Data processing

Don Li

12/06/2020

```r
library( data.table )
library( reticulate )
source( "G:/azure_hackathon/data/Don2/data_processing_functions.R" )
```

Data processing steps

- Read the `.parquet` files. Write out to .RData
- Read the .RData and combine into dataframe
- Remove variables `bearing` and `accuracy`
- Convert `pingtimestamp`
- Add `weekday`, `hour`, and `weekend`. Add `rush_hour`
- Add distances between GPS pings.
- Filter trips with large jumps between GPS pings.

Reding `.parquet`

Read the `.parquet` files, turn then into an `R` dataframe and save them as an `.RData` file.

```r
source_python( "G:/azure_hackathon/data/Don2/read_parquet_function.py" )

parquet_files = list.files( path = "G:/azure_hackathon/datasets2/parquet",
    pattern = ".parquet", full.names = T )
parquet_to_RData( parquet_files,
    "G:/azure_hackathon/datasets2/part_files/SNG_part_" )
```

Combine data frames

Read the `.RData` part files and combine them into a big dataframe.

```r
rdata_files = list.files( path = "G:/azure_hackathon/datasets2/part_files/",
    pattern = "SNG_part", full.names = T )
all_data = combine_RData_part_files( rdata_files )

dataset_convert_numeric( all_data, "trj_id" )

# Reorder variables
setorder( all_data, trj_id, pingtimestamp )
```

Remove variables

Remove bearing and accuracy because we probably won't need to use them for anything.

```
vars_ = c("bearing", "accuracy", "driving_mode")
dataset_remove_vars( all_data, vars_ )
```

Fill in time variables

Get pingtimestamp, convert to date format. Add weekday, hour, and weekend. Add rush_hour.

```
dataset_time_vars( all_data )
# Remove pingtimestamp
vars_ = c("pingtimestamp")
dataset_remove_vars( all_data, vars_ )
# Add rush_hour
dataset_add_rush_hour( all_data )
```

Add distances

Add path Haversine and reversed Haversine distance. The reversed Haversine distance swaps longitude for latitude.

```
source( "G:/azure_hackathon/data/Don2/distance_functions.R" )

dataset_add_distances( all_data )
head( all_data )
```

Add pointwise time differences

```
dataset_add_timediffs( all_data )
```

```
save( all_data,
    file = "G:/azure_hackathon/datasets2/data_processing/all_data1.RData" )
```

Data cleaning

There are trips where there is a sudden jump in the GPS location and then it snaps back on the route. Make a filter to remove this. Previously, we interpolated these jumps, but when there are 4-5-6-7 pings at random locations, it gets hard. So, just a better use of time to remove them from the trip.

Algorithm for detecting jumps

A jump in the GPS is represented by a large distance between two points, x1 and x2. If the jump only happens for one GPS sample, then we will see another large distance between x2 and x3. If there are two GPS samples out of place, then the large distance will be between x3 and x4.

We just make a filter based on the gap that we want to find the jumps for. Then we exclude the bad ones.

```r
two_window_threshold = function( x, threshold, gap = 1 ){
    (x > threshold) & (c( x[-(1:gap)], rep(F,gap) ) > threshold)
}

# Tag trips with jumps smaller than 1km. These are good trips.
good_trip_index = all_data[ , all( H_dist < 1 ), by = "trj_id" ]
good_trip_id = good_trip_index[V1==TRUE,trj_id]
bad_trip_id = good_trip_index[V1==FALSE,trj_id]
# Check the other trips based on the window filter.
good_trips = all_data[ trj_id %in% good_trip_id ]
bad_trips = all_data[ trj_id %in% bad_trip_id ]

# Tag GPS samples to remove
bad_trips[ , gap_rm := F ]

gap_seq = seq( 1, 120 )
for( gap in gap_seq ){
    cat( "Gap:", gap, "\n" )

    # Find trips with big GPS gaps
    any_gaps = bad_trips[ , which(two_window_threshold( H_dist, 2, gap )) , by = "trj_id" ]
    gap_trips = unique(any_gaps$trj_id)
    cat( gap_trips, "\n" )

    # Tag GPS samples
    bad_trips[ trj_id %in% gap_trips, gap_rm := {
        missing_dist = which( two_window_threshold( H_dist, 2, gap ) )
        for ( missing in missing_dist ){
            neighbours = missing + 0:(gap-1)
            gap_rm[ neighbours ] = T
        }
        gap_rm
    }, by = "trj_id" ]

    # Remove the bad GPS samples
    bad_trips = bad_trips[ gap_rm == FALSE ]

    # Compute the distances using the new points
    bad_trips[ trj_id %in% gap_trips, c("H_dist", "H_dist2") := {
        h_dist = haversine( rawlat, rawlng )
        h1 = c( 0, h_dist  )
        h_dist2 = haversine( rawlng, rawlat )
        h2 = c( 0, h_dist2 )
        list( h1, h2 )
    }, by = "trj_id" ]
}

# Check distances and make sure they are consistent
temp_summary = bad_trips[ , {
    path_dist = sum( H_dist )
    crow_dist = haversine( rawlat[c(1,.N)], rawlng[c(1,.N)] )
    list( path_dist = path_dist, crow_dist = crow_dist )
}, by = "trj_id" ]
```

```r
temp_summary[ , summary(path_dist - crow_dist) ]

# Combine datasets
bad_trips[ , gap_rm := NULL ]
all_data = rbindlist( list( good_trips, bad_trips ), use.names = T )
setorder( all_data, trj_id, date_ )

# Compute the distances and the times again to make sure

dataset_add_distances( all_data )

dataset_add_timediffs( all_data )
```

Missing speed data

```r
all_data[ osname == "android" & speed == 0, speed := NaN ]
all_data[ speed < 0, speed := NaN ]

save( all_data,
    file = "G:/azure_hackathon/datasets2/data_processing/all_data2_jumpclean.RData" )
```