STATS 769 - Lab 02 - bole001

Bernard O’Leary

12 August 2019

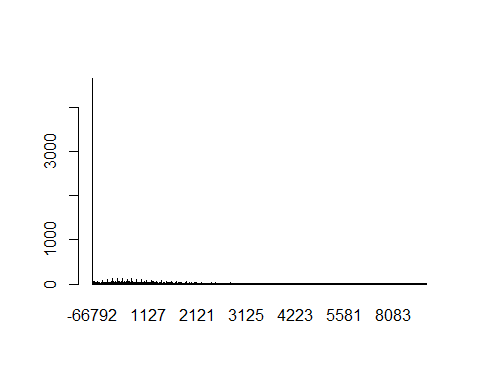
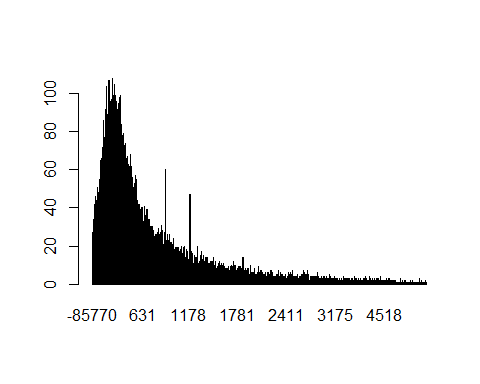
## Read in data

Date format is CSV (Comma Separated Values), reading into R using the read.csv function.

# setwd("~/Source/UoA-STATS-769/Lab02")  
# Get the files names in the directory  
directory = "."  
files.ls <- intersect(list.files(path=directory, pattern="trips"), list.files(path=directory, pattern="\*.csv"))  
# First apply read.csv, then rbind  
trips.df = do.call(rbind, lapply(files.ls, function(x) read.csv(x, stringsAsFactors = FALSE)))

## Explore duration and distance

Visualise the distance (predictor) and duration (response) features by generating a barplot of their respective values. Both variables appear to be right skewed.



## Add ‘LongTrip’ variable

Create variable named ‘LongTrip’ and add to the end of the dataframe.

trips.df$LongTrip <- trips.df$Trip.Distance > 1000

## Add logged duration variable

Name the variable Trip.Duration.Logged and add to the end of the dataframe. Note that because the data are not cleansed by this point NaNs are produced here.

trips.df$Trip.Duration.Logged <- log(trips.df$Trip.Duration)

## Warning in log(trips.df$Trip.Duration): NaNs produced

## Train/test split

Use the plyr package to split the records evenly by month (1000 of each) and take samples. Use month as divider and take a random sample of 0.8 of the dataset split evenly across the months for the training set. Capture the remainder (0.2) of the split in the second part of the output for the test set.

The plyr package can be found here: <https://www.rdocumentation.org/packages/plyr/versions/1.8.4/topics/dlply>

The sample() function enables a random selection of data from an R dataframe.

# use the plyr library to split the records evenly and take samples  
# use month as divider and take a random sample of 0.8 of the dataset split evenly across the months  
# capture the remainder (0.2) of the split in the second part of the output  
# https://www.rdocumentation.org/packages/plyr/versions/1.8.4/topics/dlply  
# https://stackoverflow.com/questions/18258690/take-randomly-sample-based-on-groups  
# https://stackoverflow.com/questions/18942792/training-and-test-set-with-respect-to-group-affiliation  
library(plyr)

## Warning: package 'plyr' was built under R version 3.6.1

split\_set = dlply(trips.df, .(Month), function(.) { s = sample(1:nrow(.), trunc(nrow(.) \* 0.8)); list(.[s, ], .[-s,]) } )  
  
# train/test split  
# https://www.rdocumentation.org/packages/plyr/versions/1.8.4/topics/ldply  
training\_set = ldply(split\_set, function(.) .[[1]])  
test\_set = ldply(split\_set, function(.) .[[2]])

## Get rid of non-positive and non-finite values in duration and distance

Remove from the training set only

training\_set <- subset(training\_set, training\_set$Trip.Duration > 0)  
training\_set <- subset(training\_set, training\_set$Trip.Distance > 0)  
training\_set <- subset(training\_set, is.finite(training\_set$Trip.Duration))  
training\_set <- subset(training\_set, is.finite(training\_set$Trip.Distance))

## Make the model

Use logged duration (rather than raw duration) to avoid warning “glm.fit: fitted probabilities numerically 0 or 1 occurred”.

y\_train = training\_set$LongTrip  
#x\_train = training\_set$Trip.Duration  
x\_train = training\_set$Trip.Duration.Logged  
fit\_mean = mean(y\_train)  
fit\_glm <- glm(formula=y~x, data.frame(y=y\_train, x=x\_train), family=binomial(link=logit))

## Get rid of non-positive and non-finite values in duration and distance

Cleanse the test\_set dataframe now.

test\_set <- subset(test\_set, test\_set$Trip.Duration > 0)  
test\_set <- subset(test\_set, test\_set$Trip.Distance > 0)  
test\_set <- subset(test\_set, is.finite(test\_set$Trip.Duration))  
test\_set <- subset(test\_set, is.finite(test\_set$Trip.Distance))

## Test results

Generate two vectors; the predicted sample mean and the prediction generated by the GLM.

y\_test <- test\_set$LongTrip  
x\_test <- test\_set$Trip.Duration.Logged  
  
# make a vector that is populated with the fit\_mean  
pred\_mean <- rep(fit\_mean, length(y\_test))  
  
# get a vector of predictions based on test data  
pred\_glm <- predict(fit\_glm, data.frame(x=x\_test))

## Define RMSE function and evaluate using test set

Take the RMSE (root mean square error) of the predicted mean and the predicted logistic regression model. The model gives us a far higher RMSE than the predicted mean (the predicted mean is an average model with random noise).

# define RMSE function  
RMSE <- function(m, o) {  
 sqrt(mean((m - o)^2))  
}  
  
# get the RMSE  
RMSE(pred\_mean, y\_test)

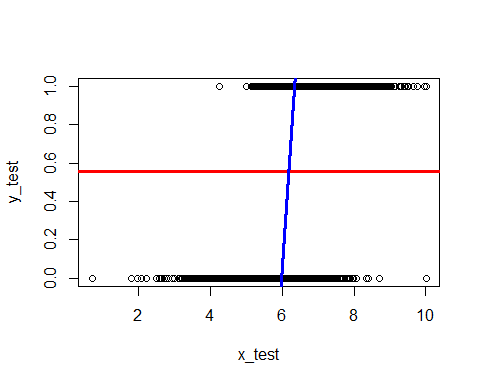
## [1] 0.4972522

RMSE(pred\_glm, y\_test)

## [1] 2.354829

## Visualise the model

Plot y\_test vs x\_test and overlay the average with random error model and our derived logistic regression model. This is to help us visualise the data and the derived model as it applies to the test dataset.



## Summary and analysis

Our model has not reduced the variability that we see by simply taking the average model with random error - in-fact it slightly increases the variablity. This is not a good model as it explains less of the variability in the data than the average model with random error. The model could be improved to better fit the data.