Anomaly detection

Jia Liu

December 22, 2014

```
# load necessary libraries
library(RSQLite)
library(dplyr)
# for plotting
library(ggplot2)
library(gridExtra)
library(GGally)
```

Abstract:

Let's investigate if the tentative features have a Gaussian shape, or having a Gaussian shape after a simple transformation. I will use the trips of driver to demonstrate this.

```
# connect to database using package 'dplyr'
driver_db = src_sqlite(file.path('drivers_dbs', paste0(driver_id_now, '.db')))
# link to data frame
driver_sqlite=tbl(driver_db, "driver_data")
# extract coordinates
coord = collect(select(driver_sqlite, x, y, trip_id))
# show some data
head(coord)
```

```
## Source: local data frame [6 x 3]
##
##     x   y trip_id
## 1 0.0 0.0       1
## 2 0.9 0.9       1
## 3 1.6 1.9       1
## 4 2.2 3.0       1
## 5 3.2 4.1       1
## 6 4.0 5.6       1
```

Possible Features

Now we think about possible features of a dangerous driver, and then quantify them use the data we have:

1. High speed/acceleration
2. High speed or acceleration when turning
3. Long drive

4. Drive at night
5. Hard breaks
6. Drive in highway

We can quantify them by:

1. Mean speed/acceleration
2. High angular speed/acceleration
3. Total trip distance
4. No way to know...
5. Total counts of hard breaks per 10 miles. Progressive has a definition for hard break
6. Total time when speed is larger than 60mph

The speed can be found by calculating numeric derivative: subtracting two consective coordinates and divided by 1s. Acceleration is calculated by implementing numerical derivative one step further. But in the post of Kaggle forum, the coordinate would jump at times, which makes the numeric derivative infinite. In order to prevent the large influence brought by these artificial large speed/acceleration, I will smooth it by substituting any speed larger than 300 miles/hour (134.112m/s) by the mean of the speed of the trip. In the function collection script `auxFunctions_AD.R`, I write a generic function which can replace the "anomaly" high value by the group mean in one group.

Find features

1. Speed The first step is to transform the Cartisan coordinates (x,y) to polar coordinate, preparing to calculate speed and acceleration.

```
# find the end point of each group
temp = coord$trip_id
temp = temp[-1]
temp[length(temp)+1]=201
# transform and create group end indicator
coord = mutate(coord, r=sqrt(x**2+y**2), theta=atan2(y,x), group_end = temp-trip_id)
rm(temp)
```

Now we calculate the speed by using the numerical derivative:

$$v = \frac{r(t+\Delta t) - r(t)}{\Delta t}$$

where Δt=1s because the time step between two GPS records is 1s. Following is the code for calculating speed.

```
# position of next time step
r_after = coord$r[-1]
r_after[nrow(coord)]=0
coord = mutate(coord, speed = r_after-r)
# fix the end point of each group
coord[which(coord$group_end==1),"speed"]=0
```

In US, most of the car cannot has a speed larger than 300 miles/hour. Any speed beyond that could be GPS error. So I set the threshold for infinity of speed to be 300 miles/hour, or 134 m/s.

```
source("auxFunctions_AD.R")
coord = fixInfinityByMean(coord, "trip_id", "speed", 134)
```

Good news! GPS works fine for this driver's trips, no hyperspace jump!

2. Acceleration In a similar way as calculating speed, we calculate the acceleration.

$$a = \frac{v(t+\Delta t) - v(t)}{\Delta t}$$

```
# position of next time step
v_after = coord$speed[-1]
v_after[nrow(coord)]=0
coord = mutate(coord, acc = v_after-speed)
# fix the end point of each group
coord[which(coord$group_end==1),"acc"]=0
```

We also want to implement an infinity check for acceleration. Suppose the maximum acceleration is 4s for 0-100mph ([ref])(http://www.zeroto60times.com/F1-0-60.html), i.e. $11m/s^2$.

```
coord = fixInfinityByMean(coord, "trip_id", "acc", 11)
```

3. Angular speed and angular acceleration In the similar steps of finding and fixing infinity of linear speed/acceleration, we calculate angular speed/acceleration, with the following assumption: - Any car **CANNOT** make a U turn in 1 second. While I do not find such a limit for angular acceleration.

```
# position of next time step
theta_after = coord$theta[-1]
theta_after[nrow(coord)]=0
coord = mutate(coord, angular_speed = theta_after-theta)
# fix the end point of each group
coord[which(coord$group_end==1),"angular_speed"]=0
# fix infinity: any car cannot make a U turn in 1 second
coord = fixInfinityByMean(coord, "trip_id", "angular_speed", pi)
```

```
# position of next time step
omega_after = coord$angular_speed[-1]
omega_after[nrow(coord)]=0
coord = mutate(coord, angular_acc = omega_after-angular_speed)
# fix the end point of each group
coord[which(coord$group_end==1),"angular_acc"]=0
```

4. Counts of hard breaks Insurance company *Progressive* defines hard break as "any decerease in speed over 7 miles per second", i.e. smaller than -3.13m/s^2. We create an indicator feature `hard_break` to flag out the acceleration smaller than this number for all the times.

```
coord$hard_break = (coord$acc< -3.13)
```

5. Time spent on highway Most of highway I drove before sets the speed limit to be equal or larger than 65 mph. Counting how many seconds when speed is larger than 65 mph gives the time spent on highway. I relax this speed limit to 60 for a little flexibility.

```
coord$on_highway = coord$speed>26.8
```

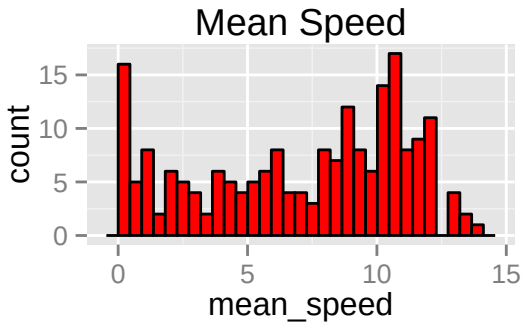
Summarize features in each group

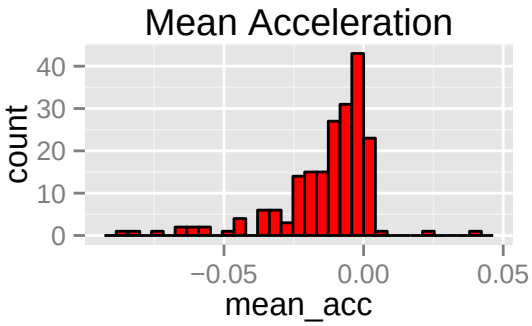
We will use mean speed, mean acceleration, mean angular speed, mean angular acceleration, trip length and total number of hard breaks per 10 miles to represent the signatures of each trip:

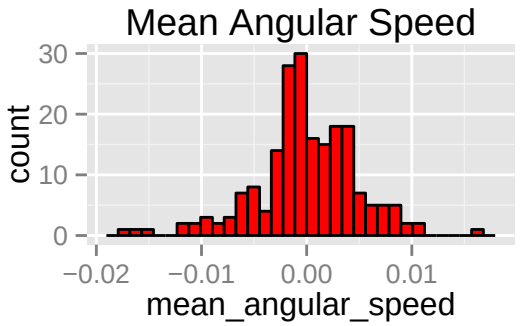
```
trip_signature = coord %>% group_by(trip_id) %>%
  summarise(mean_speed = mean(speed), mean_acc = mean(acc),
            mean_angular_speed=mean(angular_speed), mean_angular_acc=mean(angular_acc),
            trip_length= sum(speed), hardBreaks_10M= sum(hard_break)/trip_length*10,
            highway_time =sum(on_highway))
```

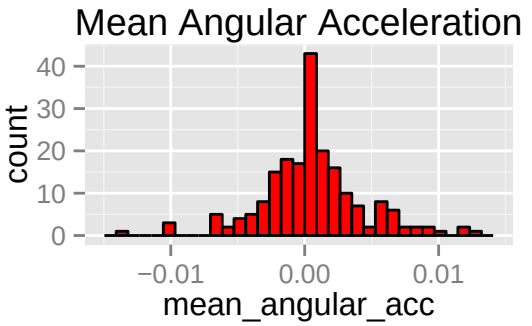
Check the distribution of each feature

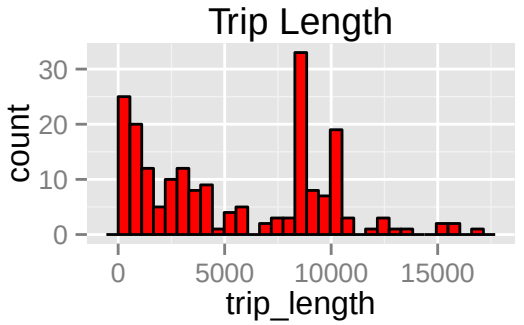
In order to implement anomaly detection, we need to see if each feature is normally distributed. Firstly we plot out the histograms of each feature:

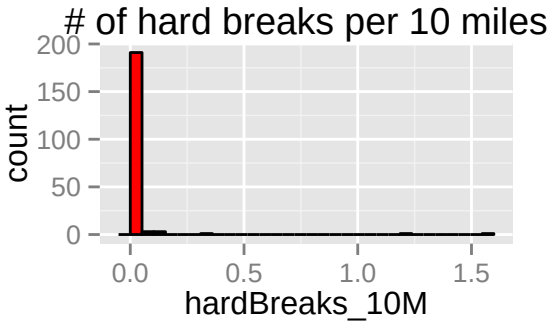


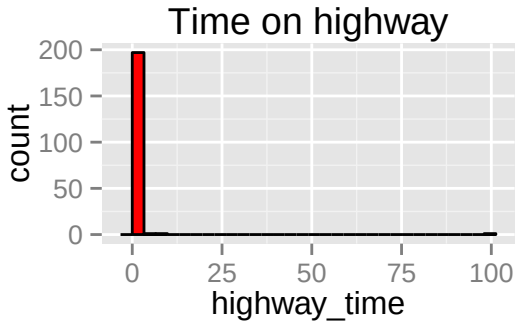




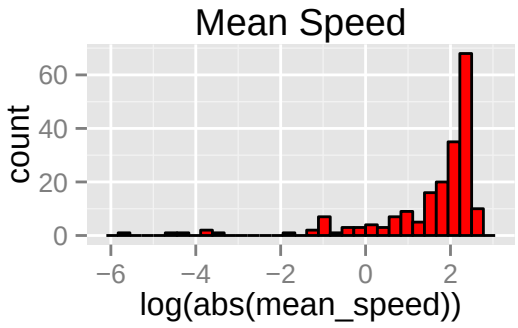


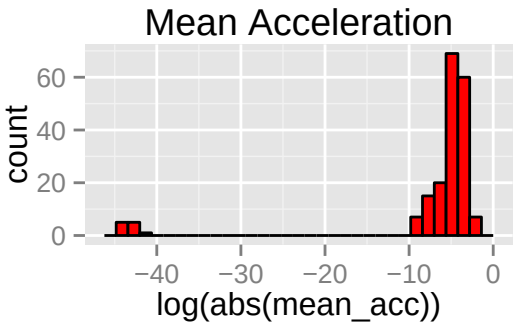


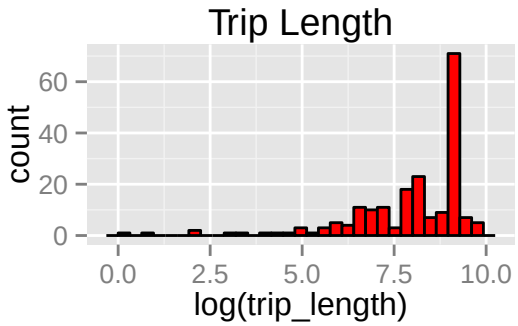


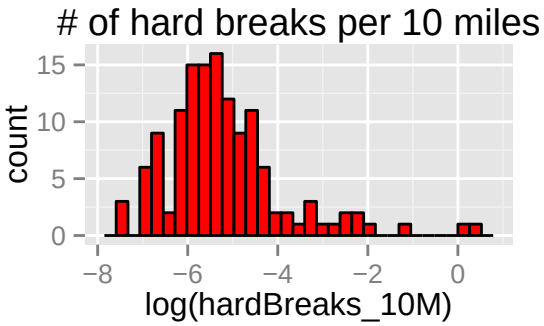


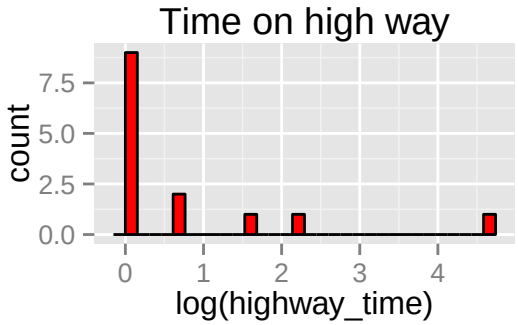
As shown above, mean speed, mean acceleration, trip length, number of hard breaks per 10 miles, and time on highway are not normally distributed. However these normally quantities can be distributed in logarithmic scale, let's see:









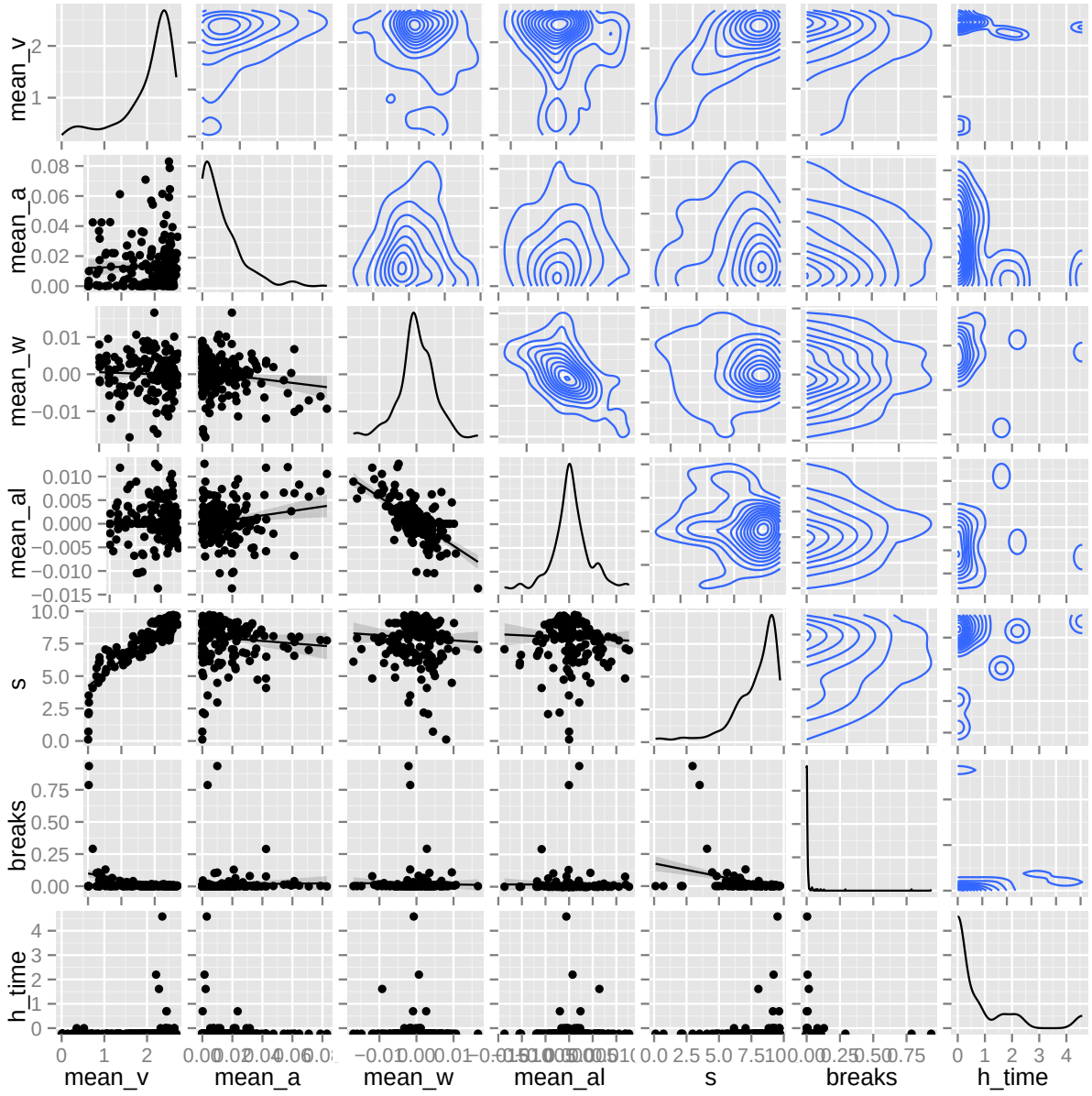


Now they look like normally distributed, and we can identify some outliers! These outlier may belong to the trips which are not made by the current driver.

Finally, we do a scatter matrix plot to visualize distribution of each 2 features.

```
trip_signature_scaled = transmute(trip_signature, trip_id=trip_id,
                                  mean_v=log(abs(mean_speed)+1),
                                  mean_a=log(abs(mean_acc)+1),
                                  mean_w=mean_angular_speed,
                                  mean_al=mean_angular_acc,
                                  s = log(trip_length),
                                  breaks=log(hardBreaks_10M+1),
                                  h_time = log(highway_time))
matrix_plot=ggpairs(trip_signature_scaled, columns=2:8,
             upper = list(continuous="density"),
             lower = list(continuous="smooth"),
             diag  = list(continuous="density", discrete="bar"),
        axisLabels='show')
```

```
print(matrix_plot)
```



The density plots show the outliers clearly, especially for feature time in highway.