Package 'hereR'

January 29, 2020

Type Package

Title 'sf'-Based Interface to the 'HERE' REST APIs

Version 0.3.1

Maintainer Merlin Unterfinger <info@munterfinger.ch>

URL https://munterfinger.github.io/hereR/,
https://github.com/munterfinger/hereR/

BugReports https://github.com/munterfinger/hereR/issues/

Description Interface to the 'HERE' REST APIs <https://developer.here.com/develop/rest-apis>:
(1) geocode and autocomplete 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) 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 (>= 4.2), data.table (>= 1.12.6), jsonlite (>= 1.6),
lwgeom (>= 0.1-7), sf (>= 0.8-0), stringr (>= 1.4.0)

Suggests ggplot2 (>= 3.2.1), leafpop (>= 0.0.1), mapview (>= 2.7.0),
testthat (>= 2.2.1), knitr (>= 1.25), rmarkdown (>= 1.16), covr
(>= 3.3.2)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

VignetteBuilder knitr

NeedsCompilation no

Author Merlin Unterfinger [aut, cre]

Repository CRAN

Date/Publication 2020-01-29 16:40:02 UTC

R topics documented:

`hereR-package` *'sf'-Based Interface to the 'HERE' REST APIs*

Description

The hereR package provides an interface to the 'HERE' REST APIs:

- Geocode and autocomplete addresses or reverse geocode POIs using the 'Geocoder' API;
- Route directions, travel distance or time matrices and isolines using the 'Routing' API;
- Request real-time traffic flow and incident information from the 'Traffic' API;
- Find request public transport connections and nearby stations from the 'Public Transit' API;
- 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.

Application credentials

This package requires an API key for a HERE project. The key is set for the current R session and is used to authenticate in the requests to the APIs. A free login and project can be created on https://developer.here.com/. In order to obtain the API key navigate to a project of your choice in the developer portal, select 'REST: Generate APP' and then 'Create API Key'. To set the API key, please use: `set_key("<YOUR API KEY>")`

Functions to access the APIs

- `autocomplete` - Get suggestions for address strings.
- `geocode` - Get coordinates from addresses.
- `reverse_geocode` - Get addresses or landmarks from locations.
- `route` - Find the fastest routes between places.
- `route_matrix` - Request a matrix of route summaries between places.
- `isoline` - Create isochrone, isodistance or isoconsumption lines around places.
- `traffic` - Get information about traffic jam and incidents in areas.
- `connection` - Request public transport connections between places.
- `station` - Find stations nearby places.
- `weather` - Get weather observations, forecasts and alerts at places.

Author(s)

Merlin Unterfinger - <info@munterfinger.ch>

See Also

- https://github.com/munterfinger/hereR/
- https://munterfinger.github.io/hereR/
- https://developer.here.com/develop/rest-apis/

---

aoi *Example Areas of Interest*

Description

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)
```

autocomplete *HERE Geocoder API: Autocomplete*

Description

Completes addresses using the HERE 'Geocoder Autocomplete' API.

Usage

```
autocomplete(addresses, results = 5, url_only = FALSE)
```

Arguments

addresses	character, addresses to autocomplete.
results	numeric, maximum number of suggestions (Valid range: 1 and 20).
url_only	boolean, only return the generated URLs (default = FALSE)?

Value

A data.table object, containing the autocomplete suggestions for the addresses.

References

HERE Geocoder API: Autocomplete

Examples

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

suggestions <- autocomplete(addresses = poi$city, url_only = TRUE)
```

connection *HERE Public Transit API: Transit Route*

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 mulitple 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.

Usage

```
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.

Note

As it is not possible to match the "maneuvers" to the "connections-sections" in the API response using the section id (sec_id), the returned geometries of walking sections are straight lines between the station (or origin and destination) points instead of routed lines on the pedestrian network. The walking segments can be routed in hindsight using the route function with mode set to "pedestrian".

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, ],
  summary = TRUE, url_only = TRUE
)
```

`flow` *HERE Traffic API: Flow*

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

```
# Provide an API Key for a HERE project
set_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

```
geocode(addresses, autocomplete = FALSE, url_only = FALSE)
```

Arguments

addresses	character, addresses to geocode.
autocomplete	boolean, use the 'Geocoder Autocomplete' API to autocomplete addresses? Note: This options doubles the amount of requests (default = FALSE).
url_only	boolean, only return the generated URLs (default = FALSE)?

Value

An sf object, containing the coordinates of the geocoded addresses.

References

HERE Geocoder API: Geocode

Examples

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

locs <- geocode(addresses = 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

```
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

```
# 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
)
```

`isoline` *HERE Routing API: Calculate Isoline*

Description

Calcuates 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

```
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
)
```

---

poi *Example Points of Interest*

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 @naturalearthdata.com

Examples

```
data(poi)
```

`reverse_geocode` *HERE Geocoder API: Reverse Geocode*

Description

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

Usage

```
reverse_geocode(poi, results = 1, landmarks = FALSE, 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 20).
`landmarks`	boolean, retrieve landmarks instead of addresses (`default = FALSE`)?.
`url_only`	boolean, only return the generated URLs (`default = FALSE`)?

Value

An `sf` object, containing the suggested addresses or landmark names of the reverse geocoded POIs.

Note

If no addresses or landmarks 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: Geocode

Examples

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

# Get addresses
addresses <- reverse_geocode(poi = poi, results = 3, landmarks = FALSE, url_only = TRUE)

# Get landmarks
landmarks <- reverse_geocode(poi = poi, results = 3, landmarks = TRUE, 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

```
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 im 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
)
```

---

route_matrix *HERE Routing API: Calculate Matrix*

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 timestamp 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 = 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

```
# 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

```
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

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

set_proxy *Proxy Configuration*

Description

If a proxy is needed, for example because the computer is behind a corporate proxy, it can be set as follows: `proxy = "http://your-proxy.net:port/"` or `"https://your-proxy.net:port/"` and `"proxyuserpwd" = "user:pwd"`.

Usage

```
set_proxy(proxy, proxyuserpwd)
```

Arguments

proxy	character, the URL of the proxy (`"https://your-proxy.net:port/"`).
proxyuserpwd	character, user and password for the authentication (`"user:pwd"`).

Value

None.

Examples

```
set_proxy(
  proxy = "https://your-proxy.net:port/",
  proxyuserpwd = "user:pwd"
)
```

`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

```
station(poi, radius = 500, results = 5, 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 = 5`).
`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: Find Stations Nearby

Examples

```
# 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

```
unset_key()
```

Value

None.

Examples

```
unset_key()
```

unset_proxy *Remove Proxy Configuration*

Description

Remove a previously set proxy configuration from the current R session.

Usage

```
unset_proxy()
```

Value

None.

Examples

```
unset_proxy()
```

`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