Package 'osrm'

May 9, 2026

Type Package

Title Interface Between R and the OpenStreetMap-Based Routing Service OSRM

Version 5.0.0

Description An interface between R and the 'OSRM' API. 'OSRM' is a routing service based on 'OpenStreetMap' data. See <http://project-osrm.org/> for more information. This package enables the computation of routes, trips, isochrones and travel distances matrices (travel time and kilometric distance).

License GPL (>= 3)

Imports RcppSimdJson, curl, utils, mapiso, googlePolylines, sf

Depends R (>= 3.5.0)

Suggests mapsf, tinytest, covr, terra

URL https://github.com/riatelab/osrm

BugReports https://github.com/riatelab/osrm/issues

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Timothée Giraud [cre, aut] (ORCID: <https://orcid.org/0000-0002-1932-3323>), Robin Cura [ctb] (ORCID: <https://orcid.org/0000-0001-5926-1828>), Matthieu Viry [ctb] (ORCID: <https://orcid.org/0000-0002-0693-8556>), Robin Lovelace [ctb] (ORCID: <https://orcid.org/0000-0001-5679-6536>)

Maintainer Timothée Giraud <timothee.giraud@cnrs.fr>

Repository CRAN

Date/Publication 2025-12-17 14:00:02 UTC

Contents

`osrm` *Shortest Paths and Travel Time from OpenStreetMap via an OSRM API*

Description

An interface between R and the OSRM API.
OSRM is a routing service based on OpenStreetMap data. See <http://project-osrm.org/> for more information. This package enables the computation of routes, trips, isochrones and travel distances matrices (travel time and kilometric distance).

- `osrmTable`: Build and send OSRM API queries to get travel time matrices between points. This function interfaces the *table* OSRM service.
- `osrmRoute`: Build and send an OSRM API query to get the travel geometry between two points. This function interfaces with the *route* OSRM service.
- `osrmTrip`: Build and send an OSRM API query to get the shortest travel geometry between multiple unordered points. This function interfaces the *trip* OSRM service. Use this function to resolve the travelling salesman problem.
- `osrmNearest`: Build and send an OSRM API query to get the nearest point on the street network. This function interfaces the *nearest* OSRM service.
- `osrmIsochrone`: This function computes areas that are reachable within a given time span from a point and returns the reachable regions as polygons. These areas of equal travel time are called isochrones.
- `osrmIsodistance`: This function computes areas that are reachable within a given road distance from a point and returns the reachable regions as polygons. These areas of equal travel distance are called isodistances.

Note

This package relies on the usage of a running OSRM service (tested with version 6.0.0 of the OSRM API).

To set the OSRM server, change the `osrm.server` option:
`options(osrm.server = "http://address.of.the.server/")`

To set the profile, use the `osrm.profile` option:
`options(osrm.profile = "name.of.the.profile")`

The "car" profile is set by default. Other possible profiles are "foot" and "bike".

A typical setup, corresponding to the Docker example, would be:
`options(osrm.server = "http://0.0.0.0:5000/", osrm.profile = "car")`

The package ships a sample dataset of 100 random pharmacies in Berlin (© OpenStreetMap contributors - https://www.openstreetmap.org/copyright/en).
The sf dataset uses the projection WGS 84 / UTM zone 34N (EPSG:32634).
The csv dataset uses WGS 84 (EPSG:4326).

Author(s)

Maintainer: Timothée Giraud <timothee.giraud@cnrs.fr> (ORCID)

Other contributors:

- Robin Cura (ORCID) [contributor]
- Matthieu Viry (ORCID) [contributor]
- Robin Lovelace (ORCID) [contributor]

See Also

Useful links:

- https://github.com/riatelab/osrm
- Report bugs at https://github.com/riatelab/osrm/issues

`osrmIsochrone` *Get Polygons of Isochrones*

Description

This function computes areas that are reachable within a given time span from a point and returns the reachable regions as polygons. These areas of equal travel time are called isochrones.

Usage

```
osrmIsochrone(
  loc,
  breaks = seq(from = 0, to = 60, length.out = 7),
  exclude,
  n = 500,
  smooth = FALSE,
  res,
  osrm.server = getOption("osrm.server"),
  osrm.profile = getOption("osrm.profile")
)
```

Arguments

loc	origin point. loc can be: • a vector of coordinates (longitude and latitude, WGS 84), • a data.frame of longitudes and latitudes (WGS 84), • a matrix of longitudes and latitudes (WGS 84), • an sfc object of type POINT, • an sf object of type POINT. If loc is a data.frame, a matrix, an sfc object or an sf object then only the first row or element is considered.
breaks	a numeric vector of break values to define isochrone areas, in minutes.
exclude	pass an optional "exclude" request option to the OSRM API.
n	number of points used to compute isochrones, possible values are c(100, 200, 500, 1000, 2000, 5000, 10000, 20000, 50000).
smooth	if TRUE a moving window with a gaussian blur is applied to durations. This option may be usefull to remove small patches of hard to reach areas. The computed isochrones are less precise but better looking.
res	deprecated
osrm.server	the base URL of the routing server. getOption("osrm.server") by default.
osrm.profile	the routing profile to use, e.g. "car", "bike" or "foot" (when using the routing.openstreetmap.de test server). getOption("osrm.profile") by default.

Value

The output of this function is an sf MULTIPOLYGON of isochrones.
It contains 3 fields:

- id, an identifier
- isomin, the minimum value of the isochrone polygon in minutes
- isomax, the maximum value of the isochrone polygon in minutes

If loc is a vector, a data.frame or a matrix the coordinate reference system (CRS) of the output is EPSG:4326 (WGS84).
If loc is an sfc or sf object, the output has the same CRS as loc.

Examples

```
## Not run:
library(sf)
apotheke.sf <- st_read(system.file("gpkg/apotheke.gpkg", package = "osrm"),
  quiet = TRUE
)
# Get isochones with lon/lat coordinates
iso <- osrmIsochrone(loc = c(13.43, 52.47), breaks = seq(0, 12, 2))
# Map
plot(iso["isomax"], breaks = sort(unique(c(iso$isomin, iso$isomax))))
```

```
# Get isochones with an sf POINT
iso2 <- osrmIsochrone(loc = apotheke.sf[11, ], breaks = seq(0, 12, 2))
# Map
if (require("mapsf")) {
  mapsf::mf_map(
    x = iso2, var = "isomin", type = "choro",
    breaks = sort(unique(c(iso2$isomin, iso2$isomax))),
    pal = "Burg", border = NA, leg_pos = "topleft",
    leg_val_rnd = 0,
    leg_frame = TRUE, leg_title = "Isochrones\n(min)"
  )
}

## End(Not run)
```

`osrmIsodistance` *Get Polygons of Isodistances*

Description

This function computes areas that are reachable within a given road distance from a point and returns the reachable regions as polygons. These areas of equal travel distance are called isodistances.

Usage

```
osrmIsodistance(
  loc,
  breaks = seq(from = 0, to = 10000, length.out = 4),
  exclude,
  n = 500,
  smooth = FALSE,
  res,
  osrm.server = getOption("osrm.server"),
  osrm.profile = getOption("osrm.profile")
)
```

Arguments

`loc` origin point. `loc` can be:

- a vector of coordinates (longitude and latitude, WGS 84),
- a data.frame of longitudes and latitudes (WGS 84),
- a matrix of longitudes and latitudes (WGS 84),
- an sfc object of type POINT,
- an sf object of type POINT.

If `loc` is a data.frame, a matrix, an sfc object or an sf object then only the first row or element is considered.

breaks	a numeric vector of break values to define isodistance areas, in meters.
exclude	pass an optional "exclude" request option to the OSRM API.
n	number of points used to compute isodistances, possible values are c(100, 200, 500, 1000, 2000, 5000, 10000, 20000, 50000).
smooth	if TRUE a moving window with a gaussian blur is applied to distances. This option may be usefull to remove small patches of hard to reach areas. The computed isodistances are less precise but better looking.
res	deprecated
osrm.server	the base URL of the routing server. getOption("osrm.server") by default.
osrm.profile	the routing profile to use, e.g. "car", "bike" or "foot" (when using the routing.openstreetmap.de test server). getOption("osrm.profile") by default.

Value

The output of this function is an sf MULTIPOLYGON of isodistances.
It contains 3 fields:

- id, an identifier
- isomin, the minimum value of the isodistance polygon in meters
- isomax, the maximum value of the isodistance polygon in meters

If loc is a vector, a data.frame or a matrix the coordinate reference system (CRS) of the output is EPSG:4326 (WGS84).
If loc is an sfc or sf object, the output has the same CRS as loc.

Examples

```
## Not run:
library(sf)
apotheke.sf <- st_read(system.file("gpkg/apotheke.gpkg", package = "osrm"),
  quiet = TRUE
)
# Get isochones with lon/lat coordinates
iso <- osrmIsodistance(loc = c(13.43, 52.47), breaks = seq(0, 500, 100))
# Map
plot(iso["isomax"], breaks = sort(unique(c(iso$isomin, iso$isomax))))

# Get isochones with an sf POINT
iso2 <- osrmIsodistance(loc = apotheke.sf[11, ], breaks = seq(0, 500, 100))
# Map
if (require("mapsf")) {
  mapsf::mf_map(
    x = iso2, var = "isomin", type = "choro",
    breaks = sort(unique(c(iso2$isomin, iso2$isomax))),
    pal = "Burg", border = NA, leg_pos = "topleft",
    leg_val_rnd = 0,
    leg_frame = TRUE, leg_title = "Isochrones\n(min)"
  )
```

```
}

## End(Not run)
```

`osrmNearest` *Get the Nearest Point on the Street Network*

Description

This function interfaces with the *nearest* OSRM service.

Usage

```
osrmNearest(
  loc,
  n = 1,
  exclude,
  osrm.server = getOption("osrm.server"),
  osrm.profile = getOption("osrm.profile")
)
```

Arguments

`loc`	a point to snap to the street network. `loc` can be: • a vector of coordinates (longitude and latitude, WGS 84), • a data.frame of longitudes and latitudes (WGS 84), • a matrix of longitudes and latitudes (WGS 84), • an sfc object of type POINT, • an sf object of type POINT. If `loc` is a data.frame, a matrix, an sfc object or an sf object then only the first row or element is considered.
`n`	number of nearest points
`exclude`	pass an optional "exclude" request option to the OSRM API.
`osrm.server`	the base URL of the routing server.
`osrm.profile`	the routing profile to use, e.g. "car", "bike" or "foot".

Value

The output of this function is an sf POINT of the point on the street network.
It contains 2 fields:

- id, the point identifier
- distance, the distance in meters to the supplied input point.

Examples

```
## Not run:
library(sf)
apotheke.sf <- st_read(system.file("gpkg/apotheke.gpkg", package = "osrm"),
  quiet = TRUE
)
pt <- osrmNearest(apotheke.sf[56, ])
pt$distance

## End(Not run)
```

`osrmRoute` *Get the Shortest Path Between Two Points*

Description

Build and send an OSRM API query to get the travel geometry between two points. This function interfaces with the *route* OSRM service.
Use `src` and `dst` to get the shortest direct route between two points.
Use `loc` to get the shortest route between two points using ordered waypoints.

Usage

```
osrmRoute(
  src,
  dst,
  loc,
  overview = "simplified",
  exclude,
  osrm.server = getOption("osrm.server"),
  osrm.profile = getOption("osrm.profile")
)
```

Arguments

`src`	starting point of the route. `src` can be: • a vector of coordinates (longitude and latitude, WGS 84), • a data.frame of longitudes and latitudes (WGS 84), • a matrix of longitudes and latitudes (WGS 84), • an sfc object of type POINT, • an sf object of type POINT. If relevant, row names are used as identifiers. If `src` is a data.frame, a matrix, an sfc object or an sf object then only the first row or element is considered.
`dst`	destination of the route. `dst` can be: • a vector of coordinates (longitude and latitude, WGS 84),

- a data.frame of longitudes and latitudes (WGS 84),
- a matrix of longitudes and latitudes (WGS 84),
- an sfc object of type POINT,
- an sf object of type POINT.

If relevant, row names are used as identifiers.
If `dst` is a data.frame, a matrix, an sfc object or an sf object then only the first row or element is considered.

`loc` starting point, waypoints (optional) and destination of the route. `loc` can be:

- a data.frame of longitudes and latitudes (WGS 84),
- a matrix of longitudes and latitudes (WGS 84),
- an sfc object of type POINT,
- an sf object of type POINT.

The first row or element is the starting point then waypoints are used in the order they are stored in `loc` and the last row or element is the destination.
If relevant, row names are used as identifiers.

`overview` "full", "simplified" or FALSE. Use "full" to return the detailed geometry, use "simplified" to return a simplified geometry, use FALSE to return only time and distance.

`exclude` pass an optional "exclude" request option to the OSRM API.

`osrm.server` the base URL of the routing server.

`osrm.profile` the routing profile to use, e.g. "car", "bike" or "foot".

Value

The output of this function is an sf LINESTRING of the shortest route.
It contains 4 fields:

- starting point identifier
- destination identifier
- travel time in minutes
- travel distance in kilometers.

If src (or loc) is a vector, a data.frame or a matrix, the coordinate reference system (CRS) of the route is EPSG:4326 (WGS84).
If src (or loc) is an sfc or sf object, the route has the same CRS as src (or loc).

If overview is FALSE, a named numeric vector is returned. It contains travel time (in minutes) and travel distance (in kilometers).

Examples

```
## Not run:
library(sf)
apotheke.df <- read.csv(system.file("csv/apotheke.csv", package = "osrm"))
apotheke.sf <- st_read(system.file("gpkg/apotheke.gpkg", package = "osrm"),
```

```
  quiet = TRUE
)
# Travel path between points
route1 <- osrmRoute(src = apotheke.sf[1, ], dst = apotheke.sf[16, ])
# Display paths
plot(st_geometry(route1))
plot(st_geometry(apotheke.sf[c(1, 16), ]), col = "red", pch = 20, add = TRUE)

# Return only duration and distance
route3 <- osrmRoute(
  src = apotheke.df[1, c("lon", "lat")],
  dst = apotheke.df[16, c("lon", "lat")],
  overview = FALSE
)
route3

# Using only coordinates
route4 <- osrmRoute(
  src = c(13.412, 52.502),
  dst = c(13.454, 52.592)
)
plot(st_geometry(route4))

# Using via points
route5 <- osrmRoute(loc = apotheke.sf[c(1, 2, 4, 3), ])
plot(st_geometry(route5), col = "red", lwd = 2)
plot(st_geometry(apotheke.sf[c(1, 2, 4, 3), ]), add = TRUE)

# Using a different routing server
u <- "https://routing.openstreetmap.de/routed-foot/"
route5 <- osrmRoute(apotheke.sf[1, ], apotheke.sf[16, ], osrm.server = u)
route5

# Using an open routing service with support for multiple modes
# see https://github.com/riatelab/osrm/issues/67
u <- "https://routing.openstreetmap.de/"
options(osrm.server = u)
route6 <- osrmRoute(apotheke.sf[1, ], apotheke.sf[16, ],
  osrm.profile = "bike"
)
route7 <- osrmRoute(apotheke.sf[1, ], apotheke.sf[16, ],
  osrm.profile = "car"
)
plot(st_geometry(route7), col = "green") # car
plot(st_geometry(route6), add = TRUE) # bike
plot(st_geometry(route5), col = "red", add = TRUE) # foot

## End(Not run)
```

`osrmTable` *Get Travel Time Matrices Between Points*

Description

Build and send OSRM API queries to get travel time matrices between points. This function interfaces the *table* OSRM service.
Use `src` and `dst` to set different origins and destinations.
Use `loc` to compute travel times or travel distances between all points.

Usage

```
osrmTable(
  src,
  dst = src,
  loc,
  exclude,
  measure = "duration",
  osrm.server = getOption("osrm.server"),
  osrm.profile = getOption("osrm.profile")
)
```

Arguments

`src`	origin points. `src` can be: • a data.frame of longitudes and latitudes (WGS 84), • a matrix of longitudes and latitudes (WGS 84), • an sfc object of type POINT, • an sf object of type POINT. If relevant, row names are used as identifiers.
`dst`	destination. `dst` can be: • a data.frame of longitudes and latitudes (WGS 84), • a matrix of longitudes and latitudes (WGS 84), • an sfc object of type POINT, • an sf object of type POINT. If relevant, row names are used as identifiers.
`loc`	points. `loc` can be: • a data.frame of longitudes and latitudes (WGS 84), • a matrix of longitudes and latitudes (WGS 84), • an sfc object of type POINT, • an sf object of type POINT. If relevant, row names are used as identifiers.
`exclude`	pass an optional "exclude" request option to the OSRM API (not allowed with the OSRM demo server).
`measure`	a character indicating what measures are calculated. It can be "duration" (in minutes), "distance" (meters), or both c('duration', 'distance').
`osrm.server`	the base URL of the routing server.
`osrm.profile`	the routing profile to use, e.g. "car", "bike" or "foot".

Value

The output of this function is a list composed of one or two matrices and 2 data.frames

- durations: a matrix of travel times (in minutes)
- distances: a matrix of distances (in meters)
- sources: a data.frame of the coordinates of the points actually used as starting points (EPSG:4326 - WGS84)
- sources: a data.frame of the coordinates of the points actually used as destinations (EPSG:4326 - WGS84)

Note

The OSRM demo server does not allow large queries (more than 10000 distances or durations).
If you use your own server and if you want to get a large number of distances make sure to set the "max-table-size" option (Max. locations supported in table) of the OSRM server accordingly.

Examples

```
## Not run:
# Inputs are data frames
apotheke.df <- read.csv(system.file("csv/apotheke.csv", package = "osrm"))
# Travel time matrix
distA <- osrmTable(loc = apotheke.df[1:50, c("lon", "lat")])
# First 5 rows and columns
distA$durations[1:5, 1:5]

# Travel time matrix with different sets of origins and destinations
distA2 <- osrmTable(
  src = apotheke.df[1:10, c("lon", "lat")],
  dst = apotheke.df[11:20, c("lon", "lat")]
)
# First 5 rows and columns
distA2$durations[1:5, 1:5]

# Inputs are sf points
library(sf)
apotheke.sf <- st_read(system.file("gpkg/apotheke.gpkg", package = "osrm"),
  quiet = TRUE
)
distA3 <- osrmTable(loc = apotheke.sf[1:10, ])
# First 5 rows and columns
distA3$durations[1:5, 1:5]

# Travel time matrix with different sets of origins and destinations
distA4 <- osrmTable(src = apotheke.sf[1:10, ], dst = apotheke.sf[11:20, ])
# First 5 rows and columns
distA4$durations[1:5, 1:5]

# Road distance matrix with different sets of origins and destinations
distA5 <- osrmTable(
  src = apotheke.sf[1:10, ], dst = apotheke.sf[11:20, ],
```

```
  measure = "distance"
)
# First 5 rows and columns
distA5$distances[1:5, 1:5]

## End(Not run)
```

`osrmTrip` *Get the Travel Geometry Between Multiple Unordered Points*

Description

Build and send an OSRM API query to get the shortest travel geometry between multiple unordered points. This function interfaces the *trip* OSRM service.
Use this function to resolve the travelling salesman problem.

Usage

```
osrmTrip(
  loc,
  exclude = NULL,
  overview = "simplified",
  osrm.server = getOption("osrm.server"),
  osrm.profile = getOption("osrm.profile")
)
```

Arguments

`loc`	starting point and waypoints to reach along the route. `loc` can be: • a data.frame of longitudes and latitudes (WGS 84), • a matrix of longitudes and latitudes (WGS 84), • an sfc object of type POINT, • an sf object of type POINT. The first row or element is the starting point. Row names, if relevant, or element indexes are used as identifiers.
`exclude`	pass an optional "exclude" request option to the OSRM API.
`overview`	"full", "simplified". Add geometry either full (detailed) or simplified according to highest zoom level it could be display on.
`osrm.server`	the base URL of the routing server.
`osrm.profile`	the routing profile to use, e.g. "car", "bike" or "foot".

Details

As stated in the OSRM API, if input coordinates can not be joined by a single trip (e.g. the coordinates are on several disconnected islands) multiple trips for each connected component are returned.

Value

A list of connected components is returned. Each component contains:

trip An sf LINESTRING. If loc is a data.frame or a matrix the coordinate reference system (CRS) of the route is EPSG:4326 (WGS84). If loc is an sfc or sf object, the route has the same CRS as loc.
Each line of the returned route is a step of the trip. The object has four columns: start (identifier of the starting point), end (identifier of the destination), duration (duration of the step in minutes), distance (length of the step in kilometers).

summary A list with 2 components: total duration (in minutes) and total distance (in kilometers) of the trip.

Examples

```
## Not run:
library(sf)
apotheke.sf <- st_read(system.file("gpkg/apotheke.gpkg", package = "osrm"),
  quiet = TRUE
)
# Get a trip with a set of points (sf POINT)
trips <- osrmTrip(loc = apotheke.sf[1:5, ])
mytrip <- trips[[1]]$trip
# Display the trip
plot(st_geometry(mytrip), col = "black", lwd = 4)
plot(st_geometry(mytrip), col = c("red", "white"), lwd = 1, add = TRUE)
plot(st_geometry(apotheke.sf[1:5, ]),
  pch = 21, bg = "red", cex = 1,
  add = TRUE
)

## End(Not run)
```

Index