Simple Geocoding Example

Mitchell J. Lovett

11/11/2020

R Markdown

We will use the openstreetmaps API for this example. This is a public API. Note you can also use the Google maps API, but that requires more setup and has a licensing aspect to it.

```
#install.packages("tmaptools")
#install.packages("here")
require(tmaptools)
```

```
## Loading required package: tmaptools
```

```
## Warning: package 'tmaptools' was built under R version 3.6.2
```

```
require(here)
```

```
## Loading required package: here
```

```
## here() starts at /Users/mitchlovett/Dropbox/Analytics Design/4. Geolocation Data and Segmentation
```

The task we will focus on is geocoding from a name and reverse coding from a lon-lat location. To get the location information for a point of interest, we need to pass the terms to search for: In this case we will pass TaiChi Bubble Tea, College Town, Rochester

```
place1 = "Taichi Bubble Tea, College Town, Rochester"
location1 = geocode_OSM(place1,details = TRUE, as.data.frame = TRUE)
place2 = "Revolution Karaoke"
location2 = geocode_OSM(place2,details = TRUE, as.data.frame = TRUE)
location1
```

```
##                                       query    lat       lon  lat_min  lat_max
## 1 Taichi Bubble Tea, College Town, Rochester 43.123 -77.61878 43.12295 43.12305
##     lon_min   lon_max place_id osm_type     osm_id place_rank
## 1 -77.61883 -77.61873 68505874     node 5925059847         30
##
## 1 Taichi bubble tea, Celebration Drive, College Town, Rochester, Monroe, New York, 14642, United Sta
##     class       type importance
## 1 amenity restaurant      0.611
##                                                                        icon
## 1 https://nominatim.openstreetmap.org/ui/mapicons//food_restaurant.p.20.png
```

```
location2
```

```
##                query      lat       lon  lat_min  lat_max   lon_min   lon_max
## 1 Revolution Karaoke 43.09082 -77.64241 43.09077 43.09087 -77.64246 -77.64236
##   place_id osm_type     osm_id place_rank
## 1 69078300     node 5924953485         30
##                                                                                display_name
```

```
## 1 Revolution karaoke, Jefferson Road, Mortimer, Henrietta, New York, 14623, United States of America
##     class       type importance
## 1 amenity restaurant      0.201
##                                                                          icon
## 1 https://nominatim.openstreetmap.org/ui/mapicons//food_restaurant.p.20.png
```

This gives us the location of the retail restaurants!

Now we reverse code this using just the lon and lat values.

```
placeReverse = rev_geocode_OSM(x = location1["lon"], y = location1["lat"],as.data.frame = TRUE)[, c("nam
placeReverse
```

```
## [1] "Taichi bubble tea, Celebration Drive, College Town, Rochester, Monroe, New York, 14642, United S
```

```
placeNearBy = rev_geocode_OSM(x = location1["lon"]-.0025, y = location1["lat"],as.data.frame = TRUE)[, c
placeNearBy
```

```
## [1] "University of Rochester Medical Center Campus, Elmwood Avenue, Upper Mount Hope, Rochester, Mon
```

We can recover the Taichi place from its lat and lon, and we can "move" around by adjusting the lat and lon and seeing what we run into. . .

We can also evaluate distances between locations. We shouldn't use simple Euclidean distance and instead use so-called Haversine distance that deals with the spherical nature of the earth.

```
#Haversine distance - measures distance between points
# on "big circle" due to Earth's spherical shape
# see https://en.wikipedia.org/wiki/Haversine_formula
my_dist <- function(long1, lat1, long2, lat2) {
  rad <- pi/180
  a1 <- lat1*rad
  a2 <- long1*rad
  b1 <- lat2*rad
  b2 <- long2*rad
  dlon <- b2 - a2
  dlat <- b1 - a1
  a <- (sin(dlat/2))^2 + cos(a1)*cos(b1)*(sin(dlon/2))^2
  if(sqrt(a)>0 & sqrt(1-a)>0){
    c<- 2*atan(sqrt(a)/sqrt(1 - a))
  } else {
    c <- 2*atan2(sqrt(a), sqrt(1 - a))
  }
  R <- 6371/1.6 # radius in miles around Washington DC
  d <- R*c
  return(d)
}

my_dist(location1["lon"],location1["lat"],location2["lon"],location2["lat"])
```

```
##        lat
## 1 2.537996
```

Get the 20m state boundary files from here: https://www.census.gov/geographies/mapping-files/2013/geo/carto-boundary-file.html

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

```
## -- Conflicts ---------------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(sf)
```

```
## Warning: package 'sf' was built under R version 3.6.2
```

```
## Linking to GEOS 3.7.2, GDAL 2.4.2, PROJ 5.2.0
```

```
usa <- "/Users/mitchlovett/Dropbox/Analytics Design/4. Geolocation Data and Segmentation/cb_2013_us_sta
  st_read()
```

```
## Reading layer `cb_2013_us_state_20m' from data source `/Users/mitchlovett/Dropbox/Analytics Design/4
## Simple feature collection with 52 features and 9 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -179.1473 ymin: 17.88481 xmax: 179.7785 ymax: 71.35256
## CRS:            4269
```

```
(usa_48 <- usa %>%
  filter(!(NAME %in% c("Alaska", "District of Columbia", "Hawaii", "Puerto Rico"))))
```

```
## Simple feature collection with 48 features and 9 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -124.7332 ymin: 24.5447 xmax: -66.9499 ymax: 49.38436
## CRS:            4269
## First 10 features:
##    STATEFP  STATENS    AFFGEOID GEOID STUSPS        NAME LSAD        ALAND
## 1       01 01779775 0400000US01    01     AL     Alabama   00 131172434095
## 2       05 00068085 0400000US05    05     AR    Arkansas   00 134772954601
## 3       06 01779778 0400000US06    06     CA  California   00 403482685922
## 4       09 01779780 0400000US09    09     CT Connecticut   00  12541965607
## 5       12 00294478 0400000US12    12     FL     Florida   00 138897453172
## 6       13 01705317 0400000US13    13     GA     Georgia   00 148962779995
## 7       16 01779783 0400000US16    16     ID       Idaho   00 214045724209
## 8       17 01779784 0400000US17    17     IL    Illinois   00 143793994610
## 9       18 00448508 0400000US18    18     IN     Indiana   00  92789545929
## 10      20 00481813 0400000US20    20     KS      Kansas   00 211752860834
##         AWATER                       geometry
## 1   4594920201 MULTIPOLYGON (((-88.31002 3...
## 2   2958815561 MULTIPOLYGON (((-94.61792 3...
## 3  20484304865 MULTIPOLYGON (((-118.6034 3...
## 4   1815409624 MULTIPOLYGON (((-73.69595 4...
## 5  31413676956 MULTIPOLYGON (((-80.6602 24...
## 6   4947803555 MULTIPOLYGON (((-85.60516 3...
## 7   2397731902 MULTIPOLYGON (((-117.2151 4...
## 8   6201680290 MULTIPOLYGON (((-91.51033 4...
## 9   1536677621 MULTIPOLYGON (((-88.09496 3...
## 10  1346718440 MULTIPOLYGON (((-102.0517 4...
```

```r
load("/Users/mitchlovett/Box Sync/Mobile Case Data - Original/Simulated Data/MobileVisits.Rdata")

ggplot(data = usa_48) +
  geom_sf() +
  geom_point(data = venues, aes(x = longitude, y = latitude,color = factor(chain)), alpha = .2) +
  coord_sf(xlim = c(-130, -60),
           ylim = c(20, 50))
```