Branch: **master** ▾

Find file | Copy path

BzztPodtaxi / FinalBzzt.R

aicasas Add files via upload

b5b3acc 21 minutes ago

1 contributor

Raw | Blame | History

206 lines (165 sloc) | 7.2 KB

```
1
2  'Bzzt Final Projct: Alexis Casas
3  Lines 5:110 are data cleaning, lines 110:133 are geocoding of ~26,00
4  saved as BzztGeocode.csv and BzztGeocodeD.csv
5  and can be ran quickly from lines 139 and on'
6
7  data1 <- read.csv('201901.csv', header=FALSE)
8  data2 <- read.csv('201902.csv', header=FALSE)
9  data3 <- read.csv('201903.csv', header=FALSE)
10
11 #Combining Datasets by columns
12 data <- rbind(data1, data2, data3)
13 rm(data1, data2, data3)
14
15 #Deleting rows with missing entries
16 data[data==""] <- NA
17 data[data=="None"] <- NA
18 which(is.na(data))
19 data<- data[complete.cases(data),]
20 which(is.na(data))
21
22 #install.packages("stringr")
23 library(stringr)
```

```r

#Data cleaning
#Split the first column by ; for time stamp at different stops
data <- data.frame(data,do.call(rbind,str_split(data$V1,";")))
data <- data.frame(data,do.call(rbind,str_split(data$V5,";")))
data <- data.frame(data,do.call(rbind,str_split(data$V3,";")))
#Deleting first columns that were split
data <- data[,-c(1, 3, 5, 14, 18)]

#Filling in column datasets that were not made included
names(data)[10] <- "OriginAddress"
names(data)[14] <- "DestinationAddress"
names(data)[3] <- "Code"
names(data)[4] <- "ActualOriginTS"
names(data)[5] <- "ActualDestinationTS"
names(data)[6] <- "EstimatedOriginTS"
names(data)[7] <- "EstimatedDestinationTS"
names(data)[8] <- "RideRequestTS"
names(data)[9] <- "RideAcceptedTS"
names(data)[1] <- "EstimatedEndAddress"
names(data)[11] <- "ZipCode"
names(data)[12] <- "ActualCost"
names(data)[13] <- "EstimatedCost"
names(data)[2] <- "EstimatedOriginAddress"

#Taking care of columns entries which do not match the others (incco
data <- data[-grep("Stockholm", data$ZipCode), ]
data <- data[grep("[0-9]", data$EstimatedEndAddress), ]
data <- data[grep("[0-9]", data$EstimatedOriginAddress), ]


#Data cleaning for Date and Time Format

data$ActualOriginTS <- sub('([^+]+).*', '\\1', data$ActualOriginTS)
data$ActualDestinationTS <- sub('([^+]+).*', '\\1', data$ActualDesti
data$EstimatedOriginTS <- sub('([^+]+).*', '\\1', data$EstimatedOrig
data$EstimatedDestinationTS <- sub('([^+]+).*', '\\1', data$Estimate
data$RideRequestTS <- sub('([^+]+).*', '\\1', data$RideRequestTS)
```

```
data$RideAcceptedTS <- sub('([^+]+).*', '\\1', data$RideAcceptedTS)

#Filling in NA
data[data==""] <- NA
data[data=="None"] <- NA
which(is.na(data))
data<- data[complete.cases(data),]

library(lubridate)
library(dplyr)

data$ActualOriginTS <- strptime(data$ActualOriginTS,format="%Y-%m-%d
data$ActualDestinationTS <- strptime(data$ActualDestinationTS,format
data$EstimatedOriginTS <- strptime(data$EstimatedOriginTS,format="%Y
data$EstimatedDestinationTS <- strptime(data$EstimatedDestinationTS,
data$RideRequestTS <- strptime(data$RideRequestTS,format="%Y-%m-%dT%
data$RideAcceptedTS <- strptime(data$RideAcceptedTS,format="%Y-%m-%d


#Encoding Weekdays and Time to see which are the bussiest
data$Weekdayy <- (weekdays(data$ActualOriginTS))
data$Hour <- hour(data$ActualDestinationTS)
data$Month <- month(data$ActualDestinationTS)

#Plotting Number of rides taken on each day
data$Weekday <- factor(data$Weekday, levels=c("Sunday","Monday", "Tu
freq_table <- table(data$Weekday)
margin.table(freq_table, 1)
plot(freq_table)
barplot(freq_table, main="Daily Ride Frequency",
        xlab="Day of Week")

#Plotting the frequency desnity at different hours of the day
install.packages('ggpubr')
library(ggpubr)
d <- density(data$Hour)
plot(d, main="Ride Density: Hour of Day")
polygon(d, col="grey", border="blue")
```

```
freq_table2 <- table(data$Hour)
margin.table(freq_table2, 1)
plot(freq_table2)
#Plot shows that there is a peak around 8am and 5-6pm (commuter hour

#####Applying K means to actual vs estimated cost



#######Here I begin Geocoding:##############
#There are too many addresses to generate 26,000 different geocoordi
#I will look at just months 1:3 of data, specifically month 1 looks
sample <- data[grep("[1:3]", data$Month), ]

#install.packages("ggplot2")
library(ggplot2)
library(ggmap)


#Getting API
api <- "AIzaSyBFmuXtB3JzjnCBOM5op0yY4rcMfJ1wL5k" # Text file with th
register_google(key = api)
#concatenate the address
start_addy <- paste(sample$OriginAddress, "Stockholm, Sweden", sampl
end_addy <- paste(sample$DestinationAddress, "Stockholm, Sweden", sa
# geocode - check for warnings
addys_coords <- geocode(start_addy)
end_addy_coords <- geocode(end_addy)
sample <- data.frame(cbind(sample, addys_coords))
sampleD <- data.frame(cbind(sample, end_addy_coords))
#write.csv(sample, file = "BzztGeocode.csv")
#write.csv(sampleD, file = "BzztGeocodeD.csv")
###########Here is the start of k means with coordinates###########
```

```
#Cleaned data for Kmeans


#Kmeans for pick up location
bzzt <- read.csv("BzztGeocode.csv")
sample <- bzzt[,-c(1:16)]
sample$Weekday <- factor(sample$Weekday, levels=c("Sunday","Monday",
sample <- sample[,4:5]
which(is.na(sample))
sample<- sample[complete.cases(sample),]
#removing outlier
#sample <- sample[-grep("-[0-9]", sample$lon), ]
#The elbow mehtod will help determine the number of clusters. This i
wcss = vector() #within cluster, sum of square. finding the closenes
for (i in 1:10) wcss[i]= sum(kmeans(sample, i)$withinss)
plot(1:10, wcss, type='b', main=paste('The Elbow Method: Pick Up Loc
     xlab = 'Number of Clusters',
     ylab = 'WCSS')

#don't need feature scaling

#apply k-means
set.seed(42)
cluster <- kmeans(sample, 2)
sample$Borough <- factor(cluster$cluster)
ggplot(sample, aes(x=lat, y=lon, color=Borough)) + geom_point(shape=
#+ coord_cartesian(xlim = c(59.2, 59.4), ylim = c(17.8,18.3))
#+ coord_cartesian(xlim = c(57.2, 59.4), ylim = c(16,19))
#+ coord_cartesian(xlim = c(59.2, 59.4), ylim = c(17.8,18.3))
ggsave("pickupKmeans.png")

freq_table <- table(sample$Borough)
margin.table(freq_table, 1)

#install.packages('RgoogleMaps')
library(RgoogleMaps)
#data(sample)
col=as.numeric(sample$Borough)
```

```
par(pty="s")
plotmap(lat, lon, zoom = 13, col = col, pch=1, data = sample)

#Kmeans for drop off location
bzztD <- read.csv("BzztGeocodeD.csv")
sampleD <- bzztD[,-c(1:16)]
sampleD <- sampleD[,4:5]
which(is.na(sampleD))
sampleD<- sampleD[complete.cases(sampleD),]

#The elbow mehtod will help determine the number of clusters. This i
wcss = vector() #within cluster, sum of square. finding the closenes
for (i in 1:10) wcss[i]= sum(kmeans(sampleD, i)$withinss)
plot(1:10, wcss, type='b', main=paste('The Elbow Method: Drop Off Lo
     xlab = 'Number of Clusters',
     ylab = 'WCSS')
#don't need feature scaling

#apply k-means
set.seed(42)
cluster <- kmeans(sampleD, 2)
sampleD$Borough <- factor(cluster$cluster)
ggplot(sampleD, aes(x=lat, y=lon, color=Borough)) + geom_point(shape
ggsave("dropoffKmeans.png")

freq_table <- table(sampleD$Borough)
margin.table(freq_table, 1)

col=as.numeric(sampleD$Borough)
par(pty="s")
plotmap(lat, lon, zoom = 13, col = col, pch=1, data = sampleD)
```