Final project R code

Yu Qin,

2019-07-23

## pre-process

Before we jump into data anyalyze, we did a pre-process of the raw data. we factor out some new data columns that will help our modeling:

* time window: a number ranged from [1:4] that represents peroid of time in a day:

- 1: 22:00 - 6:00. night and before morning  
- 2: 6:00 - 10:00. early summit of go to work  
- 3: 10:00 - 18:00. daytime  
- 4: 18:00 - 22:00. evening

* weekday: a number ranged in [1:7] to represenct Monday to Sunday.
* distance: distance between pickup location and dropoff location.

Since the input data is too big and R package readr doesn’t offer a way to read data by small chunk, we use python code to do the pre-process and you can find it [here](https://github.com/YuQin1112/HU-500-PROJECT/blob/master/pre-process/prepare_data.py).

the new data columns after pre-process define as:

|  |  |  |
| --- | --- | --- |
| column name | type | description |
| fare | float | money paid on this ride |
| pickup\_ts | int | unix timestamp of the pickup time |
| pickup\_long | float | pickup longtitude |
| pickup\_lat | float | pickup latitude |
| dropoff\_long | float | dropoff longtitude |
| dropoff\_lat | float | dropoff lattitude |
| passenger\_count | int | number of passenger |
| time\_window | int | time in range of 4 partitions in a day |
| weekday | int | Monday to Sunday as number 1 to 7 |
| distance | float | distance in mile between pickup and fropoff |

## Import library and data

library(readr)  
library(sqldf)

## Loading required package: gsubfn

## Loading required package: proto

## Loading required package: RSQLite

library(ggplot2)  
library(sp)  
library(rgdal)

## rgdal: version: 1.4-4, (SVN revision 833)  
## Geospatial Data Abstraction Library extensions to R successfully loaded  
## Loaded GDAL runtime: GDAL 2.2.3, released 2017/11/20  
## Path to GDAL shared files: C:/Users/fgh0809/Documents/R/win-library/3.6/rgdal/gdal  
## GDAL binary built with GEOS: TRUE   
## Loaded PROJ.4 runtime: Rel. 4.9.3, 15 August 2016, [PJ\_VERSION: 493]  
## Path to PROJ.4 shared files: C:/Users/fgh0809/Documents/R/win-library/3.6/rgdal/proj  
## Linking to sp version: 1.3-1

library(geosphere)  
  
data0 <- read\_csv("E:/hu500/HU500/pre-process/real\_data.csv")

## Parsed with column specification:  
## cols(  
## idx = col\_double(),  
## fare = col\_double(),  
## pickup\_ts = col\_double(),  
## pickup\_long = col\_double(),  
## pickup\_lat = col\_double(),  
## dropoff\_long = col\_double(),  
## dropoff\_lat = col\_double(),  
## passenger\_count = col\_double(),  
## time\_window = col\_double(),  
## weekday = col\_double(),  
## distance = col\_double()  
## )

### Data to begin, create a copy  
dat0 = data0

## Remove outlier 1: by human knowledge

### remove invalid coordinates.

dat0$distance[dat0$distance <= 0] = NA  
dat0$pickup\_long[dat0$pickup\_long == 0] = NA  
dat0$pickup\_lat[dat0$pickup\_lat == 0] = NA  
dat0$dropoff\_long[dat0$dropoff\_long == 0] = NA  
dat0$dropoff\_lat[dat0$dropoff\_lat == 0] = NA  
  
dat0$pickup\_long[abs(dat0$pickup\_long) > 180] = NA  
dat0$pickup\_lat[abs(dat0$pickup\_lat) > 90] = NA  
dat0$dropoff\_long[abs(dat0$dropoff\_long) > 180] = NA  
dat0$dropoff\_lat[abs(dat0$dropoff\_lat) > 90] = NA

### remove coordinates that outside of NYC

dat0$pickup\_long[dat0$pickup\_long < -100] = NA  
dat0$pickup\_long[dat0$pickup\_long > -50] = NA  
dat0$pickup\_lat[dat0$pickup\_lat < 20] = NA  
dat0$pickup\_lat[dat0$pickup\_lat > 60] = NA  
  
dat0$dropoff\_long[dat0$dropoff\_long < -100] = NA  
dat0$dropoff\_long[dat0$dropoff\_long > -50] = NA  
dat0$dropoff\_lat[dat0$dropoff\_lat < 20] = NA  
dat0$dropoff\_lat[dat0$dropoff\_lat > 60] = NA

### remove those invalid columns

no\_miss <- dat0[complete.cases(dat0), ]  
summary(no\_miss)

## idx fare pickup\_ts pickup\_long   
## Min. : 0 Min. :-44.90 Min. :1.231e+09 Min. :-80.62   
## 1st Qu.:24996 1st Qu.: 6.00 1st Qu.:1.282e+09 1st Qu.:-73.99   
## Median :49992 Median : 8.50 Median :1.332e+09 Median :-73.98   
## Mean :49990 Mean : 11.34 Mean :1.332e+09 Mean :-73.98   
## 3rd Qu.:74985 3rd Qu.: 12.50 3rd Qu.:1.382e+09 3rd Qu.:-73.97   
## Max. :99999 Max. :495.00 Max. :1.436e+09 Max. :-66.73   
## pickup\_lat dropoff\_long dropoff\_lat passenger\_count  
## Min. :37.42 Min. :-86.80 Min. :34.53 Min. :0.000   
## 1st Qu.:40.74 1st Qu.:-73.99 1st Qu.:40.74 1st Qu.:1.000   
## Median :40.75 Median :-73.98 Median :40.75 Median :1.000   
## Mean :40.75 Mean :-73.97 Mean :40.75 Mean :1.684   
## 3rd Qu.:40.77 3rd Qu.:-73.97 3rd Qu.:40.77 3rd Qu.:2.000   
## Max. :51.08 Max. :-65.24 Max. :55.78 Max. :6.000   
## time\_window weekday distance   
## Min. :1.000 Min. :1.000 Min. : 0.0001   
## 1st Qu.:2.000 1st Qu.:2.000 1st Qu.: 0.7949   
## Median :3.000 Median :4.000 Median : 1.3527   
## Mean :2.535 Mean :3.985 Mean : 2.1131   
## 3rd Qu.:3.000 3rd Qu.:6.000 3rd Qu.: 2.4577   
## Max. :4.000 Max. :7.000 Max. :1036.6964

### Kmean cluster coords (lat, long)

We want to group pick up location and dropoff location into different areas, and use Kmean algorithm to automatically divide our data into 5 clusters which represent Manhattan, Brooklyn, Flushing, Long Island and Queen.

We skip this step since cluster algorithm requires lots of memory and our computer does not support that.

A MapReduce mechanism in a distributed system (such as AWS) is a good solution. It is yet to be improved.

Here is the code we skipped:

dist = function(df) {  
 require(geosphere)  
 d <- function(i,z){  
 dist <- rep(0,nrow(z))  
 dist[i:nrow(z)] <- distHaversine(z[i:nrow(z),1:2],z[i,1:2])  
 return(dist)  
 }  
 dm <- do.call(cbind,lapply(1:nrow(df),d,df))  
 return(as.dist(dm))  
}  
  
# FAILED ON WOODEN PC FOR MEMORY INEFFICIENT  
# only\_coord <- no\_miss[,c(4,5)]  
# km <- kmeans(dist(only\_coord),centers=5)  
# hc <- hclust(dist(only\_coord)) # ALTERNINATIVE CLUSTER METHOD

## Remove outlier 2: by cutoffs

### take out irralevent columns

take out columns:

* index : provide no information.
* timestamp : time is covered and better concluded buy time\_window.
* coordinates: since we cannot process 500k entrys of data.

dat1 = no\_miss[,-c(1,3,4,5,6,7)]  
K = 5  
N = nrow(dat1)  
model1 = lm(fare ~ . ,data=dat1)

### laverage cutoff

leverage = hatvalues(model1)  
cutleverage = (2\*K + 2) / N  
badlaverage = as.numeric(leverage > cutleverage)  
table(badlaverage)

## badlaverage  
## 0 1   
## 464657 19798

### cook cutoff

cooks = cooks.distance(model1)  
cutcook = 4 / (N-K-1)  
badcooks = as.numeric(cooks > cutcook)  
table(badcooks)

## badcooks  
## 0 1   
## 465304 19151

### mahalnobis cutoff

mahal = mahalanobis(  
 dat1,  
 colMeans(dat1, na.rm=TRUE),  
 cov(dat1, use="pairwise.complete.obs")  
)  
cutmahal = qchisq(1-.001,ncol(dat1))  
badmahal = as.numeric(mahal > cutmahal)  
table(badmahal)

## badmahal  
## 0 1   
## 477187 7268

### remove outlier 2

total = badmahal + badcooks + badlaverage  
noout = subset(dat1, total < 1)

## Derive revised model and show summary

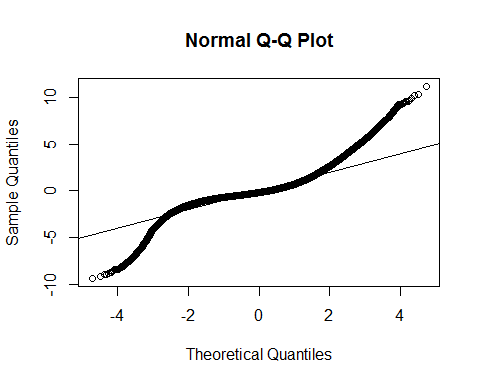
model2 = lm(fare ~ . ,data=noout)  
summary(model2, correlation = TRUE)

##   
## Call:  
## lm(formula = fare ~ ., data = noout)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -26.1473 -1.4845 -0.5114 0.9633 31.2280   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 4.1137198 0.0154012 267.104 <2e-16 \*\*\*  
## passenger\_count 0.0006432 0.0036877 0.174 0.862   
## time\_window -0.0575114 0.0041621 -13.818 <2e-16 \*\*\*  
## weekday -0.0313860 0.0021938 -14.307 <2e-16 \*\*\*  
## distance 3.3732442 0.0027574 1223.357 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.788 on 453477 degrees of freedom  
## Multiple R-squared: 0.7677, Adjusted R-squared: 0.7677   
## F-statistic: 3.747e+05 on 4 and 453477 DF, p-value: < 2.2e-16  
##   
## Correlation of Coefficients:  
## (Intercept) passenger\_count time\_window weekday  
## passenger\_count -0.32   
## time\_window -0.62 -0.05   
## weekday -0.51 -0.05 -0.05   
## distance -0.28 0.02 -0.04 -0.03

## Data screening

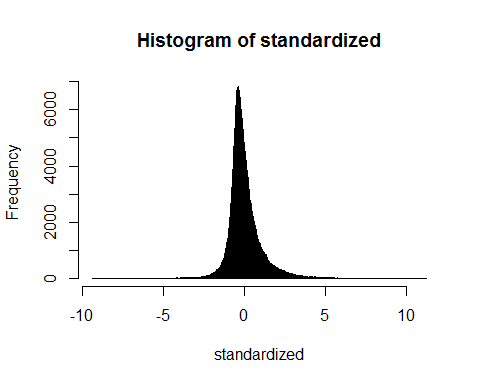
### linearity

standardized = rstudent(model2)  
fitted = scale(model2$fitted.values)  
qqnorm(standardized)  
abline(0,1)



### normality

hist(standardized, breaks = 1000)



### homogeneous

plot(fitted)  
abline(0,0)  
abline(v = 0)

