Lyft-Uber-Price-Prediction

7 November 2019

IMPORTING DATASETS AND CLEANING THEM

Importing dataset cab_rides

```
cab_rides <- read.csv("C:/Users/AJAY/Downloads/Multivariate/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 = "1970-01-0
1 00:53:20", tz="GMT")
```

Importing dataset weather

```
weather <-
read.csv("C:/Users/nisht/AJAY/Downloads/Multivariate/Project/weather.csv")
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)

merge_data$rain<-as.numeric(merge_data$rain)
merge_data$rain[is.na(merge_data$rain)]<-0

for ( i in 1:length(merge_data$rain)){
  if(merge_data$rain[i]>0 & merge_data$rain[i]<=0.30){
    merge_data$rain[i]=1
  }
```

```
}

for ( i in 1:length(merge_data$rain)){
  if(merge_data$rain[i]>=0.30 & merge_data$rain[i]!=1){
    merge_data$rain[i]=2
  }
}

merge_data$rain = factor(merge_data$rain,
                         levels = c(0,1,2),
                         labels = c(0,1,2))
```

Handling Missing values

```
#Extracting the numerical columns in a new dataframe "df"
merge_data$temp<-merge_data[,c(2)] #renaming a column
df<-merge_data[,c(4,5,8,9,10,11,22,16)]



#Data preparation
#Dealing with missing values
#summary(merge_data)
#summary(df)

merge_data$distance = ifelse(is.na(merge_data$distance),
                             ave(merge_data$distance , FUN = function
(x) mean(x, na.rm = TRUE)),
                             merge_data$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$price = ifelse(is.na(df$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(1,2,3,4,6,7,8)]
head(x)
```

```
##   clouds pressure humidity wind distance  temp    price
## 1   0.87  1014.39     0.92 1.46 2.168125 41.04 16.67376
## 2   0.86  1014.17     0.93 2.57 2.168125 40.63 16.67376
## 3   0.86  1014.17     0.93 2.59 1.440000 40.63  8.50000
## 4   0.86  1014.17     0.93 2.65 1.360000 40.61 16.50000
## 5   0.86  1014.17     0.93 2.65 1.220000 40.61 16.67376
## 6   0.95  1013.78     0.92 2.59 1.340000 40.72 26.50000
```

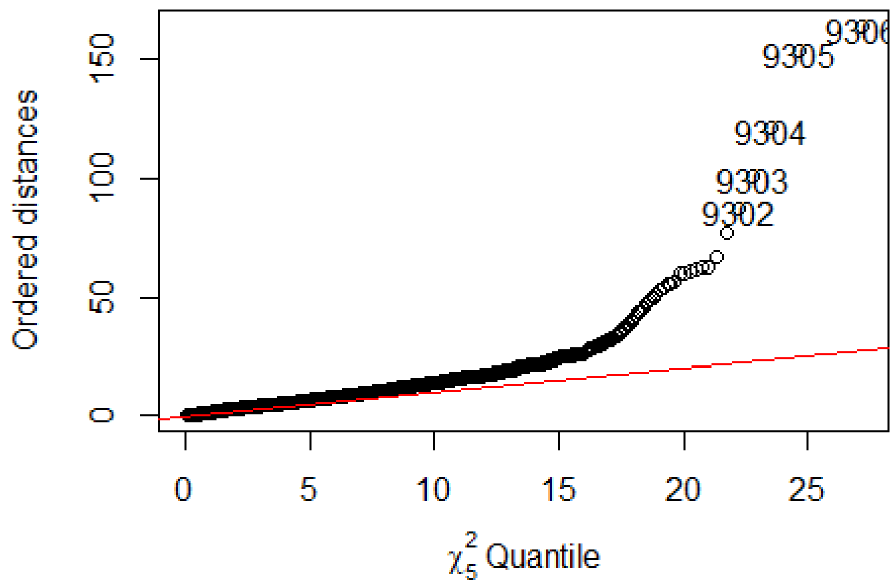
Let's check for multivariate analysis using chi-squre plot

CORRELATION, COVARIANCE AND DISTANCE

```
#We are calculating for: clouds, pressure, rain, humidity, wind, distance, te
mp
covariance<-cov(x) #variance-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

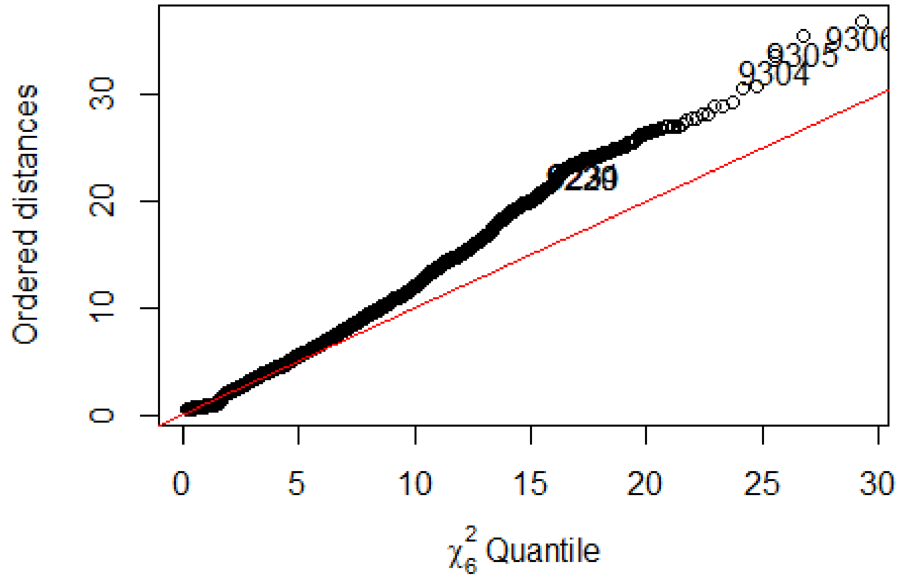
```
plot(qc <- qchisq((1:nrow(x) - 1/2) / nrow(x), df = 5), sd <- sort(d),xlab =
expression(paste(chi[5]^2, " Quantile")),ylab = "Ordered distances")
oups <- which(rank(abs(qc - sd), ties = "random") > nrow(x) - 5)
text(qc[oups], sd[oups] - 1.5,oups)
abline(a=0,b=1,col="red")
```




```
#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+1)

covariance<-cov(x_new) #variance-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(covar
iance) %*% (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

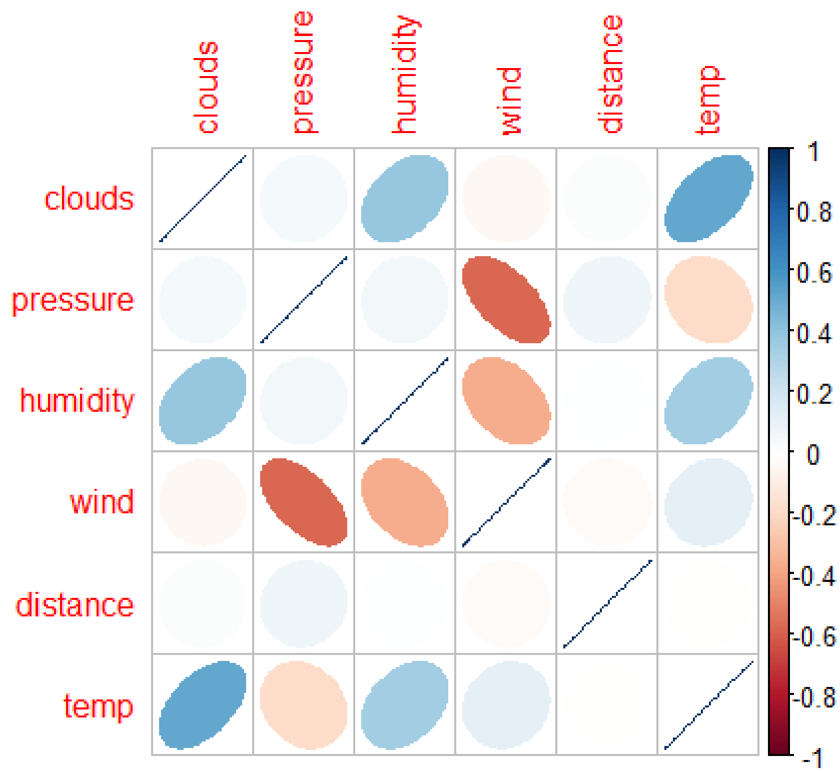
Get the Correlations between the measurements

```
x_new<-x_new[-7]
cor(x_new)

##               clouds    pressure     humidity        wind     distance
## clouds    1.00000000  0.04290237  0.380934772 -0.03065604  0.018863302
## pressure  0.04290237  1.00000000  0.051631425 -0.57889122  0.063942888
## humidity  0.38093477  0.05163143  1.000000000 -0.36402079  0.007197305
## wind     -0.03065604 -0.57889122 -0.364020786  1.00000000 -0.026571018
## distance  0.01886330  0.06394289  0.007197305 -0.02657102  1.000000000
## temp      0.51904189 -0.18953119  0.340366261  0.11773558 -0.003364190
##                 temp
## clouds    0.51904189
## pressure -0.18953119
## humidity  0.34036626
## wind      0.11773558
## distance -0.00336419
## temp      1.00000000

sapply(x_new, sd, na.rm = TRUE)
```

```
##    clouds  pressure  humidity      wind  distance      temp 
## 0.1997804 0.0124154 0.0686079 0.5304428 0.2347243 0.1437917

#There are not considerable differences between these standard deviations.. S
till let's see the PCAs.
```

Let’s Visualize Correlation..

```
library(corrplot)

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

## corrplot 0.84 loaded

corrplot(cor(x_new), method="ellipse")
```



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$rotation
summary(x_pca)

## Importance of components:
##                           PC1    PC2    PC3    PC4    PC5     PC6
## Standard deviation     1.3666 1.3004 0.9992 0.8339 0.6668 0.55060
## Proportion of Variance 0.3113 0.2818 0.1664 0.1159 0.0741 0.05053
## Cumulative Proportion  0.3113 0.5931 0.7595 0.8754 0.9495 1.00000

#x_pca$rotation
```

Each of these explains a percent of total vaiation in the dataset. PC1 explains 27.8% of total variance, PC2 explains 26% of total variance.. we need to go to PC5 to get a ver accurate view on where it stands in relation to other samples as PC1-PC5 can explain 89.9% of the variance. sample scores stored in x_pca$xsingularvalues(squarerootsofeigenvalues)storedinx_pca$sdev loadings (eigenvectors) are stored in x_pca$rotationvariablemeansstoredinx_pca$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
names(eigen_x) <- paste("PC",1:6,sep="")
#eigen_x
sumlambdas <- sum(eigen_x)
sumlambdas #total sample variance

## [1] 6

propvar <- eigen_x/sumlambdas
#propvar
cumvar_x <- cumsum(propvar)
#cumvar_x
matlambdas <- rbind(eigen_x,propvar,cumvar_x)
rownames(matlambdas) <- c("Eigenvalues","Prop. variance","Cum. prop. variance
")
round(matlambdas,4)

##                        PC1    PC2    PC3    PC4    PC5    PC6
## Eigenvalues         1.8675 1.6910 0.9983 0.6954 0.4446 0.3032
## Prop. variance      0.3113 0.2818 0.1664 0.1159 0.0741 0.0505
## Cum. prop. variance 0.3113 0.5931 0.7595 0.8754 0.9495 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.

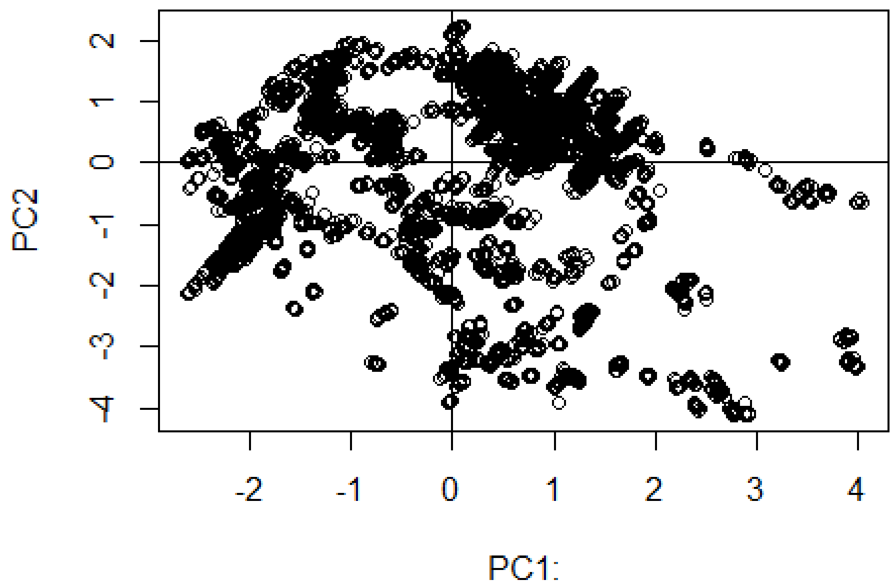
```
#x_pca$rotation
xtyp_pca <- cbind(data.frame(df$price),x_pca$x)
str(xtyp_pca)

## 'data.frame':    9306 obs. of  7 variables:
##  $ df.price: num  16.7 16.7 8.5 16.5 16.7 ...
##  $ PC1     : num  -2.03 -1.81 -1.76 -1.75 -1.74 ...
##  $ PC2     : num  -1.501 -1.095 -1.001 -0.972 -0.952 ...
##  $ PC3     : num  0.1432 0.0861 1.187 1.3248 1.5828 ...
##  $ PC4     : num  -0.474 -0.428 -0.309 -0.288 -0.261 ...
##  $ PC5     : num  0.1565 -0.0283 -0.0348 -0.0438 -0.0449 ...
##  $ PC6     : num  -0.56 -0.05892 -0.02739 -0.00333 0.0024 ...

#xtyp_pca
```

Merging price column

```
colnames(xtyp_pca)[colnames(xtyp_pca)=="df.price"] <- "price"
str(xtyp_pca)

## 'data.frame':    9306 obs. of  7 variables:
##  $ price: num  16.7 16.7 8.5 16.5 16.7 ...
##  $ PC1  : num  -2.03 -1.81 -1.76 -1.75 -1.74 ...
##  $ PC2  : num  -1.501 -1.095 -1.001 -0.972 -0.952 ...
##  $ PC3  : num  0.1432 0.0861 1.187 1.3248 1.5828 ...
##  $ PC4  : num  -0.474 -0.428 -0.309 -0.288 -0.261 ...
##  $ PC5  : num  0.1565 -0.0283 -0.0348 -0.0438 -0.0449 ...
##  $ PC6  : num  -0.56 -0.05892 -0.02739 -0.00333 0.0024 ...
```

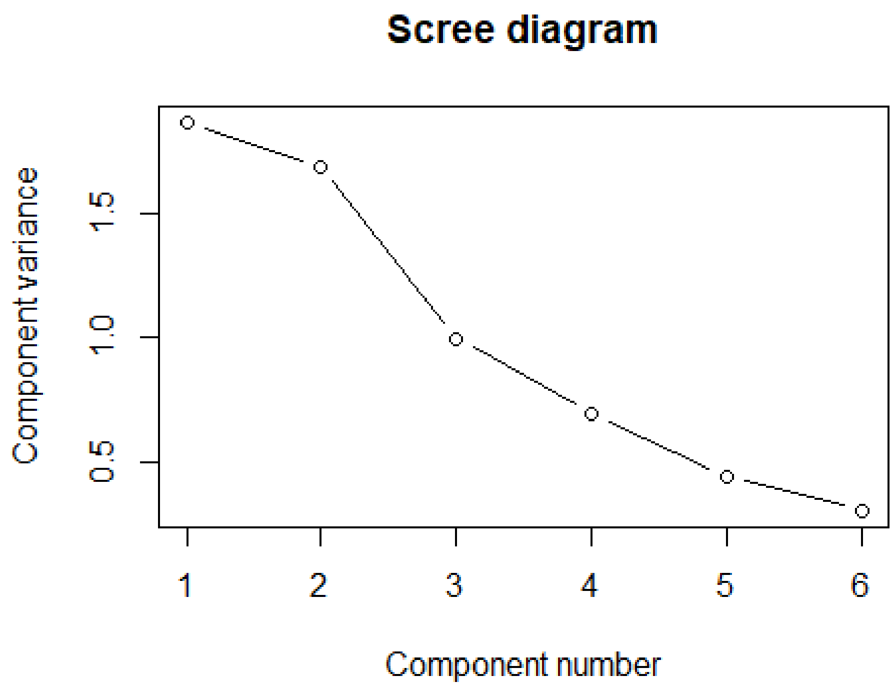
Plotting the scores of Pricipal Component 1 and Principal component 2

```
plot(xtyp_pca$PC1, xtyp_pca$PC2,xlab="PC1:", ylab="PC2")
abline(h=0)
abline(v=0)
```



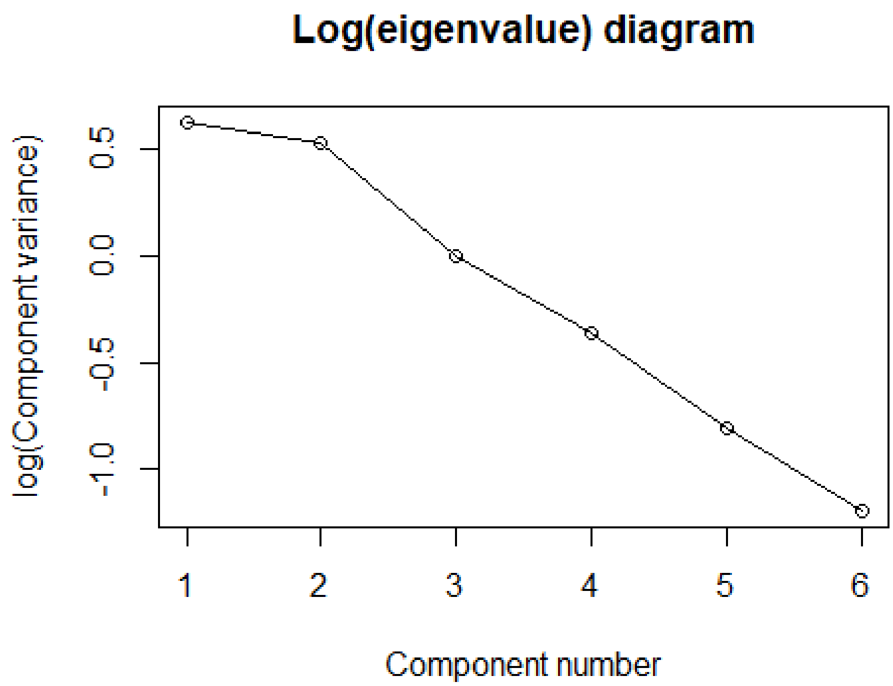
Plotting the Variance of Principal Components

```
plot(eigen_x, xlab = "Component number", ylab = "Component variance", type =
"b", main = "Scree diagram")
```



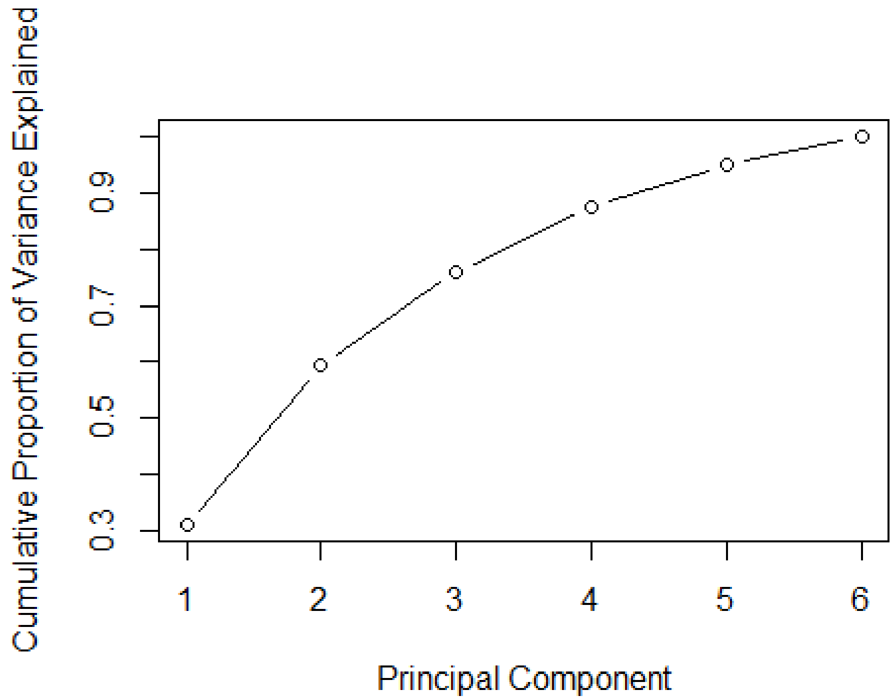
#Plotting the Log variance of COmponents

```
plot(log(eigen_x), xlab = "Component number",ylab = "log(Component variance)"
, type="o",main = "Log(eigenvalue) diagram")
```



#Cumulative scree plot

```
plot(cumsum(propvar), xlab = "Principal Component",
              ylab = "Cumulative Proportion of Variance Explained",
              type = "b")
```

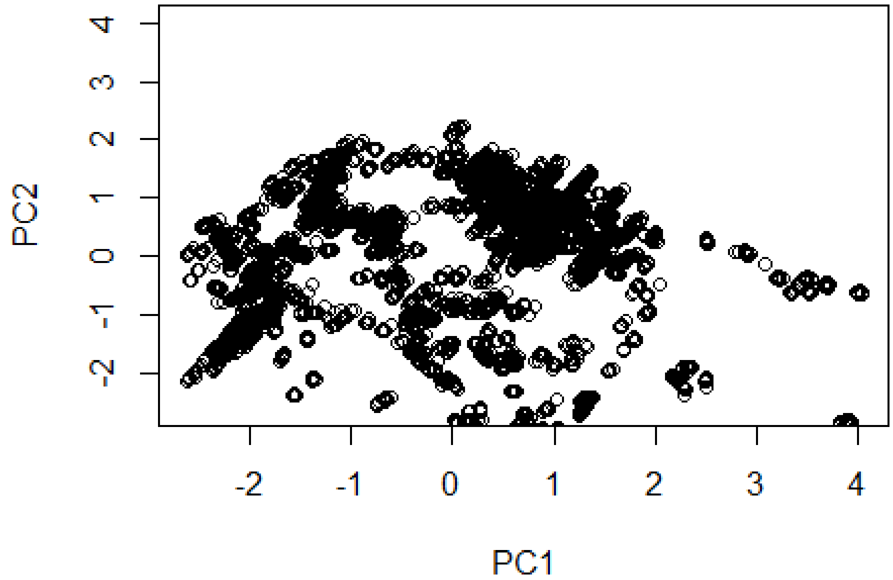


Variance of the principal components

```
#View(x_pca)
diag(cov(x_pca$x))

##       PC1       PC2       PC3       PC4       PC5       PC6
## 1.8675135 1.6909966 0.9983146 0.6953850 0.4446266 0.3031637

#x_pca$x[,1]
#x_pca$x
```

Plotting the scores

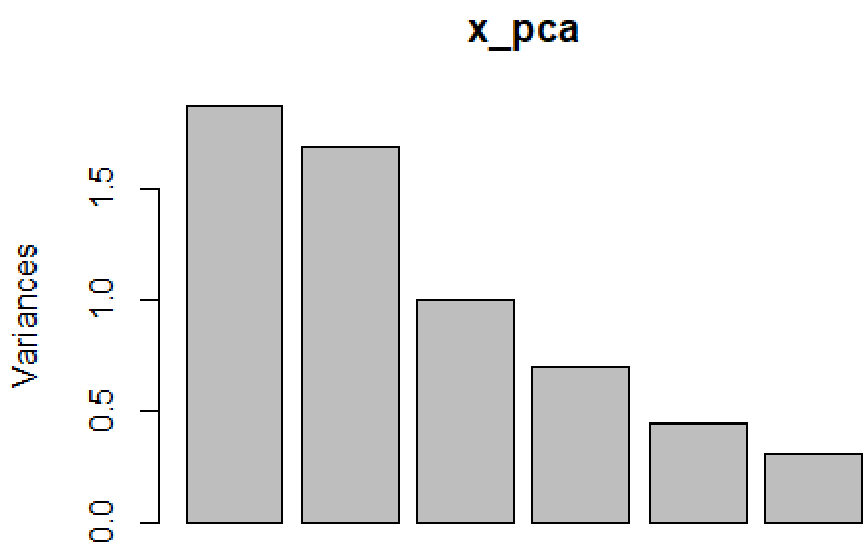
```
xlim <- range(x_pca$x[,1])
plot(x_pca$x,xlim=xlim,ylim=xlim)
```




```
#x_pca$rotation[,1]
#x_pca$rotation
```

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

```
plot(x_pca)
```



#Taking first 4 components

```
xtyp_pca<-xtyp_pca[1:4]
```

They are explaing 88% of the total variance.

Factor Analysis

```
library(psych)

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

#install.packages("psych", lib="/Library/Frameworks/R.framework/Versions/3.5/
Resources/library")
fit.pc <- principal(x_new, nfactors=4, rotate="varimax")
fit.pc

## Principal Components Analysis
## Call: principal(r = x_new, nfactors = 4, rotate = "varimax")
## Standardized loadings (pattern matrix) based upon correlation matrix
##           RC1   RC2   RC4  RC3   h2      u2 com
## clouds   0.90  0.12  0.10 0.01 0.83 0.17382 1.1
## pressure 0.01  0.95 -0.08 0.04 0.90 0.09521 1.0
## humidity 0.30  0.06  0.91 0.00 0.92 0.07603 1.2
## wind     0.13 -0.76 -0.50 0.00 0.85 0.15197 1.8
## distance 0.01  0.03  0.00 1.00 1.00 0.00015 1.0
## temp     0.81 -0.23  0.19 0.00 0.75 0.25060 1.3
##
##                        RC1  RC2  RC4  RC3
```

```
## SS loadings           1.56 1.55 1.14 1.00
## Proportion Var        0.26 0.26 0.19 0.17
## Cumulative Var        0.26 0.52 0.71 0.88
## Proportion Explained  0.30 0.30 0.22 0.19
## Cumulative Proportion 0.30 0.59 0.81 1.00
##
## Mean item complexity =  1.2
## Test of the hypothesis that 4 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.07
##  with the empirical chi square  1476.35  with prob <  NA
##
## Fit based upon off diagonal values = 0.93
```

```
round(fit.pc$values, 3)
```

```
## [1] 1.868 1.691 0.998 0.695 0.445 0.303
```

```
fit.pc$loadings
```

```
##
## Loadings:
##          RC1    RC2    RC4    RC3
## clouds    0.895  0.123
## pressure         0.947
## humidity  0.297         0.912
## wind      0.126 -0.761 -0.503
## distance                       0.999
## temp      0.811 -0.233  0.195
##
##                  RC1   RC2   RC4   RC3
## SS loadings    1.563 1.550 1.139 1.001
## Proportion Var 0.260 0.258 0.190 0.167
## Cumulative Var 0.260 0.519 0.709 0.875
```

The first 4 factors have an Eigenvalue >1 and which explains almost 88% of the variance. We can effectively reduce dimensionality from 6 to 4 while only losing about 11% of the variance.

Communalities

```
fit.pc$communality
```

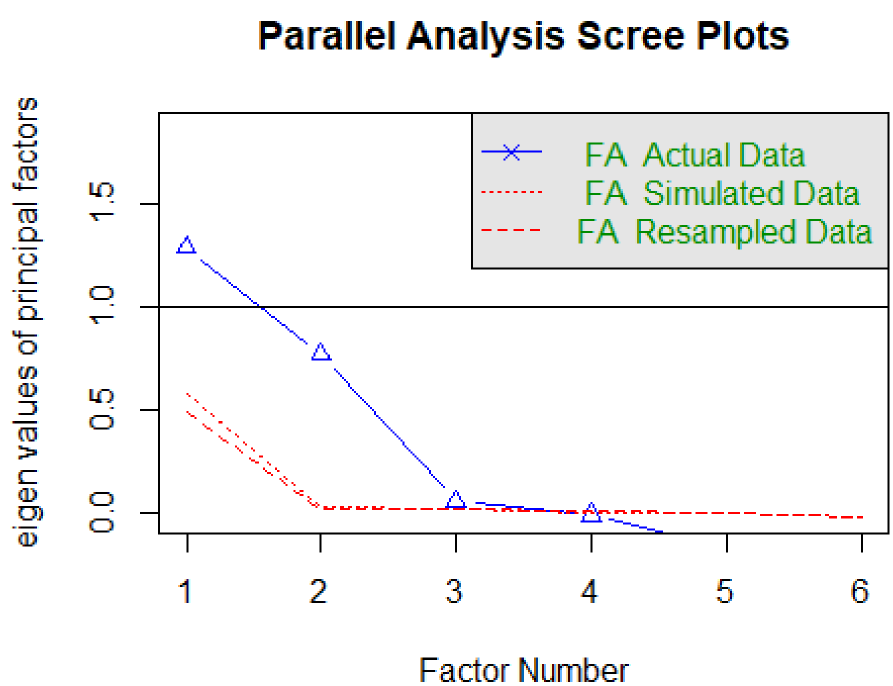
```
##    clouds  pressure  humidity      wind  distance      temp
## 0.8261773 0.9047895 0.9239694 0.8480276 0.9998461 0.7493997
```

The variance in clouds accounted by all fators is 0.87, This is the extent to which an item correlates with all other items. All comunalities are high, which means that the extracted components represent the variables well. If they are low , you may need to extract another component.

```
# Rotated factor scores, Notice the columns ordering: RC1, RC3, RC2 and RC4
fit.pc_scores<-fit.pc$scores
head(fit.pc_scores)

##            RC1       RC2      RC4         RC3
## [1,] 0.3196683 1.2040054 1.524236  0.05698478
## [2,] 0.3678396 0.9052307 1.329548  0.07812867
## [3,] 0.3823894 0.9322528 1.311205 -1.03514564
## [4,] 0.3872088 0.9242594 1.297225 -1.17642232
## [5,] 0.3902312 0.9315590 1.293890 -1.43719845
## [6,] 0.5484686 0.9711657 1.198470 -1.21705516
```

See factor recommendation

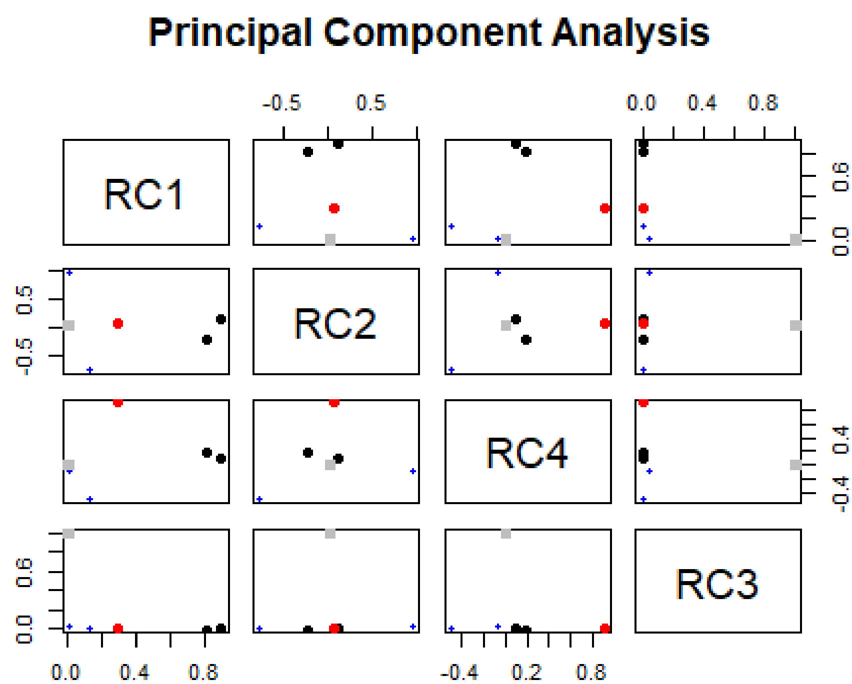
```
fa.parallel(x_new, fm='minres', fa='fa')
```




```
## Parallel analysis suggests that the number of factors =  3  and the number
of components =  NA
```

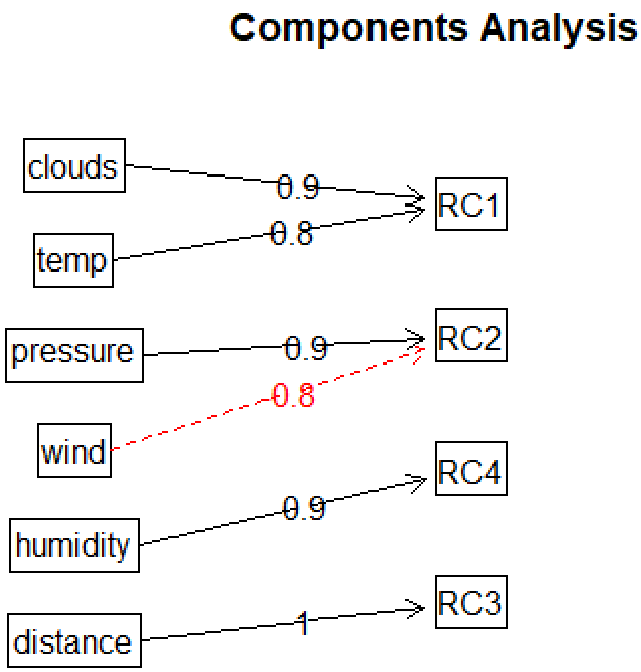
Blue line shows the eigen values of actual data and the two red lines show simulated and resampled data. Here factors between 2-4 will be a good choice..

```
fa.plot(fit.pc) # See Correlations within Factors
```



Visualize the relationship

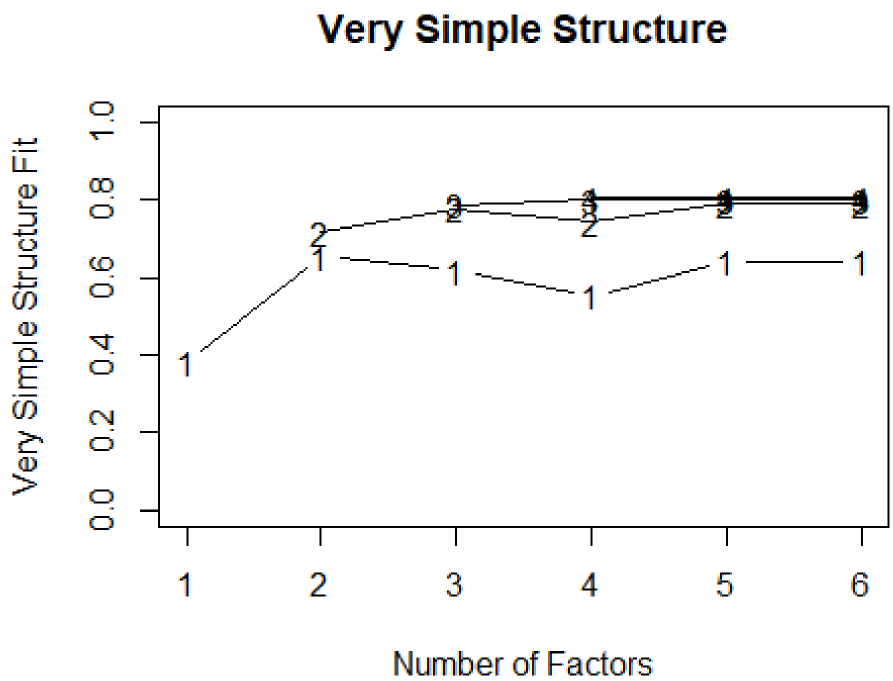
```
fa.diagram(fit.pc)
```



#Red dotted line means Wind marginally falls under the RC1 bucket.

See Factor recommendations for a simple structure

```
vss(x_new)

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate =
## rotate, : A loading greater than abs(1) was detected. Examine the loadings
## carefully.

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs
## = np.obs, : The estimated weights for the factor scores are probably
## incorrect. Try a different factor extraction method.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate =
## rotate, : An ultra-Heywood case was detected. Examine the results carefull
y
```




```
##
## Very Simple Structure
## Call: vss(x = x_new)
## VSS complexity 1 achieves a maximimum of 0.65 with 2  factors
## VSS complexity 2 achieves a maximimum of 0.79 with 5  factors
##
## The Velicer MAP achieves a minimum of NAwith    1  factors
## BIC achieves a minimum of  NA  with  2  factors
## Sample Size adjusted BIC achieves a minimum of  NA  with  2  factors
##
## Statistics by number of factors
##   vss1 vss2  map dof   chisq     prob sqresid  fit RMSEA  BIC SABIC
## 1 0.38 0.00 0.11   9 6.7e+03  0.0e+00     5.0 0.38  0.28 6597  6626
## 2 0.65 0.72 0.13   4 7.3e+02 3.3e-157     2.3 0.72  0.14  696   709
## 3 0.62 0.78 0.28   0 2.9e-01       NA     1.7 0.79    NA   NA    NA
## 4 0.55 0.74 0.44  -3 1.8e-06       NA     1.6 0.81    NA   NA    NA
## 5 0.64 0.79 1.00  -5 0.0e+00       NA     1.6 0.81    NA   NA    NA
## 6 0.64 0.79   NA  -6 0.0e+00       NA     1.6 0.81    NA   NA    NA
##   complex  eChisq    SRMR eCRMS eBIC
## 1     1.0 9.6e+03 1.8e-01 0.239 9469
## 2     1.1 3.4e+02 3.5e-02 0.068  304
## 3     1.3 2.4e-01 9.2e-04    NA   NA
## 4     1.4 1.2e-06 2.1e-06    NA   NA
## 5     1.4 1.0e-17 6.1e-12    NA   NA
## 6     1.4 1.0e-17 6.1e-12    NA   NA
```

We should continue with 4 factors as it has the max vss complexity. #Regression analysis using the factors scores as the independent variable: Let's combine the dependent variable and the factor scores into a dataset and label them.

```
cab<-cbind(x[7],fit.pc$scores)
#Labelling the data
names(cab)<-c("Price","Wind_Pressure","Temperature","Humidity","Distance")
head(cab)

##      Price Wind_Pressure Temperature Humidity    Distance
## 1 16.67376     0.3196683   1.2040054 1.524236  0.05698478
## 2 16.67376     0.3678396   0.9052307 1.329548  0.07812867
## 3  8.50000     0.3823894   0.9322528 1.311205 -1.03514564
## 4 16.50000     0.3872088   0.9242594 1.297225 -1.17642232
## 5 16.67376     0.3902312   0.9315590 1.293890 -1.43719845
## 6 26.50000     0.5484686   0.9711657 1.198470 -1.21705516
```

Let's split the dataset into training and testing dataset. (80:20)

```
set.seed(101)
Atrain<-sample(nrow(cab),nrow(cab)*0.80)
cab_train<-cab[Atrain,]
cab_test<-cab[-Atrain,]
dim(cab_train)

## [1] 7444    5

dim(cab_test)

## [1] 1862    5
```

Performing multiple regression (Taking alpha=0.1)

```
fit1 <- lm(Price~Wind_Pressure+Temperature+Humidity+Distance, data=cab_train)
#show the results
summary(fit1)

##
## Call:
## lm(formula = Price ~ Wind_Pressure + Temperature + Humidity +
##     Distance, data = cab_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.742 -0.392 -0.254  -0.063  69.318
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    16.683735   0.065216 255.823   <2e-16 ***
## Wind_Pressure  0.106653   0.065077   1.639    0.101
```

```
## Temperature    0.008777   0.065129   0.135    0.893
## Humidity      -0.036831   0.065151  -0.565    0.572
## Distance       1.919486   0.064994  29.533   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.627 on 7439 degrees of freedom
## Multiple R-squared:  0.1053, Adjusted R-squared:  0.1048
## F-statistic: 218.9 on 4 and 7439 DF,  p-value: < 2.2e-16
```

Wind_Pressure is highly insignificant

```r
fit2 <- lm(Price~Temperature+Humidity+Distance, data=cab_train)
#show the results
summary(fit2)
```

```
##
## Call:
## lm(formula = Price ~ Temperature + Humidity + Distance, data = cab_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.768  -0.316  -0.244  -0.121  69.311
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 16.68374    0.06522 255.795   <2e-16 ***
## Temperature  0.00987    0.06513   0.152    0.880
## Humidity    -0.03725    0.06516  -0.572    0.568
## Distance     1.91947    0.06500  29.530   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.627 on 7440 degrees of freedom
## Multiple R-squared:  0.105,  Adjusted R-squared:  0.1046
## F-statistic: 290.8 on 3 and 7440 DF,  p-value: < 2.2e-16
```

Humidity is insignificant

```r
fit3 <- lm(Price~Temperature+Distance, data=cab_train)
#show the results
summary(fit3)
```

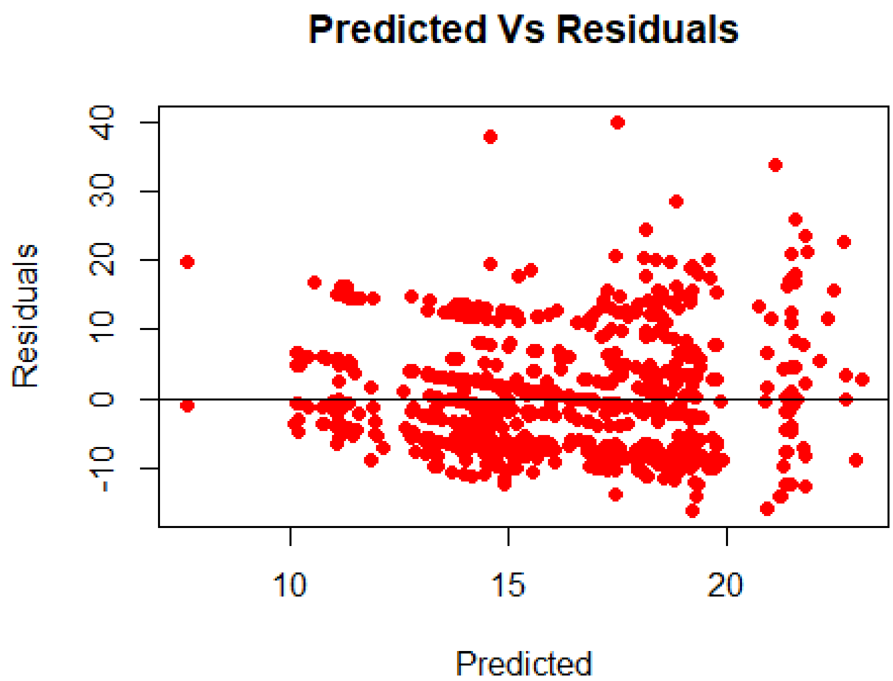
```
##
## Call:
## lm(formula = Price ~ Temperature + Distance, data = cab_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -16.779  -0.292  -0.233  -0.138  69.313
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 16.683592   0.065220 255.806   <2e-16 ***
## Temperature  0.009869   0.065130   0.152     0.88
## Distance     1.919638   0.064997  29.534   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.627 on 7441 degrees of freedom
## Multiple R-squared:  0.1049, Adjusted R-squared:  0.1047
## F-statistic: 436.1 on 2 and 7441 DF,  p-value: < 2.2e-16
```

Now we can see that all the variables are significant now. This model is explaining only 0.1% of variance only.

Predicted Test

```r
predicted_test <- predict(fit3, newdata= cab_test)
#Residuals Analysis
cab_testresults <- cab_test
cab_testresults$predicted <- predicted_test
cab_testresults$residual <- cab_testresults$Price - cab_testresults$predicted
#View(cab_testresults$residual)
plot(cab_testresults$predicted,cab_testresults$residual,xlab="Predicted",ylab
="Residuals",pch=21,bg="red",col="red",main="Predicted Vs Residuals")
abline(0,0)
```



Anova Table

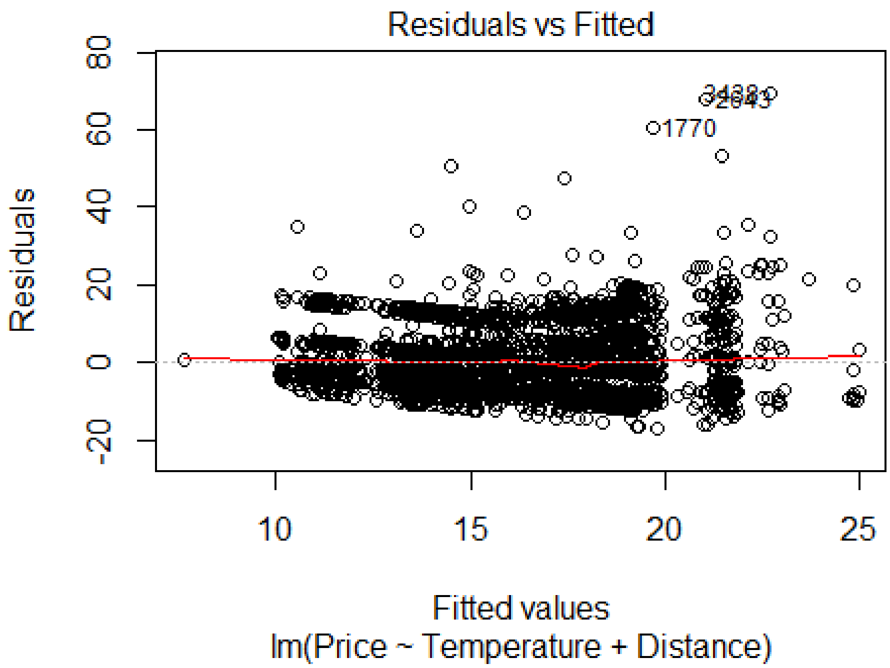
```
anova(fit3)

## Analysis of Variance Table
##
## Response: Price
##                 Df Sum Sq Mean Sq  F value Pr(>F)
## Temperature      1      0     0.5   0.0143 0.9049
## Distance         1  27619 27619.1 872.2639 <2e-16 ***
## Residuals     7441 235610    31.7
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

vcov(fit3)

##                 (Intercept)   Temperature     Distance
## (Intercept) 4.253612e-03 -9.381022e-06 -1.930894e-06
##  Temperature -9.381022e-06  4.241977e-03  4.587644e-06
## Distance -1.930894e-06 4.587644e-06  4.224652e-03

cov2cor(vcov(fit3))

##                 (Intercept)  Temperature      Distance
## (Intercept)  1.0000000000 -0.002208448 -0.0004554954
## Temperature -0.0022084476  1.000000000  0.0010837025
## Distance    -0.0004554954  0.001083703  1.0000000000
```

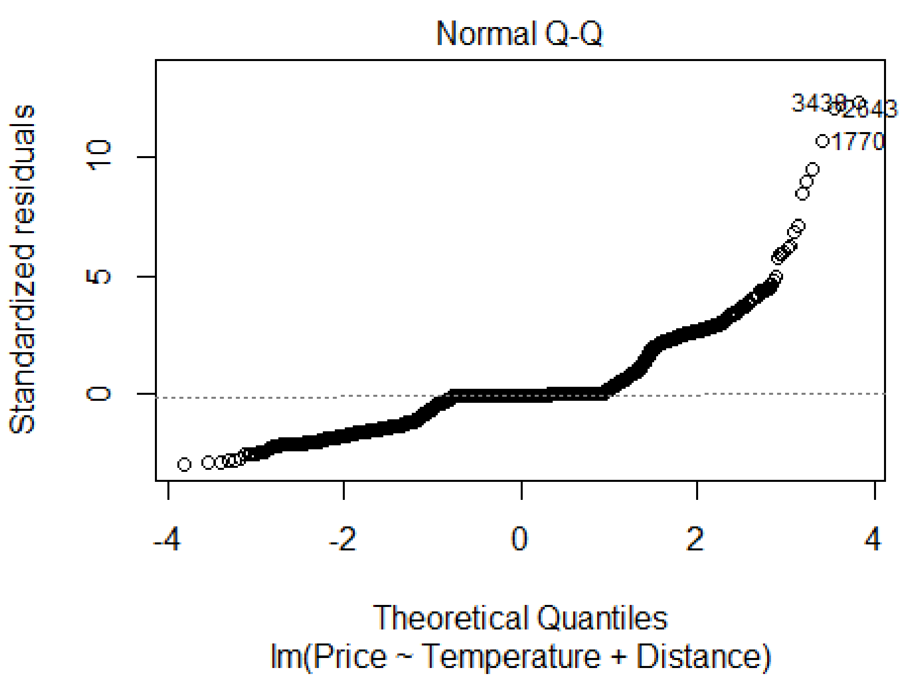
```
temp <- influence.measures(fit3)
head(temp)
```

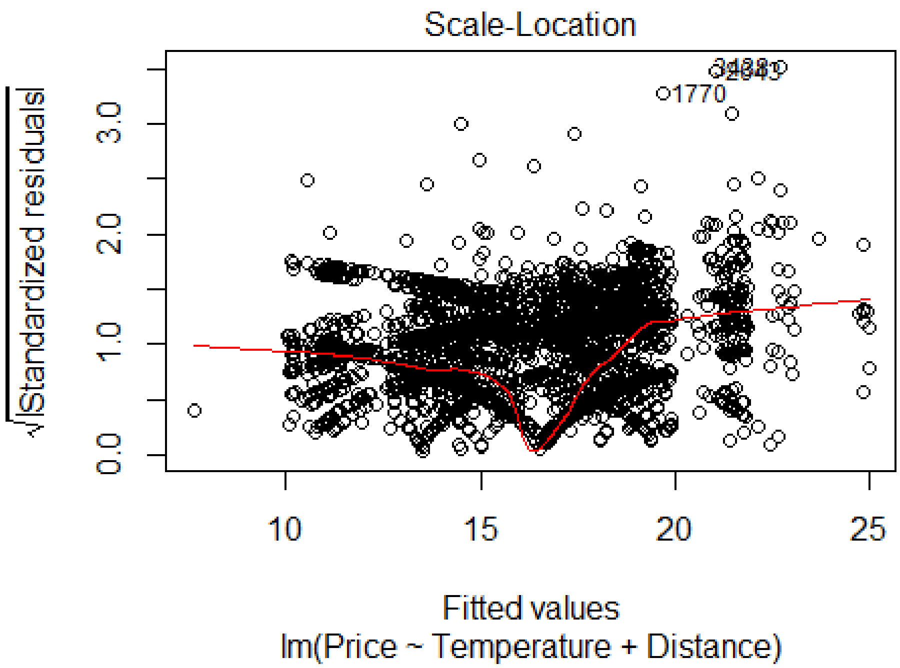
```
## Influence measures of
##   lm(formula = Price ~ Temperature + Distance, data = cab_train) :
##
##         dfb.1_  dfb.Tmpr  dfb.Dstn     dffit cov.r   cook.d      hat inf
## 3464 -5.76e-04  4.35e-04 -8.20e-05 -7.26e-04 1.001 1.76e-07 0.000214
## 408  -6.28e-03  4.60e-03 -9.91e-03 -1.26e-02 1.001 5.30e-05 0.000542
## 6603 -4.95e-04  3.34e-04 -6.01e-05 -6.00e-04 1.001 1.20e-07 0.000198
## 6119 -3.17e-04 -4.89e-04 -2.17e-05 -5.84e-04 1.001 1.14e-07 0.000453
## 2325  2.44e-02  3.13e-02  1.83e-02  4.37e-02 0.999 6.36e-04 0.000429
## 2791 -3.83e-04 -3.84e-04 -3.34e-05 -5.44e-04 1.001 9.86e-08 0.000270
## 5440 -2.48e-04 -3.22e-04 -1.29e-05 -4.07e-04 1.001 5.53e-08 0.000360
```

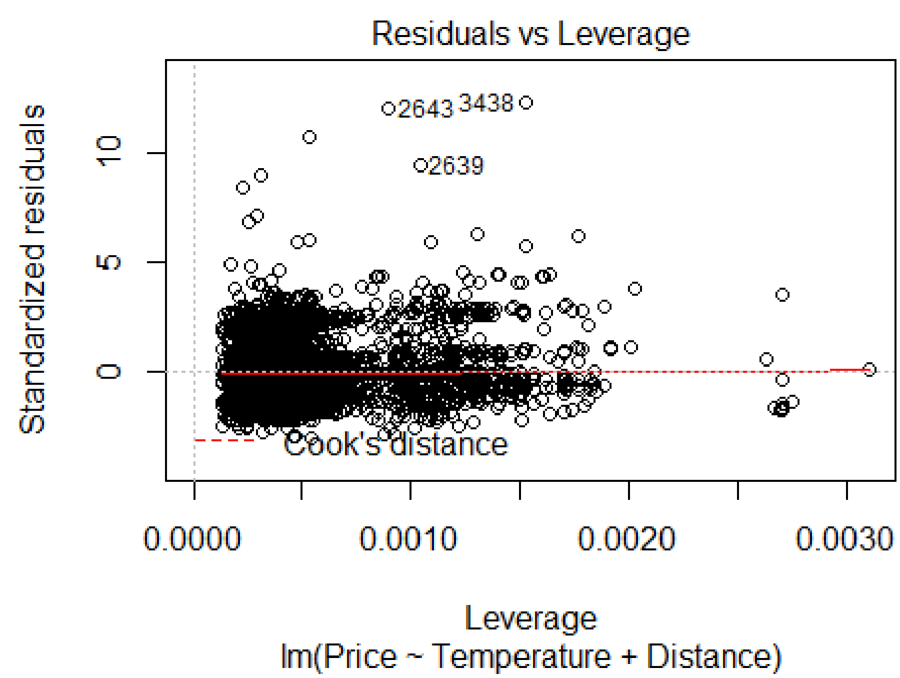
diagnostic plots

```
plotfit3)
```


Residuals vs Fitted
Residuals
80
60
40
20
0
-20
3439
2643
1770
10
15
20
25
Fitted values
lm(Price ~ Temperature + Distance)


Normal Q-Q
Standardized residuals
10
5
0
3439
2643
1770
-4
-2
0
2
4
Theoretical Quantiles
lm(Price ~ Temperature + Distance)





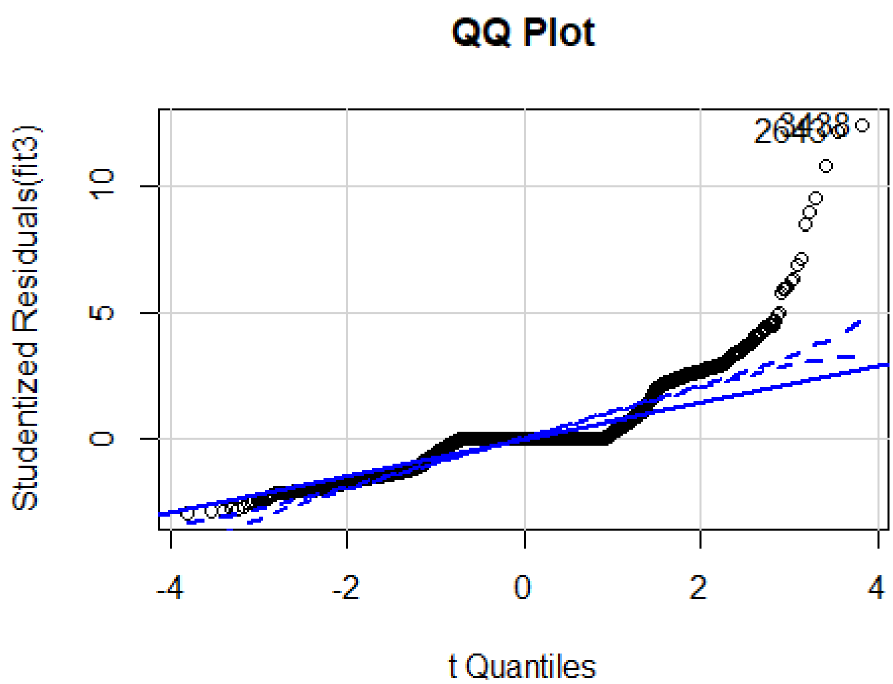
Residuals Vs Fitted: Residuals are equally spread out around a horizontal line without distinct pattern that indicates we dont have non linear relationships.

Normal Q-Q: The residuals are noramlly distributed if it has a straight linear line.

Scale-Location: Test homoscedascticity(assumption of equal variance) if you see a horizontal line with randomly spread points.

QQ-Plot

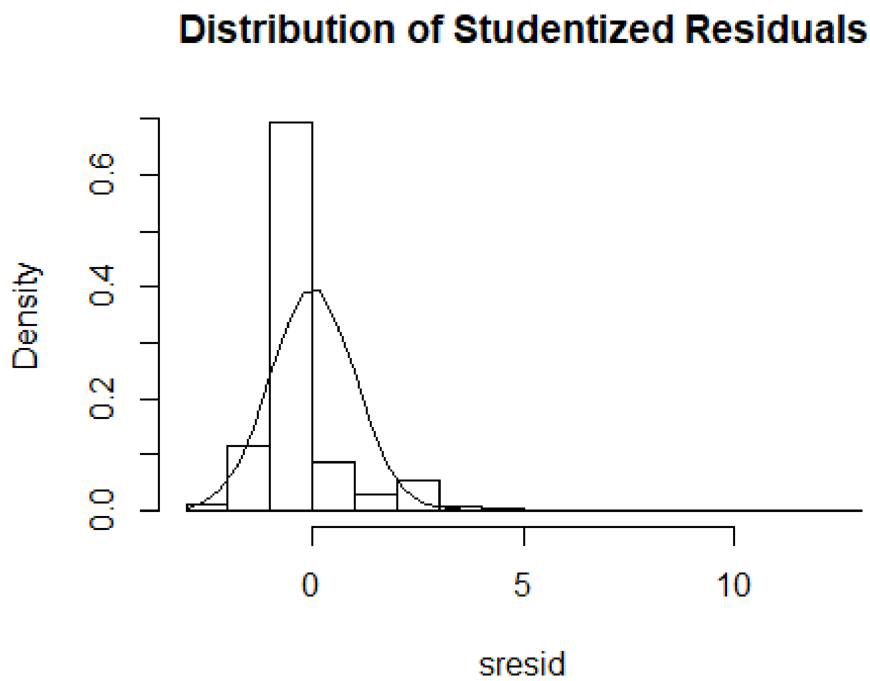
```
# Normality of Residuals
# qq plot for studentized resid
#install.packages("car")
qqPlot(fit3, main="QQ Plot")
```




```
## 2643 3438
##   643 2287
```

Distribution of studentized residuals

```
library(MASS)

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

sresid <- studres(fit3)
hist(sresid, freq=FALSE,
main="Distribution of Studentized Residuals")
xfit<-seq(min(sresid),max(sresid),length=40)
yfit<-dnorm(xfit)
lines(xfit, yfit)
```

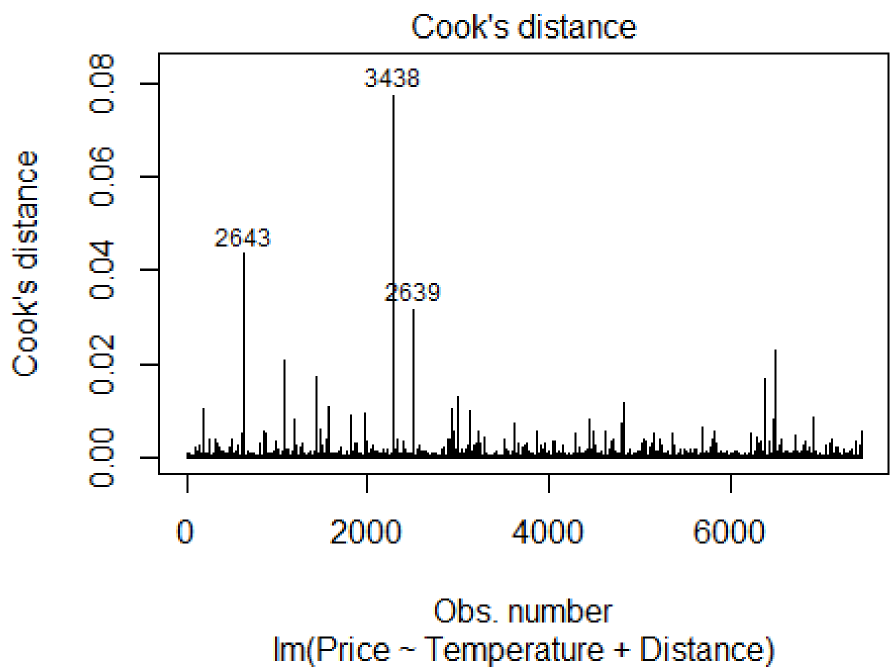


#Hence we see that residuals are normally distributed

Cook's D plot

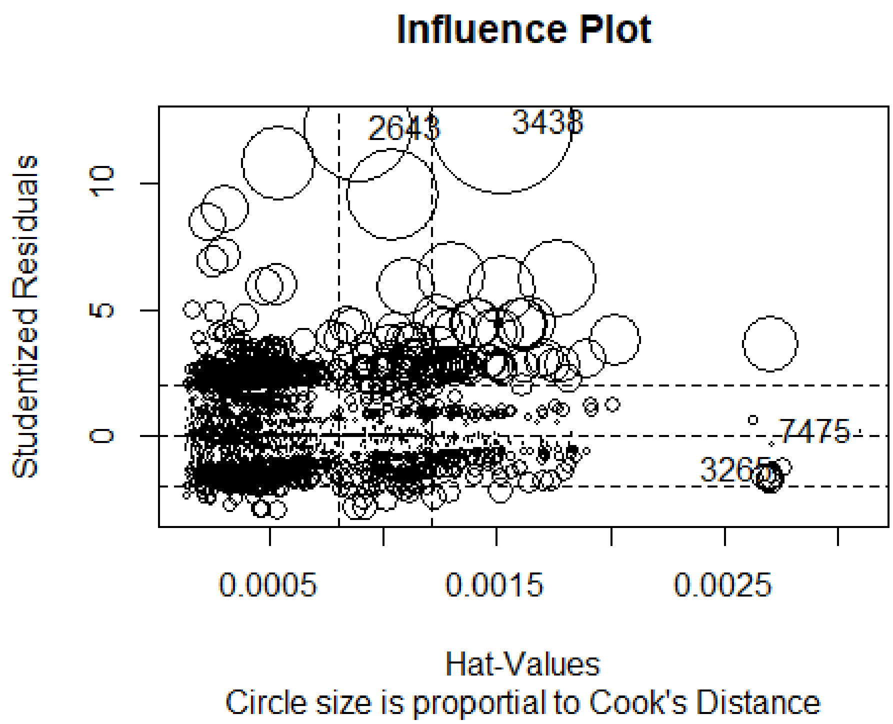
identify D values > 4/(n-k-1)

```
cutoff <- 4/((nrow(mtcars)-length(fit3$coefficients)-2))
plot(fit3, which=4, cook.levels=cutoff)
```



Influence Plot

```
influencePlot(fit3, id.method="identify", main="Influence Plot", sub="Circle
size is proportial to Cook's Distance" )

## Warning in plot.window(...): "id.method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "id.method" is not a graphical paramete
r

## Warning in axis(side = side, at = at, labels = labels, ...): "id.method" i
s
## not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "id.method" i
s
## not a graphical parameter

## Warning in box(...): "id.method" is not a graphical parameter

## Warning in title(...): "id.method" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.method" is not
a
## graphical parameter
```



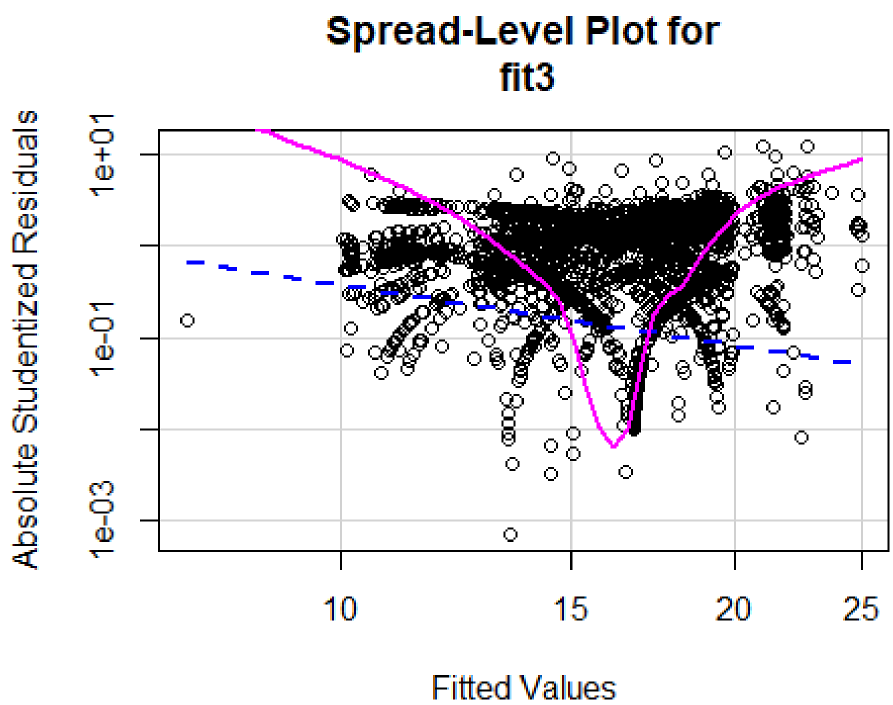

```
##          StudRes          Hat       CookD
## 2643 12.2042724 0.0008927121 4.349624e-02
## 3265 -1.3307620 0.0027551682 1.630725e-03
## 3438 12.4542355 0.0015229712 7.726163e-02
## 7475  0.1537065 0.0031021296 2.450922e-05
```

non-constant error variance test

```
ncvTest(fit3)

## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 609.4441, Df = 1, p = < 2.22e-16
```

plot studentized residuals vs. fitted values

```
spreadLevelPlot(fit3)
```




```
##
## Suggested power transformation:  3.193354
```