Trip summaries

Don Li

10/06/2020

```
source( "G:/azure_hackathon/data/Don/utility.R" )
```

Summarise trip trajectories

List of covariates:

- `crow_dist`: Distance as the crow flies (Haversine, km). Numeric.
- `path_dist`: Path distance (Haversine, km). Numeric.
- `path_dist2`: Path distance; longitude and latitude reversed (Haversine, km). Numeric.
- `timediff`: Observed arrival time. Numeric.
- `start_x`, `start_y`: Journey start longitudes and latitudes. Numeric.
- `end_x`, `end_y`: Journey end longitudes and latitudes. Numeric.
- `weekday`, `hour`: Day and hour that the trip started. Factor; Numeric.
- `rush_hour`: Whether trip started during rush hour. Factor.
- `N`: Number of GPS samples. Numeric.
- `mean_speed`, `var_speed`: Mean and variance of speed. Numeric.
- `sampling_rate`: GPS sampling rate. Numeric.

Covariates to be joined later:

- `azure_dist`, `azure_eta`: Path distance and ETA from Azure Maps.
- `OSRM_dist`, `OSRM_duration`: Path distance and ETA from OSRM.

```
load( "G:/azure_hackathon/dataset/all_SNG_2_jump_adjusted.RData" )
trip_summary = sumamrise_trips( all_data )
head(trip_summary)
```

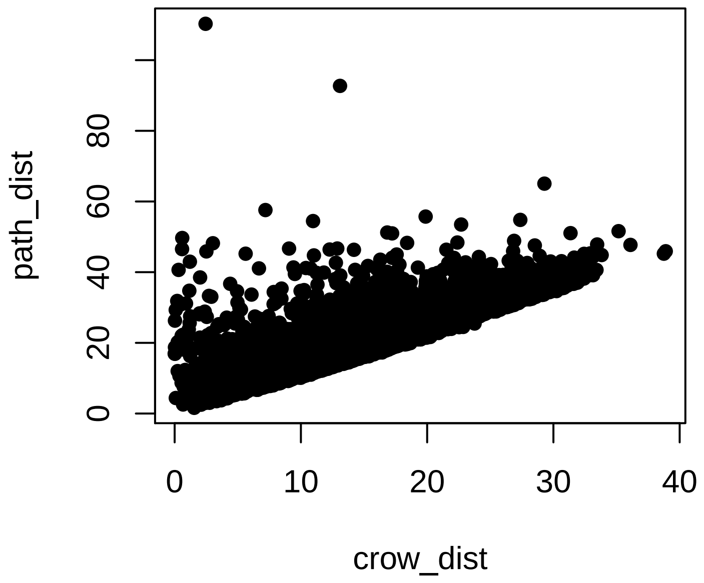
Combine external distances/times from OSRM and Azure Maps

```
load( "G:/azure_hackathon/dataset/Azure_part.RData" )
Azure_trip_summary = trip_summary
load( "G:/azure_hackathon/dataset/OSRM.RData" )
OSRM_trip_summary = trip_summary
OSRM_trip_summary[ , c("OSRM_dist", "OSRM_duration") := {
    list(OSMR_dist, OSMR_duration)
} ]
OSRM_trip_summary[ , c("OSMR_dist", "OSMR_duration") := NULL ]

join_external_distances( trip_summary, Azure_trip_summary, OSRM_trip_summary )

save( trip_summary, file = "G:/azure_hackathon/dataset/trip_summary1.RData" )
```

Trim loopy trip outliers

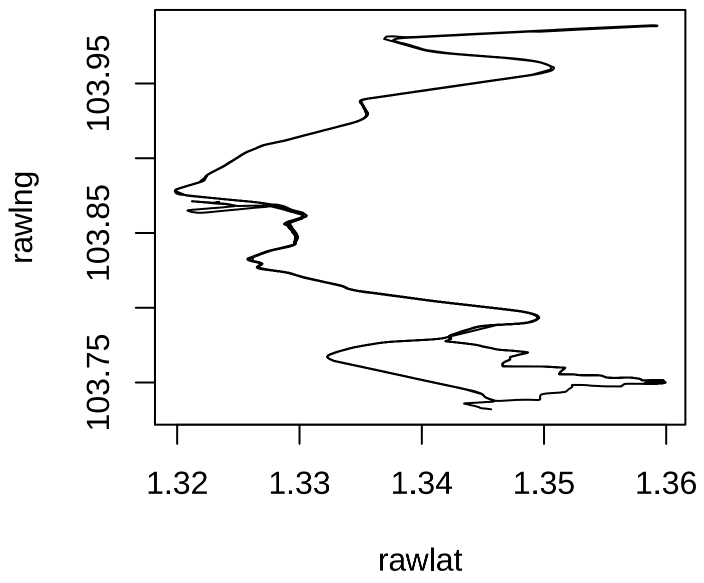
A trip where they just loop around the city. Obviously, we cannot predict the ETA of these kinds of trips using only the origin and the destination. There are some trips with very long path distances but short crow distances (top left).

```
load( "G:/azure_hackathon/dataset/trip_summary1.RData" )
load( "G:/azure_hackathon/dataset/all_SNG_2_jump_adjusted.RData" )

trip_summary[ , {
    plot( crow_dist, path_dist, pch = 16 )
} ]
```



```
## NULL
```

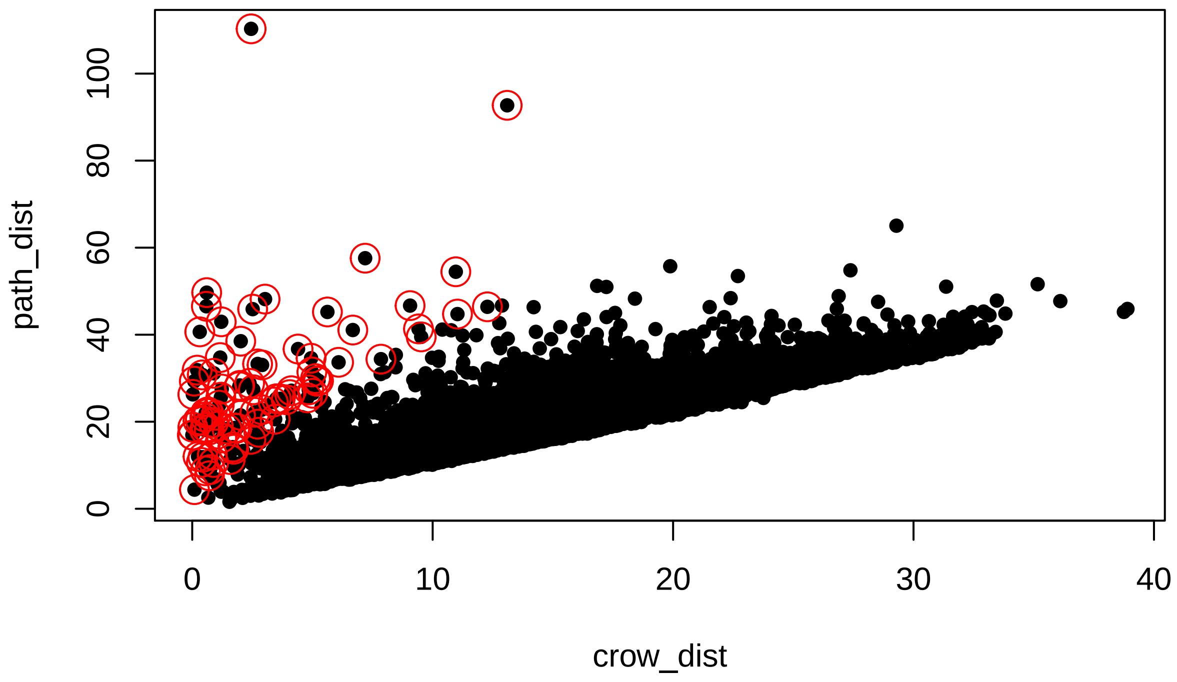
```
loopiest_trip = trip_summary[ which.max(path_dist - crow_dist) ]
all_data[ trj_id == loopiest_trip$trj_id, {
    plot( rawlat, rawlng, type = "l" )
    } ]
```



```
## NULL
```

It is somewhat hard to decide what constitutes loopiness. For example, trips that have a long crow distance are also more likely to have a long path distance. If the crow distance is short and the path distance is long, they are more likely to be doing a big loop. To detect these, we will use regression on the log-log scale and remove based on large positive residuals.

```
trip_summary[ ,{
    plot( crow_dist, path_dist, pch = 16 )
    loglog_reg = lm( log(path_dist) ~ log(crow_dist), .SD )
    remove_ = loglog_reg$residual > 1
    points( crow_dist[remove_], path_dist[remove_], cex = 2, col = "red")
    loopy_trips <<- remove_
    NULL
} ]
```



```
## NULL
```

```
trip_summary = trip_summary[ !loopy_trips ]
save( trip_summary, file =  "G:/azure_hackathon/dataset/trip_summary2_loopytrim.RData" )
```