# Package ‘hereR’

## November 26, 2020

**Type** Package  
**Title** 'sf'-Based Interface to the 'HERE' REST APIs  
**Version** 0.5.2  
**Maintainer** Merlin Unterfinger <info@munterfinger.ch>  
**BugReports** https://github.com/munterfinger/hereR/issues/  
**Description** Interface to the 'HERE' REST APIs <https://developer.here.com/develop/rest-apis>:  
(1) geocode and autosuggest addresses or reverse geocode POIs using the 'Geocoder' API;  
(2) route directions, travel distance or time matrices and isolines using the 'Routing' API;  
(3) request real-time traffic flow and incident information from the 'Traffic' API;  
(4) find request public transport connections and nearby stations from the 'Public Transit' API;  
(5) request intermodal routes using the 'Intermodal Routing' API;  
(6) get weather forecasts, reports on current weather conditions, astronomical information and alerts at a specific location from the 'Destination Weather' API. Locations, routes and isolines are returned as 'sf' objects.  

**Depends** R (>= 3.3.0)  
**Imports** curl (>= 4.3), data.table (>= 1.12.6), flexpolyline (>= 0.1.1), jsonlite (>= 1.7.0), sf (>= 0.9-0), stringr (>= 1.4.0)  
**Suggests** covr (>= 3.5.0), ggplot2 (>= 3.3.2), htmlwidgets (>= 1.5.1), knitr (>= 1.29), leafpop (>= 0.0.5), lwgeom (>= 0.2-5), mapview (>= 2.9.0), rmarkdown (>= 2.3), testthat (>= 2.3.2)  
**License** GPL-3  
**Encoding** UTF-8  
**LazyData** true  
**RoxygenNote** 7.1.1  
**VignetteBuilder** knitr  
**NeedsCompilation** no  
**Author** Merlin Unterfinger [aut, cre] (<https://orcid.org/0000-0003-2020-2366>), Daniel Possenriede [ctb] (<https://orcid.org/0000-0002-6738-9845>)
Example Areas of Interest

Some example Areas of Interest (AOIs): The boundary polygons of Switzerland and Liechtenstein.

Usage

data(aoi)

Format

An object of class "sf", "data.frame".

Source

Made with Natural Earth. Free vector and raster map data @naturalearthdata.com

Examples

data(aoi)
Description

Completes addresses using the HERE 'Geocoder Autosuggest' API.

Usage

    autosuggest(address, results = 5, url_only = FALSE)

Arguments

    address      character, address text to propose suggestions.
    results      numeric, maximum number of suggestions (Valid range: 1 and 100).
    url_only     boolean, only return the generated URLs (default = FALSE)

Value

    A data.frame object, containing the suggestions for the input addresses.

References

    HERE Geocoder API: Autosuggest

Examples

    # Provide an API Key for a HERE project
    set_key("<YOUR API KEY>")

    suggestions <- autosuggest(address = poi$city, url_only = TRUE)

connection

Description

Route public transport connections with geometries (LINESTRING) between pairs of points using the HERE 'Public Transit' API. Two modes are provided:

    • summary = FALSE: The public transport connections are returned as multiple sections with the same vehicle and transport mode. Each section has a detailed route geometry.
    • summary = TRUE: A summary of the connections is retrieved, where each connection is represented as one row with a unified and simplified geometry.
connection(  
  origin,  
  destination,  
  datetime = Sys.time(),  
  arrival = FALSE,  
  results = 3,  
  transfers = -1,  
  summary = FALSE,  
  url_only = FALSE  
)

Arguments

origin sf object, the origin locations of geometry type POINT.
destination sf object, the destination locations of geometry type POINT.
datetime POSIXct object, datetime for the departure (or arrival if arrival = TRUE).
arrival boolean, calculate connections for arrival at the defined time (default = FALSE)?
results numeric, maximum number of suggested public transport routes (Valid range: 1 and 6).
transfers numeric, maximum number of transfers allowed per route (Valid range: -1 and 6, default = -1).
summary boolean, return a summary of the public transport connections instead of the sections of the routes (default = FALSE)?
url_only boolean, only return the generated URLs (default = FALSE)?

Value

An sf object containing the requested routes.

References

HERE Public Transit API: Transit Route

Examples

# Provide an API Key for a HERE project
set_key("<YOUR API KEY>")

# Connection sections
sections <- connection(  
  origin = poi[3:4, ], destination = poi[5:6, ],  
  summary = FALSE, url_only = TRUE  
)

# Connection summary
summary <- connection(  
  origin = poi[3:4, ], destination = poi[5:6, ],
Description

Real-time traffic flow from the HERE 'Traffic' API in areas of interest (AOIs). The traffic flow data contains speed ("SP") and congestion (jam factor: "JF") information, which corresponds to the status of the traffic at the time of the query.

Usage

flow(aoi, min_jam_factor = 0, url_only = FALSE)

Arguments

- **aoi**: sf object, Areas of Interest (POIs) of geometry type POLYGON.
- **min_jam_factor**: numeric, only retrieve flow information with a jam factor greater than the value provided (default = 0).
- **url_only**: boolean, only return the generated URLs (default = FALSE)?

Value

An sf object containing the requested traffic flow information.

Note

The maximum width and height of the bounding box of the input AOIs is 10 degrees. This means that each polygon (= one row) in the AOI sf object should fit in a 10 x 10 degree bbox.

Explanation of the traffic flow variables:

- "PC": Point TMC location code.
- "DE": Text description of the road.
- "QD": Queuing direction. '+' or '-'. Note this is the opposite of the travel direction in the fully qualified ID. For example for location 107+03021 the QD would be '-'.
- "LE": Length of the stretch of road.
- "TY": Type information for the given Location Referencing container. This may be a freely defined string.
- "SP": Speed (based on UNITS) capped by speed limit.
- "FF": The free flow speed on this stretch of the road.
- "JF": The number between 0.0 and 10.0 indicating the expected quality of travel. When there is a road closure, the Jam Factor will be 10. As the number approaches 10.0 the quality of travel is getting worse. -1.0 indicates that a Jam Factor could not be calculated.
- "CN": Confidence, an indication of how the speed was determined. -1.0 road closed. 1.0=100%.
References

- HERE Traffic API: Flow
- Flow explanation, stackoverflow

Examples

```r
# Provide an API Key for a HERE project
generate_key("<YOUR API KEY>")

# Real-time traffic flow
flow <- flow(
  aoi = aoi[aoi$code == "LI", ],
  url_only = TRUE
)
```

---

**geocode**

**HERE Geocoder API: Geocode**

Description

Geocodes addresses using the HERE 'Geocoder' API.

Usage

```r
geocode(address, alternatives = FALSE, sf = TRUE, url_only = FALSE, addresses)
```

Arguments

- address: character, addresses to geocode.
- alternatives: boolean, return also alternative results (default = FALSE)?
- sf: boolean, return an sf object (default = TRUE) or a data.frame?
- url_only: boolean, only return the generated URLs (default = FALSE)?
- addresses: character, addresses to geocode (deprecated).

Value

If `sf = TRUE`, an sf object, containing the position coordinates geocoded addresses as geometry list column and the access coordinates as well-known text (WKT). If `sf = FALSE`, a data.frame containing the coordinates of the geocoded addresses as `lng`, `lat` columns.

According to the Geocoding and Search API Reference, the access coordinates are "[c]oordinates of the place you are navigating to (for example, driving or walking). This is a point on a road or in a parking lot." The position coordinates are "[t]he coordinates (latitude, longitude) of a pin on a map corresponding to the searched place."

References

HERE Geocoder API: Geocode
### incident

**Examples**

```r
# Provide an API Key for a HERE project
set_key("<YOUR API KEY>")

locs <- geocode(address = poi$city, url_only = TRUE)
```

---

**incident**  

**HERE Traffic API: Incidents**

**Description**

Traffic incident information from the HERE 'Traffic' API in areas of interest (AOIs). The incidents contain information about location, duration, severity, type, description and further details.

**Usage**

```r
incident(aoi, from = Sys.time() - 60 * 60 * 24 * 7, url_only = FALSE)
```

**Arguments**

- `aoi`  
  sf object, Areas of Interest (POIs) of geometry type POLYGON.
- `from`  
  POSIXct object, datetime of the earliest traffic incidents (default = FALSE).
- `url_only`  
  boolean, only return the generated URLs (default = FALSE)?

**Value**

An sf object containing the traffic incidents.

**Note**

The maximum width and height of the bounding box of the input AOIs is 10 degrees. This means that each polygon (= one row) in the AOI sf object should fit in a 10 x 10 degree bbox.

**References**

HERE Traffic API: Incidents

**Examples**

```r
# Provide an API Key for a HERE project
set_key("<YOUR API KEY>")

# All traffic incidents from the beginning of 2018
incidents <- incident(
  aoi = aoi,
  from = as.POSIXct("2018-01-01 00:00:00"),
  url_only = TRUE
)
```
**intermodal_route**  
**HERE Intermodal Routing API: Calculate Route**

**Description**
Calculates route geometries (LINESTRING) between given pairs of points using the HERE 'Intermodal Routing' API.

**Usage**

```r
intermodal_route(
  origin, 
  destination, 
  datetime = Sys.time(), 
  results = 3, 
  transfers = -1, 
  url_only = FALSE
)
```

**Arguments**
- **origin** sf object, the origin locations of geometry type POINT.
- **destination** sf object, the destination locations of geometry type POINT.
- **datetime** POSIXct object, datetime for the departure (default = Sys.time()).
- **results** numeric, maximum number of suggested route alternatives (Valid range: 1 and 7, default = 3).
- **transfers** numeric, maximum number of transfers allowed per route (Valid range: -1 and 6, default = -1).
- **url_only** boolean, only return the generated URLs (default = FALSE).

**Value**
An sf object containing the requested intermodal routes.

**References**
HERE Intermodal Routing API: Routes

**Examples**

```r
# Provide an API Key for a HERE project
set_key("<YOUR API KEY>")

# Intermodal routing
routes <- intermodal_route(
  origin = poi[1:3, ],
  destination = poi[4:6, ],
```
**Description**

Calculates isolines (POLYGON or MULTIPOLYGON) using the HERE 'Routing' API that connect the end points of all routes leaving from defined centers (POIs) with either a specified length, a specified travel time or consumption.

**Usage**

```r
isoline(
  poi,
  datetime = Sys.time(),
  arrival = FALSE,
  range = seq(5, 30, 5) * 60,
  range_type = "time",
  type = "fastest",
  mode = "car",
  traffic = FALSE,
  aggregate = TRUE,
  url_only = FALSE
)
```

**Arguments**

- **poi**: sf object, Points of Interest (POIs) of geometry type POINT.
- **datetime**: POSIXct object, datetime for the departure (or arrival if `arrival = TRUE`).
- **arrival**: boolean, are the provided Points of Interest (POIs) the origin or destination locations (default = FALSE)?
- **range**: numeric, a vector of type integer containing the breaks for the generation of the isolines: (1) time in seconds; (2) distance in meters; (3) consumption in costfactor.
- **range_type**: character, unit of the isolines: "distance", "time" or "consumption".
- **type**: character, set the routing type: "fastest" or "shortest".
- **mode**: character, set the transport mode: "car", "pedestrian" or "truck".
- **traffic**: boolean, use real-time traffic or prediction in routing (default = FALSE)? If no datetime is set, the current timestamp at the moment of the request is used for datetime.
- **aggregate**: boolean, aggregate (with function `min`) and intersect the isolines from geometry type POLYGON to geometry type MULTIPOLYGON (default = TRUE)?
- **url_only**: boolean, only return the generated URLs (default = FALSE)?
Value

An sf object containing the requested isolines.

References

HERE Routing API: Calculate Isoline

Examples

# Provide an API Key for a HERE project
set_key("<YOUR API KEY>")

# Isochrone for 5, 10, 15, 20, 25 and 30 minutes driving time
isolines <- isoline(
  poi = poi,
  range = seq(5, 30, 5) * 60,
  url_only = TRUE
)

---

<table>
<thead>
<tr>
<th>poi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example Points of Interest</td>
</tr>
</tbody>
</table>

Description

Some example Points of Interest (POIs): Cities in Switzerland and Liechtenstein with more than 100'000 inhabitants.

Usage

data(poi)

Format

An object of class "sf", "data.frame".

Source

Made with Natural Earth. Free vector and raster map data @naturaleartheadata.com

Examples

data(poi)
Description

Get addresses from locations using the HERE 'Geocoder' API. The return value is an sf object, containing point geometries with suggestions for addresses near the provided POIs.

Usage

reverse_geocode(poi, results = 1, sf = TRUE, url_only = FALSE)

Arguments

- poi sf object, Points of Interest (POIs) of geometry type POINT.
- results numeric, maximum number of results (Valid range: 1 and 100).
- sf boolean, return an sf object (default = TRUE) or a data.frame?
- url_only boolean, only return the generated URLs (default = FALSE)?

Value

If sf = TRUE, an sf object, containing the position coordinates of the reverse geocoded POIs as geometry list column and the access coordinates as well-known text (WKT). If sf = FALSE, a data.frame containing the coordinates of the reverse geocoded POIs as lng, lat columns.

Note

If no addresses are found near a POI, NULL for this POI is returned. In this case the rows corresponding to this particular POI are missing and merging the POIs by row is not possible. However, in the returned sf object, the column “id” matches the rows of the input POIs. The “id” column can be used to join the original POIs.

References

HERE Geocoder API: Reverse Geocode

Examples

# Provide an API Key for a HERE project
set_key("<YOUR API KEY>")

# Get addresses
addresses <- reverse_geocode(poi = poi, results = 3, url_only = TRUE)
route HERE Routing API: Calculate Route

Description

Calculates route geometries (LINESTRING) between given pairs of points using the HERE 'Routing' API. Routes can be created for various transport modes, as for example 'car' or 'public transport', incorporating current traffic information, if available. For routes using the transport mode "car" a vehicle type can be specified, to obtain an estimate of the consumption.

Usage

```r
route(
  origin,
  destination,
  datetime = Sys.time(),
  arrival = FALSE,
  type = "fastest",
  mode = "car",
  traffic = FALSE,
  vehicle_type = "diesel,5.5",
  url_only = FALSE
)
```

Arguments

- **origin**: sf object, the origin locations of geometry type POINT.
- **destination**: sf object, the destination locations of geometry type POINT.
- **datetime**: POSIXct object, datetime for the departure (or arrival if arrival = TRUE).
- **arrival**: boolean, calculate routes for arrival at the defined time (default = FALSE)?
- **type**: character, set the routing type: "fastest", "shortest" or "balanced".
- **mode**: character, set the transport mode: "car", "pedestrian", "carHOV", "publicTransport", "truck" or "bicycle".
- **traffic**: boolean, use real-time traffic or prediction in routing (default = FALSE)? If no datetime is set, the current timestamp at the moment of the request is used for datetime.
- **vehicle_type**: character, specify the motor type of the vehicle: "diesel", "gasoline" or "electric". And set the consumption per 100km in liters (default = "diesel,5.5").
- **url_only**: boolean, only return the generated URLs (default = FALSE)?

Value

An sf object containing the requested routes.
Note

The public transport routes (mode = "publicTransport") provided by route are not considering
the time tables of the public transport providers. Use connection for public transport routes that
consider time tables.

References

HERE Routing API: Calculate Route

Examples

# Provide an API Key for a HERE project
set_key("<YOUR API KEY>")

# Get all from - to combinations from POIs
to <- poi[rep(seq_len(nrow(poi)), nrow(poi)), ]
from <- poi[rep(seq_len(nrow(poi)), each = nrow(poi)),]
idx <- apply(to != from, any, MARGIN = 1)
to <- to[idx, ]
from <- from[idx, ]

# Routing
routes <- route(
  origin = from, destination = to,
  mode = "car", type = "fastest", traffic = TRUE,
  vehicle_type = "diesel,5.5",
  url_only = TRUE
)

Description

Calculates a matrix of M:N, M:1 or 1:N route summaries between given points of interest (POIs)
using the HERE 'Routing' API. Various transport modes and traffic information at a provided times-
tamp are supported. The requested matrix is split into (sub-)matrices of dimension 15x100 to use the
maximum matrix size per request and thereby minimize the number of overall needed requests. The
result is one route summary matrix, that fits the order of the provided POIs: origIndex, destIndex.

Usage

route_matrix(
  origin,
  destination = origin,
  datetime = Sys.time(),
  type = "fastest",
  mode = "car",
  traffic = TRUE
)
route_matrix

```r
traffic = FALSE,
search_range = 99999999,
attribute = c("distance", "traveltime"),
url_only = FALSE
)
```

**Arguments**

- **origin** sf object, the origin locations (M) of geometry type POINT.
- **destination** sf object, the destination locations (N) of geometry type POINT.
- **datetime** POSIXct object, datetime for the departure.
- **type** character, set the routing type: "fastest", "shortest" or "balanced".
- **mode** character, set the transport mode: "car", "pedestrian", "carHOV" or "truck".
- **traffic** boolean, use real-time traffic or prediction in routing (default = FALSE)? If no datetime is set, the current timestamp at the moment of the request is used for datetime.
- **search_range** numeric, value in meters to limit the search radius in the route generation (default = 99999999).
- **attribute** character, attributes to be calculated on the routes: "distance" or "traveltime" (default = c("distance", "traveltime").
- **url_only** boolean, only return the generated URLs (default = FALSE)?

**Value**

A data.frame, which is an edge list containing the requested M:N route combinations.

**References**

HERE Routing API: Calculate Matrix

**Examples**

```r
# Provide an API Key for a HERE project
set_key("<YOUR API KEY>")

# Create routes summaries between all POIs
mat <- route_matrix(
  origin = poi,
  traffic = TRUE,
  url_only = TRUE
)
```
set_key

**Set HERE Application Credentials**

**Description**

Provide an API Key for a HERE project of type 'REST'. The key is set for the current R session and is used to authenticate in the requests to the APIs.

**Usage**

```r
set_key(api_key)
```

**Arguments**

- `api_key` character, the API key from a HERE project.

**Details**

No login yet? Get a free login and key here: [klick](#)

**Value**

None.

**Examples**

```r
set_key("<YOUR API KEY>")
```

---

setVerbose

**Verbose API usage of hereR**

**Description**

If set to TRUE the hereR package is messaging information about the amount of requests sent to the APIs and data size received.

**Usage**

```r
setVerbose(ans = FALSE)
```

**Arguments**

- `ans` boolean, verbose or not (default = FALSE)?

**Value**

None.
Examples

```r
setVerbose(TRUE)
```

---

**station**  
*HERE Public Transit API: Find Stations Nearby*

**Description**

Retrieve stations with the corresponding line information around given locations using the HERE 'Public Transit' API.

**Usage**

```r
station(poi, radius = 500, results = 50, url_only = FALSE)
```

**Arguments**

- `poi` sf object, Points of Interest (POIs) of geometry type `POINT`.
- `radius` numeric, the search radius in meters (default: 500).
- `results` numeric, maximum number of suggested public transport stations (Valid range: 1 and 50, default: 50).
- `url_only` boolean, only return the generated URLs (default: FALSE)?

**Value**

An sf object containing the requested stations with the corresponding line information.

**References**

*HERE Public Transit API: Station Search*

**Examples**

```r
# Provide an API Key for a HERE project
set_key("<YOUR API KEY>")

# Stations
stations <- station(poi = poi, url_only = TRUE)
```
**unset_key**  
Remove HERE Application Credentials

**Description**
Remove previously set HERE API key from the current R session.

**Usage**

```r
unset_key()
```

**Value**
None.

**Examples**

```r
unset_key()
```

**unset_proxy**

# @rdname hereR-defunct

**Description**

# @rdname hereR-defunct

**Usage**

```r
unset_proxy()
```

**unset_proxy**

For `unset_proxy`, configure a global proxy for R in ‘~/.Renviron’ instead.
weather

**HERE Destination Weather API: Observations, Forecasts, Astronomy and Alerts**

**Description**

Weather forecasts, reports on current weather conditions, astronomical information and alerts at a specific location (coordinates or location name) based on the HERE 'Destination Weather' API. The information comes from the nearest available weather station and is not interpolated.

**Usage**

weather(poi, product = "observation", url_only = FALSE)

**Arguments**

- **poi** sf object or character, Points of Interest (POIs) of geometry type POINT or location names (e.g. cities or regions).
- **product** character, weather product of the 'Destination Weather API'. Supported products: "observation", "forecast_hourly", "forecast_astronomy" and "alerts".
- **url_only** boolean, only return the generated URLs (default = FALSE)?

**Value**

An sf object containing the requested weather information at the nearest weather station. The point geometry in the sf object is the location of the weather station.

**References**

HERE Destination Weather API: Observation

**Examples**

# Provide an API Key for a HERE project
set_key("<YOUR API KEY>")

# Observation
observation <- weather(poi = poi, product = "observation", url_only = TRUE)

# Forecast
forecast <- weather(poi = poi, product = "forecast_hourly", url_only = TRUE)

# Astronomy
astronomy <- weather(poi = poi, product = "forecast_astronomy", url_only = TRUE)

# Alerts
alerts <- weather(poi = poi, product = "alerts", url_only = TRUE)
Index

* datasets
  aoi, 2
  poi, 10

aoi, 2
autosuggest, 3
connection, 3, 13
flow, 5
geocode, 6
incident, 7
intermodal_route, 8
isoline, 9

poi, 10

reverse_geocode, 11
route, 12, 13
route_matrix, 13

set_key, 15
set_verbose, 15
station, 16

unset_key, 17
unset_proxy, 17

weather, 18