Lyft-Uber-Price-Prediction

Nishtha Chaudhary

24 October 2019

# IMPORTING DATASETS AND CLEANING THEM

# Importing dataset cab\_rides

cab\_rides <- read.csv("C:/Users/nisht/Desktop/MITA/Fall/MVA/Final Project/cab\_rides.csv")  
summary(cab\_rides)

## distance cab\_type time\_stamp   
## Min. :0.020 Lyft:307408 Min. :1.543e+12   
## 1st Qu.:1.280 Uber:385663 1st Qu.:1.543e+12   
## Median :2.160 Median :1.544e+12   
## Mean :2.189 Mean :1.544e+12   
## 3rd Qu.:2.920 3rd Qu.:1.545e+12   
## Max. :7.860 Max. :1.545e+12   
##   
## destination source price   
## Financial District: 58851 Financial District: 58857 Min. : 2.50   
## Theatre District : 57798 Theatre District : 57813 1st Qu.: 9.00   
## Back Bay : 57780 Back Bay : 57792 Median :13.50   
## Boston University : 57764 Boston University : 57764 Mean :16.55   
## Haymarket Square : 57764 North End : 57763 3rd Qu.:22.50   
## Fenway : 57757 Fenway : 57757 Max. :97.50   
## (Other) :345357 (Other) :345325 NA's :55095   
## surge\_multiplier id   
## Min. :1.000 00005b8c-5647-4104-9ac6-94fa6a40f3c3: 1   
## 1st Qu.:1.000 00006eeb-0183-40c1-8198-c441d3c8a734: 1   
## Median :1.000 00008b42-5ecc-4f66-b4b9-b22a331634e6: 1   
## Mean :1.014 000094c0-00c4-43f1-ae1b-4693eec2a580: 1   
## 3rd Qu.:1.000 0000a8b2-e4d3-4227-8374-af8a2366e475: 1   
## Max. :3.000 0000b5d6-59be-4534-b371-8214334d94f0: 1   
## (Other) :693065   
## product\_id name   
## 6d318bcc-22a3-4af6-bddd-b409bfce1546: 55096 Black SUV: 55096   
## 6f72dfc5-27f1-42e8-84db-ccc7a75f6969: 55096 UberXL : 55096   
## 9a0e7b09-b92b-4c41-9779-2ad22b4d779d: 55096 WAV : 55096   
## 6c84fd89-3f11-4782-9b50-97c468b19529: 55095 Black : 55095   
## 8cf7e821-f0d3-49c6-8eba-e679c0ebcf6a: 55095 Taxi : 55095   
## 55c66225-fbe7-4fd5-9072-eab1ece5e23e: 55094 UberX : 55094   
## (Other) :362499 (Other) :362499

cab\_data<-cab\_rides

# Creating a date\_time column

cab\_data$date\_time<-as.POSIXct((cab\_data$time\_stamp/1000),origin = "1970-01-01 00:53:20", tz="GMT")

# Importing dataset weather

weather <- read.csv("C:/Users/nisht/Desktop/MITA/Fall/MVA/Final Project/weather.xls")  
summary(weather)

## ï..temp location clouds   
## Min. :19.62 Back Bay : 523 Min. :0.0000   
## 1st Qu.:36.08 Beacon Hill : 523 1st Qu.:0.4400   
## Median :40.13 Boston University : 523 Median :0.7800   
## Mean :39.09 Fenway : 523 Mean :0.6778   
## 3rd Qu.:42.83 Financial District: 523 3rd Qu.:0.9700   
## Max. :55.41 Haymarket Square : 523 Max. :1.0000   
## (Other) :3138   
## pressure rain time\_stamp humidity   
## Min. : 988.2 Min. :0.000 Min. :1.543e+09 Min. :0.450   
## 1st Qu.: 997.7 1st Qu.:0.005 1st Qu.:1.543e+09 1st Qu.:0.670   
## Median :1007.7 Median :0.015 Median :1.544e+09 Median :0.760   
## Mean :1008.4 Mean :0.058 Mean :1.544e+09 Mean :0.764   
## 3rd Qu.:1018.5 3rd Qu.:0.061 3rd Qu.:1.545e+09 3rd Qu.:0.890   
## Max. :1035.1 Max. :0.781 Max. :1.545e+09 Max. :0.990   
## NA's :5382   
## wind   
## Min. : 0.290   
## 1st Qu.: 3.518   
## Median : 6.570   
## Mean : 6.803   
## 3rd Qu.: 9.920   
## Max. :18.180   
##

str(weather)

## 'data.frame': 6276 obs. of 8 variables:  
## $ ï..temp : num 42.4 42.4 42.5 42.1 43.1 ...  
## $ location : Factor w/ 12 levels "Back Bay","Beacon Hill",..: 1 2 3 4 5 6 7 8 9 10 ...  
## $ clouds : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ pressure : num 1012 1012 1012 1012 1012 ...  
## $ rain : num 0.1228 0.1846 0.1089 0.0969 0.1786 ...  
## $ time\_stamp: int 1545003901 1545003901 1545003901 1545003901 1545003901 1545003901 1545003901 1545003901 1545003901 1545003901 ...  
## $ humidity : num 0.77 0.76 0.76 0.77 0.75 0.77 0.77 0.77 0.78 0.75 ...  
## $ wind : num 11.2 11.3 11.1 11.1 11.5 ...

weather\_data<-weather

# creating a date\_time column in weather\_data

weather\_data$date\_time<-as.POSIXct(weather\_data$time\_stamp,origin = "1970-01-01 00:53:20", tz="GMT")  
str(weather\_data)

## 'data.frame': 6276 obs. of 9 variables:  
## $ ï..temp : num 42.4 42.4 42.5 42.1 43.1 ...  
## $ location : Factor w/ 12 levels "Back Bay","Beacon Hill",..: 1 2 3 4 5 6 7 8 9 10 ...  
## $ clouds : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ pressure : num 1012 1012 1012 1012 1012 ...  
## $ rain : num 0.1228 0.1846 0.1089 0.0969 0.1786 ...  
## $ time\_stamp: int 1545003901 1545003901 1545003901 1545003901 1545003901 1545003901 1545003901 1545003901 1545003901 1545003901 ...  
## $ humidity : num 0.77 0.76 0.76 0.77 0.75 0.77 0.77 0.77 0.78 0.75 ...  
## $ wind : num 11.2 11.3 11.1 11.1 11.5 ...  
## $ date\_time : POSIXct, format: "2018-12-17 00:38:21" "2018-12-17 00:38:21" ...

# merge the datasets to reflect the same time for a location

cab\_data$merge\_date<-paste(cab\_data$source,"-",as.Date(cab\_data$date\_time),"-",format(cab\_data$date\_time,"%H:%M:%S"))  
weather\_data$merge\_date<-paste(weather\_data$location,"-",as.Date(weather\_data$date\_time),"-",format(weather\_data$date\_time,"%H:%M:%S"))

#making those values as characters  
weather\_data$merge\_date<-as.character(weather\_data$merge\_date)  
cab\_data$merge\_date<-as.character(cab\_data$merge\_date)

# verify that merge\_date has unique values.

weather\_data<-subset(weather\_data,!duplicated(weather\_data$merge\_date))  
isTRUE(duplicated(weather\_data$merge\_date))

## [1] FALSE

# Merging both the dataframes.

merge\_data<-merge(x=weather\_data, y=cab\_data,by='merge\_date', all.x=TRUE)  
str(merge\_data)

## 'data.frame': 9306 obs. of 21 variables:  
## $ merge\_date : chr "Back Bay - 2018-11-26 - 04:34:05" "Back Bay - 2018-11-26 - 05:34:13" "Back Bay - 2018-11-26 - 05:34:58" "Back Bay - 2018-11-26 - 05:36:38" ...  
## $ ï..temp : num 41 40.6 40.6 40.6 40.6 ...  
## $ location : Factor w/ 12 levels "Back Bay","Beacon Hill",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ clouds : num 0.87 0.86 0.86 0.86 0.86 0.95 0.95 0.94 0.93 0.93 ...  
## $ pressure : num 1014 1014 1014 1014 1014 ...  
## $ rain : num NA NA NA NA NA NA NA NA NA NA ...  
## $ time\_stamp.x : int 1543203645 1543207253 1543207298 1543207398 1543207398 1543207777 1543207777 1543208142 1543208578 1543209183 ...  
## $ humidity : num 0.92 0.93 0.93 0.93 0.93 0.92 0.92 0.92 0.92 0.92 ...  
## $ wind : num 1.46 2.57 2.59 2.65 2.65 2.59 2.59 2.83 3 3.01 ...  
## $ date\_time.x : POSIXct, format: "2018-11-26 04:34:05" "2018-11-26 05:34:13" ...  
## $ distance : num NA NA 1.44 1.36 1.22 1.34 1.1 NA NA NA ...  
## $ cab\_type : Factor w/ 2 levels "Lyft","Uber": NA NA 2 1 2 2 2 NA NA NA ...  
## $ time\_stamp.y : num NA NA 1.54e+12 1.54e+12 1.54e+12 ...  
## $ destination : Factor w/ 12 levels "Back Bay","Beacon Hill",..: NA NA 3 10 9 4 9 NA NA NA ...  
## $ source : Factor w/ 12 levels "Back Bay","Beacon Hill",..: NA NA 1 1 1 1 1 NA NA NA ...  
## $ price : num NA NA 8.5 16.5 NA 26.5 7.5 NA NA NA ...  
## $ surge\_multiplier: num NA NA 1 1 1 1 1 NA NA NA ...  
## $ id : Factor w/ 693071 levels "00005b8c-5647-4104-9ac6-94fa6a40f3c3",..: NA NA 548701 610037 513190 566219 94420 NA NA NA ...  
## $ product\_id : Factor w/ 13 levels "55c66225-fbe7-4fd5-9072-eab1ece5e23e",..: NA NA 7 10 5 3 1 NA NA NA ...  
## $ name : Factor w/ 13 levels "Black","Black SUV",..: NA NA 13 4 9 2 11 NA NA NA ...  
## $ date\_time.y : POSIXct, format: NA NA ...

# Handling Missing values

#Filling NA values in price  
merge\_data$rain[is.na(merge\_data$rain)]<-0  
  
#Extracting the numerical columns in a new dataframe "df"  
merge\_data$temp<-merge\_data[,c(2)] #renaming a column  
df<-merge\_data[,c(4,5,6,8,9,10,11,17,22,16)]  
  
#Data preparation  
#Dealing with missing values  
summary(merge\_data)

## merge\_date ï..temp location   
## Length:9306 Min. :19.62 Haymarket Square : 843   
## Class :character 1st Qu.:36.74 North Station : 801   
## Mode :character Median :39.73 Theatre District : 800   
## Mean :39.12 Northeastern University: 788   
## 3rd Qu.:41.86 North End : 772   
## Max. :55.41 Fenway : 771   
## (Other) :4531   
## clouds pressure rain time\_stamp.x   
## Min. :0.0000 Min. : 988.2 Min. :0.00000 Min. :1.543e+09   
## 1st Qu.:0.4500 1st Qu.: 992.2 1st Qu.:0.00000 1st Qu.:1.543e+09   
## Median :0.7700 Median :1002.2 Median :0.00000 Median :1.543e+09   
## Mean :0.6799 Mean :1005.2 Mean :0.01197 Mean :1.544e+09   
## 3rd Qu.:0.9700 3rd Qu.:1014.4 3rd Qu.:0.00000 3rd Qu.:1.544e+09   
## Max. :1.0000 Max. :1035.1 Max. :0.78070 Max. :1.545e+09   
##   
## humidity wind date\_time.x   
## Min. :0.4500 Min. : 0.290 Min. :2018-11-26 04:34:04   
## 1st Qu.:0.6700 1st Qu.: 4.183 1st Qu.:2018-11-28 01:38:42   
## Median :0.7500 Median : 7.490 Median :2018-11-28 23:55:29   
## Mean :0.7623 Mean : 7.212 Mean :2018-12-01 23:49:51   
## 3rd Qu.:0.8800 3rd Qu.: 9.990 3rd Qu.:2018-12-02 09:31:14   
## Max. :0.9900 Max. :18.180 Max. :2018-12-18 19:38:22   
##   
## distance cab\_type time\_stamp.y destination   
## Min. :0.020 Lyft:1732 Min. :1.543e+12 Fenway : 344   
## 1st Qu.:1.250 Uber:2134 1st Qu.:1.543e+12 Financial District: 342   
## Median :2.140 NA's:5440 Median :1.543e+12 Back Bay : 337   
## Mean :2.168 Mean :1.543e+12 Beacon Hill : 335   
## 3rd Qu.:2.947 3rd Qu.:1.543e+12 South Station : 334   
## Max. :7.460 Max. :1.545e+12 (Other) :2174   
## NA's :5440 NA's :5440 NA's :5440   
## source price surge\_multiplier  
## Haymarket Square : 392 Min. : 2.50 Min. :1.000   
## North Station : 351 1st Qu.: 9.00 1st Qu.:1.000   
## Theatre District : 344 Median :13.50 Median :1.000   
## Northeastern University: 329 Mean :16.67 Mean :1.018   
## North End : 316 3rd Qu.:22.50 3rd Qu.:1.000   
## (Other) :2134 Max. :92.00 Max. :2.000   
## NA's :5440 NA's :5758 NA's :5440   
## id   
## 000baa63-5e1c-4f9d-891c-e4e78e830199: 1   
## 002b15bc-b433-44a4-8174-b8ac95caebf8: 1   
## 00423464-fb1b-4e96-9154-b55a00854181: 1   
## 00552d6f-c5fa-4006-962a-4613097afabe: 1   
## 005ca94d-9dad-4b34-a8ce-82a6de9058b4: 1   
## (Other) :3861   
## NA's :5440   
## product\_id name   
## 8cf7e821-f0d3-49c6-8eba-e679c0ebcf6a: 318 Taxi : 318   
## 6d318bcc-22a3-4af6-bddd-b409bfce1546: 308 Black SUV: 308   
## 6c84fd89-3f11-4782-9b50-97c468b19529: 307 Black : 307   
## 6f72dfc5-27f1-42e8-84db-ccc7a75f6969: 306 UberPool : 306   
## 997acbb5-e102-41e1-b155-9df7de0a73f2: 306 UberXL : 306   
## (Other) :2321 (Other) :2321   
## NA's :5440 NA's :5440   
## date\_time.y temp   
## Min. :2018-11-26 04:34:06 Min. :19.62   
## 1st Qu.:2018-11-27 03:08:42 1st Qu.:36.74   
## Median :2018-11-28 14:25:28 Median :39.73   
## Mean :2018-11-28 08:15:46 Mean :39.12   
## 3rd Qu.:2018-11-29 00:42:54 3rd Qu.:41.86   
## Max. :2018-12-16 20:38:27 Max. :55.41   
## NA's :5440

summary(df)

## clouds pressure rain humidity   
## Min. :0.0000 Min. : 988.2 Min. :0.00000 Min. :0.4500   
## 1st Qu.:0.4500 1st Qu.: 992.2 1st Qu.:0.00000 1st Qu.:0.6700   
## Median :0.7700 Median :1002.2 Median :0.00000 Median :0.7500   
## Mean :0.6799 Mean :1005.2 Mean :0.01197 Mean :0.7623   
## 3rd Qu.:0.9700 3rd Qu.:1014.4 3rd Qu.:0.00000 3rd Qu.:0.8800   
## Max. :1.0000 Max. :1035.1 Max. :0.78070 Max. :0.9900   
##   
## wind date\_time.x distance   
## Min. : 0.290 Min. :2018-11-26 04:34:04 Min. :0.020   
## 1st Qu.: 4.183 1st Qu.:2018-11-28 01:38:42 1st Qu.:1.250   
## Median : 7.490 Median :2018-11-28 23:55:29 Median :2.140   
## Mean : 7.212 Mean :2018-12-01 23:49:51 Mean :2.168   
## 3rd Qu.: 9.990 3rd Qu.:2018-12-02 09:31:14 3rd Qu.:2.947   
## Max. :18.180 Max. :2018-12-18 19:38:22 Max. :7.460   
## NA's :5440   
## surge\_multiplier temp price   
## Min. :1.000 Min. :19.62 Min. : 2.50   
## 1st Qu.:1.000 1st Qu.:36.74 1st Qu.: 9.00   
## Median :1.000 Median :39.73 Median :13.50   
## Mean :1.018 Mean :39.12 Mean :16.67   
## 3rd Qu.:1.000 3rd Qu.:41.86 3rd Qu.:22.50   
## Max. :2.000 Max. :55.41 Max. :92.00   
## NA's :5440 NA's :5758

merge\_data$surge\_multiplier = ifelse(is.na(merge\_data$surge\_multiplier),  
 ave(merge\_data$surge\_multiplier , FUN = function(x) mean(x, na.rm = TRUE)),  
 merge\_data$surge\_multiplier)  
  
merge\_data$price = ifelse(is.na(merge\_data$price),  
 ave(merge\_data$price , FUN = function(x) mean(x, na.rm = TRUE)),  
 merge\_data$price)  
  
df$distance = ifelse(is.na(df$distance),  
 ave(df$distance , FUN = function(x) mean(x, na.rm = TRUE)),  
 df$distance)  
  
df$surge\_multiplier = ifelse(is.na(df$surge\_multiplier),  
 ave(df$surge\_multiplier , FUN = function(x) mean(x, na.rm = TRUE)),  
 df$surge\_multiplier)  
  
df$price = ifelse(is.na(df$price),  
 ave(df$price , FUN = function(x) mean(x, na.rm = TRUE)),  
 df$price)

# Checking for null values

any(is.na(df))

## [1] FALSE

# Adding date and time column in the df data set

df$day<-weekdays(df$date\_time)  
df$time<-format(df$date\_time.x,"%H:%M:%S")  
df$date\_time<-as.Date(df$date\_time.x)  
merge\_data$day=weekdays(merge\_data$date\_time.x)

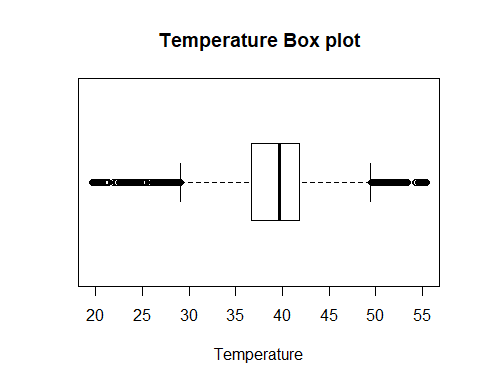
# Creating a Numeric dataframe

x<-df[,c(1,2,3,4,5,7,9)]  
str(x)

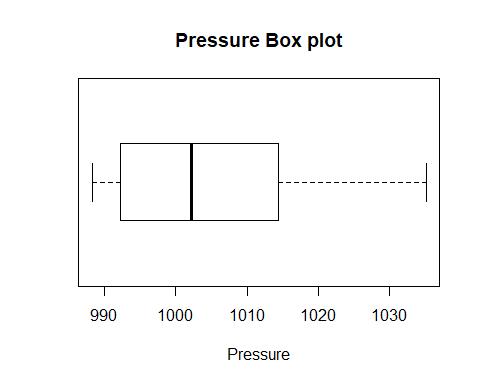
## 'data.frame': 9306 obs. of 7 variables:  
## $ clouds : num 0.87 0.86 0.86 0.86 0.86 0.95 0.95 0.94 0.93 0.93 ...  
## $ pressure: num 1014 1014 1014 1014 1014 ...  
## $ rain : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ humidity: num 0.92 0.93 0.93 0.93 0.93 0.92 0.92 0.92 0.92 0.92 ...  
## $ wind : num 1.46 2.57 2.59 2.65 2.65 2.59 2.59 2.83 3 3.01 ...  
## $ distance: num 2.17 2.17 1.44 1.36 1.22 ...  
## $ temp : num 41 40.6 40.6 40.6 40.6 ...

# BOXPLOT

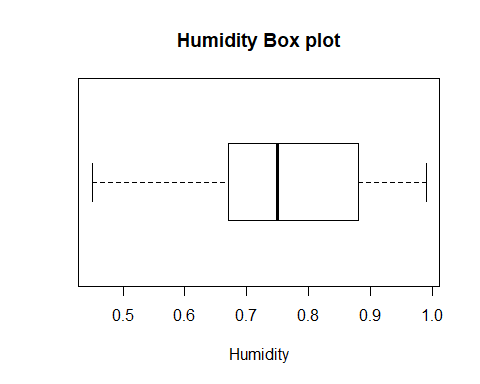
boxplot(x$temp, main="Temperature Box plot",yaxt="n", xlab="Temperature", horizontal=TRUE)



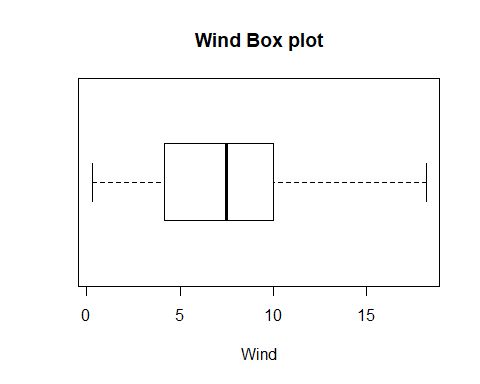
boxplot(x$pressure, main="Pressure Box plot",yaxt="n", xlab="Pressure", horizontal=TRUE)



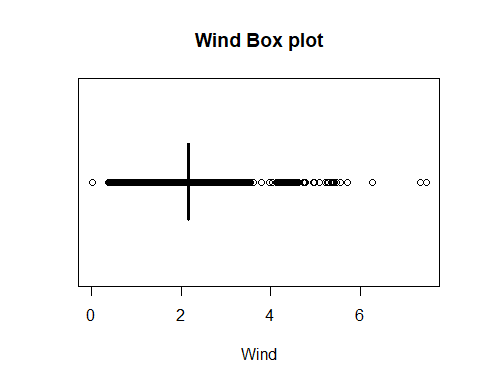
boxplot(x$humidity, main="Humidity Box plot",yaxt="n", xlab="Humidity", horizontal=TRUE)



boxplot(x$wind, main="Wind Box plot",yaxt="n", xlab="Wind", horizontal=TRUE)



boxplot(x$distance, main="Wind Box plot",yaxt="n", xlab="Wind", horizontal=TRUE)

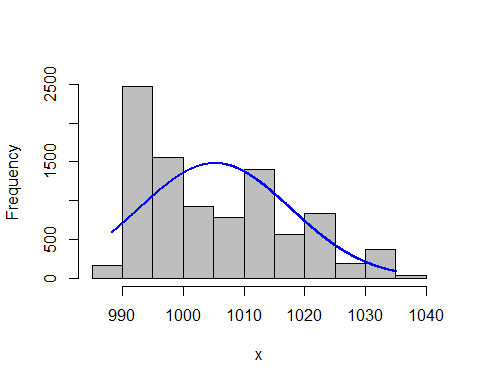


#Q-Q Plot to check normality..

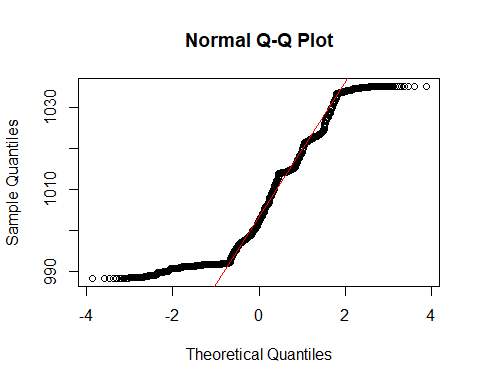
library(rcompanion)

## Warning: package 'rcompanion' was built under R version 3.5.3

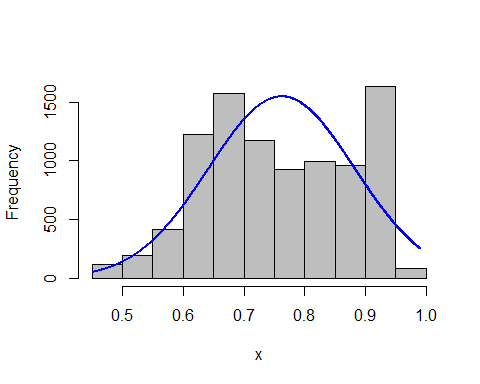
plotNormalHistogram(x$pressure)



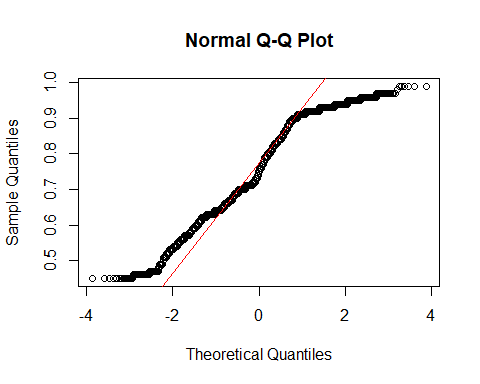
qqnorm(df$pressure)  
qqline(df$pressure, col="red")



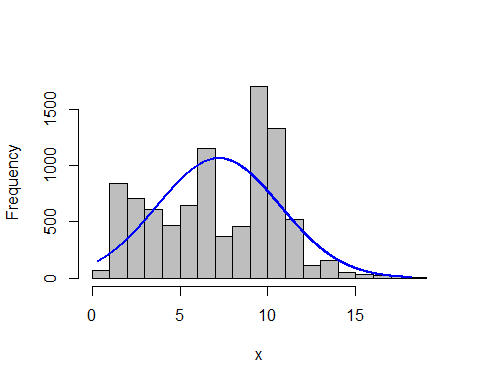
plotNormalHistogram(x$humidity)



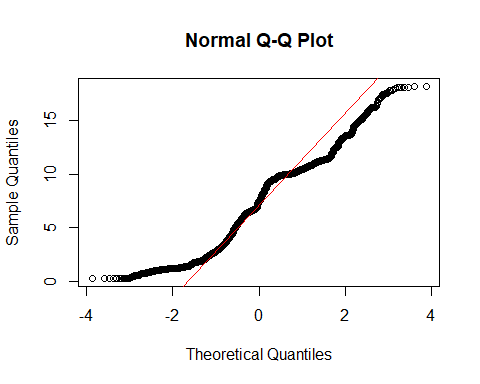
qqnorm(df$humidity)  
qqline(df$humidity, col="red")



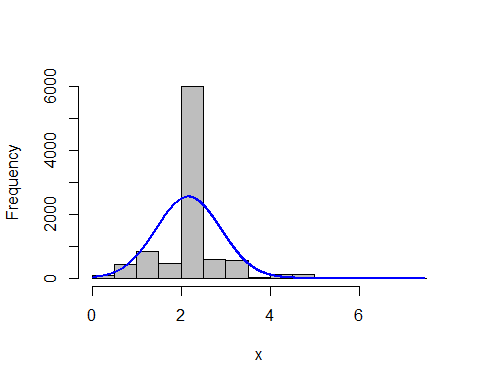
plotNormalHistogram(x$wind)



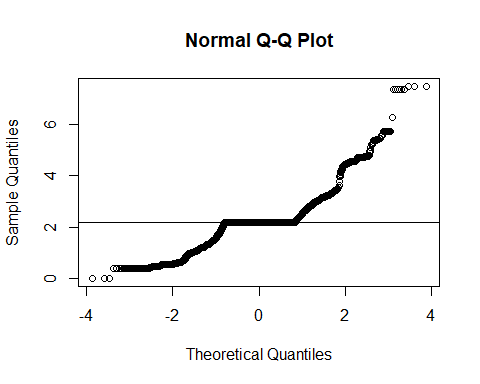
qqnorm(df$wind)  
qqline(df$wind, col="red")



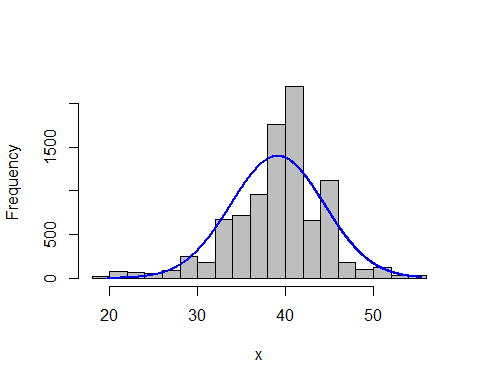
plotNormalHistogram(x$distance)



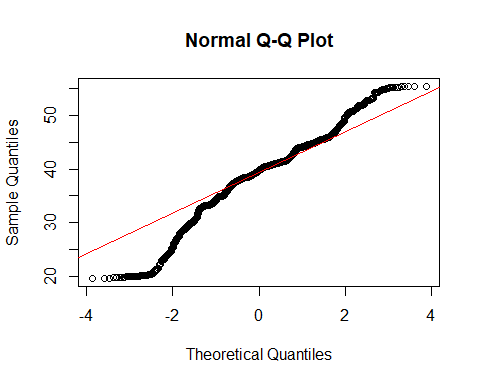
qqnorm(df$distance)  
qqline(df$distance)



plotNormalHistogram(x$temp)



qqnorm(df$temp)  
qqline(df$temp, col="red")



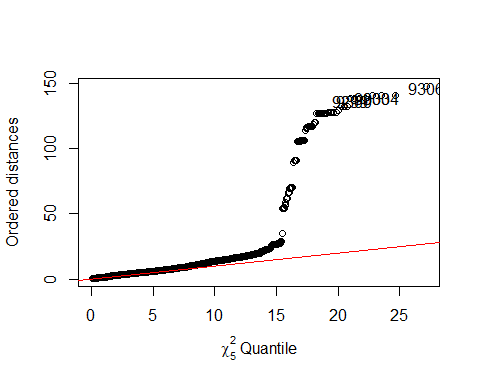
# Deviation from normality can be observed in our variables. Let’s check for multivariate analysis using chi-squre plot

## CORRELATION, COVARIANCE AND DISTANCE

#We are calculating for: clouds, pressure, rain, humidity, wind, distance, surge\_multiplier, temp, price  
covariance<-cov(x) #variamce-covariance matrix created  
correlation<-cor(x) #standardized  
#colmeans  
cm<-colMeans(x)  
distance<-dist(scale(x,center=FALSE))  
#Calculating di(generalized distance for all observations of our data)  
d <- apply(x, MARGIN = 1, function(x) + t(x - cm) %\*% solve(covariance) %\*% (x - cm))

## The sorted distance are now plotted against the appropriate quantiles of the chi-distribution

plot(qc <- qchisq((1:nrow(x) - 1/2) / nrow(x), df = 5), sd <- sort(d),xlab = expression(paste(chi[5]^2, " Quantile")),ylab = "Ordered distances")  
oups <- which(rank(abs(qc - sd), ties = "random") > nrow(x) - 5)  
text(qc[oups], sd[oups] - 1.5,oups)  
abline(a=0,b=1,col="red")



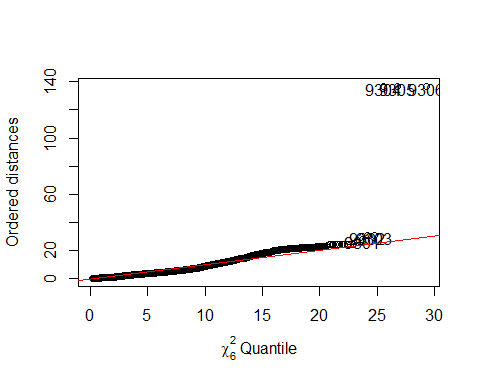
#Our observations seems to deviate from linearity after a certain point

# There is a complete deviation from Normality. We will apply the log transformation on our dataset.

#x\_new<-x+1  
#x\_new=log(x - (min(x) - 1))  
x\_new<-log(x[,c(2,4,5,6,7)])

covariance<-cov(x\_new) #variamce-covariance matrix created  
correlation<-cor(x\_new) #standardized  
#colmeans  
cm<-colMeans(x\_new)  
distance<-dist(scale(x\_new,center=FALSE))  
#Calculating di(generalized distance for all observations of our data)  
d <- apply(x\_new, MARGIN = 1, function(x\_new) + t(x\_new - cm) %\*% solve(covariance) %\*% (x\_new - cm))

plot(qc <- qchisq((1:nrow(x\_new) - 1/2) / nrow(x\_new), df = 6), sd <- sort(d),xlab = expression(paste(chi[6]^2, " Quantile")),ylab = "Ordered distances")  
oups <- which(rank(abs(qc - sd), ties = "random") > nrow(x) - 6)  
text(qc[oups], sd[oups] - 1.5,oups)  
abline(a=0,b=1,col="red")



# We have normalized the data..

## Pca || T-test || F-test

# Get the Correlations between the measurements

cor(x\_new)

## pressure humidity wind distance temp  
## pressure 1.00000000 0.037667720 -0.57053758 0.091084564 -0.190802751  
## humidity 0.03766772 1.000000000 -0.34918388 0.007457245 0.342394254  
## wind -0.57053758 -0.349183876 1.00000000 -0.036561758 0.107101055  
## distance 0.09108456 0.007457245 -0.03656176 1.000000000 -0.002908013  
## temp -0.19080275 0.342394254 0.10710106 -0.002908013 1.000000000

sapply(x\_new, sd, na.rm = TRUE)

## pressure humidity wind distance temp   
## 0.01242771 0.16241660 0.67116505 0.39696563 0.14798758

#**There are not considerable differences between these standard deviations..** Still let's see the PCAs.

# Using prcomp to compute the principal components (eigenvalues and eigenvectors).

# With scale=TRUE, variable means are set to zero, and variances set to one

x\_pca <- prcomp(x\_new,scale=TRUE)  
x\_pca

## Standard deviations (1, .., p=5):  
## [1] 1.3050862 1.1732928 0.9966622 0.7718227 0.5754028  
##   
## Rotation (n x k) = (5 x 5):  
## PC1 PC2 PC3 PC4 PC5  
## pressure -0.6258199 0.23938719 -0.01737613 0.51939957 -0.53006170  
## humidity -0.3194217 -0.65993093 -0.04083935 -0.52331376 -0.43236070  
## wind 0.6908793 0.04300622 0.11994313 0.09716528 -0.70498852  
## distance -0.1208578 0.04613105 0.98636820 -0.09381744 0.03926031  
## temp 0.1199934 -0.70937108 0.10354529 0.66190935 0.18316287

summary(x\_pca)

## Importance of components:  
## PC1 PC2 PC3 PC4 PC5  
## Standard deviation 1.3051 1.1733 0.9967 0.7718 0.57540  
## Proportion of Variance 0.3407 0.2753 0.1987 0.1191 0.06622  
## Cumulative Proportion 0.3407 0.6160 0.8146 0.9338 1.00000

#x\_pca$rotation

# We see that the first four components account for nearly 80% of the total variance.

# sample scores stored in x\_pca$x # singular values (square roots of eigenvalues) stored in x\_pca$sdev

# loadings (eigenvectors) are stored in x\_pca$rotation # variable means stored in x\_pca$center

# variable standard deviations stored in x\_pca$scale

# A table containing eigenvalues and %’s accounted, follows

# Eigenvalues are sdev^2

(eigen\_x <- x\_pca$sdev^2)

## [1] 1.7032500 1.3766159 0.9933355 0.5957103 0.3310884

names(eigen\_x) <- paste("PC",1:5,sep="")  
eigen\_x

## PC1 PC2 PC3 PC4 PC5   
## 1.7032500 1.3766159 0.9933355 0.5957103 0.3310884

sumlambdas <- sum(eigen\_x)  
sumlambdas #total sample variance

## [1] 5

propvar <- eigen\_x/sumlambdas  
propvar

## PC1 PC2 PC3 PC4 PC5   
## 0.34065000 0.27532318 0.19866709 0.11914205 0.06621768

cumvar\_x <- cumsum(propvar)  
cumvar\_x

## PC1 PC2 PC3 PC4 PC5   
## 0.3406500 0.6159732 0.8146403 0.9337823 1.0000000

matlambdas <- rbind(eigen\_x,propvar,cumvar\_x)  
rownames(matlambdas) <- c("Eigenvalues","Prop. variance","Cum. prop. variance")  
round(matlambdas,4)

## PC1 PC2 PC3 PC4 PC5  
## Eigenvalues 1.7033 1.3766 0.9933 0.5957 0.3311  
## Prop. variance 0.3407 0.2753 0.1987 0.1191 0.0662  
## Cum. prop. variance 0.3407 0.6160 0.8146 0.9338 1.0000

# Sample scores stored in x\_pca$x

# We need to calculate the scores on each of these components for each individual in our sample.

#x\_pca$x  
xtyp\_pca <- cbind(data.frame(df$price),x\_pca$x)  
str(xtyp\_pca)

## 'data.frame': 9306 obs. of 6 variables:  
## $ df.price: num 16.7 16.7 8.5 16.5 16.7 ...  
## $ PC1 : num -2.29 -1.73 -1.6 -1.56 -1.52 ...  
## $ PC2 : num -1.003 -0.967 -1.014 -1.017 -1.029 ...  
## $ PC3 : num -0.1144 -0.0228 -1.0382 -1.1765 -1.4464 ...  
## $ PC4 : num -0.225 -0.232 -0.134 -0.12 -0.094 ...  
## $ PC5 : num 0.647 0.021 -0.0276 -0.0579 -0.0686 ...

#xtyp\_pca

# Merging price column

colnames(xtyp\_pca)[colnames(xtyp\_pca)=="df.price"] <- "price"  
str(xtyp\_pca)

## 'data.frame': 9306 obs. of 6 variables:  
## $ price: num 16.7 16.7 8.5 16.5 16.7 ...  
## $ PC1 : num -2.29 -1.73 -1.6 -1.56 -1.52 ...  
## $ PC2 : num -1.003 -0.967 -1.014 -1.017 -1.029 ...  
## $ PC3 : num -0.1144 -0.0228 -1.0382 -1.1765 -1.4464 ...  
## $ PC4 : num -0.225 -0.232 -0.134 -0.12 -0.094 ...  
## $ PC5 : num 0.647 0.021 -0.0276 -0.0579 -0.0686 ...

# Sample scores stoted. x\_pca$x

# T-Test– We see that true difference in all the means is different from zero.

t.test(xtyp\_pca$PC1,xtyp\_pca$price,var.equal = TRUE)

##   
## Two Sample t-test  
##   
## data: xtyp\_pca$PC1 and xtyp\_pca$price  
## t = -265.73, df = 18610, p-value < 2.2e-16  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -16.79675 -16.55077  
## sample estimates:  
## mean of x mean of y   
## -1.534642e-14 1.667376e+01

t.test(xtyp\_pca$PC2,xtyp\_pca$price,var.equal = TRUE)

##   
## Two Sample t-test  
##   
## data: xtyp\_pca$PC2 and xtyp\_pca$price  
## t = -266.92, df = 18610, p-value < 2.2e-16  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -16.79620 -16.55132  
## sample estimates:  
## mean of x mean of y   
## 4.850155e-15 1.667376e+01

t.test(xtyp\_pca$PC3,xtyp\_pca$price,var.equal = TRUE)

##   
## Two Sample t-test  
##   
## data: xtyp\_pca$PC3 and xtyp\_pca$price  
## t = -268.34, df = 18610, p-value < 2.2e-16  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -16.79555 -16.55197  
## sample estimates:  
## mean of x mean of y   
## -3.485127e-16 1.667376e+01

t.test(xtyp\_pca$PC4,xtyp\_pca$price,var.equal = TRUE)

##   
## Two Sample t-test  
##   
## data: xtyp\_pca$PC4 and xtyp\_pca$price  
## t = -269.84, df = 18610, p-value < 2.2e-16  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -16.79488 -16.55264  
## sample estimates:  
## mean of x mean of y   
## 1.371754e-14 1.667376e+01

t.test(xtyp\_pca$PC5,xtyp\_pca$price,var.equal = TRUE)

##   
## Two Sample t-test  
##   
## data: xtyp\_pca$PC5 and xtyp\_pca$price  
## t = -270.85, df = 18610, p-value < 2.2e-16  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -16.79443 -16.55309  
## sample estimates:  
## mean of x mean of y   
## -1.304992e-14 1.667376e+01

#F-Test #Testing Variation

# Variance Test- Test for variance

var.test(xtyp\_pca$PC1,xtyp\_pca$price)

##   
## F test to compare two variances  
##   
## data: xtyp\_pca$PC1 and xtyp\_pca$price  
## F = 0.048752, num df = 9305, denom df = 9305, p-value < 2.2e-16  
## alternative hypothesis: true ratio of variances is not equal to 1  
## 95 percent confidence interval:  
## 0.04681082 0.05077444  
## sample estimates:  
## ratio of variances   
## 0.04875236

var.test(xtyp\_pca$PC2,xtyp\_pca$price)

##   
## F test to compare two variances  
##   
## data: xtyp\_pca$PC2 and xtyp\_pca$price  
## F = 0.039403, num df = 9305, denom df = 9305, p-value < 2.2e-16  
## alternative hypothesis: true ratio of variances is not equal to 1  
## 95 percent confidence interval:  
## 0.03783386 0.04103737  
## sample estimates:  
## ratio of variances   
## 0.03940307

var.test(xtyp\_pca$PC3,xtyp\_pca$price)

##   
## F test to compare two variances  
##   
## data: xtyp\_pca$PC3 and xtyp\_pca$price  
## F = 0.028432, num df = 9305, denom df = 9305, p-value < 2.2e-16  
## alternative hypothesis: true ratio of variances is not equal to 1  
## 95 percent confidence interval:  
## 0.02730007 0.02961165  
## sample estimates:  
## ratio of variances   
## 0.02843238

var.test(xtyp\_pca$PC4,xtyp\_pca$price)

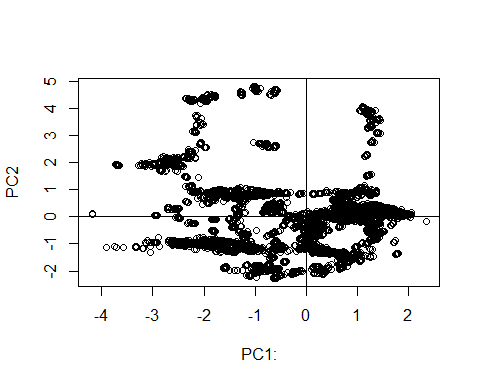
##   
## F test to compare two variances  
##   
## data: xtyp\_pca$PC4 and xtyp\_pca$price  
## F = 0.017051, num df = 9305, denom df = 9305, p-value < 2.2e-16  
## alternative hypothesis: true ratio of variances is not equal to 1  
## 95 percent confidence interval:  
## 0.01637204 0.01775832  
## sample estimates:  
## ratio of variances   
## 0.0170511

var.test(xtyp\_pca$PC5,xtyp\_pca$price)

##   
## F test to compare two variances  
##   
## data: xtyp\_pca$PC5 and xtyp\_pca$price  
## F = 0.0094768, num df = 9305, denom df = 9305, p-value < 2.2e-16  
## alternative hypothesis: true ratio of variances is not equal to 1  
## 95 percent confidence interval:  
## 0.009099379 0.009869852  
## sample estimates:  
## ratio of variances   
## 0.009476789

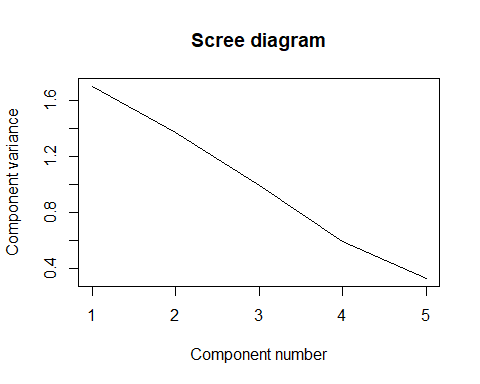
# Plotting the scores of Pricipal Component 1 and Principal component 2

plot(xtyp\_pca$PC1, xtyp\_pca$PC2,xlab="PC1:", ylab="PC2")  
abline(h=0)  
abline(v=0)

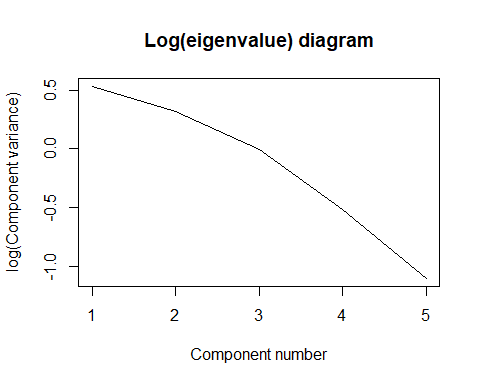


# Plotting the Variance of Principal Components

plot(eigen\_x, xlab = "Component number", ylab = "Component variance", type = "l", main = "Scree diagram")

 #Plotting the Log variance of COmponents

plot(log(eigen\_x), xlab = "Component number",ylab = "log(Component variance)", type="l",main = "Log(eigenvalue) diagram")

 #Variance of the principal components

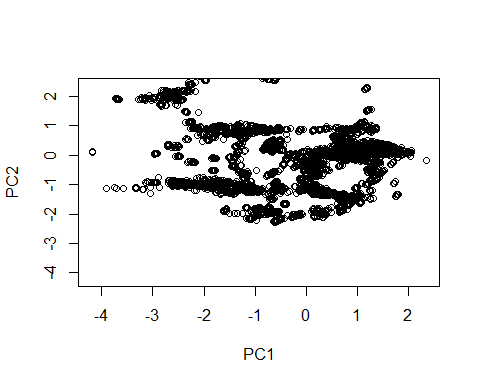
#View(x\_pca)  
diag(cov(x\_pca$x))

## PC1 PC2 PC3 PC4 PC5   
## 1.7032500 1.3766159 0.9933355 0.5957103 0.3310884

#x\_pca$x[,1]  
#x\_pca$x

# Plotting the scores

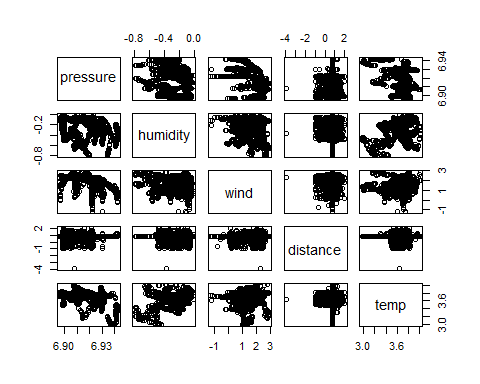
xlim <- range(x\_pca$x[,1])  
plot(x\_pca$x,xlim=xlim,ylim=xlim)



#x\_pca$rotation[,1]  
#x\_pca$rotation

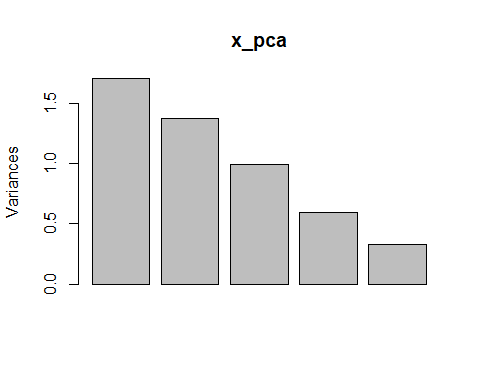
# Scatter plot matrix of the actual data

plot(x\_new)

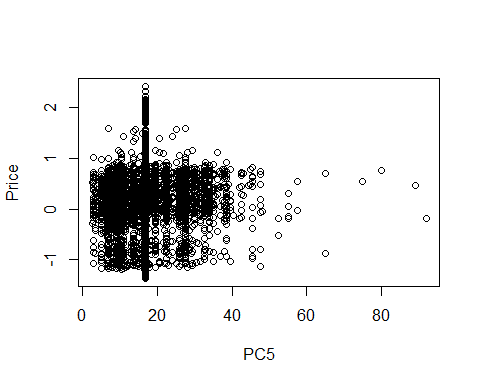
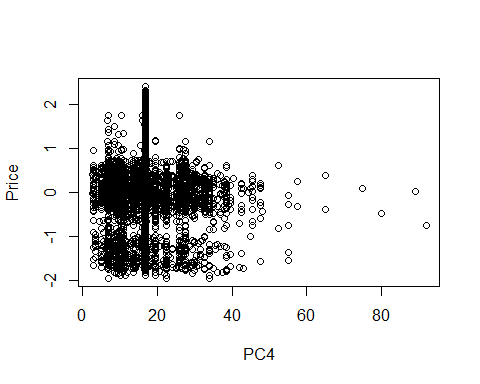
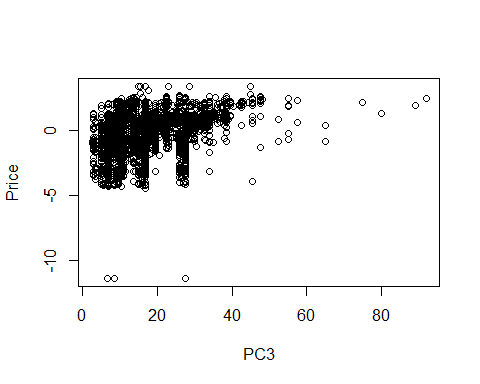
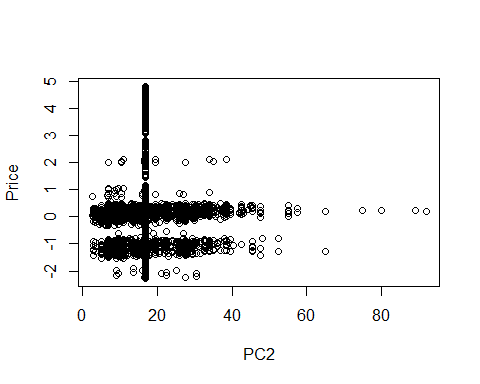
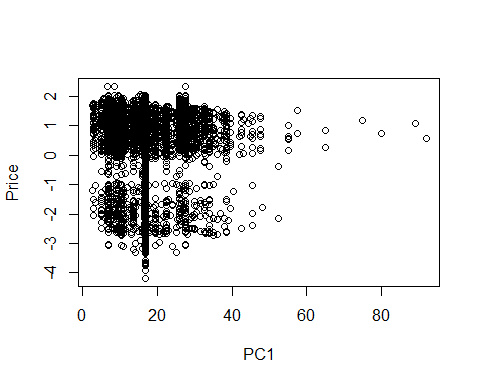


# Variance plot for each component. We can see that all components play a dominant role.

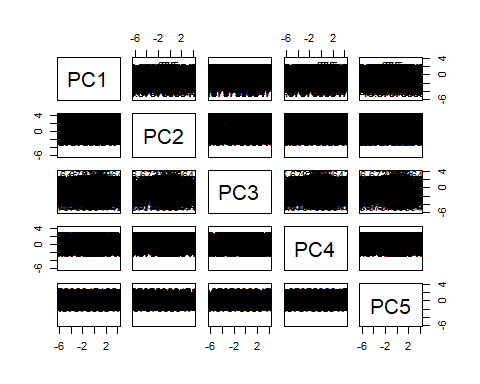
plot(x\_pca)



#get the original value of the data based on PCA  
center <- x\_pca$center  
scale <- x\_pca$scale  
new\_x <- as.matrix(x\_new)  
#drop(scale(new\_x,center=center, scale=scale)%\*%x\_pca$rotation[,1])  
#predict(x\_pca)[,1]  
#The aboved two gives us the same thing. predict is a good function to know.  
  
x\_new$price<-df$price  
out <- sapply(1:5, function(i){plot(x\_new$price,x\_pca$x[,i],xlab=paste("PC",i,sep=""),  
 ylab="Price")})



pairs(x\_pca$x[,1:5], ylim = c(-6,4),xlim = c(-6,4),panel=function(x,y,...){text(x,y,x\_new$price)})



# CLuster Analysis

#install.packages("cluster", #lib="/Library/Frameworks/R.framework/Versions/3.5/Resources/library")  
library(cluster)

## Warning: package 'cluster' was built under R version 3.5.3

# Pulling the numerical variables in the “Cluster” dataframe. Scaling the values..

cluster <- df[,c(1,2,4,5,7,9)]  
matstd.cluster <- scale(cluster)  
dim(matstd.cluster)

## [1] 9306 6

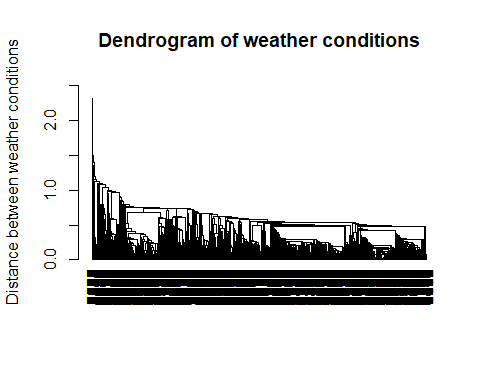
# Calculating the distance between all observations..

dist.cluster <- dist(matstd.cluster, method="euclidean")  
length(dist.cluster)

## [1] 43296165

# Invoking hclust command (cluster analysis by single linkage method)

hclust\_cluster <- hclust(dist.cluster, method = "single")  
par(mar=c(6, 4, 4, 2) + 0.1)  
plot(as.dendrogram(hclust\_cluster),ylab="Distance between weather conditions",ylim=c(0,2.5),main="Dendrogram of weather conditions")



# K-means Clustering for k=2 and then computing the percentage variance

#attach(cluster)  
matstd.cluster <- scale(cluster)  
# Computing the percentage of variation accounted for. Two clusters  
kmeans2.cluster <- kmeans(matstd.cluster,2,nstart = 10)  
perc.var.2 <- round(100\*(1 - kmeans2.cluster$betweenss/kmeans2.cluster$totss),1)  
names(perc.var.2) <- "Perc. 2 clus"  
perc.var.2

## Perc. 2 clus   
## 75.4

# Computing the percentage of variation accounted for. Three clusters

kmeans3.cluster <- kmeans(matstd.cluster,3,nstart = 10)  
perc.var.3 <- round(100\*(1 - kmeans3.cluster$betweenss/kmeans3.cluster$totss),1)  
names(perc.var.3) <- "Perc. 3 clus"  
perc.var.3

## Perc. 3 clus   
## 58.3

# Computing the percentage of variation accounted for. Four clusters

kmeans4.cluster <- kmeans(matstd.cluster,4,nstart = 10)  
perc.var.4 <- round(100\*(1 - kmeans4.cluster$betweenss/kmeans4.cluster$totss),1)  
names(perc.var.4) <- "Perc. 4 clus"  
perc.var.4

## Perc. 4 clus   
## 50.9

# Computing the percentage of variation accounted for. Five clusters

kmeans5.cluster <- kmeans(matstd.cluster,5,nstart = 10)  
perc.var.5 <- round(100\*(1 - kmeans5.cluster$betweenss/kmeans5.cluster$totss),1)  
names(perc.var.5) <- "Perc. 5 clus"  
perc.var.5

## Perc. 5 clus   
## 44.5

# Computing the percentage of variation accounted for. Six clusters

kmeans6.cluster <- kmeans(matstd.cluster,6,nstart = 10)  
perc.var.6 <- round(100\*(1 - kmeans6.cluster$betweenss/kmeans6.cluster$totss),1)  
names(perc.var.6) <- "Perc. 6 clus"  
perc.var.6

## Perc. 6 clus   
## 38.7

# plots to compare

#install.packages("VIM")  
library(VIM)

## Warning: package 'VIM' was built under R version 3.5.3

## Loading required package: colorspace

## Loading required package: grid

## Loading required package: data.table

## VIM is ready to use.   
## Since version 4.0.0 the GUI is in its own package VIMGUI.  
##   
## Please use the package to use the new (and old) GUI.

## Suggestions and bug-reports can be submitted at: https://github.com/alexkowa/VIM/issues

##   
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':  
##   
## sleep

#install.packages("tidyverse")  
library(tidyverse) # data manipulation

## Warning: package 'tidyverse' was built under R version 3.5.3

## -- Attaching packages ----------------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.1.1 v purrr 0.3.0   
## v tibble 2.0.1 v dplyr 0.8.0.1  
## v tidyr 0.8.3 v stringr 1.3.1   
## v readr 1.3.1 v forcats 0.3.0

## Warning: package 'ggplot2' was built under R version 3.5.3

## Warning: package 'tidyr' was built under R version 3.5.3

## -- Conflicts -------------------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::between() masks data.table::between()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::first() masks data.table::first()  
## x dplyr::lag() masks stats::lag()  
## x dplyr::last() masks data.table::last()  
## x purrr::transpose() masks data.table::transpose()

#install.packages("cluster")  
library(cluster) # clustering algorithms  
#install.packages("factoextra")  
library(factoextra)

## Warning: package 'factoextra' was built under R version 3.5.3

## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ

p1 <- fviz\_cluster(kmeans2.cluster, geom = "point", data = cluster) + ggtitle("k = 2")  
p2 <- fviz\_cluster(kmeans3.cluster, geom = "point", data = cluster) + ggtitle("k = 3")  
p3 <- fviz\_cluster(kmeans4.cluster, geom = "point", data = cluster) + ggtitle("k = 4")  
p4 <- fviz\_cluster(kmeans5.cluster, geom = "point", data = cluster) + ggtitle("k = 5")  
p5 <- fviz\_cluster(kmeans6.cluster, geom = "point", data = cluster) + ggtitle("k = 6")

# Grid plot

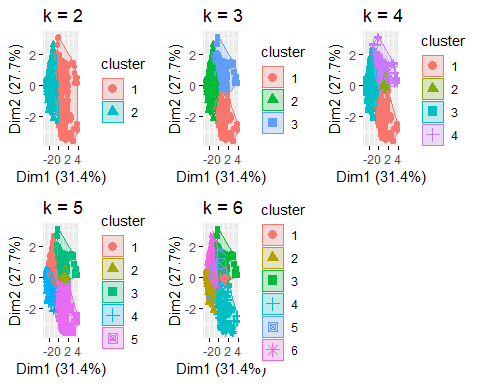
library(gridExtra)

## Warning: package 'gridExtra' was built under R version 3.5.3

##   
## Attaching package: 'gridExtra'

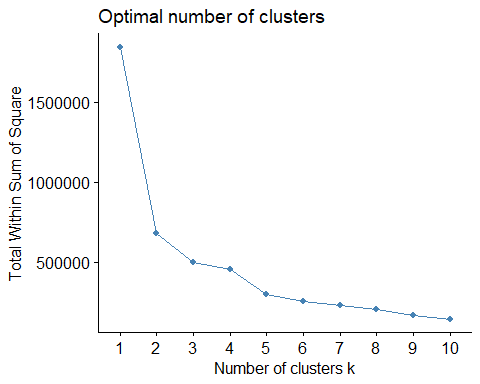
## The following object is masked from 'package:dplyr':  
##   
## combine

grid.arrange(p1, p2, p3, p4,p5, nrow = 2)



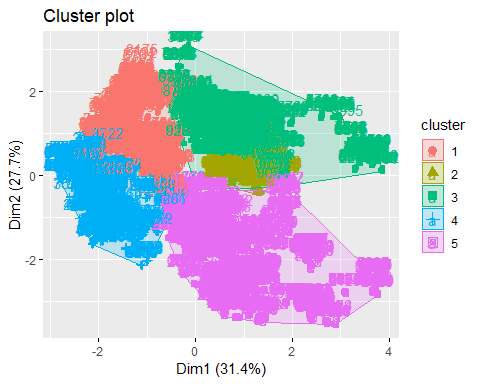
# Determining Optimal Clusters

set.seed(123)  
fviz\_nbclust(cluster, kmeans, method = "wss")

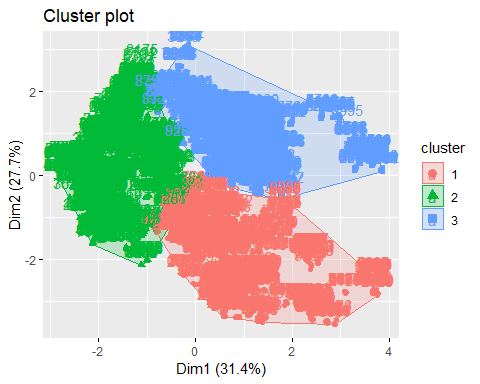


# K=5 seems optimal number of clusters

fviz\_cluster(kmeans5.cluster, data = cluster)



fviz\_cluster(kmeans3.cluster, data = cluster)



# Adding cluster number to the file for each observation-

clusterFile <- cbind(df, clusterNum = kmeans5.cluster$cluster)  
head(clusterFile)

## clouds pressure rain humidity wind date\_time.x distance  
## 1 0.87 1014.39 0 0.92 1.46 2018-11-26 04:34:05 2.168125  
## 2 0.86 1014.17 0 0.93 2.57 2018-11-26 05:34:13 2.168125  
## 3 0.86 1014.17 0 0.93 2.59 2018-11-26 05:34:58 1.440000  
## 4 0.86 1014.17 0 0.93 2.65 2018-11-26 05:36:38 1.360000  
## 5 0.86 1014.17 0 0.93 2.65 2018-11-26 05:36:38 1.220000  
## 6 0.95 1013.78 0 0.92 2.59 2018-11-26 05:42:57 1.340000  
## surge\_multiplier temp price day time date\_time clusterNum  
## 1 1.018365 41.04 16.67376 Monday 04:34:05 2018-11-26 4  
## 2 1.018365 40.63 16.67376 Monday 05:34:13 2018-11-26 4  
## 3 1.000000 40.63 8.50000 Monday 05:34:58 2018-11-26 4  
## 4 1.000000 40.61 16.50000 Monday 05:36:38 2018-11-26 4  
## 5 1.000000 40.61 16.67376 Monday 05:36:38 2018-11-26 4  
## 6 1.000000 40.72 26.50000 Monday 05:42:57 2018-11-26 4

**Factor Analysis:**

We concluded during Principal Component Analysis that all the variables of our dataset are not highly correlated and all are significant. Hence, we did not apply Factor Analysis on our data.

