How to access OpenStreetMapApi with R

2019-10-01T13:09:13-06:00

Introduction

In this techreport we are going to use the open street maps, exploring various API calls possible with the platform. We do this by

The database of Open Street Maps

Since Google closed its maps API behind a credit card paywall, developers need to migrate to other services. Open Street Maps (OSM) provides some of the functions google maps had for free. In this tech report, we will use the Nominatim API for OpenStreetMap data to help us solve various questions, to get more information about Nominatim see its documentation at the wiki page: https://wiki.openstreetmap.org/wiki/Nominatim. To get an overview of Nominatim makes it able to interrogate its API to retrieve OSM data check: https://nominatim.org/release-docs/develop/api/Overview/. The database of nominatim can also be downloaded be used on servers. Interrogating the API this way, your script can use the data without constraints

```
# Used libraries
library(jsonlite)   # Used for calling web API's and parsin JSON files
library(gepaf)      # Encode Coordinates to Google Polylines, used for routing calls
library(leaflet)
```

```
# First we will define some the URL base for some of the API operations
reverse_url = "https://nominatim.openstreetmap.org/reverse?format=jsonv2&"
search_url = "https://nominatim.openstreetmap.org/search.php?format=jsonv2&q="
routing_url = "http://router.project-osrm.org/route/v1/driving/polyline("
```

Retrieving data from coordinates

In this function, we will interrogate the Nominatim API to retrieve location data according to the longitudinal and latitudinal points given. lon=4.8945 and lat=52.3667, the coordinates for some random place in Amsterdam.

```
CoordinatesToLocation <- function(lon = NULL,lat=NULL) {
  url <- paste0(reverse_url,"&lat=",lat,"&lon=",lon)
  system.time({
    d <- fromJSON(url)
  })
  return(d)
}
CoordinatesToLocation(4.8945, 52.3667)$display_name
```

```
## [1] "McDonald's, , Grachtengordel, Amsterdam, Noord-Holland, Nederland, 1017CK, Nederland"
```

For this tech report, we have only chosen to show the `$display\_name$` attribute from the returned data frame `$d$`. This indicates anything was found while querying the Nominatim API. There resides much more data available of use in the complete frame.

Retrieving data from location

We can also send a search query to OSM, to retrieve location data according to the location information we have. The API will return a data frame with various found places sorted according to importance. We return the first found result which OSM has determined to be determined.

```
LocationToCoordinates <- function(location) {
  url <- paste0(search_url, location)
  system.time({
    d <- fromJSON(url)[1,]
  })
  return (data.frame(as.numeric(as.character(d['lat'])), as.numeric(as.character(d['lon']))))
}
LocationToCoordinates("Amsterdam")
```

```
##   as.numeric.as.character.d..lat.... as.numeric.as.character.d..lon....
## 1                           52.37454                            4.897976
```

Car route between locations

To find the distance between multiple locations we use another API, in this case we use the API call from Open Source Routing Machine (OSRM), another API service working with the data from the OpenStreetMap project. Another library that is used `$gepaf$`, which is an encoder to encode coordinates into Google's Encoded Polyline Format.

```
GetWalkingRoute <- function(encpolied_position) {
  url <- paste0(routing_url,encpolied_position,")")
  system.time({
    d <- fromJSON(url)
  })
  return(d)
}

location_utrecht <- LocationToCoordinates("Utrecht")
location_eindhoven <- LocationToCoordinates("Eindhoven")
coords <- data.frame(lat = c(location_utrecht[1,1], location_eindhoven[1,1]),lon = c(location_utrecht[1
result <- GetWalkingRoute(encodePolyline(coords))
print(result[["routes"]][["distance"]])
```

```
## [1] 88675.9
```

Mapping the route

The result that is returned by the OSRM api call, is able to tell us all the route nodes for finding the the destination the API needs to find.

```
route_df <- decodePolyline(result[["routes"]][["geometry"]])

leaflet(route_df) %>%
  addTiles() %>%
  addProviderTiles(providers$CartoDB.Positron) %>%
  addCircles(lng = ~lon, lat = ~lat, weight = 2, fillOpacity = 0.5,
             radius = 10)
```