GEO_Code - GPS Trajectory Data Example

J. Dayton

5/13/2020

Source of data and the following data description: https://archive.ics.uci.edu/ml/datasets/GPS+ Trajectories.

Abstract: The dataset has been feed by Android app called Go!Track. It is available at Goolge Play Store.

STEP 0: Load Libraries

STEP 1: Import relevant data set.

Get the GPS Activity Data

Source of data and the following data description: https://archive.ics.uci.edu/ml/datasets/GPS+ Trajectories.

Data Set Information:

The dataset is composed by two tables. The first table go_track_tracks presents general attributes and each instance has one trajectory that is represented by the table go_track_trackspoints.

Attribute Information:

(1) go_track_tracks.csv: a list of trajectories

- id_android - it represents the device used to capture the instance;
- speed - it represents the average speed (Km/H)
- distance - it represent the total distance (Km)
- rating - it is an evaluation parameter. Evaluation the traffic is a way to verify the volunteers perception about the traffic during the travel, in other words, if volunteers move to some place and face traffic jam, maybe they will evaluate 'bad'. (3- good, 2- normal, 1-bad).
- rating_bus - it is other evaluation parameter. (1 - The amount of people inside the bus is little, 2 - The bus is not crowded, 3- The bus is crowded.
- rating_weather - it is another evaluation parameter. (2- sunny, 1- raining).
- car_or_bus - (1 - car, 2-bus)
- linha - information about the bus that does the pathway

(2) go_track_trackspoints.csv: localization points of each trajectory

- id: unique key to identify each point
- latitude: latitude from where the point is
- longitude: longitude from where the point is
- track_id: identify the trajectory which the point belong

- time: datetime when the point was collected (GMT-3)

```
##Load the data
dfTracks <- read.csv("./GPS_Trajectory/go_track_tracks.csv",
                     stringsAsFactors = FALSE)
dfPoints <- read.csv("./GPS_Trajectory/go_track_trackspoints.csv",
                     stringsAsFactors = FALSE)
```

Clean Data

Check for NA values and richness of 'tracks'.

```
kable(head(dfTracks))
```

id	id_android	speed	time	distance	rating	rating_bus	rating_weather	car_or_bus	linha
1	0	19.210586	0.1380489	2.652	3	0	0	1	
2	0	30.848229	0.1714847	5.290	3	0	0	1	
3	1	13.560101	0.0676986	0.918	3	0	0	2	
4	1	19.766679	0.3895444	7.700	3	0	0	2	
8	0	25.807401	0.1548006	3.995	2	0	0	1	
10	2	1.346913	0.0066819	0.009	2	0	0	1	

```
kable(head(dfPoints))
```

id	latitude	longitude	track_id	time
1	-10.93934	-37.06274	1	2014-09-13 07:24:32
2	-10.93934	-37.06274	1	2014-09-13 07:24:37
3	-10.93932	-37.06276	1	2014-09-13 07:24:42
4	-10.93921	-37.06284	1	2014-09-13 07:24:47
5	-10.93894	-37.06288	1	2014-09-13 07:24:53
6	-10.93854	-37.06284	1	2014-09-13 07:24:59

```
#Rename 'id' to 'index' for dfPoints to elleviate confusion, for this file we want 'track_id'
dfPoints <- dfPoints %>% rename(index = id, id = track_id)

#order the two data frames and check to ensure that the respective ID's align
dfPoints <- dfPoints %>% arrange(id)
dfTracks <- dfTracks %>% arrange(id)

#Check which id's will not align, and what index they are for diagnosis
which(!(unique(dfTracks$id) == unique(dfPoints$id)))
```

```
## integer(0)
```

```
sum(!(unique(dfTracks$id) == unique(dfPoints$id)))
```

```
## [1] 0
```

```
# unique(dfTracks$id)
# unique(dfPoints$track_id)
# summary(dfTracks)
# summary(dfPoints)
```

STEP 2. Aggregate data by country (or other location).

With the respective IDs aligned between two df's, join the two together

```
#Join based on 'id'
df <- left_join(dfPoints, dfTracks, by = "id")
rm(dfPoints); rm(dfTracks)

df <- df %>%
   rename(user = id_android, time = time.x, lat = latitude, lon = longitude) %>%
   select(id, user, lat, lon, time)
#Fix the time variable
df$time <- ymd_hms(df$time)
#Fix user from 0-27, to 1-28 and make catagorical (factor)
df$user <- as.factor(df$user + 1)

dim(df)
```

```
## [1] 18107     5
```

```
kable(head(df))
```

id	user	lat	lon	time
1	1	-10.93934	-37.06274	2014-09-13 07:24:32
1	1	-10.93934	-37.06274	2014-09-13 07:24:37
1	1	-10.93932	-37.06276	2014-09-13 07:24:42
1	1	-10.93921	-37.06284	2014-09-13 07:24:47
1	1	-10.93894	-37.06288	2014-09-13 07:24:53
1	1	-10.93854	-37.06284	2014-09-13 07:24:59

STEP 3. Merge data from data set to map data.

STEP 4. Create the plot(s).