimensions.
`simple`	Logical; whether to return a vector of outputs or an sf object containing both inputs and outputs.

Value

By default, a list of `length(h3_address)`. Each list element contains a character vector of H3 cell indices belonging to that geometry. A result of NA indicates that no H3 cell indices of the chosen resolution are centered over the geometry.

Note

This function will be slow with a large number of polygons, and/or polygons that are large relative to the hexagon area at the chosen resolution. A message is printed to console where the total input area is (roughly) > 100000x the area of the chosen H3 resolution.

Examples

```
# Which level 5 H3 cell indices have centers inside County Ashe, NC?
nc <- sf::st_read(system.file("shape/nc.shp", package="sf"), quiet = TRUE)
nc1 <- nc[1, ]
fillers <- polygon_to_cells(geometry = nc1, res = 5)
```

`rads_to_degs` *Convert radians to degrees*

Description

Convert radians to degrees.

Usage

```
rads_to_degs(radian = NULL, lang = c("r", "h3"), simple = TRUE)
```

Arguments

`radian`	Numeric, value in radians
`lang`	Character; whether to perform the conversion using base R or the H3 library. Defaults to R for speed.
`simple`	Logical; whether to return a vector or a data frame containing both inputs and outputs.

Value

Numeric, value in degrees

Examples

```
rads_to_degs(1.5)
```

res_area *Get average cell area*

Description

This function returns the average area of an H3 cell at a given resolution.

Usage

```
res_area(res = NULL, units = c("m2", "km2"), fast = TRUE)
```

Arguments

res	Integer; Desired H3 resolution. See https://h3geo.org/docs/core-library/restable/ for allowable values and related dimensions.
units	Areal unit to report in. Options are square meters or square kilometers.
fast	Logical; whether to retrieve values from a locally stored table or reclaculate from source.

Value

Numeric; average H3 cell area.

Examples

```
# Return average H3 cell area at each resolution in square meters
res_area(res = seq(0, 15), units = 'm2')
```

res_cendist *Get average distance between H3 cell centers*

Description

This function returns the average distance between the center of H3 cells at a given resolution.

Usage

```
res_cendist(res = NULL, units = c("m", "km"), fast = TRUE)
```

Arguments

res	Integer; Desired H3 resolution. See https://h3geo.org/docs/core-library/restable/ for allowable values and related dimensions.
units	Length unit to report in, either meters or kilometers.
fast	Logical; whether to retrieve values from a locally stored table or recalculate from source.

Value

Numeric; H3 cell center separation distance.

Note

This isn't in the core library but may be useful.

Examples

```
# Return average H3 cell separation distance at each resolution in kilometers
res_cendist(res = seq(0, 15), units = 'km')
```

`res_length` *Get average cell edge length*

Description

This function returns the average edge length of an H3 cell edge at a given resolution.

Usage

```
res_length(res = NULL, units = c("m", "km"), fast = TRUE)
```

Arguments

`res`	Integer; Desired H3 resolution. See https://h3geo.org/docs/core-library/restable/ for allowable values and related dimensions.
`units`	Length unit to report in. Options are meters or kilometers.
`fast`	Logical; whether to retrieve values from a locally stored table or recalculate from source.

Value

Numeric; H3 cell edge length

Note

This value is also the hexagon circumradius.

Examples

```
# Return average H3 cell edge length at each resolution in kilometers
res_length(res = seq(0, 15), units = 'km')
```

`splitlong_to_cell` *Split long to H3 cell*

Description

Convert a "split long" - a pair of 32-bit integers - into an H3 cell index.

Usage

```
splitlong_to_cell(split_lower = NULL, split_upper = NULL, simple = TRUE)
```

Arguments

`split_lower`	Integer; Lower 32 bits of an H3 index.
`split_upper`	Integer; Upper 32 bits of an H3 index.
`simple`	Logical; whether to return a vector or a data frame containing both inputs and outputs.

Value

Vector of H3 addresses, one for each split long pair supplied.

Examples

```
x <- cell_to_splitlong(h3_address = '8abe8d12acaffff')

splitlong_to_cell(split_lower = x[[1]][1], split_upper = x[[1]][2])
```

`udedge_to_line` *Get the geometry of an H3 edge*

Description

This function takes an H3 unidirectional edge address and returns the coordinates of its geometry in WGS84.

Usage

```
udedge_to_line(h3_edge = NULL, simple = TRUE)
```

Arguments

`h3_edge`	Character; address of unidirectional edge.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, an object of type ‘sfc_LINESTRING‘.

Examples

```
# get me the shape of this edge
udedge_to_line(h3_edge = '166be8d12fffffff')
```

---

uncompact — *Uncompact H3 cell indices*

Description

This function uncompacts a compacted set of H3 cells to indices of the target resolution.

Usage

```
uncompact(h3_addresses = NULL, res = NULL, simple = TRUE)
```

Arguments

`h3_addresses`	Character vector or list of 15-character cell indices generated by H3.
`res`	Integer; Desired H3 resolution. See `https://h3geo.org/docs/core-library/restable/` for allowable values and related dimensions.
`simple`	Logical; whether to return a vector of outputs or a list object containing both inputs and outputs.

Value

A list of H3 cell indices of the chosen resolution.

Examples

```
## Not run:
# Give me a compacted representation of County Ashe, NC
nc <- sf::st_read(system.file("shape/nc.shp", package="sf"), quiet = TRUE)
nc1 <- nc[1, ]
nc1 <- sf::st_cast(nc1, 'POLYGON')
fillers <- polygon_to_cells(geometry = nc1, res = 6)
compacted <- compact(fillers)
# uncompact to resolution 7
uncompacted <- uncompact(compacted, res = 7)

## End(Not run)
```

`vertex_to_point` *Convert H3 cell vertex index to point location*

Description

This function takes a H3 cell vertex index and returns its coordinates in WGS84.

Usage

```
vertex_to_point(h3_vertex = NULL, simple = TRUE)
```

Arguments

`h3_vertex`	Character; vertex address or addresses.
`simple`	Logical; whether to return a vector of outputs or a data frame containing both inputs and outputs.

Value

By default, an `sfc_POINT` object of `length(h3_address)`. EPSG:WGS84.

Examples

```
# Convert this vertex to a point
vertex_to_point('246be8d127ffffff')
```

Index