Package 'osrm'

August 10, 2017

Type Package

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

Version 3.0.2

Date 2017-08-10

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. A public API exists but one can run its own instance. This package allows to compute distances (travel time and kilometric distance) between points and travel time matrices.

License GPL-3

LazyData TRUE

Imports jsonlite, RCurl, utils, stats, rgeos, methods, gepaf, raster

Depends R (>= 2.10), sp

Suggests cartography

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

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

Encoding UTF-8

RoxygenNote 6.0.1

NeedsCompilation no

Author Timothée Giraud [cre, aut],
Robin Cura [ctb],
Matthieu Viry [ctb]

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

Repository CRAN

Date/Publication 2017-08-10 15:35:17 UTC

R topics documented:

com . . . 2
dst . . . 2
osrm . . . 3
osrmIsochrone . . . 3
osrmRoute . . . 5
osrmTable . . . 6
osrmTrip . . . 8
src . . . 9

Index **10**

`com` *Communes Coordinates*

Description

Coordinates of a set of communes in France, Belgium and Luxembourg. Coordinates are in WGS84.

Source

UMS RIATE

`dst` *SpatialPointsDataFrame of 10 Communes in France*

Description

10 communes in France. The projection is RGF93 / Lambert-93.

Source

UMS RIATE

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. A public API exists but one can run its own instance. This package allows to compute distances (travel time and kilometric distance) between points and travel time matrices.

- `osrmTable` Get travel time matrices between points.
- `osrmRoute` Get the shortest path between two points.
- `osrmTrip` Get the travel geometry between multiple unordered points.
- `osrmIsochrone` Get a SpatialPolygonsDataFrame of isochrones.

Note

This package relies on the usage of a running OSRM service (tested with version 5.0.0 of the OSRM API).
By default, this service is the OSRM demo server (http://router.project-osrm.org/). If you plan to use the OSRM demo server you should read the OSRM API Usage Policy.
You should also take into account "that there are no guarantees regarding availability, stability or correctness of results. It's server to demonstrate OSRM, not a production-ready API." Demo Server

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

To change the profile (driving is set by default), use the `osrm.profile` option:
`options(osrm.profile = "name.of.the.profile")`

osrmIsochrone *Get a SpatialPolygonsDataFrame of Isochrones*

Description

Based on `osrmTable`, this function buids a SpatialPolygonsDataFrame of isochrones.

Usage

```
osrmIsochrone(loc, breaks = seq(from = 0, to = 60, length.out = 7),
  res = 30)
```

Arguments

loc	a numeric vector of longitude and latitude (WGS84) or a SpatialPointsDataFrame or a SpatialPolygonsDataFrame of the origine point.
breaks	a numeric vector of isochrone values (in minutes).
res	number of points used to compute isochrones, one side of the square grid, the total number of points will be res*res.

Value

A SpatialPolygonsDateFrame of isochrones is returned. The data frame of the output contains four fields: id (id of each polygon), min and max (minimum and maximum breaks of the polygon), center (central values of classes).

See Also

osrmTable

Examples

```
## Not run:
# Load data
data("com")

# Get isochones with lon/lat coordinates, default breaks
iso <- osrmIsochrone(loc = c(5.936036, 49.24882))
plot(iso)
points(5.936036, 49.24882, pch = 20, col = "red")

# Map
if(require("cartography")){
  osm <- getTiles(spdf = iso, crop = TRUE, type = "osmgrayscale")
  tilesLayer(osm)
  breaks <- sort(c(unique(iso$min), max(iso$max)))
  cartography::choroLayer(spdf = iso, df = iso@data,
                          var = "center", breaks = breaks,
                          border = NA,
                          legend.pos = "topleft",legend.frame = TRUE,
                          legend.title.txt = "Isochrones\n(min)",
                          add = TRUE)
}

# Get isochones with a SpatialPointsDataFrame, custom breaks
iso2 <- osrmIsochrone(loc = src[7,], breaks = seq(from = 0,to = 30, by = 5))

# Map
if(require("cartography")){
  osm2 <- getTiles(spdf = iso2, crop = TRUE, type = "osmgrayscale")
  tilesLayer(osm2)
  breaks2 <- sort(c(unique(iso2$min), max(iso2$max)))
  cartography::choroLayer(spdf = iso2, df = iso2@data,
                          var = "center", breaks = breaks2,
```

```
                          border = NA,
                          legend.pos = "topleft",legend.frame = TRUE,
                          legend.title.txt = "Isochrones\n(min)",
                          add = TRUE)
}

## 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 the *route* OSRM service.

Usage

```
osrmRoute(src, dst, overview = "simplified", sp = FALSE)
```

Arguments

`src`	a numeric vector of identifier, longitude and latitude (WGS84), a SpatialPointsDataFrame or a SpatialPolygonsDataFrame of the origine point.
`dst`	a numeric vector of identifier, longitude and latitude (WGS84), a SpatialPointsDataFrame or a SpatialPolygonsDataFrame of the destination point.
`overview`	"full", "simplified" or FALSE. Add geometry either full (detailed), simplified according to highest zoom level it could be display on, or not at all.
`sp`	if sp is TRUE the function returns a SpatialLinesDataFrame.

Value

If sp is FALSE, a data frame is returned. It contains the longitudes and latitudes of the travel path between the two points.
If sp is TRUE a SpatialLinesDataFrame is returned. It contains 4 fields : identifiers of origine and destination, travel time in minutes and travel distance in kilometers.
If overview is FALSE, a named numeric vector is returned. It contains travel time (in minutes) and travel distance (in kilometers).

Examples

```
## Not run:

# Load data
data("com")
# Travel path between points
route <- osrmRoute(src = com[1, c("comm_id", "lon","lat")],
                   dst = com[15, c("comm_id", "lon","lat")])
```

```
# Display the path
plot(com[c(1,15),3:4], asp =1, col = "red", pch = 20, cex = 1.5)
points(route[,1:2], type = "l", lty = 2)
text(com[c(1,15),3:4], labels = com[c(1,15),2], pos = 2)

# Travel path between points - output a SpatialLinesDataFrame
route2 <- osrmRoute(src=c("Bethune", 2.64781, 50.5199),
                    dst = c("Cassel", 2.486388, 50.80016),
                    sp = TRUE)
# Display the path
plot(com[c(1,16),3:4], asp =1, col = "red", pch = 20, cex = 1.5)
plot(route2, lty = 2, add=TRUE)
text(com[c(1,16),3:4], labels = com[c(1,16),2], pos = 2)

# Input is SpatialPointsDataFrames
route3 <- osrmRoute(src = src[1,], dst = dst[1,], sp = TRUE)
route3@data

## 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.

Usage

```
osrmTable(loc, src = NULL, dst = NULL)
```

Arguments

`loc`	a data frame containing 3 fields: points identifiers, longitudes and latitudes (WGS84). It can also be a SpatialPointsDataFrame or a SpatialPolygonsDataFrame, then row names are used as identifiers. If loc parameter is used, all pair-wise distances are computed.
`src`	a data frame containing origin points identifiers, longitudes and latitudes (WGS84). It can also be a SpatialPointsDataFrame or a SpatialPolygonsDataFrame, then row names are used as identifiers. If dst and src parameters are used, only pairs between scr/dst are computed.
`dst`	a data frame containing destination points identifiers, longitudes and latitudes (WGS84). It can also be a SpatialPointsDataFrame or a SpatialPolygonsDataFrame, then row names are used as identifiers.

Details

If loc, src or dst are data frames we assume that the 3 first columns of the data.frame are: identifiers, longitudes and latitudes.

Value

A list containing 3 data frames is returned. durations is the matrix of travel times (in minutes), sources and destinations are the coordinates of the origin and destination points actually used to compute the travel times (WGS84).

Note

The public OSRM API does not allow more than 10 000 distances in query result.
If you use an other OSRM API service, make sure that more distances are allowed in results (i.e. the "max-table-size" argument, Max. locations supported in distance table query).

See Also

osrmIsochrone

Examples

```
## Not run:
# Load data
data("com")

# Inputs are data frames
# Travel time matrix
distCom <- osrmTable(loc = com[1:50, c("comm_id","lon","lat")])
# First 5 rows and columns
distCom$durations[1:5,1:5]

# Travel time matrix with different sets of origins and destinations
distCom2 <- osrmTable(src = com[1:10,c("comm_id","lon","lat")],
                      dst = com[11:20,c("comm_id","lon","lat")])
# First 5 rows and columns
distCom2$durations[1:5,1:5]


# Inputs are SpatialPointsDataFrames
distCom <- osrmTable(loc = src)
# First 5 rows and columns
distCom$durations[1:5,1:5]

# Travel time matrix with different sets of origins and destinations
distCom2 <- osrmTable(src = src, dst = dst)
# First 5 rows and columns
distCom2$durations[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 points. This function interfaces the *trip* OSRM service.

Usage

```
osrmTrip(loc, overview = "simplified")
```

Arguments

`loc`	a SpatialPointsDataFrame of the waypoints, or a data.frame with points as rows and 3 columns: identifier, longitudes and latitudes (WGS84 decimal degrees).
`overview`	"full", "simplified". Add geometry either full (detailed) or simplified according to highest zoom level it could be display on.

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 disconnecte islands) multiple trips for each connected component are returned.

Value

A list of connected components. Each component contains:

trip A SpatialLinesDataFrame (loc's CRS if there is one, WGS84 else) containing a line for each step of the trip.

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

See Also

osrmRoute

Examples

```
## Not run:
# Load data
data("com")

# Get a trip with a id lat lon data.frame
trips <- osrmTrip(loc = com[1101:1150, c(1,3,4)])

# Display the trip
plot(trips[[1]]$trip , col = 1:5)
points(com[1101:1150, 3:4], pch = 20, col = "red", cex = 0.5)
```

```
# Map
if(require("cartography")){
  osm <- getTiles(spdf = trips[[1]]$trip, crop = TRUE, type = "osmgrayscale")
  tilesLayer(osm)
  plot(trips[[1]]$trip, col = 1:5, add = TRUE)
  points(com[1101:1150, 3:4], pch = 20, col = "red", cex = 0.5)
}

# Get a trip with a SpatialPointsDataFrame
trips <- osrmTrip(loc = src)

# Map
if(require("cartography")){
  osm <- getTiles(spdf = trips[[1]]$trip, crop = TRUE, type = "osmgrayscale")
  tilesLayer(osm)
  plot(src, pch = 20, col = "red", cex = 2, add = TRUE)
  plot(trips[[1]]$trip, col = 1:5, add = TRUE, lwd=2)
}

## End(Not run)
```

---

src *SpatialPointsDataFrame of 10 Communes in France And Belgium*

Description

10 communes in France & Belgium. The projection is RGF93 / Lambert-93.

Source

UMS RIATE

Index

com, 2

dst, 2

osrm, 3
osrm-package *(osrm)*, 3
osrmIsochrone, *3*, 3, *7*
osrmRoute, *3*, 5, *8*
osrmTable, *3*, *4*, 6
osrmTrip, *3*, 8

src, 9