## UNDERSTANDING DATA AND ITS ENVIRONMENT: REPORT ASSESSMENT

## 1. Introduction

This report explains the approach taken to forecast sales across stores. It is reproduble based on code available at <https://github.com/eugenividal/Understanding-data-report>

The task is to estimate sales for the departments of each store based on the historical training data.

The first stage is to load in the data. We will use the tidyverse package and load it with the library() function:

library(tidyverse)  
# load data  
source("code/load-data.R")

We can check that these files have been loaded into the R envionment with the following command:

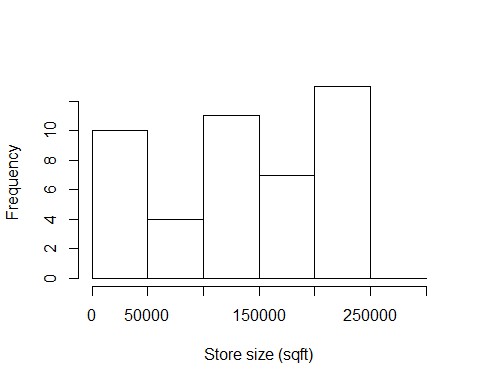
ls()

## [1] "features" "stores" "test" "train"

## 2. Reviewing and describing the data

The data is described on a histogram.

b = seq(from = 0, to = 300000, by = 50000)  
hist(stores$`Size (sq ft)`, main = "", xlab = "Store size (sqft)", breaks = b)



### 3. Joining the data

Data sets are combined together into a unified view in order to facilitate the tasks of pre-processing and identifying the key factors.

No difficulties of Data Integration were found.

The three data sets (stores, train and features) are linked in this order using the ‘store’ variable as ID.

# join store-level data onto training dataset (so we know size)  
train\_joined = inner\_join(train, y = stores)

## Joining, by = "Store"

# would use rename() function to rename columns if needed (not needed)  
train\_joined = inner\_join(train\_joined, y = features)

## Joining, by = c("Store", "Date", "IsHoliday")

## 4. Pre-processing the data

## 5. Identifying the key factors

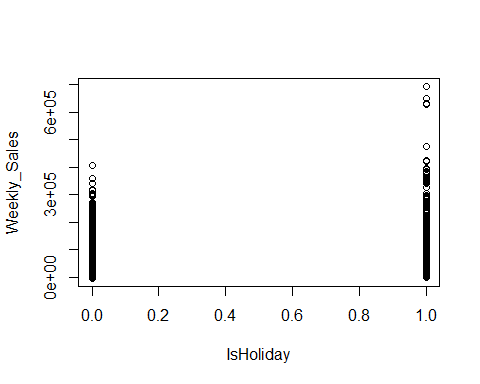
## 6. Creating the predictive model

There are many packages and approach for forecasting. We could use the lm() function to do a linear regression, for example. Here we use the xgboost package

# install.packages("xgboost")  
m1 = lm(Weekly\_Sales ~ IsHoliday + Fuel\_Price, data = train\_joined)  
summary(m1)

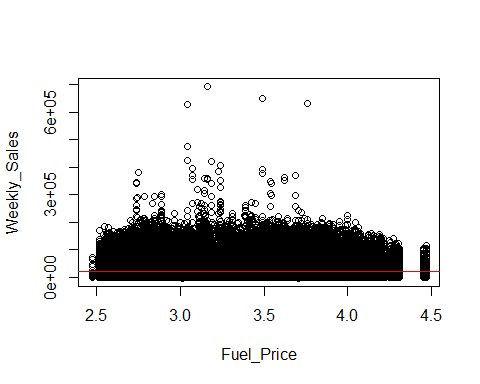
##   
## Call:  
## lm(formula = Weekly\_Sales ~ IsHoliday + Fuel\_Price, data = train\_joined)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -20875 -13896 -8357 4224 676067   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 15753.66 260.47 60.482 <2e-16 \*\*\*  
## IsHolidayTRUE 1140.53 137.18 8.314 <2e-16 \*\*\*  
## Fuel\_Price 43.84 76.52 0.573 0.567   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 22710 on 421567 degrees of freedom  
## Multiple R-squared: 0.000164, Adjusted R-squared: 0.0001592   
## F-statistic: 34.57 on 2 and 421567 DF, p-value: 9.763e-16

# Plot regression line  
plot(Weekly\_Sales ~ IsHoliday + Fuel\_Price, data = train\_joined)



abline(m1, col="red")

## Warning in abline(m1, col = "red"): only using the first two of 3  
## regression coefficients



# predicted\_sales = predict(m1, train\_joined)  
# plot(train\_joined$Weekly\_Sales, predicted\_sales)

Other option would be to use the ISLR package

# library("ISLR")  
# install.packages("ISLR")  
library("MASS")

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':  
##   
## select

library("ISLR")  
m2 = lm(Weekly\_Sales ~ Temperature, data = train\_joined)  
summary (m2)

##   
## Call:  
## lm(formula = Weekly\_Sales ~ Temperature, data = train\_joined)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -20937 -13906 -8368 4224 677105   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 16152.325 119.184 135.524 <2e-16 \*\*\*  
## Temperature -2.847 1.896 -1.501 0.133   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 22710 on 421568 degrees of freedom  
## Multiple R-squared: 5.347e-06, Adjusted R-squared: 2.975e-06   
## F-statistic: 2.254 on 1 and 421568 DF, p-value: 0.1332

# Plot regression line  
plot(Weekly\_Sales ~ Temperature, data = train\_joined)  
abline(m2, col="red")

