

# Package ‘hereR’

October 4, 2021

**Type** Package

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

**Version** 0.8.1

**Maintainer** Merlin Unterfinger <info@munterfing.ch>

**URL** <https://munterfi.github.io/hereR/>,  
<https://github.com/munterfi/hereR/>

**BugReports** <https://github.com/munterfi/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', 'Matrix Routing' and 'Isoline Routing' APIs;
  - (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** crul (>= 1.1.0), curl (>= 4.3), data.table (>= 1.13.0),  
flexpolyline (>= 0.2.0), 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** 7.1.2

**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>>)

**Repository** CRAN

**Date/Publication** 2021-10-04 09:30:02 UTC

## R topics documented:

aoi . . . . .	2
autosuggest . . . . .	3
connection . . . . .	4
flow . . . . .	5
geocode . . . . .	6
incident . . . . .	7
intermodal_route . . . . .	8
isoline . . . . .	9
poi . . . . .	11
reverse_geocode . . . . .	11
route . . . . .	12
route_matrix . . . . .	14
set_freemium . . . . .	15
set_key . . . . .	16
set_verbose . . . . .	16
station . . . . .	17
unset_key . . . . .	18
weather . . . . .	18
<b>Index</b>	<b>20</b>

---

aoi	<i>Example Areas of Interest</i>
-----	----------------------------------

---

### 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](https://www.naturalearthdata.com)

## Examples

```
data(aoi)
```

---

autosuggest

*HERE Geocoding & Search API: Autosuggest*

---

## 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

---

*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 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.

### Usage

```
connection(
  origin,
  destination,
  datetime = Sys.time(),
  arrival = FALSE,
  results = 3,
  transfers = -1,
  transport_mode = NULL,
  summary = FALSE,
  url_only = FALSE
)
```

### Arguments

<code>origin</code>	sf object, the origin locations of geometry type POINT.
<code>destination</code>	sf object, the destination locations of geometry type POINT.
<code>datetime</code>	POSIXct object, datetime for the departure (or arrival if <code>arrival = TRUE</code> ).
<code>arrival</code>	boolean, calculate connections for arrival at the defined time (default = FALSE)?
<code>results</code>	numeric, maximum number of suggested public transport routes (Valid range: 1 and 6).
<code>transfers</code>	numeric, maximum number of transfers allowed per route (Valid range: -1 and 6, whereby the default = -1 allows for unlimited transfers).
<code>transport_mode</code>	character, enable or disable ("-" prefix) transport modes. Note: Do not enable and disable modes at the same time (default = NULL).
<code>summary</code>	boolean, return a summary of the public transport connections instead of the sections of the routes (default = FALSE)?
<code>url_only</code>	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, ],
  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 Geocoding & Search API: Geocode*


---

**Description**

Geocodes addresses using the HERE 'Geocoding & Search API' API.

**Usage**

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

**Arguments**

address	character, addresses to geocode or a list containing qualified queries with the keys "country", "state", "county", "city", "district", "street", "houseNumber" or "postalCode".
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)?

**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 Geocoding & Search API: Geocode](#)

**Examples**

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

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

---

incident	<i>HERE Traffic API: Incidents</i>
----------	------------------------------------

---

**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
)
```

---

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**

```
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**

```
# 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, ],
  url_only = TRUE
)
```

---

isoline

---

*HERE Isoline Routing API: Calculate Isoline*


---

**Description**

Calculates isolines (POLYGON or MULTIPOLYGON) using the HERE 'Isoline 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 (only the default E-car available).

**Usage**

```
isoline(
  poi,
  datetime = Sys.time(),
  arrival = FALSE,
  range = seq(5, 30, 5) * 60,
```

```

range_type = "time",
routing_mode = "fast",
transport_mode = "car",
traffic = TRUE,
optimize = "balanced",
consumption_model = NULL,
aggregate = TRUE,
url_only = FALSE
)

```

### Arguments

<code>poi</code>	sf object, Points of Interest (POIs) of geometry type POINT.
<code>datetime</code>	POSIXct object, datetime for the departure (or arrival if <code>arrival = TRUE</code> ).
<code>arrival</code>	boolean, are the provided Points of Interest (POIs) the origin or destination locations (default = FALSE)?
<code>range</code>	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 Wh.
<code>range_type</code>	character, unit of the isolines: "distance", "time" or "consumption".
<code>routing_mode</code>	character, set the routing mode: "fast" or "short".
<code>transport_mode</code>	character, set the transport mode: "car", "pedestrian" or "truck".
<code>traffic</code>	boolean, use real-time traffic or prediction in routing (default = TRUE)? If no traffic is selected, the datetime is set to "any" and the request is processed independently from time.
<code>optimize,</code>	character, specifies how isoline calculation is optimized: "balanced", "quality" or "performance" (default = "balanced").
<code>consumption_model</code>	character, specify the consumption model of the vehicle, see <a href="#">consumption model</a> for more information (default = NULL a average electric car is set).
<code>aggregate</code>	boolean, aggregate (with function min) and intersect the isolines from geometry type POLYGON to geometry type MULTIPOLYGON (default = TRUE)?
<code>url_only</code>	boolean, only return the generated URLs (default = FALSE)?

### Value

An sf object containing the requested isolines.

### References

[HERE Isoline Routing API](#)

### 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](https://www.naturalearthdata.com)

### Examples

```
data(poi)
```

---

reverse\_geocode

*HERE Geocoding & Search API: Reverse Geocode*

---

### 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**

<code>poi</code>	<code>sf</code> object, Points of Interest (POIs) of geometry type POINT.
<code>results</code>	numeric, maximum number of results (Valid range: 1 and 100).
<code>sf</code>	boolean, return an <code>sf</code> object (default = TRUE) or a <code>data.frame</code> ?
<code>url_only</code>	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 'bicycle', incorporating current traffic information, if available. For routes using the transport mode "car" a vehicle consumption model can be specified, to obtain an estimate of the consumption.

**Usage**

```
route(
  origin,
  destination,
  datetime = Sys.time(),
  arrival = FALSE,
  results = 1,
  routing_mode = "fast",
  transport_mode = "car",
  traffic = TRUE,
  avoid_area = NULL,
  avoid_feature = NULL,
  consumption_model = NULL,
  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)?
results	numeric, maximum number of suggested routes (Valid range: 1 and 7).
routing_mode	character, set the routing type: "fast" or "short" (default = "fast").
transport_mode	character, set the transport mode: "car", "truck", "pedestrian", "bicycle" or scooter (default = "car").
traffic	boolean, use real-time traffic or prediction in routing (default = TRUE)? If no traffic is selected, the datetime is set to "any" and the request is processed independently from time.
avoid_area,	sf object, area (only bounding box is taken) to avoid in routes (default = NULL).
avoid_feature	character, transport network features to avoid, e.g. "tollRoad" or "ferry" (default = NULL).
consumption_model	character, specify the consumption model of the vehicle, see <a href="#">consumption model</a> for more information (default = NULL a average electric car is set).
url_only	boolean, only return the generated URLs (default = FALSE)?

**Value**

An sf object containing the requested routes.

**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, results = 3,
  transport_mode = "car", url_only = TRUE
)
```

---

route\_matrix

*HERE Matrix 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 'Matrix 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: orig\_id, dest\_id.

**Usage**

```
route_matrix(
  origin,
  destination = origin,
  datetime = Sys.time(),
  routing_mode = "fast",
  transport_mode = "car",
  traffic = TRUE,
  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.
routing_mode	character, set the routing type: "fast" or "short" (default = "fast").

transport_mode	character, set the transport mode: "car", "truck", "pedestrian" or "bicycle" (default = "car").
traffic	boolean, use real-time traffic or prediction in routing (default = TRUE)? If no traffic is selected, the datetime is set to "any" and the request is processed independently from time.
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 Matrix Routing API](#)

**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,
  url_only = TRUE
)
```

---

set_freemium	<i>Set whether plan is freemium or not</i>
--------------	--

---

**Description**

If set to TRUE the hereR package limits the requests per second (RPS) sent to the APIs and routing matrices will be chopped up into submatrices of size 15x100. This option is necessary for freemium licenses to avoid hitting the rate limit of the APIs with status code 429. Deactivate this option to increase speed of requests for paid plans.

**Usage**

```
set_freemium(ans = TRUE)
```

**Arguments**

ans                   boolean, use limits or not (default = TRUE)?

**Value**

None.

**Examples**

```
set_freemium(FALSE)
```

---

set_key	<i>Set HERE Application Credentials</i>
---------	---

---

**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 login and key here: [klick](#)

**Value**

None.

**Examples**

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

---

set_verbose	<i>Verbose API usage of hereR</i>
-------------	-----------------------------------

---

**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**

```
set_verbose(ans = FALSE)
```

**Arguments**

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



**Value**

None.

**Examples**

```
set_verbose(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**

```
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**

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

# Stations
stations <- station(poi = poi, url_only = TRUE)
```

---

unset_key	<i>Remove HERE Application Credentials</i>
-----------	--

---

**Description**

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

**Usage**

```
unset_key()
```

**Value**

None.

**Examples**

```
unset_key()
```

---

weather	<i>HERE Destination Weather API: Observations, Forecasts, Astronomy and Alerts</i>
---------	--

---

**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, [11](#)

aoi, [2](#)  
autosuggest, [3](#)

connection, [4](#)

flow, [5](#)

geocode, [6](#)

incident, [7](#)  
intermodal\_route, [8](#)  
isoline, [9](#)

poi, [11](#)

reverse\_geocode, [11](#)  
route, [12](#)  
route\_matrix, [14](#)

set\_freemium, [15](#)  
set\_key, [16](#)  
set\_verbose, [16](#)  
station, [17](#)

unset\_key, [18](#)

weather, [18](#)