Package 'openrouteservice'

May 9, 2026

Title An 'openrouteservice' API Client

Version 0.6.2

Description The client streamlines access to the services provided by <https://api.openrouteservice.org>.
It allows you to painlessly query for directions, isochrones, time-distance matrices, geocoding, elevation, points of interest, and more.

URL https://github.com/GIScience/openrouteservice-r

BugReports https://github.com/GIScience/openrouteservice-r/issues

Imports geojsonsf, httr, jsonlite, jsonvalidate, keyring, leaflet, utils, V8, xml2

Suggests covr, ggplot2, googlePolylines, lwgeom, knitr, mapview, pkgdown, RColorBrewer, rmarkdown, roxygen2, sf, testthat, units

License Apache License 2.0

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.3.2

NeedsCompilation no

Author Heidelberg Institute for Geoinformation Technology (HeiGIT) gGmbH [cph], Andrzej K. Oleś [aut, cre] (ORCID: <https://orcid.org/0000-0003-0285-2787>)

Maintainer Andrzej K. Oleś <andrzej.oles@gmail.com>

Repository CRAN

Date/Publication 2025-02-08 01:10:01 UTC

Contents

`fitBBox` *Set Bounds of a Map Widget*

Description

Helper function to set the bounds of a leaflet map widget.

Usage

```
fitBBox(map, bbox)
```

Arguments

`map`	a map widget object created from `leaflet()`
`bbox`	A vector `c(lng1, lat1, lng2, lat2)` specifying the bounding box coordinates

Value

The modified map widget.

Author(s)

Andrzej Oleś andrzej.oles@gmail.com

`ors_api_key` *API key management*

Description

Get/set openrouteservice API key.

Usage

```
ors_api_key(key, service = "openrouteservice", username = NULL, keyring = NULL)
```

Arguments

`key`	API key value provided as a character scalar
`service`	Service name, a character scalar.
`username`	Username, a character scalar, or `NULL` if the key is not associated with a username.
`keyring`	For systems that support multiple keyrings, specify the name of the keyring to use here. If `NULL`, then the default keyring is used. See also `has_keyring_support()`.

Details

To set the key provide it in the `key` argument. To retrieve the current value call the function with `key` unset.

Typically the key is saved in the system credential store. Once the key is defined, it persists in the keyring store of the operating system so it doesn't need to be set again in a new R session.

Internally the function uses `\link[keyring]{key_set}` and `\link[keyring]{key_get}`. The use of keyring package can be bypassed by providing the key in the environment variable ORS_API_KEY. The value from the environment variable takes precedence over the value stored in the system credential store. The default environment variable name used to retrieve the openrouteservice api key can be overridden by specifying it in `options("openrouteservice.api_key_env")`.

Value

API Key value when called without `key`.

Author(s)

Andrzej Oleś andrzej.oles@gmail.com

`ors_directions` *Openrouteservice Directions*

Description

Get directions for different modes of transport.

Usage

```
ors_directions(
  coordinates,
  profile = ors_profile(),
  format = c("geojson", "json", "gpx"),
  ...,
  api_key = ors_api_key(),
  output = c("parsed", "text", "sf")
)
```

Arguments

`coordinates`	List of `longitude, latitude` coordinate pairs visited in order, alternatively a two column `matrix` or `data.frame`.
`profile`	Route profile, defaults to `"driving-car"`.
`format`	Response format, defaults to `"geojson"`
`...`	Optional parameters as described here
`api_key`	Character scalar containing openrouteservice API key
`output`	Output format. By default the response is being parsed to a list-based R object

Value

Route between two or more locations in the selected `format` structured according to `output`:

- for `"text"`, a character vector of length 1 re-encoded to UTF-8.
- for `"parsed"`, a parsed R object.
- for `"sf"`, a simple features `sf` object.

Author(s)

Andrzej Oleś andrzej.oles@gmail.com

Examples

```
# These examples might require interaction to query the local keyring, or
# might fail due to network issues, so they are not run by default
## Not run:
coordinates <- list(c(8.34234, 48.23424), c(8.34423, 48.26424))

# simple call
try( ors_directions(coordinates, preference="fastest") )

# customized options
try( ors_directions(coordinates, profile="cycling-mountain", elevation=TRUE) )

# list of locations as `data.frame` output as simple features `sf` object
locations <- data.frame(lng = c(8.34234, 8.327807, 8.34423),
                        lat = c(48.23424, 48.239368, 48.26424))
try( ors_directions(locations, output = "sf") )

## End(Not run)
```

`ors_elevation` *Openrouteservice Elevation*

Description

Get elevation data for points or lines

Usage

```
ors_elevation(
  format_in = c("geojson", "point", "polyline", "encodedpolyline", "encodedpolyline6"),
  geometry,
  format_out = format_in,
  ...,
  api_key = ors_api_key(),
  output = c("parsed", "text", "sf")
)
```

Arguments

`format_in`	input format
`geometry`	`longitude, latitude` coordinate pairs
`format_out`	output format
`...`	Optional parameters as described here
`api_key`	Character scalar containing openrouteservice API key
`output`	Output format. By default the response is being parsed to a list-based R object

Details

A GeoJSON based service to query SRTM elevation for Point or LineString 2D geometries and return 3D geometries in various formats.

Value

3D point or line geometry structured according to `output`:

- for `"text"`, a character vector of length 1 re-encoded to UTF-8.
- for `"parsed"`, a parsed R object.
- for `"sf"`, a simple features `sf` object.

Author(s)

Andrzej Oleś andrzej.oles@gmail.com

Examples

```
# These examples might require interaction to query the local keyring, or
# might fail due to network issues, so they are not run by default
## Not run:
# point coordinates
coordinates <- c(13.349762, 38.11295)
try( ors_elevation("point", coordinates) )

# geojson as input
point <- '{ "type": "Point", "coordinates": [13.349762, 38.11295] }'
try( ors_elevation("geojson", point) )

# line geometry returned as encoded polyline
coordinates <- list(
  c(13.349762, 38.11295),
  c(12.638397, 37.645772)
)
try( ors_elevation("polyline", coordinates, format_out = "encodedpolyline") )

## End(Not run)
```

`ors_export` *Openrouteservice Export*

Description

Export the base graph for different modes of transport.

Usage

```
ors_export(
  bbox,
  profile = ors_profile(),
  ...,
  api_key = ors_api_key(),
  output = c("parsed", "text")
)
```

Arguments

bbox	List of `longitude, latitude` coordinate pairs defining the SW and NE corners of a rectangular area of interest, alternatively a two column `matrix` or `data.frame`.
profile	Route profile, defaults to `"driving-car"`.
...	Optional parameters as described here
api_key	Character scalar containing openrouteservice API key
output	Output format. By default the response is being parsed to a list-based R object

Value

Lists of graph nodes and edges contained in the provided bounding box and relevant for the given routing profile. The edge property `weight` represents travel time in seconds. The response is structured according to `output`:

- for `"text"`, a character vector of length 1 re-encoded to UTF-8.
- for `"parsed"`, a parsed R object.

Author(s)

Andrzej Oleś andrzej.oles@gmail.com

Examples

```
## Not run:
bbox <- list(
  c(8.681495, 49.41461),
  c(8.686507, 49.41943)
)

res <- ors_export(bbox)

## End(Not run)
```

`ors_geocode` *Openrouteservice Geocoding*

Description

Resolve input coordinates to addresses and vice versa.

Usage

```
ors_geocode(
  query,
  location,
  ...,
  api_key = ors_api_key(),
  output = c("parsed", "text", "sf")
)
```

Arguments

`query`	Name of location, street address or postal code. For a structured geocoding request a named list of parameters.
`location`	Coordinates to be inquired provided in the form `c(longitude, latitude)`
`...`	Optional parameters as described here
`api_key`	Character scalar containing openrouteservice API key
`output`	Output format. By default the response is being parsed to a list-based R object

Details

This endpoint can be used for geocoding (specified `query`) and reverse geocoding requests (specified `location`). Either `query` or `location` has to be specified for a valid request. If both parameters are specified location takes precedence.

Value

Geocoding: a JSON formatted list of objects corresponding to the search input. Reverse geocoding: the next enclosing object with an address tag which surrounds the given coordinate.

Author(s)

Andrzej Oleś andrzej.oles@gmail.com

Examples

```
# These examples might require interaction to query the local keyring, or
# might fail due to network issues, so they are not run by default
## Not run:
## locations of Heidelberg around the globe
x <- ors_geocode("Heidelberg")

## set the number of results returned
x <- ors_geocode("Heidelberg", size = 1)

## search within a particular country
ors_geocode("Heidelberg", boundary.country = "DE")

## structured geocoding
x <- ors_geocode(list(locality="Heidelberg", county="Heidelberg"))

## reverse geocoding
location <- x$features[[1L]]$geometry$coordinates
y <- ors_geocode(location = location, layers = "locality", size = 1)

## End(Not run)
```

`ors_isochrones` *Openrouteservice Isochrones*

Description

Obtain areas of reachability from given locations.

Usage

```
ors_isochrones(
  locations,
  profile = ors_profile(),
  range = 60,
  ...,
  api_key = ors_api_key(),
  output = c("parsed", "text", "sf")
)
```

Arguments

`locations`	List of `longitude, latitude` coordinate pairs, alternatively a two column `matrix` or `data.frame`.
`profile`	Route profile, defaults to `"driving-car"`.
`range`	Maximum range value of the analysis in seconds for time and meters for distance. Alternatively a comma separated list of specific single range values.

...	Optional parameters as described here
api_key	Character scalar containing openrouteservice API key
output	Output format. By default the response is being parsed to a list-based R object

Details

The Isochrone Service supports time and distance analyses for one single or multiple locations. You may also specify the isochrone interval or provide multiple exact isochrone range values.

Value

A GeoJSON object containing a FeatureCollection of Polygons

- for "text", a character vector of length 1 re-encoded to UTF-8.
- for "parsed", a parsed R object.
- for "sf", a simple features sf object.

Author(s)

Andrzej Oleś andrzej.oles@gmail.com

Examples

```
# These examples might require interaction to query the local keyring, or
# might fail due to network issues, so they are not run by default
## Not run:
ors_isochrones(c(8.34234, 48.23424), interval=20)

locations <- list(c(8.681495, 49.41461), c(8.686507,49.41943))
ors_isochrones(locations, range=c(300, 200))

## End(Not run)
```

---

ors_matrix *Openrouteservice Matrix*

Description

Obtain one-to-many, many-to-one and many-to-many matrices for time and distance.

Usage

```
ors_matrix(
  locations,
  profile = ors_profile(),
  ...,
  api_key = ors_api_key(),
  output = c("parsed", "text")
)
```

Arguments

locations	List of longitude, latitude coordinate pairs, alternatively a two column matrix or data.frame.
profile	Route profile, defaults to "driving-car".
...	Optional parameters as described here
api_key	Character scalar containing openrouteservice API key
output	Output format. By default the response is being parsed to a list-based R object

Value

Duration or distance matrix for multiple source and destination

- for "text", a character vector of length 1 re-encoded to UTF-8.
- for "parsed", a parsed R object.

Author(s)

Andrzej Oleś andrzej.oles@gmail.com

Examples

```
# These examples might require interaction to query the local keyring, or
# might fail due to network issues, so they are not run by default
## Not run:
coordinates <- list(
  c(9.970093, 48.477473),
  c(9.207916, 49.153868),
  c(37.573242, 55.801281),
  c(115.663757,38.106467)
)

# query for duration and distance in km
res <- ors_matrix(coordinates, metrics = c("duration", "distance"), units = "km")

# duration in hours
res$durations / 3600

# distance in km
res$distances

## End(Not run)
```

`ors_optimization` *Openrouteservice Optimization*

Description

Optimize a fleet of vehicles on a number of jobs. For more information, see the Vroom project API documentation.

The helper functions `jobs()` and `vehicles()` create data.frames which can be used as arguments to `ors_optimization()`.

Usage

```
ors_optimization(
  jobs,
  vehicles,
  matrix = NULL,
  ...,
  api_key = ors_api_key(),
  output = c("parsed", "text")
)

jobs(
  id,
  location,
  location_index,
  service,
  amount,
  skills,
  priority,
  time_windows
)

vehicles(
  id,
  profile,
  start,
  start_index,
  end,
  end_index,
  capacity,
  skills,
  time_window
)
```

Arguments

`jobs`	`data.frame` describing the places to visit

vehicles	data.frame describing the available vehicles
matrix	Optional two-dimensional array describing a custom travel-time matrix
...	Optional parameters as described here
api_key	Character scalar containing openrouteservice API key
output	Output format. By default the response is being parsed to a list-based R object
id	An integer used as unique identifier
location	Coordinates array
location_index	Index of relevant row and column in custom matrix
service	Job service duration (defaults to 0)
amount	An array of integers describing multidimensional quantities
skills	An array of integers defining skills
priority	An integer in the [0, 10] range describing priority level (defaults to 0)
time_windows	An array of time_window objects describing valid slots for job service start
profile	routing profile (defaults to car)
start	coordinates array
start_index	index of relevant row and column in custom matrix
end	coordinates array
end_index	index of relevant row and column in custom matrix
capacity	an array of integers describing multidimensional quantities
time_window	a time_window object describing working hours

Value

Solution computed by the optimization endpoint formatted as described here and structured according to output:

- for "text", a character vector of length 1 re-encoded to UTF-8.
- for "parsed", a parsed R object.

Author(s)

Andrzej Oleś andrzej.oles@gmail.com

Examples

```
# These examples might require interaction to query the local keyring, or
# might fail due to network issues, so they are not run by default
## Not run:
home_base <- c(2.35044, 48.71764)

vehicles <- vehicles(
  id = 1:2,
  profile = "driving-car",
```

```
  start = home_base,
  end = home_base,
  capacity = 4,
  skills = list(c(1, 14), c(2, 14)),
  time_window = c(28800, 43200)
)

locations <- list(
  c(1.98935, 48.701),
  c(2.03655, 48.61128),
  c(2.39719, 49.07611),
  c(2.41808, 49.22619),
  c(2.28325, 48.5958),
  c(2.89357, 48.90736)
)

jobs <- jobs(
  id = 1:6,
  service = 300,
  amount = 1,
  location = locations,
  skills = list(1, 1, 2, 2, 14, 14)
)

try( ors_optimization(jobs, vehicles) )

## End(Not run)
```

`ors_pois` *Openrouteservice POIs*

Description

Search for points of interest around points or in geometries.

Usage

```
ors_pois(
  request = c("pois", "stats", "list"),
  geometry,
  ...,
  api_key = ors_api_key(),
  output = c("parsed", "text", "sf")
)
```

Arguments

`request`	One of the following: `"pois"`, `"stats"` or `"list"`

geometry	named list containing either a geojson geometry object (GeoJSON Point, LineString or Polygon) or a bbox, optionally buffered by a value provided buffer
...	Optional request attributes as described here
api_key	Character scalar containing openrouteservice API key
output	Output format. By default the response is being parsed to a list-based R object

Details

There are three different request types: pois, stats and list.

pois returns a GeoJSON FeatureCollection in the bounding box specified in geometry$bbox or a GeoJSON geometry provided in geometry$geojson. stats does the same but groups by categories, ultimately returning a JSON object with the absolute numbers of POIs of a certain group.

list returns a list of category groups and their ids.

Value

A list of points of interest in the area specified in geometry structured according to output:

- for "text", a character vector of length 1 re-encoded to UTF-8.
- for "parsed", a parsed R object.
- for "sf", a simple features sf object. Valid only for argument request = "pois".

Author(s)

Andrzej Oleś andrzej.oles@gmail.com

Examples

```
# These examples might require interaction to query the local keyring, or
# might fail due to network issues, so they are not run by default
## Not run:
# POI categories list
ors_pois('list')

# POIs around a buffered point
geometry <- list(geojson = list(type = "Point",
                                coordinates = c(8.8034, 53.0756)),
                 buffer = 100)
ors_pois(geometry = geometry)

# alternative specification via bounding box
ors_pois(geometry = list(bbox = list(c(8.8034, 53.0756), c(8.8034, 53.0756)),
                         buffer = 100))

# POIs of given categories
ors_pois(geometry = geometry,
         limit = 200,
         sortby = "distance",
```

```
          filters = list(
            category_ids = c(180, 245)
          ))

# POIs of given category groups
ors_pois(geometry = geometry,
          limit = 200,
          sortby = "distance",
          filters = list(
            category_group_ids = 160
          ))

# POI Statistics
ors_pois("stats", geometry = geometry)

## End(Not run)
```

`ors_profile` *Openrouteservice Profiles*

Description

List of available modes of transport.

Usage

```
ors_profile(
  mode = c("car", "hgv", "bike", "roadbike", "mtb", "e-bike", "walking", "hiking",
     "wheelchair")
)
```

Arguments

`mode`	Profile label.

Details

Convenience function for specifying the profile in `ors_directions()`, `ors_isochrones()` and `ors_matrix()`.

Value

Profile name, or named vector of available profiles.

Author(s)

Andrzej Oleś andrzej.oles@gmail.com

See Also

ors_directions(), ors_isochrones(), ors_matrix()

Examples

```
# list availbale profiles
ors_profile()

# retrieve full profile name based on label
ors_profile("car")
```

---

ors_snap *Openrouteservice Snapping*

Description

Snap coordinates to road network

Usage

```
ors_snap(
  locations,
  profile = ors_profile(),
  radius,
  format = c("geojson", "json"),
  ...,
  api_key = ors_api_key(),
  output = c("parsed", "text", "sf")
)
```

Arguments

locations	List of longitude, latitude coordinate pairs, alternatively a two column matrix or data.frame.
profile	Route profile, defaults to "driving-car".
radius	Maximum radius in meters around given coordinates to search for graph edges
format	Response format, defaults to "geojson"
...	Optional parameters as described here
api_key	Character scalar containing openrouteservice API key
output	Output format. By default the response is being parsed to a list-based R object

Value

Coordinates of snapped location(s) and distance to the original point(s) structured according to `output`:

- for `"text"`, a character vector of length 1 re-encoded to UTF-8.
- for `"parsed"`, a parsed R object.
- for `"sf"`, a simple features `sf` object.

Author(s)

Andrzej Oleś andrzej.oles@gmail.com

Examples

```
# These examples might require interaction to query the local keyring, or
# might fail due to network issues, so they are not run by default
## Not run:
locations <- list(
  c(8.669629, 49.413025),
  c(8.675841, 49.418532),
  c(8.665144, 49.415594)
)

# query for locations snapped onto the OpenStreetMap road network
res <- ors_snap(locations, radius = 350)

## End(Not run)
```

`print.ors_api` *Print a Compact Summary of the API Response*

Description

`print.ors_api` uses str to compactly display the structure of the openrouteservice API response object.

Usage

```
## S3 method for class 'ors_api'
print(x, give.attr = FALSE, list.len = 6L, ...)
```

Arguments

`x`	object of class `ors_api`.
`give.attr`	logical; if `TRUE` (default), show attributes as sub structures.
`list.len`	numeric; maximum number of list elements to display within a level.
`...`	further arguments passed to str.

Value

`print.ors_api` prints its argument and returns it *invisibly*.

Index