STAT290 Project Telematics

Niraj Juneja, Shiro Miyao

March 18, 2015

The goal of this project is to create an R package that can visualize and feature engineer telematics data for a set of car drivers. The package also provides the ability to create a predictive model that can uniquely identify a driver based on driving data.

```
> library(niraj9.telematics)

> # a sample trip that contains the x-y coordinates of a certain
> # trip from a driver. Each row is interval of 1 second.
>
> str(sampleTrip)

'data.frame':	863 obs. of  2 variables:
 $ x: num  0 18.6 36.1 53.7 70.1 ...
 $ y: num  0 -11.1 -21.9 -32.6 -42.8 -52.6 -62.3 -71.6 -80.4 -88.7 ...

> head(sampleTrip) # example of the dataset of a trip

     x     y
1  0.0   0.0
2 18.6 -11.1
3 36.1 -21.9
4 53.7 -32.6
5 70.1 -42.8
6 86.5 -52.6
```

1. Create features based on X-Y coordinates

Based on the X-Y coordinates create features for velocity, acceleration, breaks, and fast turns and speed limit breaks. Note that all the features listed are considered important for getting to a unique driver signature.

1(a) velocity10

Calculate velocity based on moving average of 10 second intervals using trip data.

```
> velocity10  # Function of velocity10
```

```
function (trip)
{
    x = diff(trip$x, 8, 1)
    y = diff(trip$y, 8, 1)
    x = MovingAverage(xv = x, n = 8, centered = TRUE)
    y = MovingAverage(xv = y, n = 8, centered = TRUE)
    vel = 3.6 * sqrt(x^2 + y^2)/8
    return(vel)
}
<environment: namespace:niraj9.telematics>
```

```
> sampleVelocity10 <- velocity10(sampleTrip) #sampleTrip dataset
> head(sampleVelocity10)
```

```
[1] 64.48817 63.12732 61.67922 60.24410 57.58293 54.94462
```

1(b) velocity4

Calculate velocity based on moving average of 4 second interval using trip data.

```
> velocity4 # Function of velocity4
```

```
function (trip)
{
    x = diff(trip$x, 4, 1)
    y = diff(trip$y, 4, 1)
    x = MovingAverage(xv = x, n = 4, centered = TRUE)
    y = MovingAverage(xv = y, n = 4, centered = TRUE)
    vel = 3.6 * sqrt(x^2 + y^2)/4
    return(vel)
}
<environment: namespace:niraj9.telematics>
```

```
> sampleVelocity4 <- velocity4(sampleTrip) #sampleTrip dataset
> head(sampleVelocity4)
```

```
[1] 71.62593 70.44811 68.11024 65.76801 63.61646 61.19181
```

1(c) acceleration

Calculate acceleration based on moving average of 10 seconds using trip data.

```
> acceleration # Function of acceleration
```

```
function (trip)
{
    velocities = velocity10(trip)
    acc = diff(velocities, 15)
    return(acc)
}
<environment: namespace:niraj9.telematics>
```

```
> sampleAcceleration <- acceleration(sampleTrip) #sampleTrip dataset
> head(sampleAcceleration)
```

```
[1] -20.226719 -18.057450 -15.689448 -13.352892  -9.751113  -6.489604
```

1(d) breaks

Calculate breaks based on moving average of 4 seconds using trip data.

```
> breaks # Function of breaks

function (trip)
{
    velocities = velocity4(trip)
    br = diff(velocities, 4)
    return(br)
}
<environment: namespace:niraj9.telematics>

> sampleBreaks <- breaks(sampleTrip) #sampleTrip dataset
> head(sampleBreaks)

[1]  -8.009477  -9.256297  -9.619776 -10.240916 -11.962867 -13.043492
```

1(e) MovingAverage

Define the moving average.

```
> MovingAverage # Function of MovingAverage

function (xv, n = 1, centered = FALSE)
{
    if (centered) {
        before <- floor((n - 1)/2)
        after <- ceiling((n - 1)/2)
    }
    else {
        before <- n - 1
        after <- 0
    }
    s <- rep(0, length(xv))
    count <- rep(0, length(xv))
    new <- xv
    count <- count + (!is.na(new))
    new[is.na(new)] <- 0
    s <- s + new
    i <- 1
    while (i <= before) {
        new <- c(rep(NA, i), xv[1:(length(xv) - i)])
        count <- count + (!is.na(new))
        new[is.na(new)] <- 0
        s <- s + new
        i <- i + 1
    }
    i <- 1
    while (i <= after) {
        new <- c(xv[(i + 1):length(xv)], rep(NA, i))
        count <- count + (!is.na(new))
        new[is.na(new)] <- 0
        s <- s + new
        i <- i + 1
    }
    zz = s/count
    return(zz)
}
<environment: namespace:niraj9.telematics>
```

```
> sampleMvAvg <- MovingAverage(sampleTrip) #sampleTrip dataset
> head(sampleMvAvg)
```

```
     x     y
1  0.0   0.0
2 18.6 -11.1
3 36.1 -21.9
4 53.7 -32.6
5 70.1 -42.8
6 86.5 -52.6
```

2. Visualization of each driver trip

Provide functions that can visualize each driver trip.

2(a) Plot X-Y coordinate movements for each trip

```
> PlotggCoordinates(trip=sampleTrip)
```

2(b) Plot Velocity of the driver with time

```
> PlotggVelocity(trip=sampleTrip)
```

Based on the different colors in the plot which is segmented by the difference in levels of velocity,we created a plot to distinguish where the drivers possibly might be.

For high velocities above 75km/h which is plotted in red, we plot it as on a freeway. The blue plots with a range of more than 15km/h and less than 75km/h will most probably be on the street while the green which is <15km/h will most probably be slowing down or stopping. Please note that it is based on a moving average of 10 seconds.

2(c) Plot acceleration of the driver with time

```
> PlotggAcceleration(trip=sampleTrip)
```

Based on the different colors in the plot which is segmented by the difference in levels of acceleration, we created a plot to see how often the driver is doing high acceleration where we defined high acceleration to be above 15 km/h2. Please note that it is based on a moving average of 10 seconds.

2(d) Plot breaks of the driver with time

```
> PlotggBreaks(trip=sampleTrip)
```

The above plot is similar to plotggAcceleration but this plot is for the deceleration. Based on the different colors in the plot which is segmented by the difference in levels of deceleration, we created a plot to see how often the driver is doing braking where we defined high levels to be below -8 km/h2. Please note that it is based on a moving average of 4 seconds.

2(e) Plot quantile charts for velocity, acceleration and breaks

```
> PlotggTripNow(trip=sampleTrip)
```

The above plot is a combination of the above 4 plots. (Trip movements, velocities, acccceleration and breaks) On one glance, we can visualize how the trip was and what kind of driving was done during that trip.

3. Master Summary file

Create a Mastersummary.csv file that summarizes data from all the trips based on the features created.

```
> # Example,
> #
> # telematics <- Telematics$new()
> # telematics$createMasterSummary(
> #  master.file = "../data/master.csv",
> #  file.list = "../data/fileList.csv" ,
> #  root.folder = "../data/drivers",
> #  subset.driver.count = 10 ,
> #  cores =2)

> masterPath1 <- system.file("extdata", "master.csv", package="niraj9.telematics")
> master <- read.csv(masterPath1)
> str(master)
```

```
'data.frame':	6000 obs. of  66 variables:
 $ file  : Factor w/ 6000 levels "/home/niraj9/public/drivers/1080/1.csv",..: 801 912 924
 $ time  : int  233 465 353 438 187 275 655 369 1131 650 ...
 $ dis   : num  519.72 4143.29 1588.86 1913.1 8.96 ...
 $ vecDis: num  376.9 2759.8 848.1 878.3 6.4 ...
 $ v1    : num  0.006 0 0.082 0.034 0.062 0.09 0.058 0.139 0.054 0.034 ...
 $ v2    : num  0.023 0.017 0.141 0.051 0.073 0.317 0.078 0.296 0.104 0.112 ...
 $ v3    : num  0.028 0.034 0.199 0.091 0.079 ...
 $ v4    : num  0.034 0.195 0.251 0.138 0.09 ...
 $ v5    : num  0.041 0.47 0.321 0.183 0.102 ...
 $ v6    : num  0.051 0.971 0.368 0.232 0.111 ...
 $ v7    : num  0.059 2.264 0.47 0.283 0.122 ...
 $ v8    : num  0.072 9.268 1.235 0.753 0.125 ...
 $ v9    : num  0.091 22.108 3.789 2.484 0.133 ...
 $ v10   : num  0.139 28.218 6.238 7.408 0.139 ...
 $ v11   : num  0.225 31.974 10.781 15.241 0.148 ...
 $ v12   : num  3.358 36.117 14.357 19.466 0.161 ...
 $ v13   : num  7.357 45.672 19.747 23.979 0.168 ...
 $ v14   : num  8.881 53.47 27.617 25.665 0.174 ...
 $ v15   : num  12.843 58.221 35.263 27.176 0.186 ...
 $ v16   : num  16.23 64.812 38.483 30.069 0.201 ...
 $ v17   : num  20.572 69.291 40.309 35.813 0.207 ...
 $ v18   : num  28.43 79.448 45.101 45.044 0.222 ...
 $ v19   : num  36.76 90.82 48.51 48.53 0.24 ...
 $ v20   : num  40.23 91.44 50.4 49.45 0.31 ...
 $ a1    : num  -22.037 -43.621 -30.541 -17.206 -0.134 ...
 $ a2    : num  -8.25 -27.761 -22.48 -10.28 -0.112 ...
 $ a3    : num  -6.199 -18.402 -16.273 -7.325 -0.082 ...
 $ a4    : num  -3.741 -12.586 -11.89 -4.711 -0.061 ...
 $ a5    : num  -0.957 -7.302 -8.346 -2.206 -0.048 ...
 $ a6    : num  -0.176 -3.73 -4.847 -1.064 -0.043 ...
 $ a7    : num  -0.113 -0.624 -1.6 -0.494 -0.037 ...
 $ a8    : num  -0.049 -0.256 -0.627 -0.157 -0.033 ...
 $ a9    : num  -0.035 -0.15 -0.172 -0.127 -0.029 ...
 $ a10   : num  -0.018 -0.044 -0.135 -0.076 -0.022 ...
 $ a11   : num  0 -0.017 -0.079 -0.041 -0.01 ...
 $ a12   : num  0.014 0.003 -0.014 0.006 -0.001 ...
 $ a13   : num  0.027 0.23 0.15 0.057 0.018 ...
 $ a14   : num  0.038 2.647 0.287 0.142 0.039 ...
 $ a15   : num  0.09 5.184 1.036 0.289 0.05 ...
 $ a16   : num  0.202 11.454 6.173 1.16 0.064 ...
 $ a17   : num  5.353 19.035 9.66 4.178 0.078 ...
 $ a18   : num  10.538 31.47 18.47 10.234 0.094 ...
 $ a19   : num  13.423 43.202 30.252 18.237 0.141 ...
 $ a20   : num  22.02 61.137 35.749 36.199 0.226 ...
 $ b1    : num  -8.086 -17.086 -10.811 -7.55 -0.218 ...
 $ b2    : num  -5.736 -10.605 -8.407 -5.667 -0.103 ...
 $ b3    : num  -2.935 -6.661 -5.017 -4.069 -0.084 ...
 $ b4    : num  -1.651 -3.047 -2.623 -2.485 -0.064 ...
 $ b5    : num  -0.357 -1.508 -1.355 -1.4 -0.045 ...
 $ b6    : num  -0.11 -0.406 -0.532 -0.373 -0.027 ...
 $ b7    : num  -0.068 -0.23 -0.208 -0.202 -0.022 -0.932 -0.11 -0.489 -0.378 -0.347 ...
 $ b8    : num  -0.045 -0.067 -0.114 -0.127 -0.015 -0.517 -0.064 -0.208 -0.127 -0.143 ..
 $ b9    : num  -0.027 -0.022 -0.042 -0.048 -0.001 -0.369 -0.034 -0.03 -0.023 -0.045 ...
 $ b10   : num  0 0 0 0 0 -0.129 -0.008 0.088 0.05 0 ...
 $ b11   : num  0 0.005 0.024 0.023 0 -0.015 0.003 0.282 0.151 0.046 ...
 $ b12   : num  0 0.073 0.076 0.091 0.021 0.193 0.026 0.612 0.687 0.195 ...
 $ b13   : num  0.026 0.246 0.15 0.186 0.022 ...
 $ b14   : num  0.045 0.796 0.343 0.366 0.026 ...
 $ b15   : num  0.09 2.942 1.101 1.361 0.045 ...
 $ b16   : num  0.251 5.745 2.357 2.412 0.064 ...
 $ b17   : num  2.755 7.479 4.714 3.356 0.084 ...
 $ b18   : num  5.186 10.813 8.295 5.755 0.113 ...
 $ b19   : num  8.162 15.65 11.873 7.99 0.224 ...
 $ b20   : num  17.596 22.545 21.076 17.489 0.541 ...
 $ driver: int  14 14 14 14 14 14 14 14 14 14 ...
 $ trip  : int  1 2 3 4 5 6 7 8 9 10 ...
```

This will create a master summary file based on the features created. (Velocities, acceleration, breaks)

4. Predictive Model that identifies a driver signature

Create predictive model that can
a. segment drivers into various driver types
b. uniquely identify a driver signature. i.e Given a new trip identify if the trip was from a particular driver or not.

```
> # Example,
> #
> # d1793$createDriverSignature()
> #
> # gbm(formula = target ~ ., distribution = "gaussian", data = train,
> #    n.trees = trees, interaction.depth = 5, shrinkage = shrinkage)
> # A gradient boosted model with gaussian loss function.
> # 100 iterations were performed.
> # There were 63 predictors of which 57 had non-zero influence.
```

5. Using R6 Class (Driver, Telematics)

Similary, methods in the Telematics and Driver class can be used to obtain the plots of the different features and also to create the master summary files. After loading the database for Telematics, users can specify which driver's information to retrieve and to refer to which trip number.

5(a) Telematics class

Methods
load : Load the telematics database
createMasterSummary : Create the master summary file of the drivers.

```
> # Example,
> #
> # telematics <- Telematics$new()
> # telematics$createMasterSummary()
> #
> #  master.file = "../data/master.csv",
> #  file.list = "../data/fileList.csv" ,
> #  root.folder = "../data/drivers",
> #  subset.driver.count = 10 ,
> #  cores =2)
```

5(b) Driver class

Methods

ShowTrip : Load the telematics database
ShowCoordinates : Create the master summary file of the drivers.
ShowVelocity : plot velocity
ShowAcceleration : plot acceleration
ShowBreaks : plot breaks
ShowSpeedBreaks : plot speed breaks
ShowQuantiles : plot quantiles
CreateDriverSignature : create driver's signature

```
> # Example,
> #
> # telematics$load(verbose = FALSE)
> # d1793 = Driver$new(db = telematics, driver.name = "1793")
> # d1793$ShowTrip("55")
```