Package ‘graphhopper’

October 13, 2022

Title An R Interface to the 'GraphHopper' Directions API
Version 0.1.2
Date 2021-02-06
Maintainer Stefan Kuethe <crazycapivara@gmail.com>
Description Provides a quick and easy access to the 'GraphHopper' Directions API.
  'GraphHopper' <https://www.graphhopper.com/> itself is a routing engine based on 'OpenStreetMap' data.
  API responses can be converted to simple feature (sf) objects in a convenient way.
License MIT + file LICENSE
Encoding UTF-8
LazyData true
Imports magrittr, httr, googlePolylines, jsonlite, tibble, dplyr
Suggests sf, geojsonsf, ggplot2, testthat
RoxygenNote 6.1.1
URL https://github.com/crazycapivara/graphhopper-r
BugReports https://github.com/crazycapivara/graphhopper-r/issues
NeedsCompilation no
Author Stefan Kuethe [aut, cre]
Repository CRAN
Date/Publication 2021-02-06 16:50:02 UTC

R topics documented:

gh_as_sf .............................................................. 2
gh_available_spt_columns ........................................ 3
gh_bbox ............................................................... 3
gh_get_info ............................................................ 4
gh_get_isochrone ...................................................... 4
gh_get_route ............................................................ 5
gh_get_routes .......................................................... 6
`gh_as_sf`  

Convert a gh object into an sf object

Description

Convert a gh object into an sf object

Usage

```r
gh_as_sf(data, ...)  
## S3 method for class 'gh_route'
gh_as_sf(data, ..., geom_type = c("linestring", "point"))  
## S3 method for class 'gh_spt'
gh_as_sf(data, ...)  
## S3 method for class 'gh_isochrone'
gh_as_sf(data, ...)
```

Arguments

- `data`  
  A gh_route or gh_spt object.  
- `...`  
  ignored  
- `geom_type`  
  Use geom_type = point to return the points of the route with ids corresponding to the instruction ids.

Examples

```r
if (FALSE) {  
  start_point <- c(52.592204, 13.414307)  
  end_point <- c(52.539614, 13.364868)  
  route_sf <- gh_get_route(list(start_point, end_point)) %>%  
    gh_as_sf()
}
```
**gh_available_spt_columns**

*Get a vector with available columns of the spt endpoint*

---

**Description**

Get a vector with available columns of the spt endpoint

**Usage**

gh_available_spt_columns()

---

**gh_bbox**

*Extract the bounding box from a gh object*

---

**Description**

Extract the bounding box from a gh object

**Usage**

gh_bbox(data)

```r
## S3 method for class 'gh_route'
gh_bbox(data)
```

```r
## S3 method for class 'gh_info'
gh_bbox(data)
```

**Arguments**

```r
data # A gh_route or gh_info object.
```
### gh_get_info

**Get information about the GraphHopper instance**

**Description**

Get information about the GraphHopper instance

**Usage**

gh_get_info()

**Examples**

```r
if (FALSE) {
  info <- gh_get_info()
  message(info$version)
  message(info$data_date)
  print(gh_bbox(info))
}
```

### gh_get_isochrone

**Get isochrones for a given start point**

**Description**

Get isochrones for a given start point

**Usage**

gh_get_isochrone(start_point, time_limit = 180, distance_limit = -1, ...

**Arguments**

- **start_point** The start point as (lat, lon) pair.
- **time_limit** The travel time limit in seconds. Ignored if distance_limit > 0.
- **distance_limit** The distance limit in meters.
- **...** Additional parameters. See https://docs.graphhopper.com/#operation/getIsochrone.
gh_get_route

Examples

```r
if (FALSE) {
  start_point <- c(52.53961, 13.36487)

  isochrone_sf <- gh_get_isochrone(start_point, time_limit = 180) %>%
    gh_as_sf()
}
```

---

gh_get_route  

*Get a route for a given set of points*

Description

Get a route for a given set of points

Usage

```r
gh_get_route(points, ..., response_only = FALSE)
```

Arguments

- **points**  
  A list of 2 or more points as (lat, lon) pairs.

- **...**  
  Optional parameters that are passed to the query.

- **response_only**  
  Whether to return the raw response object instead of just its content.

See Also

[https://docs.graphhopper.com/#tag/Routing-API](https://docs.graphhopper.com/#tag/Routing-API) for optional parameters.

Examples

```r
if (FALSE) {
  start_point <- c(52.592204, 13.414307)
  end_point <- c(52.539614, 13.364868)

  route_sf <- gh_get_route(list(start_point, end_point)) %>%
    gh_as_sf()
}
```
gh_get_routes

Get multiple routes

Description

Internally it just calls gh_get_route several times. See also gh_get_spt.

Usage

gh_get_routes(x, y, ..., callback = NULL)

Arguments

x
A single start point as (lat, lon) pair

y
A matrix or a data frame containing columns with latitudes and longitudes that
are used as endpoints. Needs (lat, lon) order.

... Parameters that are passed to gh_get_route.

callback A callback function that is applied to every calculated route.

Examples

if (FALSE) {
  start_point <- c(52.519772, 13.392334)

  end_points <- rbind(
    c(52.564665, 13.42083),
    c(52.564456, 13.342724),
    c(52.489261, 13.324871),
    c(52.48738, 13.454647)
  )

  time_distance_table <- gh_get_routes(
    start_point, end_points, calc_points = FALSE,
    callback = gh_time_distance
  )

  dplyr::bind_rows() %>%
  do.call(rbind, .)

  routes_sf <- gh_get_routes(start_point, end_points, callback = gh_as_sf) %>%
  do.call(rbind, .)
}
**gh_get_spt**

Get the shortest path tree for a given start point

**Description**

Get the shortest path tree for a given start point

**Usage**

```r
gh_get_spt(start_point, time_limit = 600, distance_limit = -1,
            columns = gh_spt_columns(), reverse_flow = FALSE, profile = "car")
```

**Arguments**

- `start_point`: The start point as (lat, lon) pair.
- `time_limit`: The travel time limit in seconds. Ignored if `distance_limit > 0`.
- `distance_limit`: The distance limit in meters.
- `columns`: The columns to be returned. See `gh_spt_columns` and `gh_available_spt_columns` for available columns.
- `reverse_flow`: Use `reverse_flow = TRUE` to change the flow direction.
- `profile`: The profile for which the spt should be calculated.

**Examples**

```r
if (FALSE) {
  start_point <- c(52.53961, 13.36487)

  columns <- gh_spt_columns(
    prev_longitude = TRUE,
    prev_latitude = TRUE,
    prev_time = TRUE
  )

  points_sf <- gh_get_spt(start_point, time_limit = 180, columns = columns) %>%
    gh_as_sf()
}
```
gh_instructions  

*Extract the instructions from a gh route object*

**Description**

Extract the instructions from a gh route object

**Usage**

```r
gh_instructions(data, instructions_only = FALSE)
```

**Arguments**

- `data`: A `gh_route` object.
- `instructions_only`: Whether to return the instructions without the corresponding points.

**See Also**

`gh_get_route`

---

gh_points  

*Extract the points from a gh route object*

**Description**

Extract the points from a gh route object

**Usage**

```r
gh_points(data)
```

**Arguments**

- `data`: A `gh_route` object.
gh_set_api_url

Set gh API base url

**Description**

Set gh API base url

**Usage**

```r
gh_set_api_url(api_url)
```

**Arguments**

- `api_url` API base url

**Note**

Internally it calls `Sys.setenv` to store the API url in an environment variable called `GH_API_URL`.

**Examples**

```r
gh_set_api_url("http://localhost:8989")
```

---

gh_spt_as_linestrings_sf

*Build lines from a gh spt object*

**Description**

Build lines from a gh spt object

**Usage**

```r
gh_spt_as_linestrings_sf(data)
```

**Arguments**

- `data` A gh_spt object.
Examples

```r
if (FALSE) {
    start_point <- c(52.53961, 13.36487)

    columns <- gh_spt_columns(
        prev_longitude = TRUE,
        prev_latitude = TRUE,
        prev_time = TRUE
    )

    lines_sf <- gh_get_spt(start_point, time_limit = 180, columns = columns) %>%
        gh_spt_as_linestrings_sf()
}
```

---

**gh_spt_columns**  
Select the columns to be returned by a spt request

**Description**
Times are returned in milliseconds and distances in meters.

**Usage**

```r
gh_spt_columns(longitude = TRUE, latitude = TRUE, time = TRUE,
    distance = TRUE, prev_longitude = FALSE, prev_latitude = FALSE,
    prev_time = FALSE, prev_distance = FALSE, node_id = FALSE,
    prev_node_id = FALSE, edge_id = FALSE, prev_edge_id = FALSE)
```

**Arguments**

- `longitude, latitude`  
  The longitude, latitude of the node.
- `time, distance`  
  The travel time, distance to the node.
- `prev_longitude, prev_latitude`  
  The longitude, latitude of the previous node.
- `prev_time, prev_distance`  
  The travel time, distance to the previous node.
- `node_id, prev_node_id`  
  The ID of the node, previous node.
- `edge_id, prev_edge_id`  
  The ID of the edge, previous edge.
**Description**

Extract time and distance from a gh route object

**Usage**

```r
gh_time_distance(data)
```

**Arguments**

- `data` A gh_route object.
Index

gh_as_sf, 2
gh_available_spt_columns, 3, 7
gh_bbox, 3
gh_get_info, 4
gh_get_isochrone, 4
gh_get_route, 5, 6, 8
gh_get_routes, 6
gh_get_spt, 6, 7
gh_instructions, 8
gh_points, 8
gh_set_api_url, 9
gh_spt_as_linestrings_sf, 9
gh_spt_columns, 7, 10
gh_time_distance, 11