AllabtML

Vikas Bansode
5/14/2020

DATA COLLECTION

Data Collection may require knowledge of below technologies

1. SQL, NOSQL

2. WEB SCRAPING

3. TEXT MINING

4. UNIX, SHELL SCRIPTTING, AWK, SED, GREP

```
data("iris")
data <- iris
```

EXPLORE DATA

str(data)
dim(data)
length(data)
head(data)
tail(data)
names(data)
levels(data)
class(data$Species)
class(data$Sepal.Length)
summary(data)

DATA CLEANING

Detect Trailing and leading spaces

```
data <- read.csv("Data_Preparation.txt",header = TRUE,sep = "\t")
summary(data)
```

Handling Trailing and Leading space

Here we observed trailing and leading spaces for Gender attribute, lets clean them

```
data$Gender <- trimws(data$Gender,which = c('right'))
data$Gender <- trimws(data$Gender,which = c('left'))
summary(data)
```

Detect Punctuation marks

```
summary(data)
##     LungCap            Age              Height          Smoke
##  Min.   : 0.507   Min.   :  3.00   Min.   :45.30   no     :642
##  1st Qu.: 6.150   1st Qu.:  9.00   1st Qu.:59.90   yes    : 76
```

```
##  Median : 8.000   Median : 13.00   Median :65.40   %no    :  1
##  Mean   : 7.863   Mean   : 13.01   Mean   :64.82   no ?   :  1
##  3rd Qu.: 9.800   3rd Qu.: 15.00   3rd Qu.:70.22   no#    :  1
##  Max.   :14.675   Max.   :190.00   Max.   :81.80   no$    :  1
##                   NA's   :3        NA's   :1       (Other):  3
##     Gender          Caesarean
##  Length:725         no :561
##  Class :character   yes:164
##  Mode  :character
##
##
##
##
```

Handling Punctuation marks

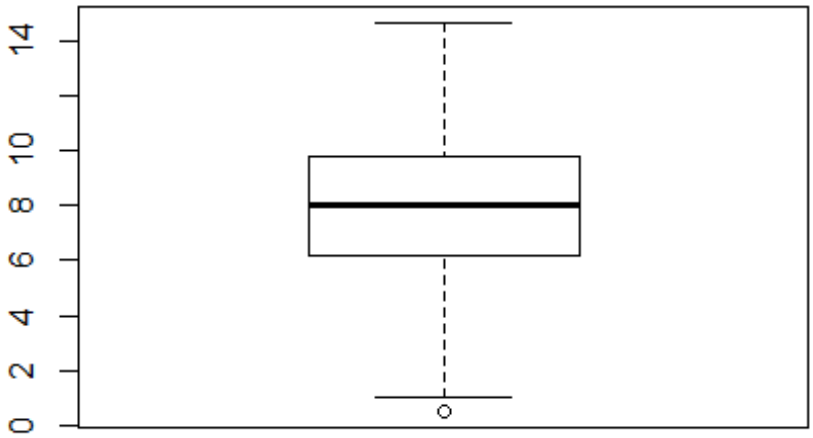
Here we see some punctuation marks for Smoke Atrribute, lets clean them

```
data$smoke <- gsub("[[:punct:]]|[[:digit:]]|(http[[:alpha:]]*:\\/\\/)","",dat
a$Smoke)
summary(data)
##     LungCap           Age             Height          Smoke
##  Min.   : 0.507   Min.   :  3.00   Min.   :45.30   no     :642
##  1st Qu.: 6.150   1st Qu.:  9.00   1st Qu.:59.90   yes    : 76
##  Median : 8.000   Median : 13.00   Median :65.40   %no    :  1
##  Mean   : 7.863   Mean   : 13.01   Mean   :64.82   no ?   :  1
##  3rd Qu.: 9.800   3rd Qu.: 15.00   3rd Qu.:70.22   no#    :  1
##  Max.   :14.675   Max.   :190.00   Max.   :81.80   no$    :  1
##                   NA's   :3        NA's   :1       (Other):  3
##     Gender          Caesarean    smoke
##  Length:725         no :561   Length:725
##  Class :character   yes:164   Class :character
##  Mode  :character             Mode  :character
##
##
##
##
library(dplyr)
data1 <- select(data,c(1,2,3,7,5,6))
rm(data)
data <- data1
rm(data1)
summary(data)
##     LungCap           Age             Height          smoke
##  Min.   : 0.507   Min.   :  3.00   Min.   :45.30   Length:725
##  1st Qu.: 6.150   1st Qu.:  9.00   1st Qu.:59.90   Class :character
##  Median : 8.000   Median : 13.00   Median :65.40   Mode  :character
##  Mean   : 7.863   Mean   : 13.01   Mean   :64.82
##  3rd Qu.: 9.800   3rd Qu.: 15.00   3rd Qu.:70.22
##  Max.   :14.675   Max.   :190.00   Max.   :81.80
##                   NA's   :3        NA's   :1
##     Gender          Caesarean
```

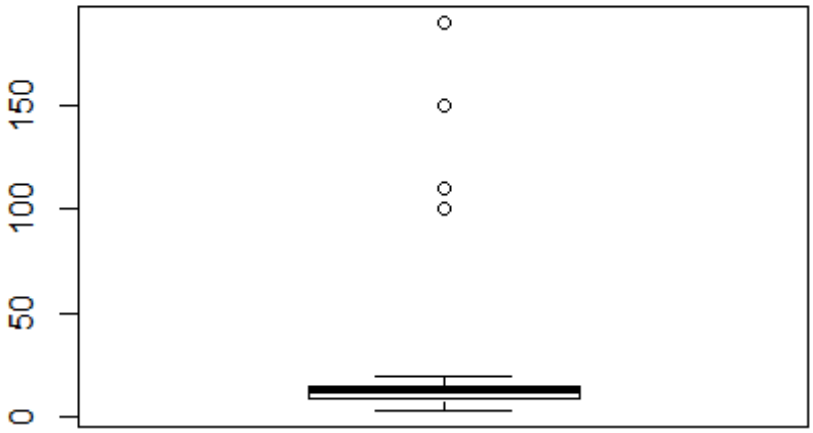
```
##  Length:725         no :561
##  Class :character   yes:164
##  Mode  :character
##
##
##
##
```

DETECT OUTLIERS

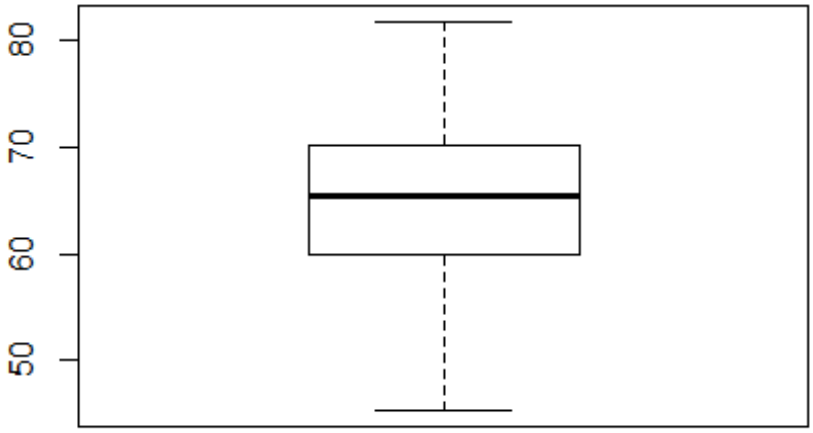
```
data <- read.csv("Data_Preparation.txt",header = T,sep = "\t")
boxplot(data$LungCap) # No outliers
```




```
boxplot(data$Age)    # In age we see lot of outliers
```




```r
boxplot(data$Height) # No outliers
```




```r
summary(data$Age)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    3.00    9.00   13.00   13.01   15.00  190.00       3
```

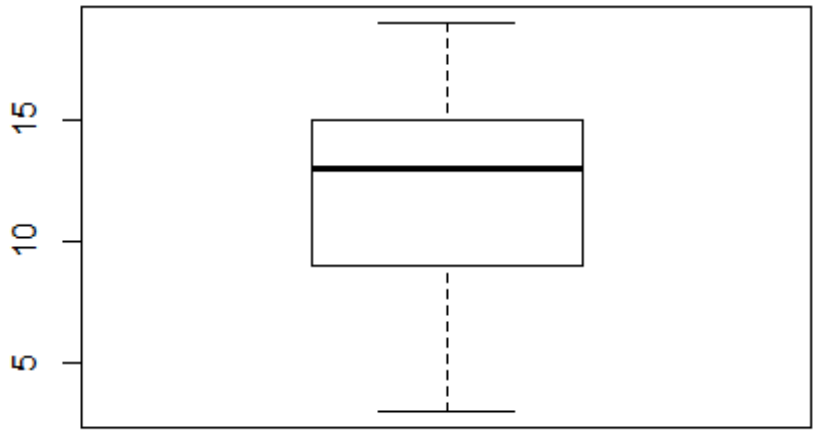
Handling Outliers

```
IQR <- 15.00 - 9.00
bench <- 15.00 + 1.5*IQR
bench
## [1] 24
data$Age[data$Age > bench] # Display outliers
## [1]  NA  NA  NA 100 110 150 190
```

Replace outliers with NA values

NOTE : You can replace , omit, remove outliers from data according to your requirment

```
data$Age[data$Age > bench]  <- NA
boxplot(data$Age)
```

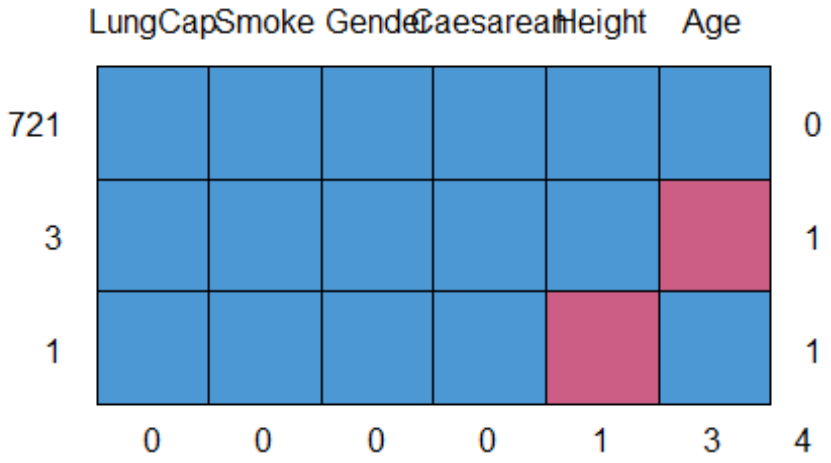


MISSING VALUES

```
library(mice)
library(VIM)
```

Detect Missing Values

```
data <- read.csv("Data_Preparation.txt",header = T,sep = "\t")
md.pattern(data)
```




```
##     LungCap Smoke Gender Caesarean Height Age
## 721       1     1      1         1      1   1 0
## 3         1     1      1         1      1   0 1
## 1         1     1      1         1      0   1 1
##           0     0      0         0      1   3 4
md.pairs(data)
## $rr
##           LungCap Age Height Smoke Gender Caesarean
## LungCap       725 722    724   725    725       725
## Age           722 722    721   722    722       722
## Height        724 721    724   724    724       724
## Smoke         725 722    724   725    725       725
## Gender        725 722    724   725    725       725
## Caesarean     725 722    724   725    725       725
##
## $rm
##           LungCap Age Height Smoke Gender Caesarean
```

```
## LungCap         0   3      1     0      0         0
## Age             0   0      1     0      0         0
## Height          0   3      0     0      0         0
## Smoke           0   3      1     0      0         0
## Gender          0   3      1     0      0         0
## Caesarean       0   3      1     0      0         0
##
## $mr
##           LungCap Age Height Smoke Gender Caesarean
## LungCap         0   0      0     0      0         0
## Age             3   0      3     3      3         3
## Height          1   1      0     1      1         1
## Smoke           0   0      0     0      0         0
## Gender          0   0      0     0      0         0
## Caesarean       0   0      0     0      0         0
##
## $mm
##           LungCap Age Height Smoke Gender Caesarean
## LungCap         0   0      0     0      0         0
## Age             0   3      0     0      0         0
## Height          0   0      1     0      0         0
## Smoke           0   0      0     0      0         0
## Gender          0   0      0     0      0         0
## Caesarean       0   0      0     0      0         0
p <- function(x){sum(is.na(x))/length(x)*100}
apply(data,2,p)
##   LungCap       Age    Height     Smoke    Gender Caesarean
## 0.0000000 0.4137931 0.1379310 0.0000000 0.0000000 0.0000000
# OR to get exact rows of missing values
apply(is.na(data),2,which)
## $LungCap
## integer(0)
##
## $Age
## [1]  4 19 35
##
## $Height
## [1] 58
##
## $Smoke
## integer(0)
##
## $Gender
## integer(0)
##
## $Caesarean
## integer(0)
```

Handling Missing Values

imputation with mean / median / mode

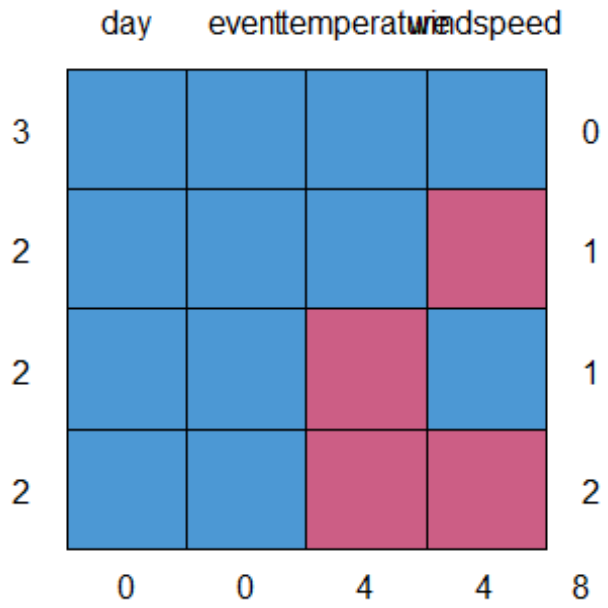
```
mean(data$Height[!is.na(data$Height)])
## [1] 64.81657
mean(data$Height,na.rm = TRUE)
## [1] 64.81657
round(mean(data$Height,na.rm = TRUE),2)
## [1] 64.82
data$Height[which(is.na(data$Height))] <- round(mean(data$Height,na.rm = TRUE
),2)
round(mean(data$Age,na.rm = TRUE))
## [1] 13
data$Age[which(is.na(data$Age))] <- round(mean(data$Age,na.rm = TRUE))
```

Example 2 : Missing Value

```
df <- read.csv("missing_values.csv",header = T,sep = ",")
```

Detect Missing Values

```
md.pattern(df)
```




```
##   day event temperature windspeed
## 3   1     1           1         1 0
## 2   1     1           1         0 1
## 2   1     1           0         1 1
## 2   1     1           0         0 2
##     0     0           4         4 8
md.pairs(df)
```

```
## $rr
##             day temperature windspeed event
## day           9           5         5     9
## temperature   5           5         3     5
## windspeed     5           3         5     5
## event         9           5         5     9
##
## $rm
##             day temperature windspeed event
## day           0           4         4     0
## temperature   0           0         2     0
## windspeed     0           2         0     0
## event         0           4         4     0
##
## $mr
##             day temperature windspeed event
## day           0           0         0     0
## temperature   4           0         2     4
## windspeed     4           2         0     4
## event         0           0         0     0
##
## $mm
##             day temperature windspeed event
## day           0           0         0     0
## temperature   0           4         2     0
## windspeed     0           2         4     0
## event         0           0         0     0
p <- function(x){sum(is.na(x))/length(x)*100}
apply(df,2,p)
##         day temperature   windspeed       event
##     0.00000    44.44444    44.44444     0.00000
apply(is.na(df),2,which)
## $day
## integer(0)
##
## $temperature
## [1] 2 4 6 7
##
## $windspeed
## [1] 3 5 6 7
##
## $event
## integer(0)
```

Transform Dataset as Day column in factor transform it into date type

```
df
##         day temperature windspeed  event
## 1  1/1/2017          32         6   Rain
## 2  1/4/2017          NA         9  Sunny
## 3  1/5/2017          28        NA   Snow
```

```
## 4  1/6/2017          NA         7
## 5  1/7/2017          32        NA   Rain
## 6  1/8/2017          NA        NA  Sunny
## 7  1/9/2017          NA        NA
## 8 1/10/2017          34         8 Cloudy
## 9 1/11/2017          40        12  Sunny
class(df$day)
## [1] "factor"
library(lubridate)
##
## Attaching package: 'lubridate'
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
## The following objects are masked from 'package:dplyr':
##
##     intersect, setdiff, union
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
df$day <- mdy(as.character(df$day))
df
##          day temperature windspeed  event
## 1 2017-01-01          32         6   Rain
## 2 2017-01-04          NA         9  Sunny
## 3 2017-01-05          28        NA   Snow
## 4 2017-01-06          NA         7
## 5 2017-01-07          32        NA   Rain
## 6 2017-01-08          NA        NA  Sunny
## 7 2017-01-09          NA        NA
## 8 2017-01-10          34         8 Cloudy
## 9 2017-01-11          40        12  Sunny
df$event[df$event ==''] <- NA
df
##          day temperature windspeed  event
## 1 2017-01-01          32         6   Rain
## 2 2017-01-04          NA         9  Sunny
## 3 2017-01-05          28        NA   Snow
## 4 2017-01-06          NA         7   <NA>
## 5 2017-01-07          32        NA   Rain
## 6 2017-01-08          NA        NA  Sunny
## 7 2017-01-09          NA        NA   <NA>
## 8 2017-01-10          34         8 Cloudy
## 9 2017-01-11          40        12  Sunny
```

Below functions can be used for Handling Missing Values

na.fill(df1$temperature,"extend") ##
na.fill(df1$temperature,c(“extend”,NA))
na.fill(df$temperature,list(NA,NULL,NA)) ##
na.aggregate(df$temperature)
na.action()
na.aggregate()
na.mean()
na.sd()
na.aggregate.default()
na.approx()
na.approx.default()
na.contiguous()
na.exclude()
na.fail()
na.fill()
na.fill.default()
na.fill0()
na.replace()

DATA TRANSFORMATION

```
data <- read.csv("Cardiotocographic.csv",header = T,sep = ",")
data$NSPF <- factor(data$NSP)
str(data)
data <- as.matrix(data)
class(data)
```

EXPLORATORY DATA ANALYSIS

Summarization

```
data("iris")
```

Univariate Analysis

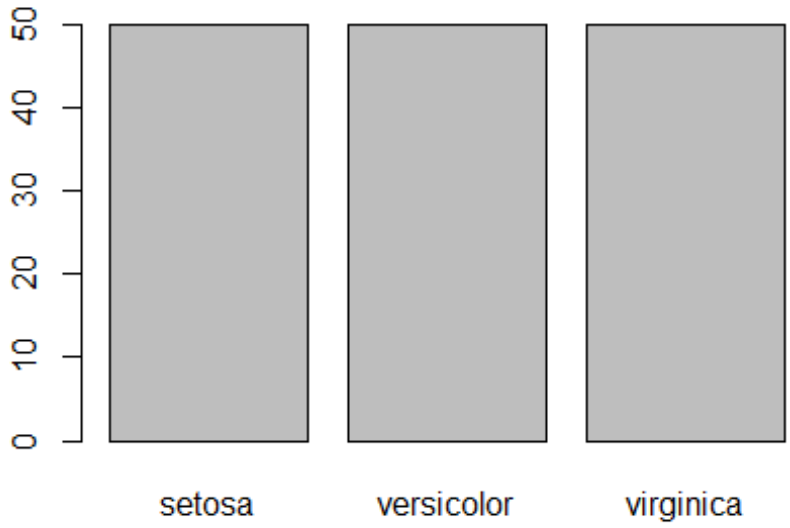
```
summary(iris)
table(iris$Species)
```

Bivariate Analysis

```
cor(iris$Sepal.Length,iris$Sepal.Width)
cor(iris$Petal.Length,iris$Petal.Width)
```

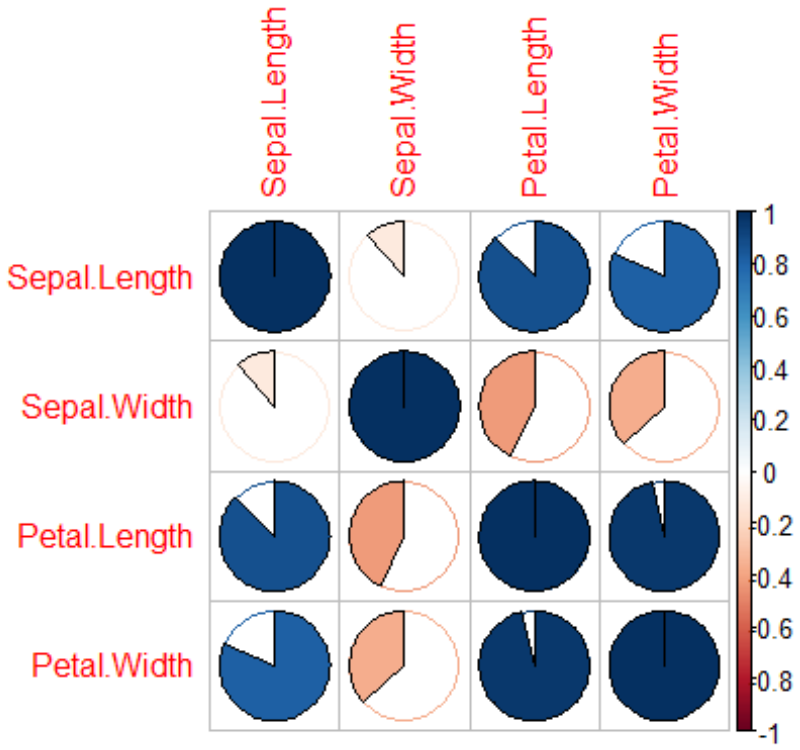
Multivariate Analysis

```
aggregate(iris$Sepal.Length, by=list(Species=iris$Species), FUN=mean)
aggregate(iris$Sepal.Width, by=list(Species=iris$Species), FUN=mean)
aggregate(iris$Petal.Length, by=list(Species=iris$Species), FUN=mean)
aggregate(iris$Petal.Width, by=list(Species=iris$Species), FUN=mean)
aggregate(data.frame("SepalLength(cm)"=iris$Sepal.Length,"SepalWidth(cm)"=iri
s$Sepal.Width,"PetalLength(cm)"=iris$Petal.Length,
                     "PetalWidth(cm)"=iris$Petal.Width), by=list(Species=iris
$Species), FUN=mean)
barplot(table(iris$Species))
```

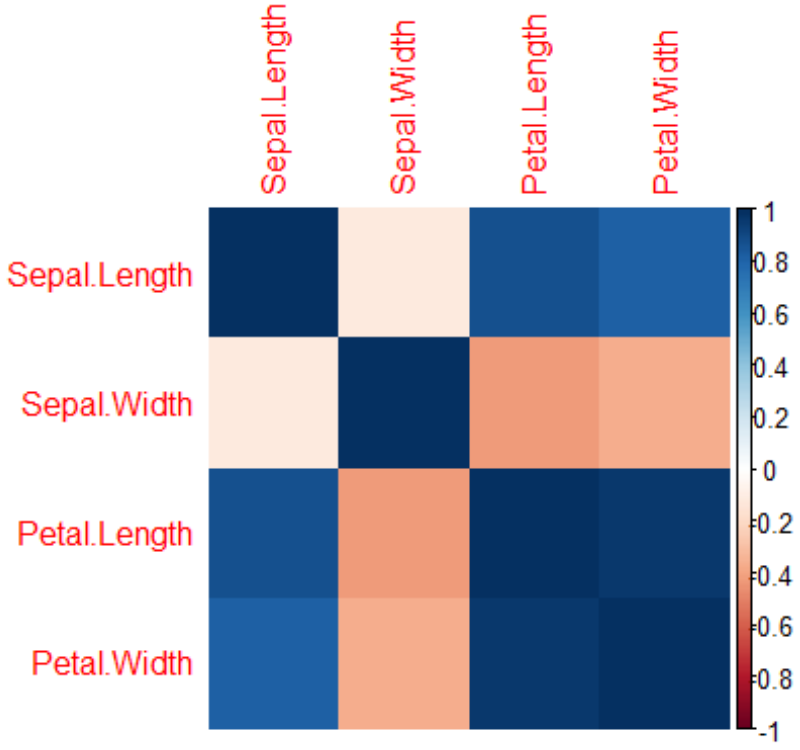


Correlation Matrix

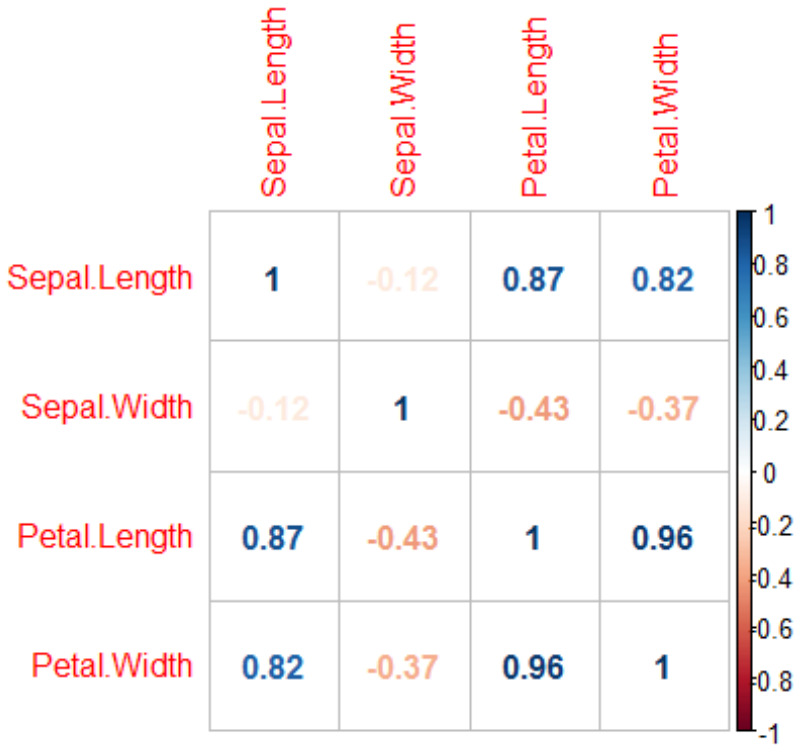
```
library(corrplot)
cor(iris[-5])
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
## Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
## Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
## Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
corrplot(cor(iris[-5]),method = 'pie')
```



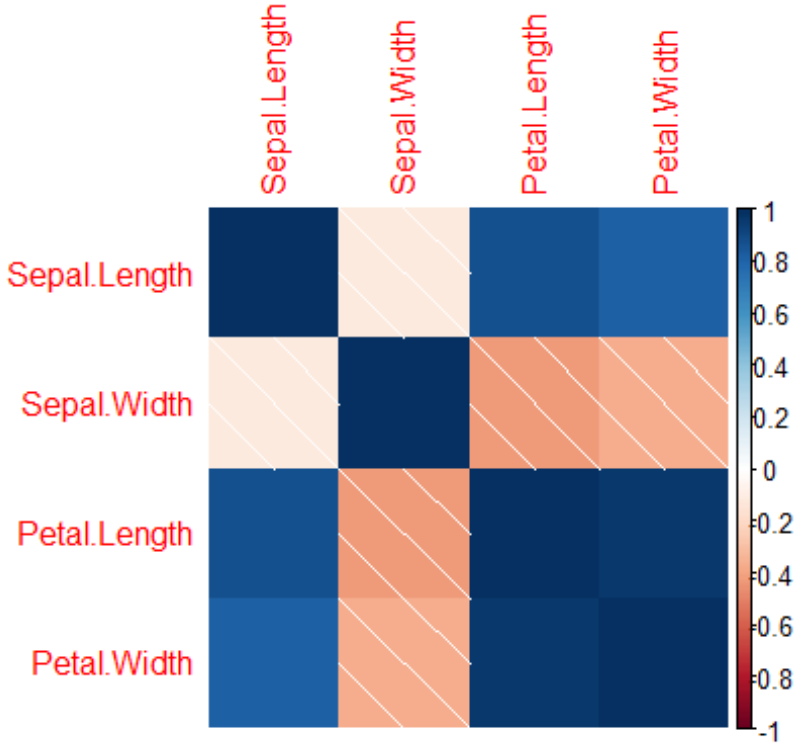

```
corrplot(cor(iris[-5]),method = 'color')
```



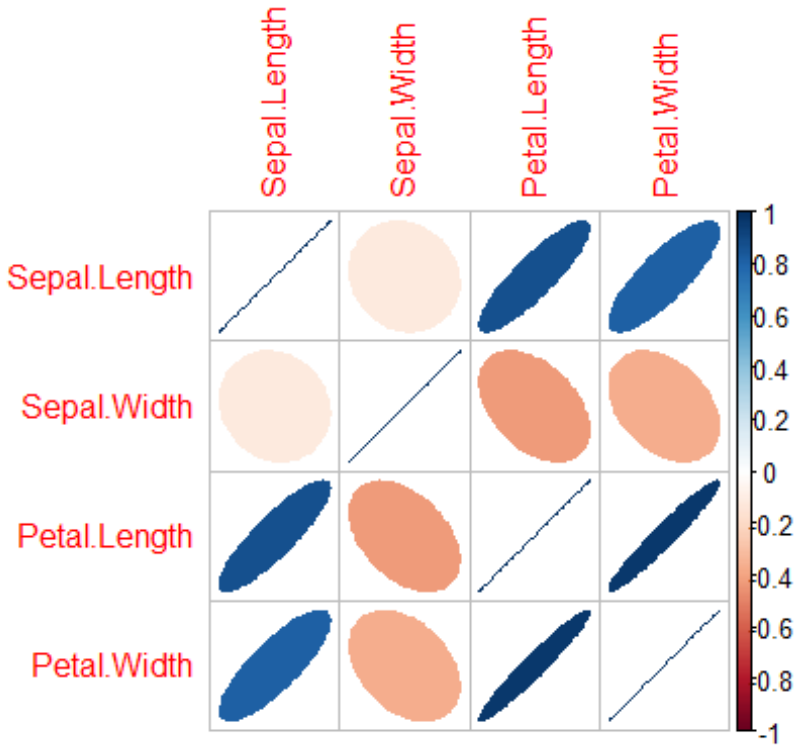

```
corrplot(cor(iris[-5]),method = 'number')
```



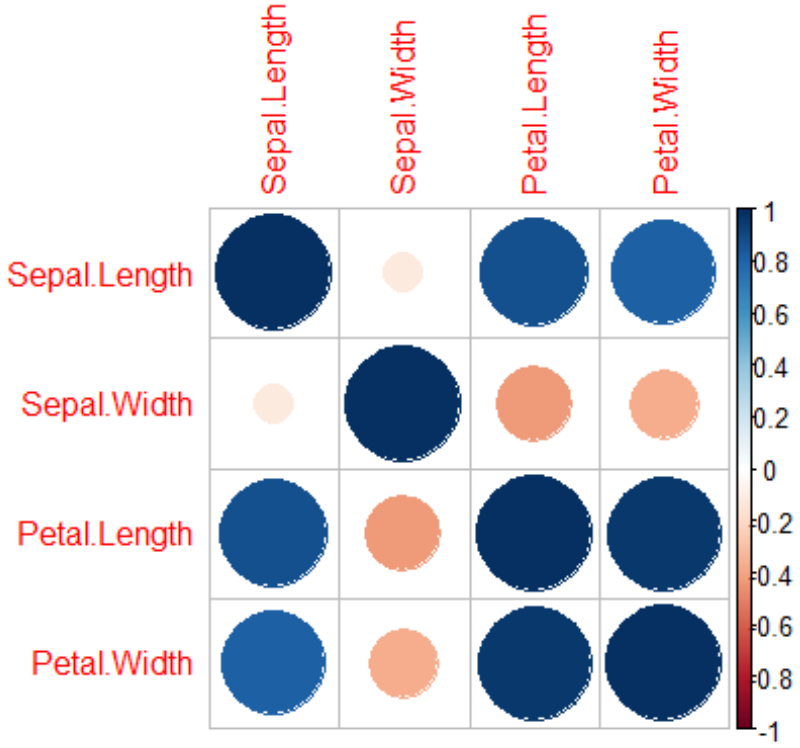

```
corrplot(cor(iris[-5]),method = 'shade')
```




```
corrplot(cor(iris[-5]),method = 'ellipse')
```



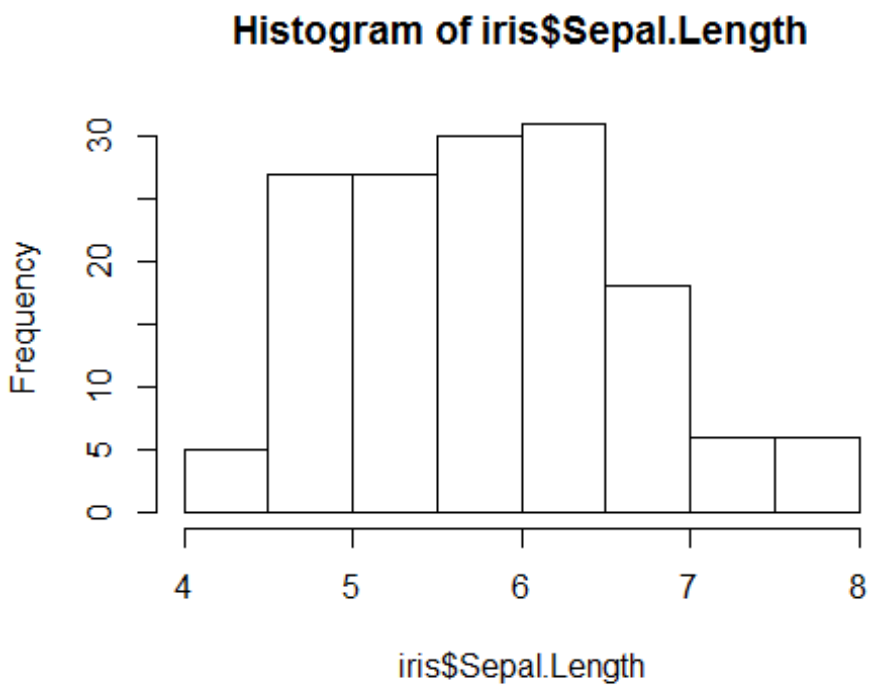

```
corrplot(cor(iris[-5]),method = 'circle')
```



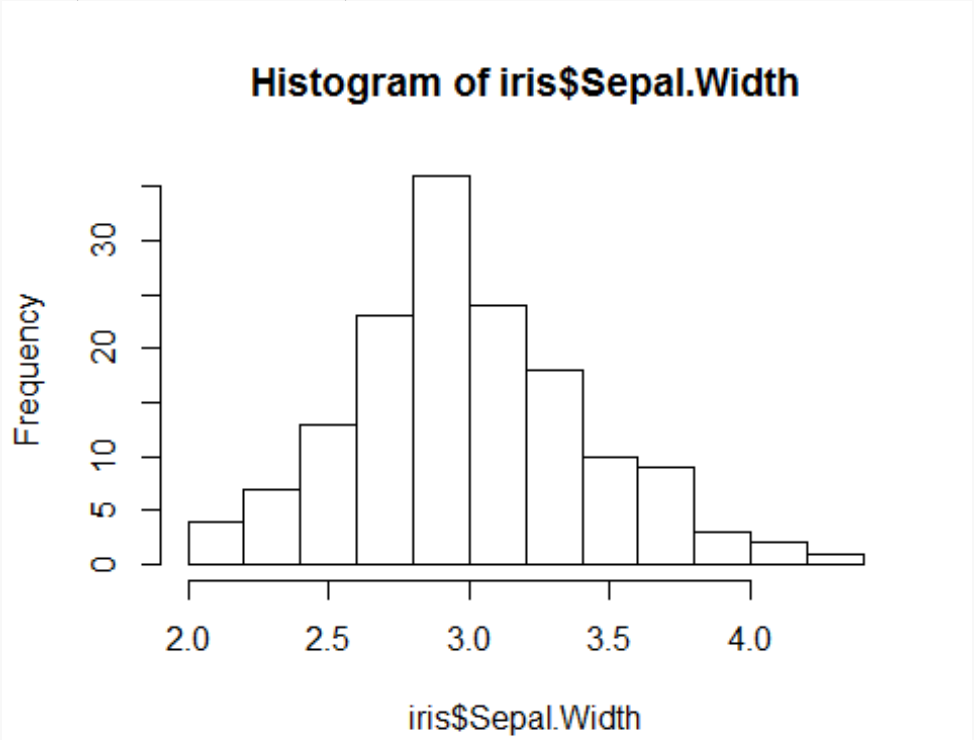
Visualization

Histogram

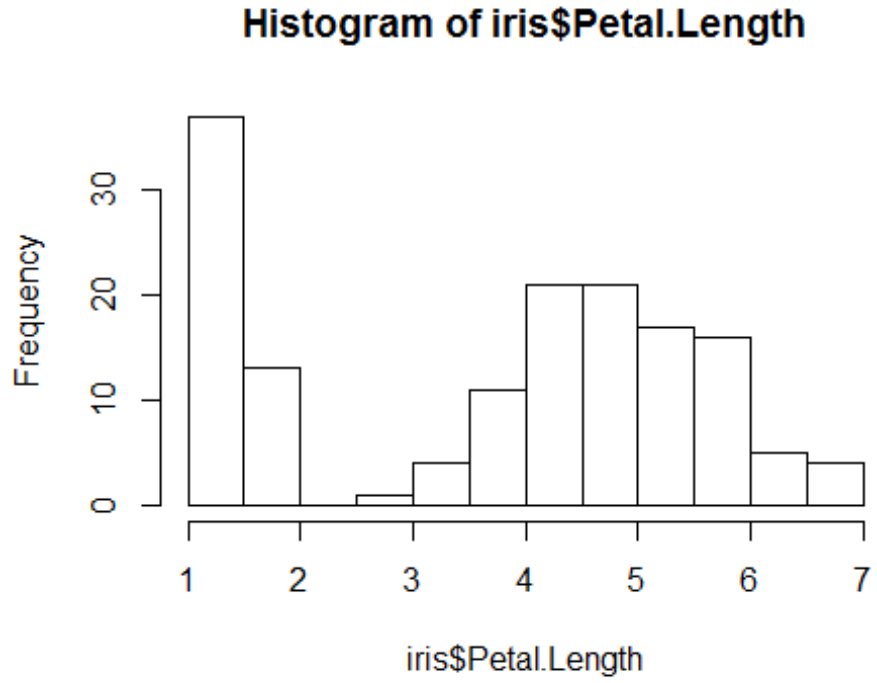
```
hist(iris$Sepal.Length)
```



```r
hist(iris$Sepal.Width)
```

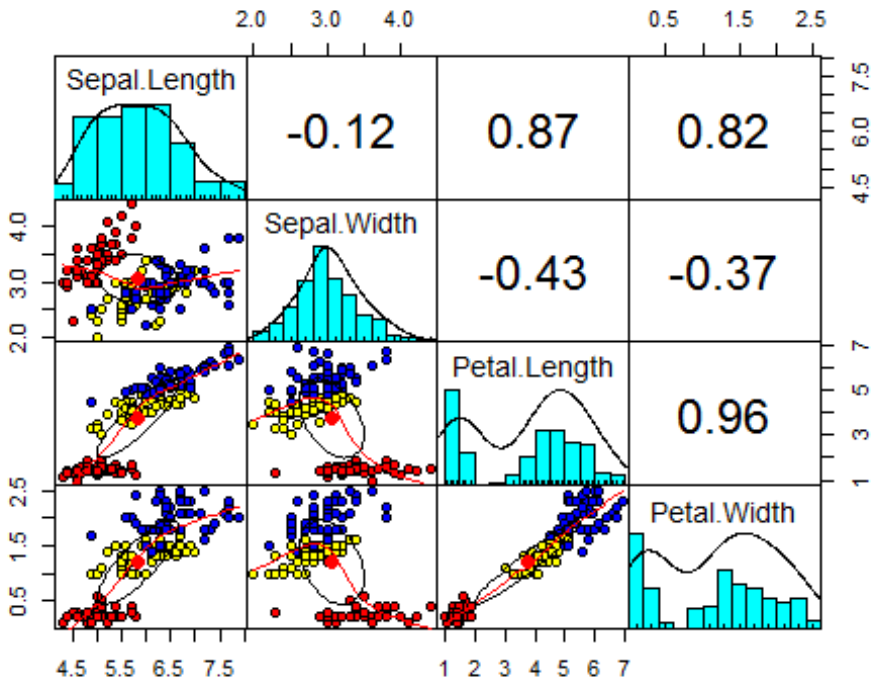



```r
hist(iris$Petal.Length)
hist(iris$Petal.Length)
```

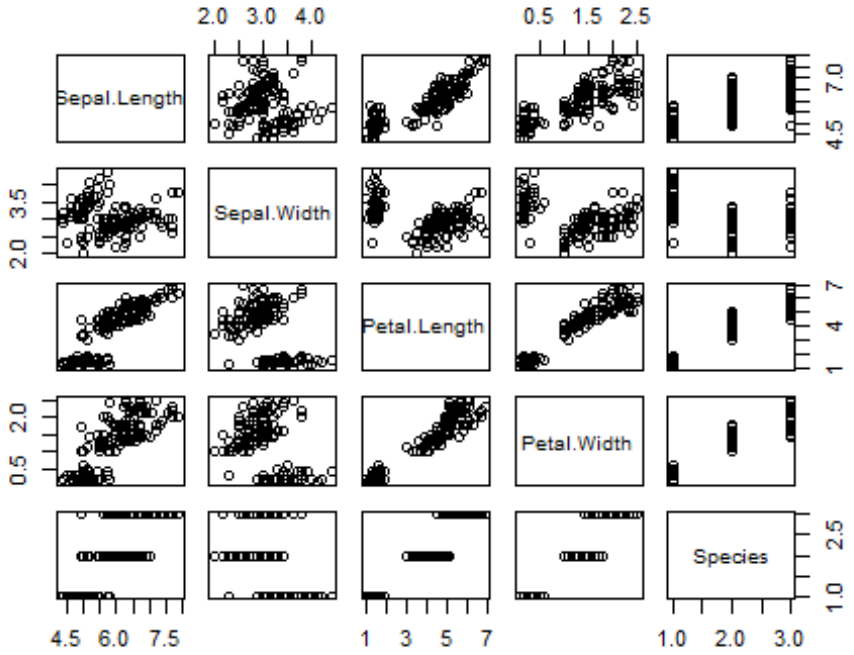


Pair Plots

```
library(psych)
library(ggplot2)
pairs.panels(iris[-5],gap=0,bg=c("red","yellow","blue")[iris$Species],pch=21)
```

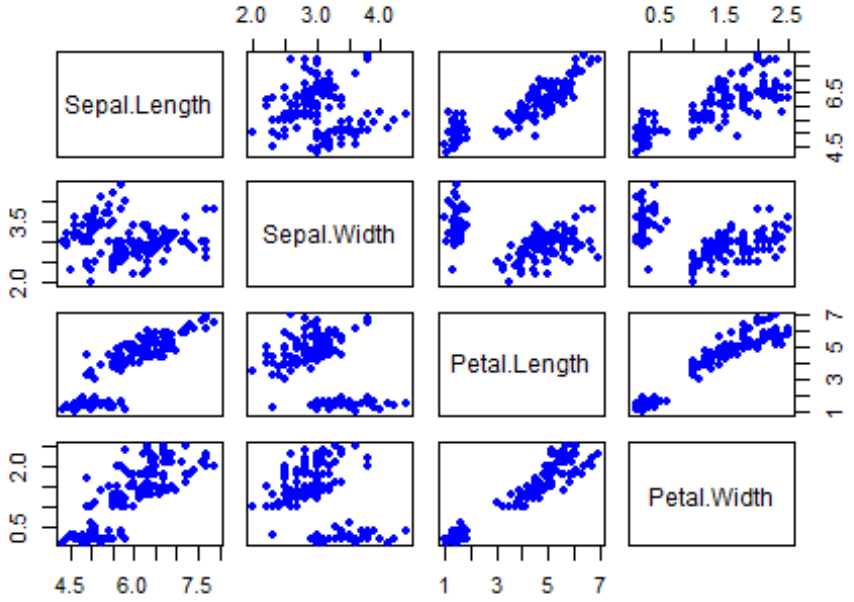



```
pairs(iris)
```



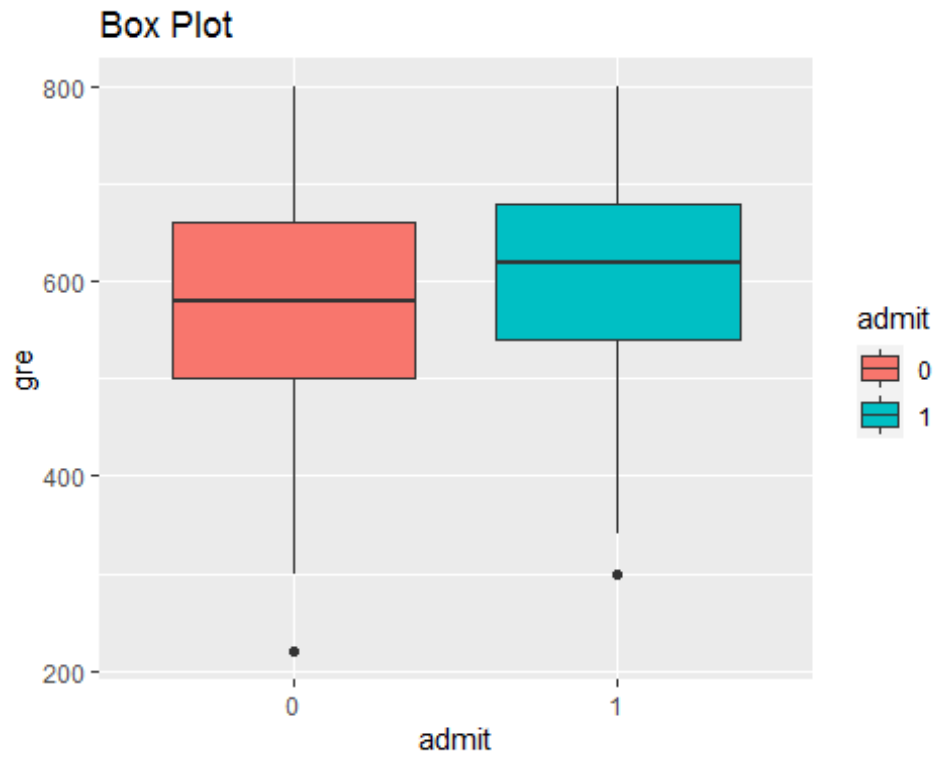

```
plot(iris[,1:4],
     main="Relationships between characteristics of iris flowers",
     pch=19,
     col="blue",
     cex=0.9)
```

Relationships between characteristics of iris flowers

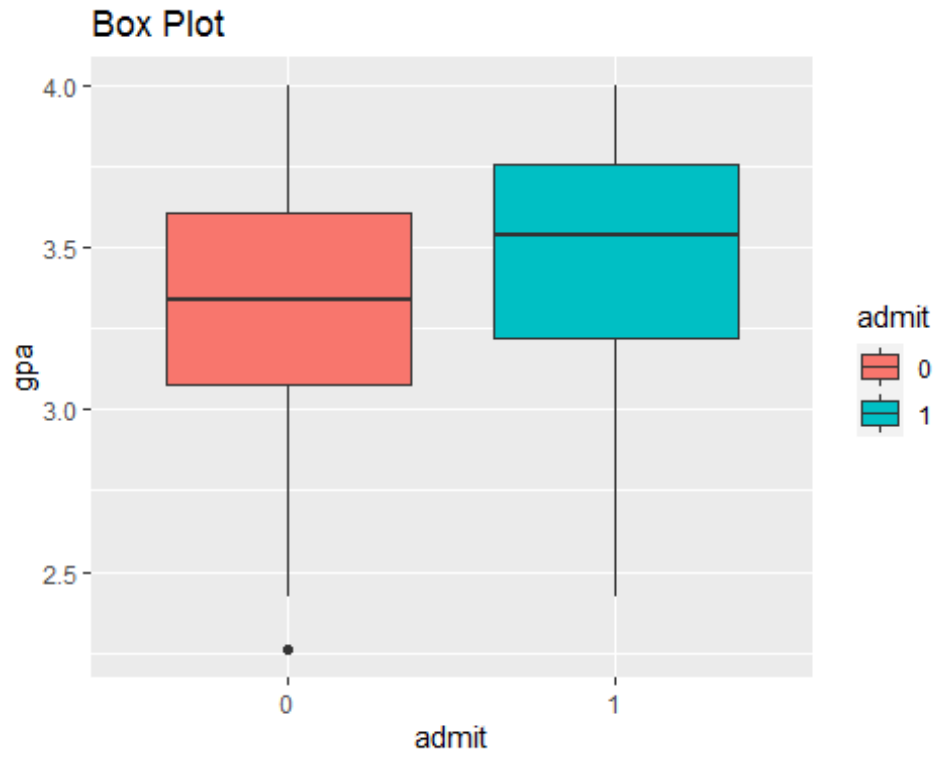


Boxplot to check overlap

```
data <- read.csv("binary.csv",header = T,sep = ",")
str(data)
## 'data.frame':    400 obs. of  4 variables:
##  $ admit: int  0 1 1 1 0 1 1 0 1 0 ...
##  $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
##  $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
##  $ rank : int  3 3 1 4 4 2 1 2 3 2 ...
data$rank <- as.factor(data$rank)
data$admit <- as.factor(data$admit)
str(data)
## 'data.frame':    400 obs. of  4 variables:
##  $ admit: Factor w/ 2 levels "0","1": 1 2 2 2 1 2 2 1 2 1 ...
##  $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
##  $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
##  $ rank : Factor w/ 4 levels "1","2","3","4": 3 3 1 4 4 2 1 2 3 2 ...
data %>%
  ggplot(aes(x=admit,y=gre,fill=admit)) +
  geom_boxplot() +
  ggtitle("Box Plot")
```

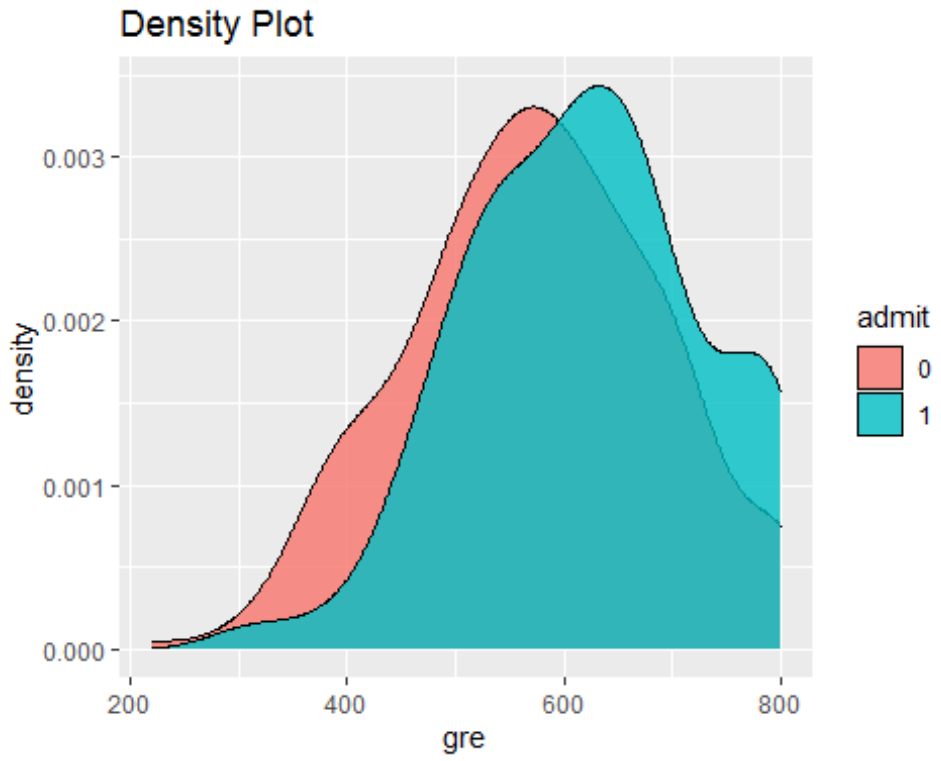



```
data %>%
  ggplot(aes(x=admit,y=gpa,fill=admit)) +
  geom_boxplot() +
  ggtitle("Box Plot")
```

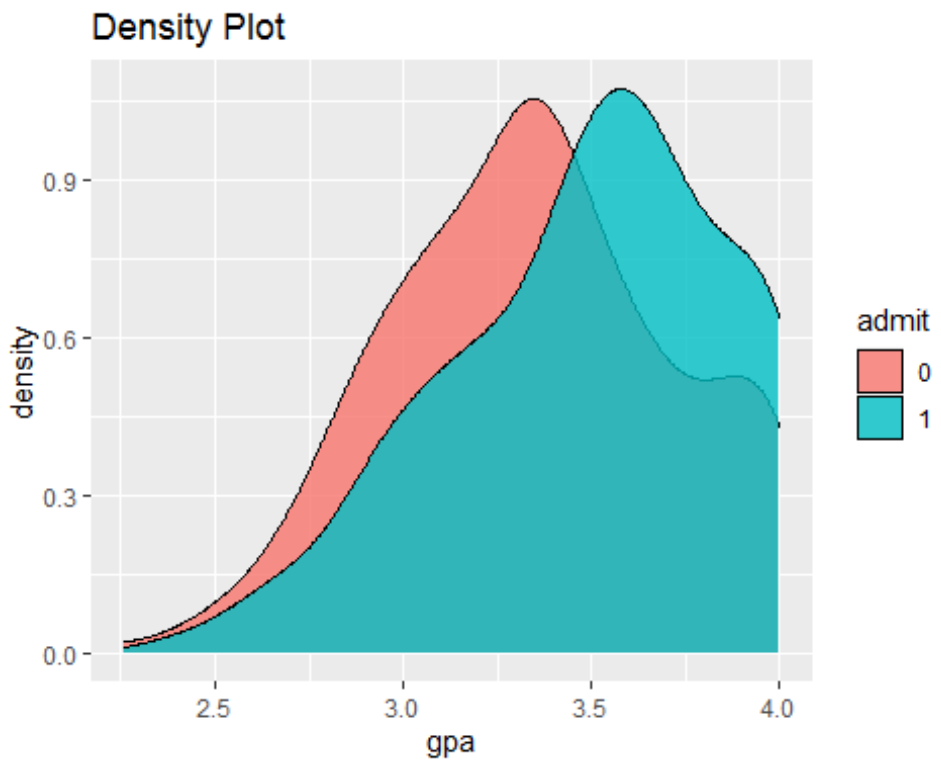


Density plot to check Overlap

```
data %>%
  ggplot(aes(x=gre,fill=admit)) +
  geom_density(alpha=0.8,color='black') +
  ggtitle('Density Plot')
```




```
data %>%
  ggplot(aes(x=gpa,fill=admit)) +
  geom_density(alpha=0.8,color='black') +
  ggtitle('Density Plot')
```



DATA NORMALIZATION

Min max Normalization

```
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
df <- data.frame(ctl,trt)
df$ctl <- (df$ctl - min(df$ctl))/(max(df$ctl)-min(df$ctl))
df$trt <- (df$trt - min(df$trt))/(max(df$trt)-min(df$trt))
df
##          ctl        trt
## 1  0.0000000 0.50000000
## 2  0.7268041 0.23770492
## 3  0.5206186 0.33606557
## 4  1.0000000 0.00000000
## 5  0.1701031 0.93442623
## 6  0.2268041 0.09836066
## 7  0.5154639 1.00000000
## 8  0.1855670 0.53278689
## 9  0.5979381 0.29918033
## 10 0.5000000 0.45081967
```

Z-Score Normalization

```
df1 <- data.frame(ctl,trt)
df1
##     ctl  trt
## 1  4.17 4.81
## 2  5.58 4.17
## 3  5.18 4.41
## 4  6.11 3.59
## 5  4.50 5.87
## 6  4.61 3.83
## 7  5.17 6.03
## 8  4.53 4.89
## 9  5.33 4.32
## 10 5.14 4.69
df1$ctl <- (df1$ctl - mean(df1$ctl))/sd(df1$ctl)
df1$trt <- (df1$trt - mean(df1$trt))/sd(df1$trt)
df1
##           ctl         trt
## 1  -1.4783275  0.18773411
## 2   0.9398184 -0.61864059
## 3   0.2538196 -0.31625008
## 4   1.8487668 -1.34941766
## 5  -0.9123784  1.52329220
## 6  -0.7237288 -1.04702715
## 7   0.2366696  1.72488588
## 8  -0.8609285  0.28853095
## 9   0.5110691 -0.42964652
## 10  0.1852197  0.03653885
```

Decimal Scaling Normalization

```
df2 <- data.frame(ctl,trt)
df2$ctl <- (df2$ctl)/(10^2)
df2$trt <- (df2$trt)/(10^2)
df2
##       ctl    trt
## 1  0.0417 0.0481
## 2  0.0558 0.0417
## 3  0.0518 0.0441
## 4  0.0611 0.0359
## 5  0.0450 0.0587
## 6  0.0461 0.0383
## 7  0.0517 0.0603
## 8  0.0453 0.0489
## 9  0.0533 0.0432
## 10 0.0514 0.0469
```

DATA PARTITION

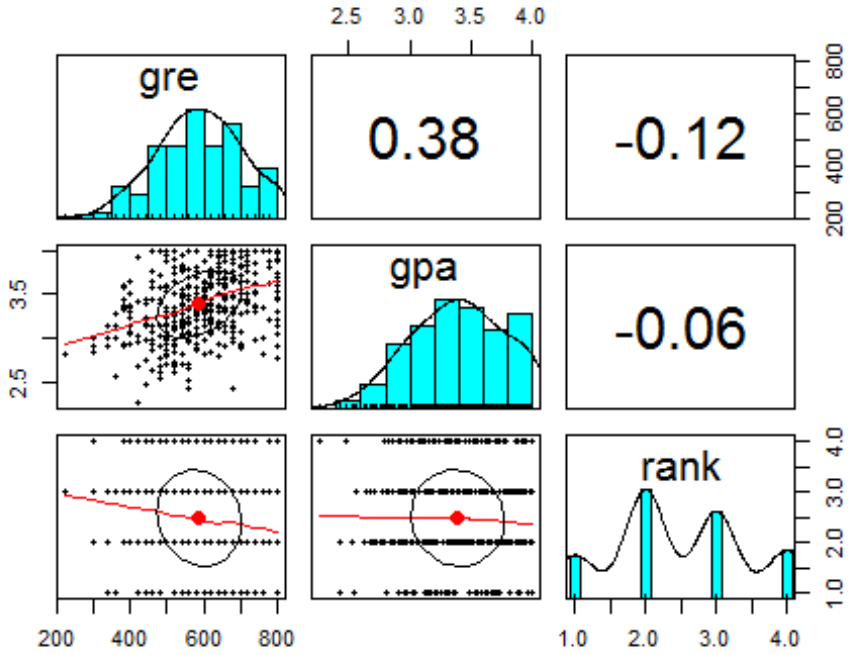
```
data <- read.csv("binary.csv",header = T,sep = ",")
str(data)
## 'data.frame':    400 obs. of  4 variables:
##  $ admit: int  0 1 1 1 0 1 1 0 1 0 ...
##  $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
##  $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
##  $ rank : int  3 3 1 4 4 2 1 2 3 2 ...
```

do the Crosstabulation for admit and Rank

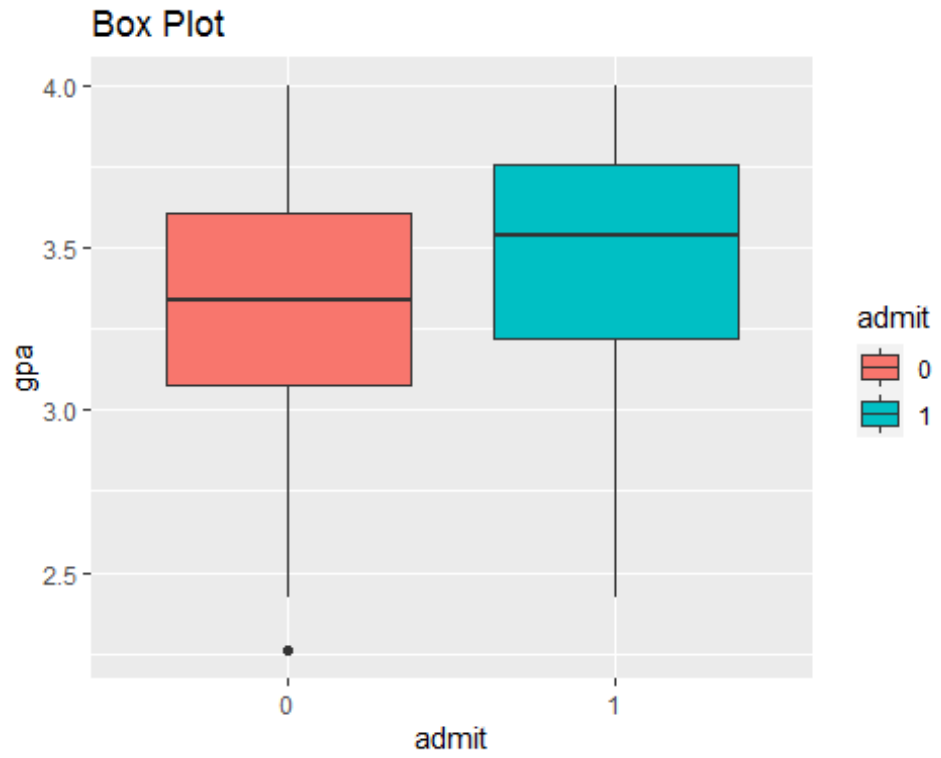
```
xtabs(~admit+rank,data = data)
##      rank
## admit  1  2  3  4
##     0 28 97 93 55
##     1 33 54 28 12
```

Convert rank and admit variable to factor

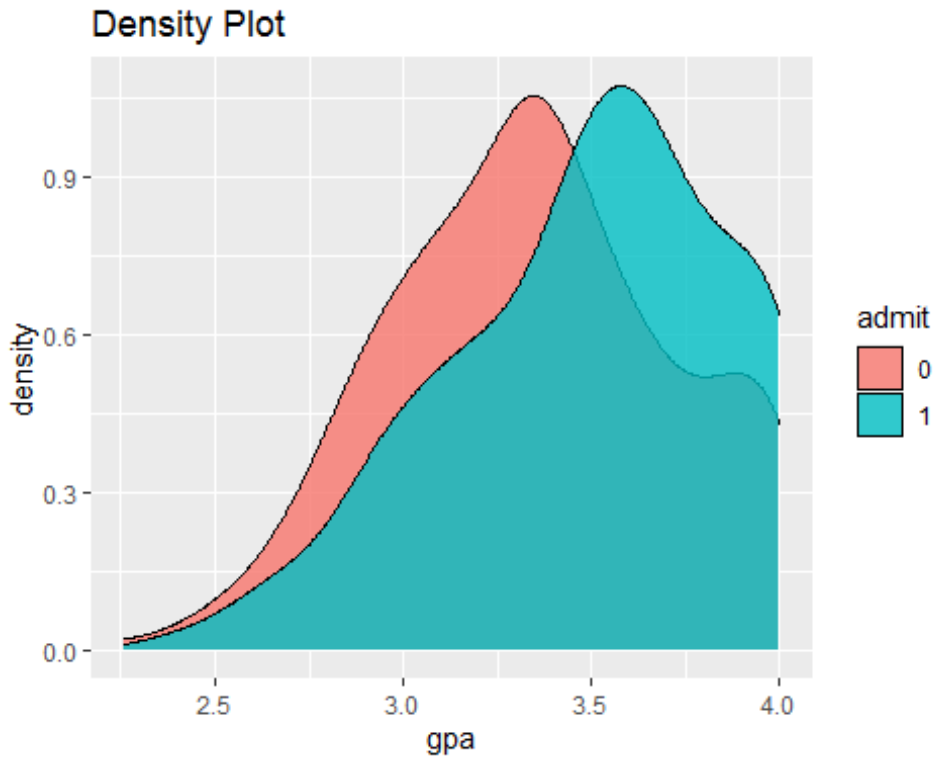
```
data$rank <- as.factor(data$rank)
data$admit <- as.factor(data$admit)
str(data)
## 'data.frame':    400 obs. of  4 variables:
##  $ admit: Factor w/ 2 levels "0","1": 1 2 2 2 1 2 2 1 2 1 ...
##  $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
##  $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
##  $ rank : Factor w/ 4 levels "1","2","3","4": 3 3 1 4 4 2 1 2 3 2 ...
# Visualization
pairs.panels(data[-1])
```




```
data %>%
  ggplot(aes(x=admit,y=gpa,fill=admit)) +
  geom_boxplot()+
  ggtitle("Box Plot")
```



```
data %>%
  ggplot(aes(x=gpa,fill=admit))+
  geom_density(alpha=0.8,color='black')+
  ggtitle('Density Plot')
```




```
set.seed(1234)
ind = sample(2,nrow(data),replace = T,prob = c(0.8,0.2))
train <- data[ind == 1,]
test <- data[ind == 2, ]
```

BUILD MODEL

```
library(naivebayes)
model <- naive_bayes(admit ~., data = train)
model

train %>%
  filter(admit == "0") %>%
  summarise(mean(gre),sd(gre))
##   mean(gre) sd(gre)
## 1  578.6547 116.325
```

PREDICT MODEL

```
p <- predict(model,train,type = 'prob')
## Warning: predict.naive_bayes(): more features in the newdata are provided
as
## there are probability tables in the object. Calculation is performed based
on
## features to be found in the tables.
```

```
head(cbind(p,train))
##           0         1 admit gre  gpa rank
## 1 0.8449088 0.1550912     0 380 3.61    3
## 2 0.6214983 0.3785017     1 660 3.67    3
## 3 0.2082304 0.7917696     1 800 4.00    1
## 4 0.8501030 0.1498970     1 640 3.19    4
## 6 0.6917580 0.3082420     1 760 3.00    2
## 7 0.6720365 0.3279635     1 560 2.98    1
```

Confusion Matrix and Misclassification - train Data

```
p1 <- predict(model,train)
## Warning: predict.naive_bayes(): more features in the newdata are provided
as
## there are probability tables in the object. Calculation is performed based
on
## features to be found in the tables.
tab1 <- table(p1,train$admit)
tab1
##
## p1    0   1
##   0 196  69
##   1  27  33
1-sum(diag(tab1))/sum(tab1)
## [1] 0.2953846
```

Confusion matrix and Misclassification - test Data

```
p2 <- predict(model,test)
## Warning: predict.naive_bayes(): more features in the newdata are provided
as
## there are probability tables in the object. Calculation is performed based
on
## features to be found in the tables.
(tab2 <- table(p2,test$admit))
##
## p2   0  1
##   0 47 21
##   1  3  4
1-sum(diag(tab2))/sum(tab2)
## [1] 0.32
```

PROBLEM 1 : OVERFITTING AND UNDERFITTING

Overfitting: Good performance on the training data and poor generliazation to other data.

Underfitting: Poor performance on the training data and poor generalization to other data.

Split Data training and testing and check if the model performance

Confusion Matrix - train Data

```
p1 <- predict(model,train)
tab1 <- table(p1,train$admit)
tab1
##
## p1    0   1
##   0 196  69
##   1  27  33
1-sum(diag(tab1))/sum(tab1)
## [1] 0.2953846
```

Confusion Matrix – Test Data

```
p2 <- predict(model,test)
(tab2 <- table(p2,test$admit))
##
## p2   0  1
##   0 47 21
##   1  3  4
1-sum(diag(tab2))/sum(tab2)
## [1] 0.32
```

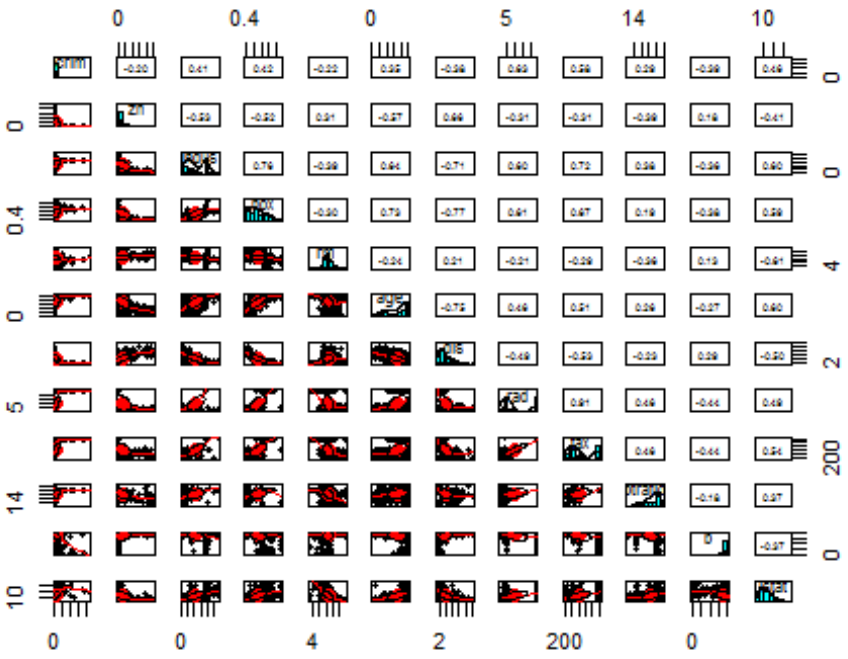
HANDLING OVERFITTING AND UNDERFITTING

Overfitting and UnderFitting -> use Regularization or Cross validation Methods

Regularization

```
library(caret)
## Loading required package: lattice
library(glmnet)
## Loading required package: Matrix
## Loaded glmnet 3.0-2
library(mlbench)
library(psych)
data("BostonHousing")
data <- BostonHousing
str(data)
```

```
pairs.panels(data[c(-4,-14)])
```




```
set.seed(222)
ind <- sample(2,nrow(data),replace = T,prob = c(0.7,0.3))
train <- data[ind == 1,]
test <- data[ind == 2,]
```

Cross validation

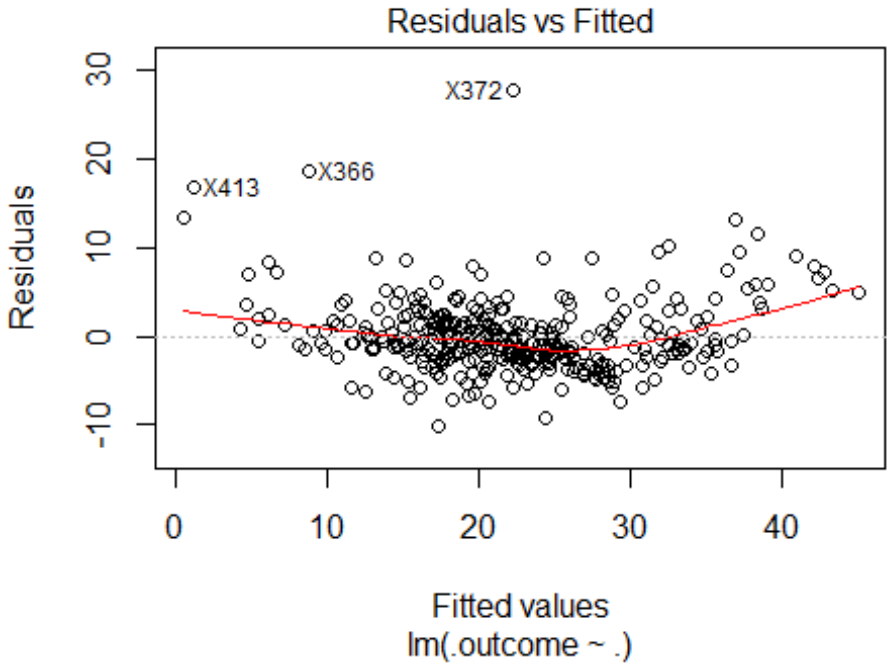
```
custom <- trainControl(method = "repeatedcv",number = 10,repeats = 5,verboseI
ter = T)
set.seed(1234)
lm <- train(medv ~., train,method="lm",trControl=custom)
## - Fold09.Rep5: intercept=TRUE
## + Fold10.Rep5: intercept=TRUE
## - Fold10.Rep5: intercept=TRUE
## Aggregating results
## Fitting final model on full training set
```

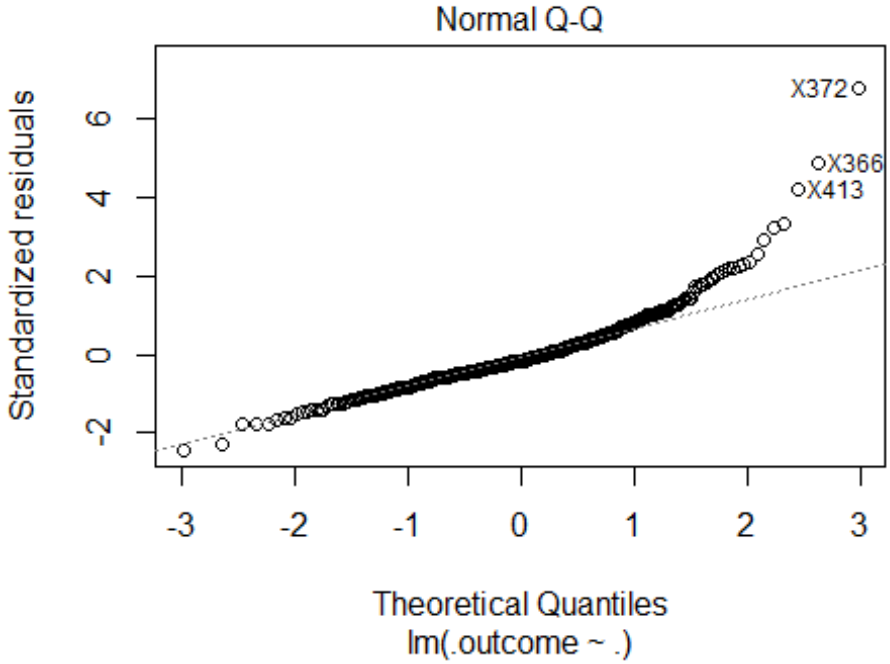
results

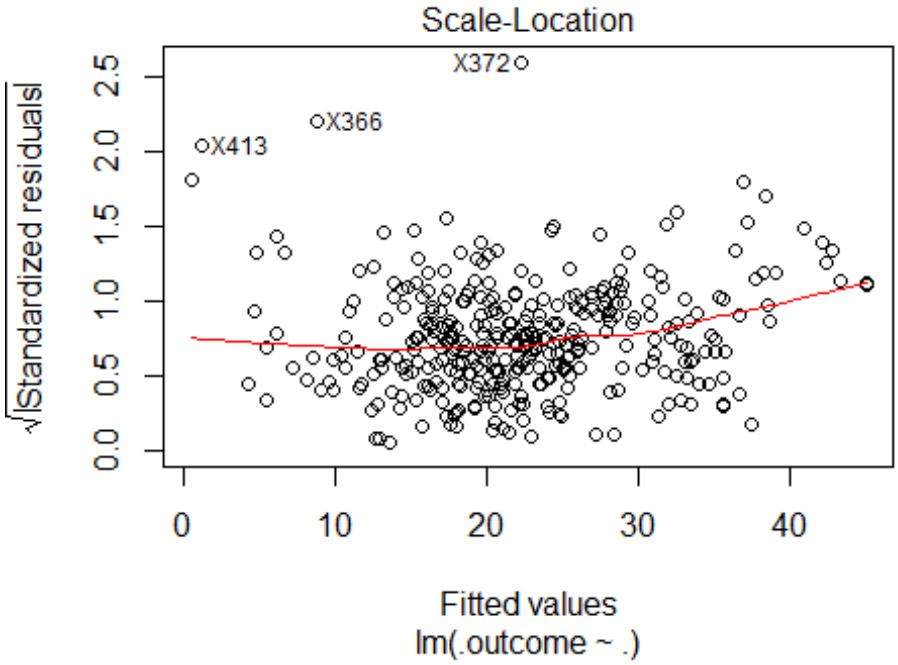
```
lm$results
##   intercept    RMSE Rsquared      MAE    RMSESD RsquaredSD     MAESD
## 1      TRUE 4.23222 0.778488 3.032342 0.9833981 0.09350015 0.4154734
lm
## Linear Regression
##
## 353 samples
##  13 predictor
##
## No pre-processing
```

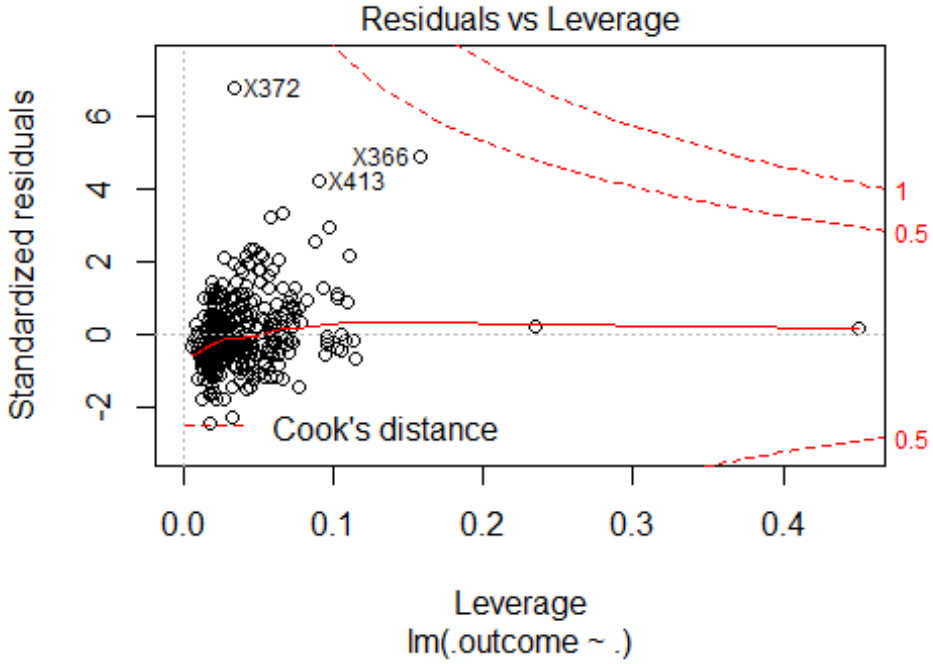
```
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 316, 318, 318, 319, 317, 318, ...
## Resampling results:
##
##   RMSE     Rsquared  MAE
##   4.23222  0.778488  3.032342
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
summary(lm)
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.1018  -2.3528  -0.7279   1.7047  27.7868
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  25.742808   5.653389   4.554 7.37e-06 ***
## crim         -0.165452   0.036018  -4.594 6.15e-06 ***
## zn            0.047202   0.015401   3.065 0.002352 **
## indus         0.013377   0.067401   0.198 0.842796
## chas1         1.364633   0.947288   1.441 0.150630
## nox         -13.065313   4.018576  -3.251 0.001264 **
## rm            5.072891   0.468889  10.819  < 2e-16 ***
## age          -0.028573   0.013946  -2.049 0.041247 *
## dis          -1.421107   0.208908  -6.803 4.66e-11 ***
## rad           0.260863   0.070092   3.722 0.000232 ***
## tax          -0.013556   0.004055  -3.343 0.000922 ***
## ptratio      -0.906744   0.139687  -6.491 3.03e-10 ***
## b             0.008912   0.002986   2.985 0.003040 **
## lstat        -0.335149   0.056920  -5.888 9.40e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.192 on 339 degrees of freedom
## Multiple R-squared:  0.7874, Adjusted R-squared:  0.7793
## F-statistic: 96.59 on 13 and 339 DF,  p-value: < 2.2e-16
```

```r
plot(lm$finalModel)
```






Scale-Location
X372
X366
X413
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
0
10
20
30
40
Fitted values
lm(.outcome ~ .)


Residuals vs Leverage
X372
X366
X413
Standardized residuals
-2
0
2
4
6
1
0.5
0.5
Cook's distance
0.0
0.1
0.2
0.3
0.4
Leverage
lm(.outcome ~ .)

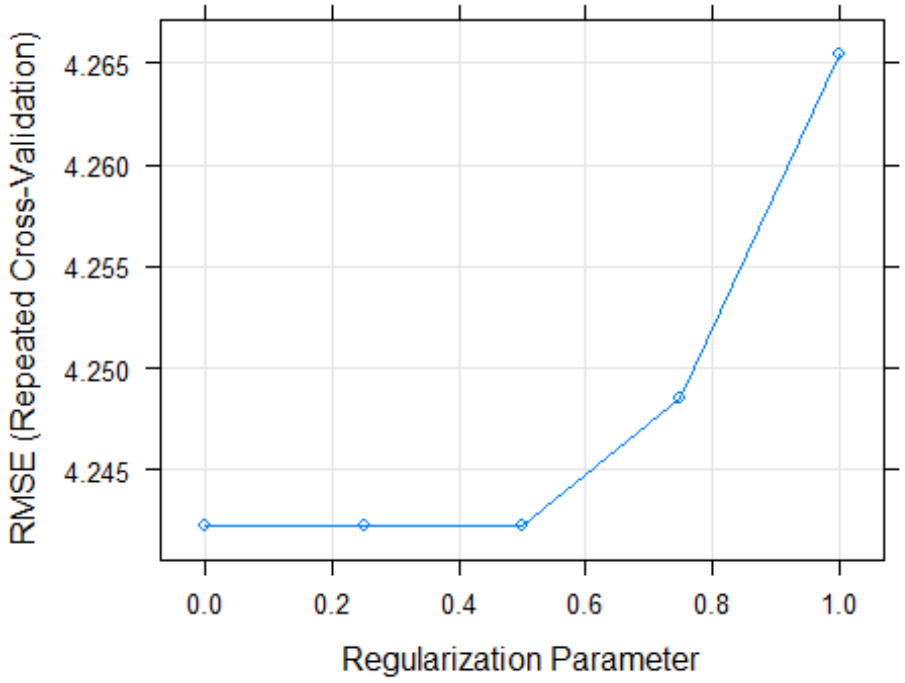
Ridge Regression

```
set.seed(1234)
ridge <- train(medv ~., train,method='glmnet',tuneGrid = expand.grid(alpha =
0,
                                                                     lambda =
seq(0.0001,1,length=5)),
               trControl= custom)

## - Fold10.Rep5: alpha=0, lambda=1
## Aggregating results
## Selecting tuning parameters
## Fitting alpha = 0, lambda = 0.5 on full training set
```

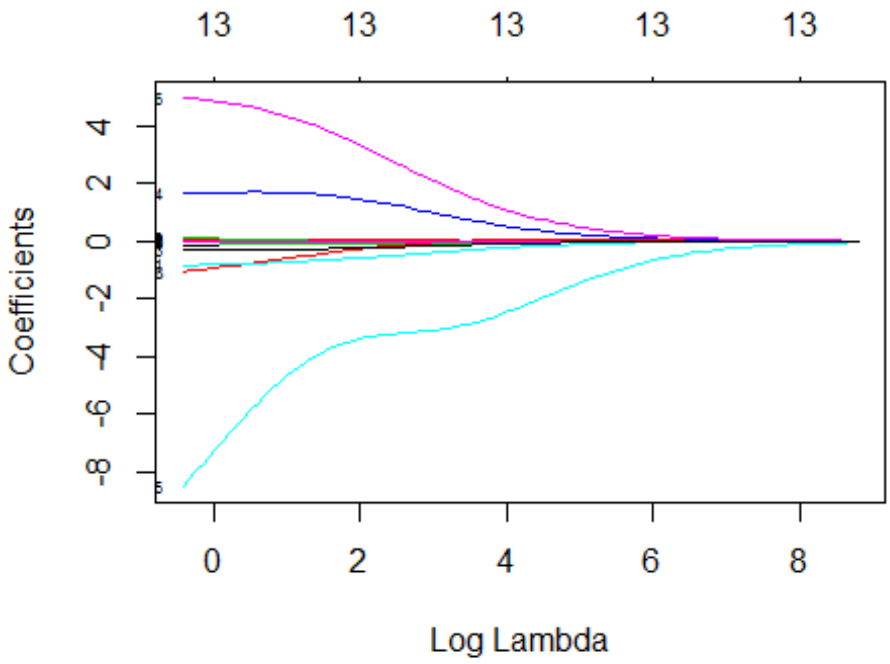
Ridge results

```
plot(ridge)
```

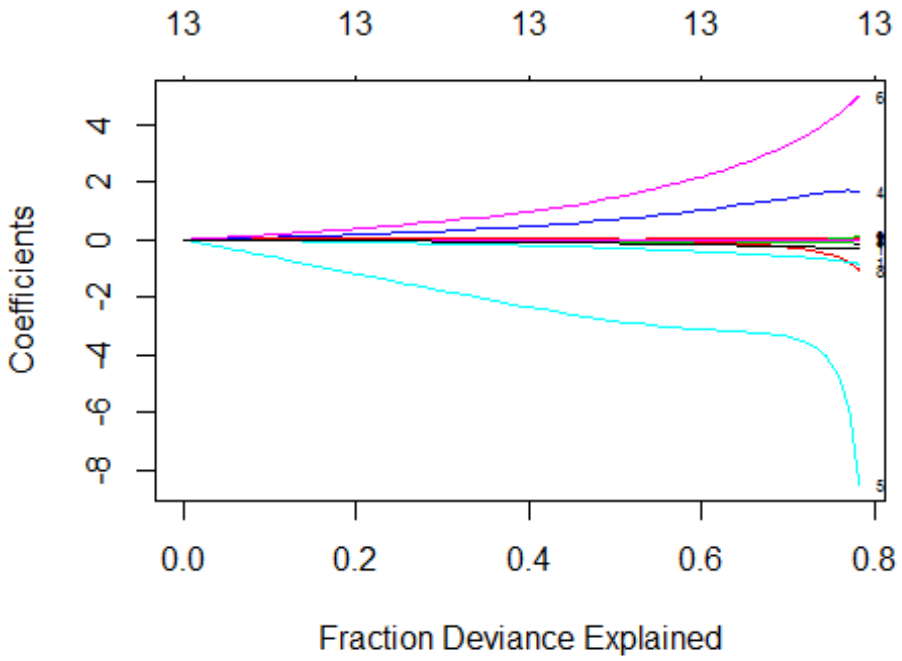



```
ridge
## glmnet
##
## 353 samples
##  13 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 316, 318, 318, 319, 317, 318, ...
## Resampling results across tuning parameters:
##
##   lambda    RMSE      Rsquared   MAE
```

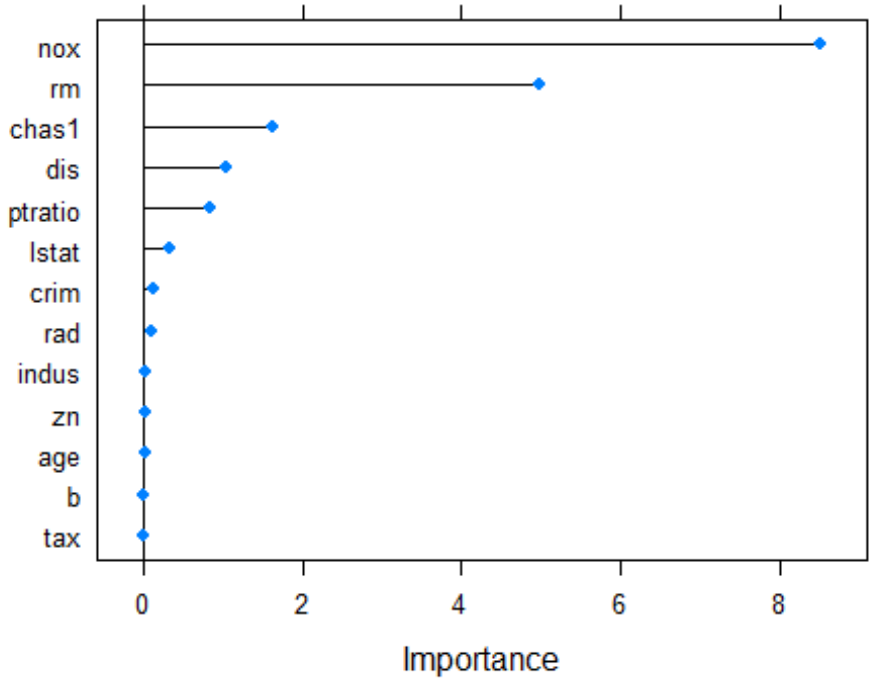
```
##   0.000100  4.242204  0.7782278  3.008339
##   0.250075  4.242204  0.7782278  3.008339
##   0.500050  4.242204  0.7782278  3.008339
##   0.750025  4.248536  0.7779462  3.012397
##   1.000000  4.265479  0.7770264  3.023091
##
## Tuning parameter 'alpha' was held constant at a value of 0
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0 and lambda = 0.50005.
plot(ridge$finalModel,xvar = "lambda",label = T)
```



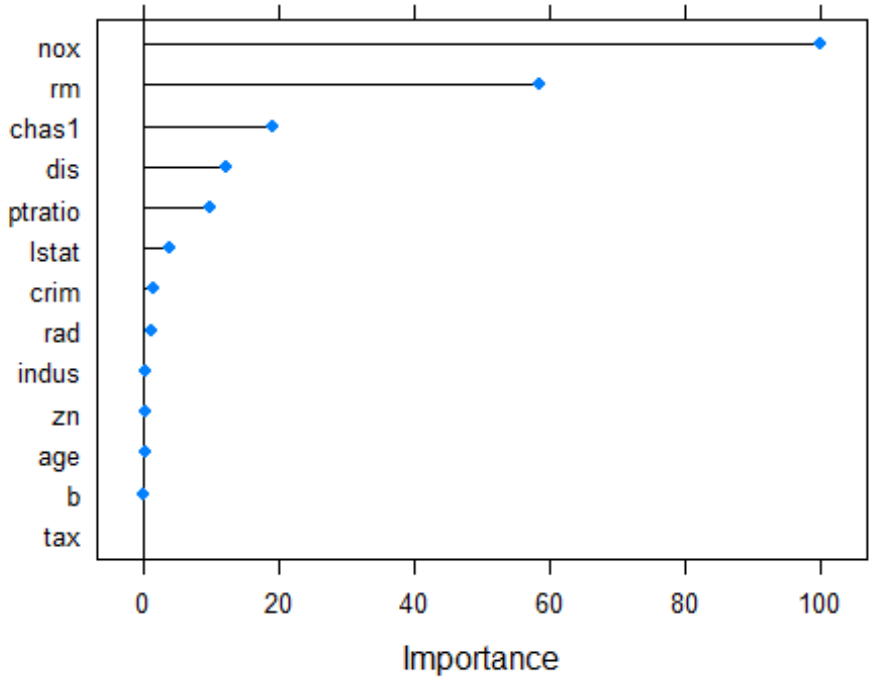

```
plot(ridge$finalModel,xvar = "dev",label = T)
```




```
plot(varImp(ridge,scale = F))
```




```
plot(varImp(ridge,scale = T))
```

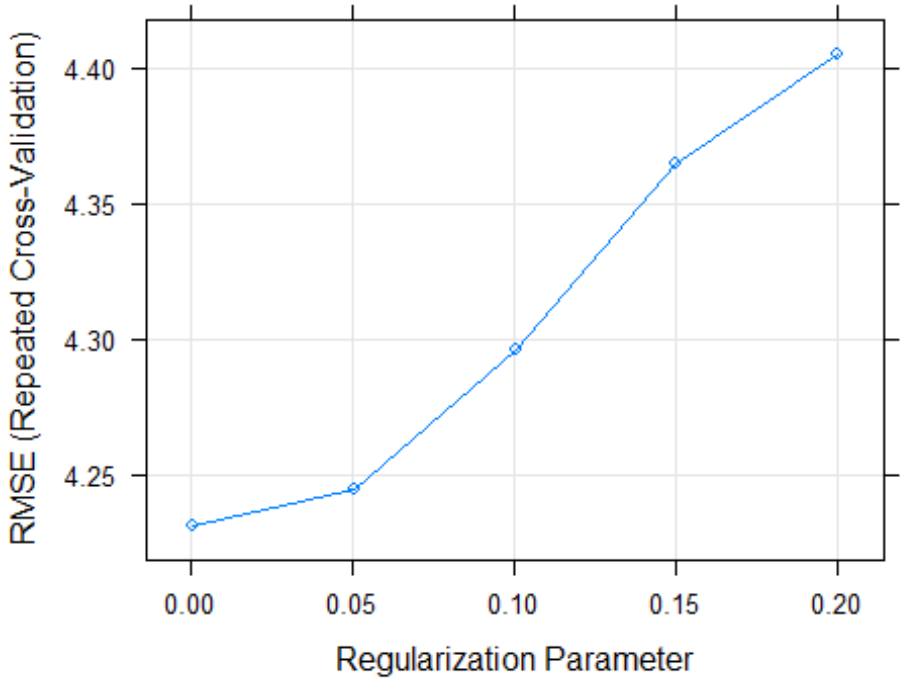

nox
rm
chas1
dis
ptratio
lstat
crim
rad
indus
zn
age
b
tax
0
20
40
60
80
100
Importance

Lasso Regression

```
set.seed(1234)
lasso <- train(medv ~., train,method='glmnet',tuneGrid=expand.grid(alpha=1,
                                                                   lambda=seq
(0.0001,0.2,length=5)),
               trControl=custom)
## + Fold10.Rep5: alpha=1, lambda=0.2
## - Fold10.Rep5: alpha=1, lambda=0.2
## Aggregating results
## Selecting tuning parameters
## Fitting alpha = 1, lambda = 1e-04 on full training set
```

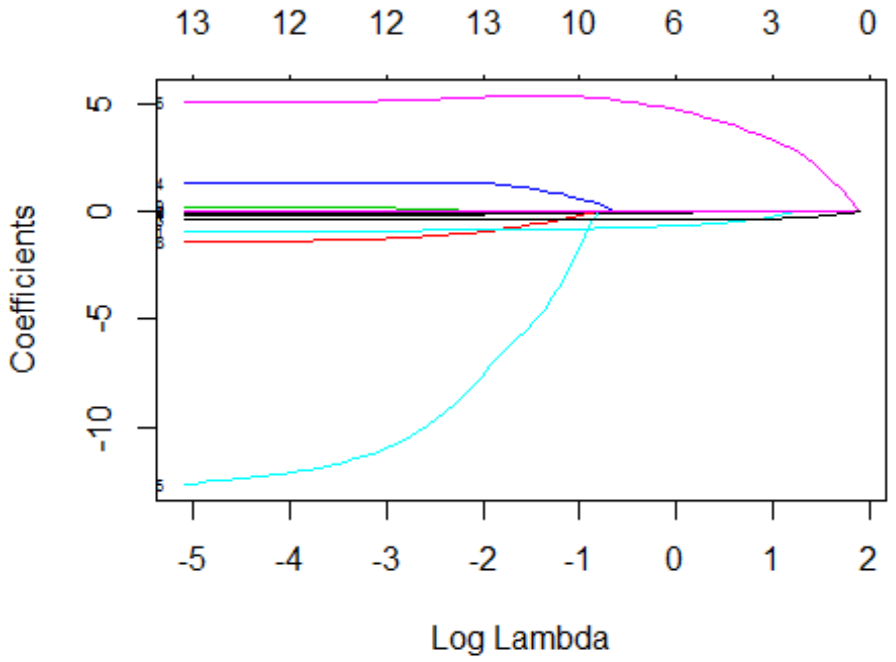
LASSO RESULT

```
plot(lasso)
```

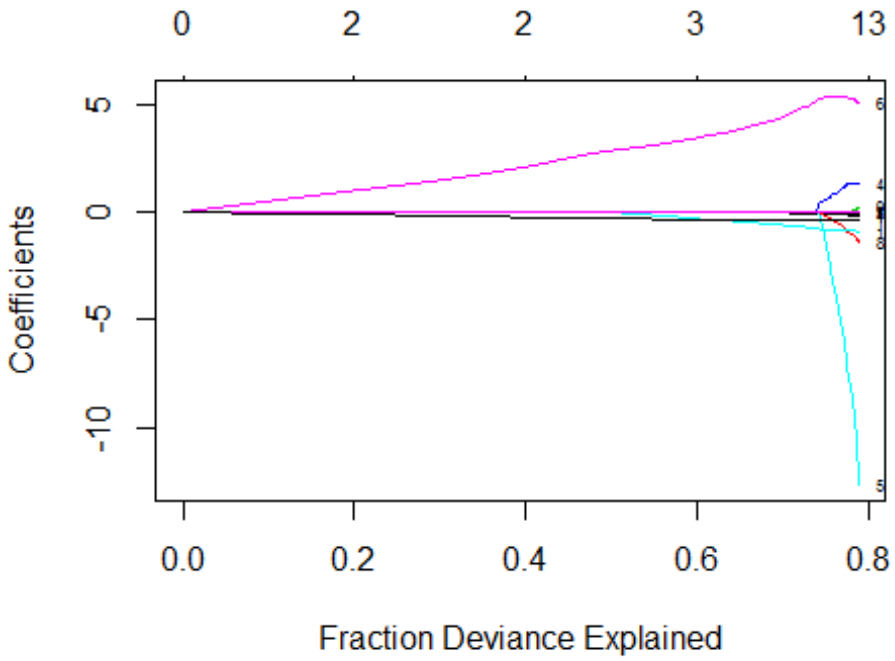



```
lasso
## glmnet
##
## 353 samples
##  13 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 316, 318, 318, 319, 317, 318, ...
## Resampling results across tuning parameters:
##
##   lambda    RMSE      Rsquared   MAE
##   0.000100  4.230700  0.7785841  3.025998
```

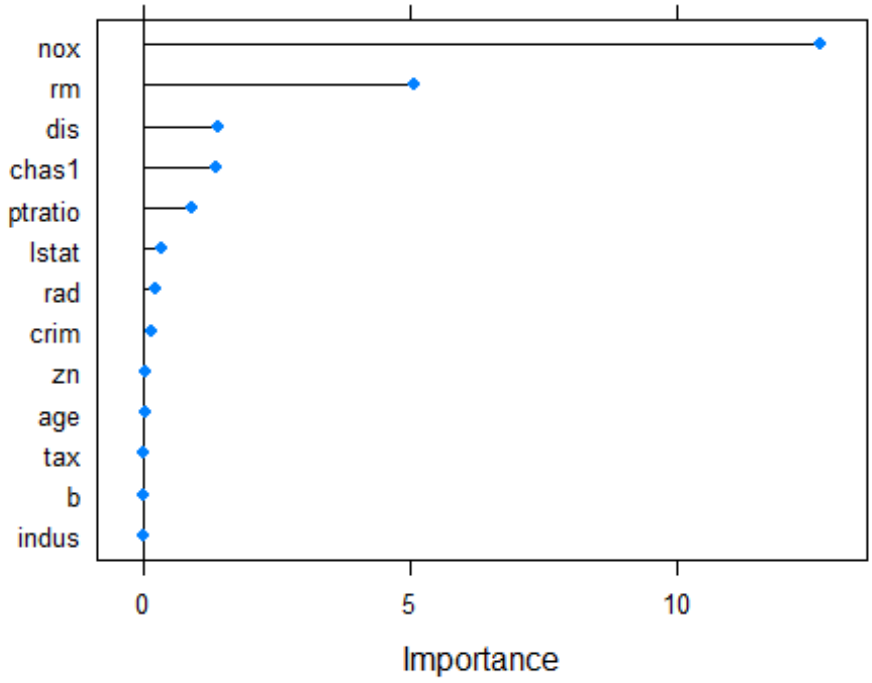
```
##   0.050075  4.244334  0.7770330  3.011344
##   0.100050  4.295773  0.7719043  3.035321
##   0.150025  4.364484  0.7651821  3.076854
##   0.200000  4.405206  0.7617009  3.102491
##
## Tuning parameter 'alpha' was held constant at a value of 1
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 1 and lambda = 1e-04.
plot(lasso$finalModel,xvar = "lambda",label = T)
```



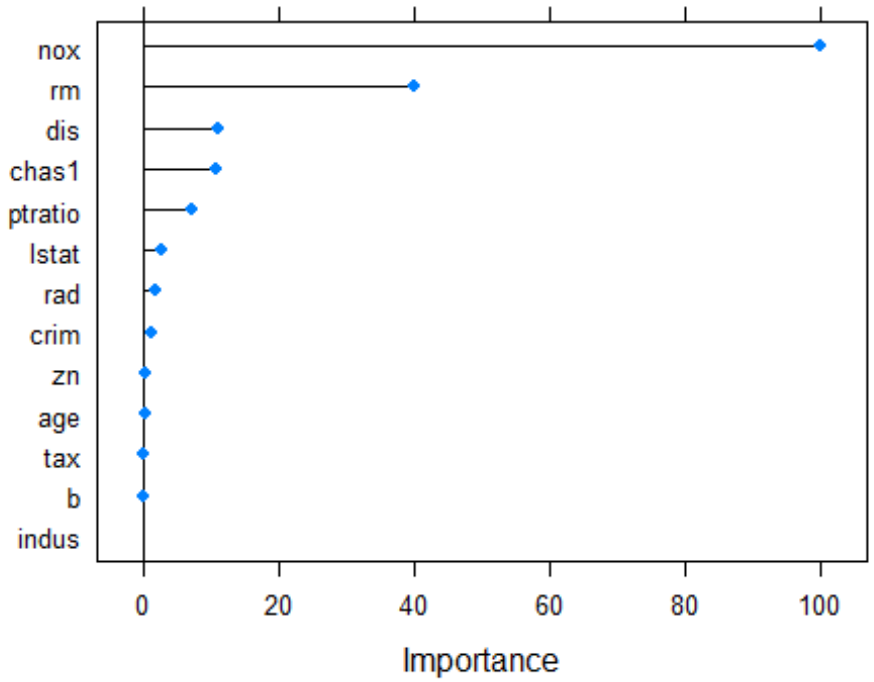

```
plot(lasso$finalModel,xvar = "dev",label = T)
```




```
plot(varImp(lasso,scale = F))
```




```
plot(varImp(lasso,scale = T))
```


nox
rm
dis
chas1
ptratio
lstat
rad
crim
zn
age
tax
b
indus
0
20
40
60
80
100
Importance

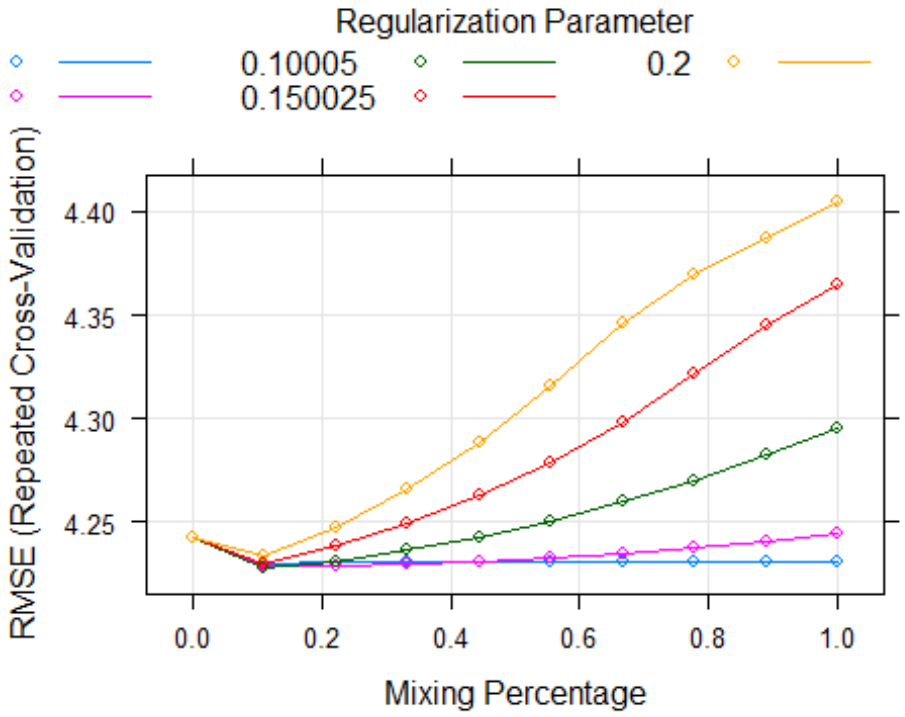
Elastic Net Regression

```
set.seed(1234)
en <- train(medv ~., train,method='glmnet',tuneGrid=expand.grid(alpha=seq(0,1
,length=10),
                                                                lambda=seq
(0.0001,0.2,length=5)),
                trControl=custom)

## - Fold10.Rep5: alpha=1.0000, lambda=0.2
## Aggregating results
## Selecting tuning parameters
## Fitting alpha = 0.111, lambda = 0.1 on full training set
```

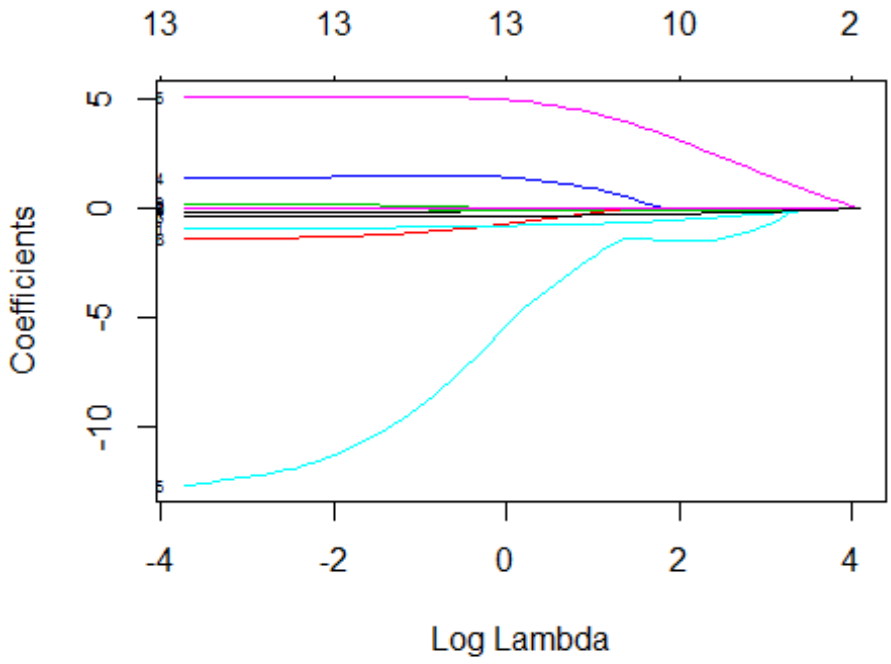
ELASTIC NET RESULT

```
plot(en)
```

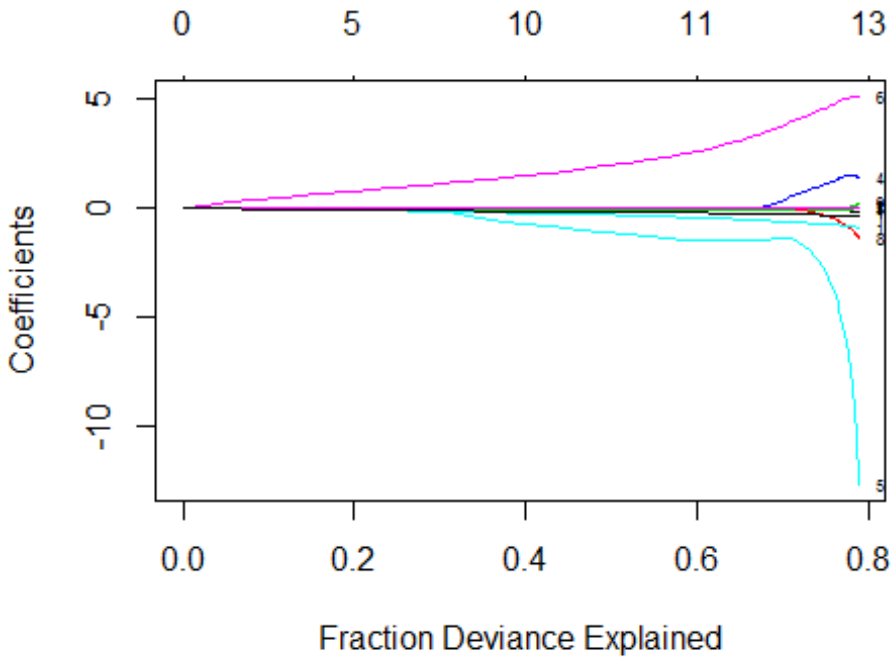



```
en
## glmnet
##
## 353 samples
##  13 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 316, 318, 318, 319, 317, 318, ...
## Resampling results across tuning parameters:
##
##   alpha      lambda     RMSE      Rsquared   MAE
```

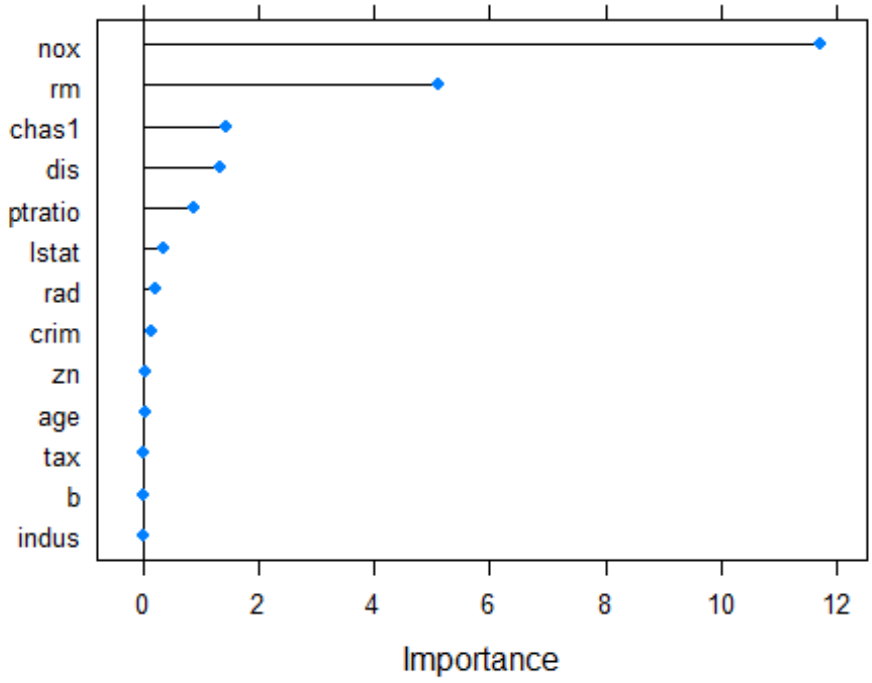
```
##   0.0000000  0.000100  4.242204  0.7782278  3.008339
##   0.0000000  0.050075  4.242204  0.7782278  3.008339
##   0.0000000  0.100050  4.242204  0.7782278  3.008339
##   0.0000000  0.150025  4.242204  0.7782278  3.008339
##   0.0000000  0.200000  4.242204  0.7782278  3.008339
##   0.1111111  0.000100  4.230292  0.7786226  3.025857
##   0.1111111  0.050075  4.228437  0.7787777  3.019236
##   0.1111111  0.100050  4.227739  0.7788251  3.010332
##   0.1111111  0.150025  4.229814  0.7786315  3.005266
##   0.1111111  0.200000  4.233949  0.7782676  3.003662
##   0.2222222  0.000100  4.230694  0.7785669  3.026161
##   0.2222222  0.050075  4.228863  0.7787036  3.017107
##   0.2222222  0.100050  4.231209  0.7784424  3.008141
##   0.2222222  0.150025  4.238397  0.7777559  3.006343
##   0.2222222  0.200000  4.247863  0.7768865  3.010248
##   0.3333333  0.000100  4.230795  0.7785677  3.026282
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0.1111111 and lambda = 0.
10005.
plot(en$finalModel,xvar = "lambda",label = T)
```



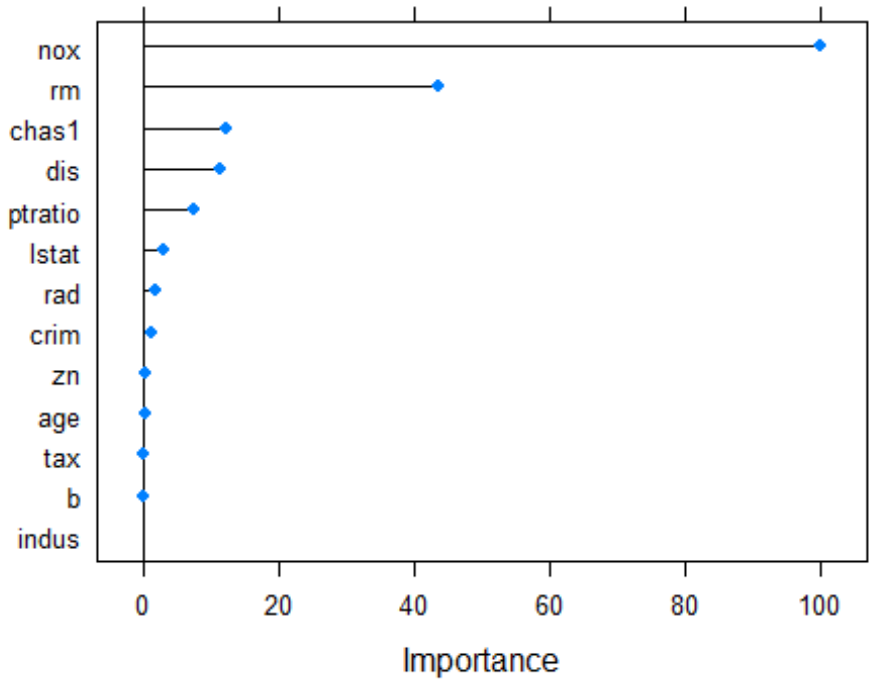

```
plot(en$finalModel,xvar = "dev",label = T)
```




```
plot(varImp(en,scale = F))
```




```
plot(varImp(en,scale = T))
```

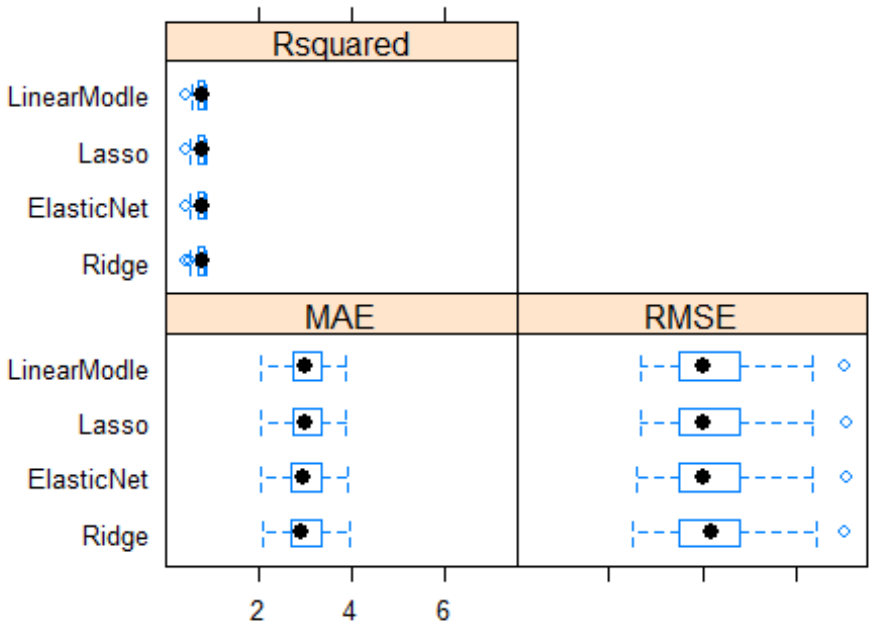


Compare Models

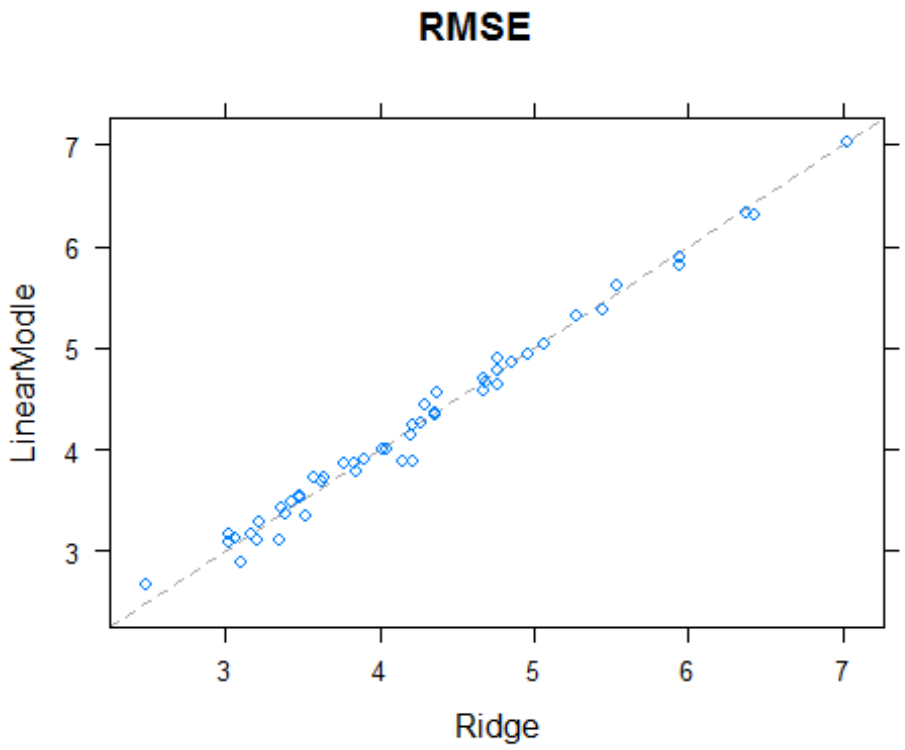
```
model_list <- list(LinearModle = lm,Ridge=ridge,Lasso=lasso,ElasticNet=en)

res <- resamples(model_list)
summary(res)
##
## Call:
## summary.resamples(object = res)
##
## Models: LinearModle, Ridge, Lasso, ElasticNet
## Number of resamples: 50
##
## MAE
##                 Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## LinearModle 2.080208 2.767061 3.002455 3.032342 3.355281 3.874270    0
## Ridge       2.094151 2.736246 2.934350 3.008339 3.366834 3.971337    0
## Lasso       2.072408 2.764289 2.988132 3.025998 3.346437 3.882800    0
## ElasticNet  2.059901 2.733120 2.959587 3.010332 3.340553 3.907873    0
##
## RMSE
##                 Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## LinearModle 2.673817 3.495197 3.998562 4.232220 4.751509 7.027551    0
## Ridge       2.478993 3.477912 4.169422 4.242204 4.759265 7.035089    0
## Lasso       2.650331 3.490881 3.993362 4.230700 4.748958 7.040494    0
## ElasticNet  2.595730 3.475199 3.993399 4.227739 4.741667 7.049079    0
##
## Rsquared
```

```
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA
's
## LinearModle 0.4865769 0.7269864 0.7991104 0.7784880 0.8472274 0.9128278
0
## Ridge       0.4796929 0.7339342 0.8018589 0.7782278 0.8459744 0.9141020
0
## Lasso       0.4848588 0.7272700 0.8002386 0.7785841 0.8475939 0.9138499
0
## ElasticNet  0.4826785 0.7281150 0.8028115 0.7788251 0.8487030 0.9149131
0
bwplot(res)
```




```
xyplot(res,metric = 'RMSE')
```



CHOOSE BEST MODEL

```
en$bestTune
##       alpha  lambda
## 8 0.1111111 0.10005
best <- en$finalModel
coef(best,s=en$bestTune$lambda)
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)  23.836156580
## crim         -0.155545721
## zn            0.042093859
## indus        -0.003887724
## chas1         1.430596788
## nox         -11.717596093
## rm            5.096463815
## age          -0.027710003
## dis          -1.316292494
## rad           0.209120525
## tax          -0.011052950
## ptratio      -0.889605615
## b             0.008657381
## lstat        -0.334763519
```

Save Final Model for Later Use

```
saveRDS(en,"Final_model.rds")
```

Read Model

```
fm <- readRDS("Final_model.rds")
print(fm)
##   1.0000000  0.150025  4.364484  0.7651821  3.076854
##   1.0000000  0.200000  4.405206  0.7617009  3.102491
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0.1111111 and lambda = 0.
10005.
```

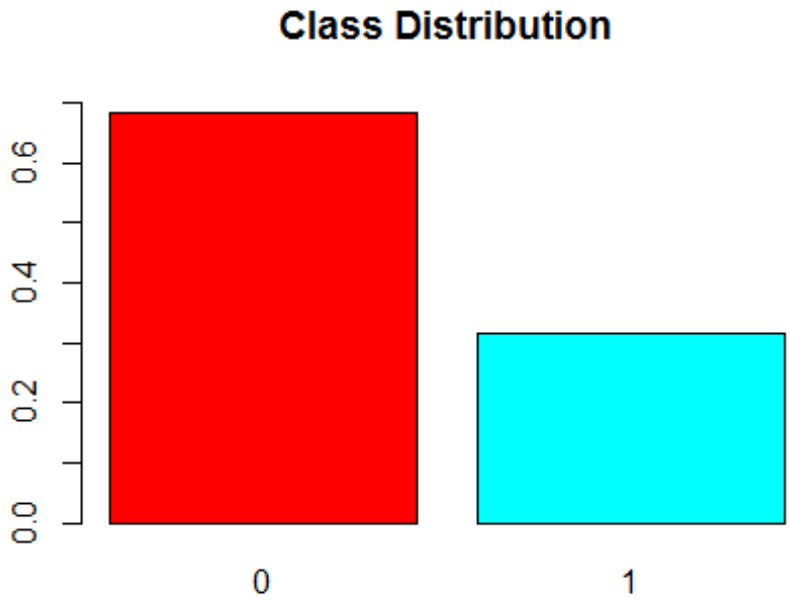
Prediction

```
p1 <- predict(fm,train)
sqrt(mean((train$medv-p1)^2))
## [1] 4.113352
p2 <- predict(fm,test)
sqrt(mean((test$medv-p2)^2))
## [1] 6.154483
```

PROBLEM 2 : CLASS IMBALANCE

DETECT CLASS IMBALANCE

```
data <- read.csv("binary.csv",header = T,sep = ",")
str(data)
## 'data.frame':    400 obs. of  4 variables:
##  $ admit: int  0 1 1 1 0 1 1 0 1 0 ...
##  $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
##  $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
##  $ rank : int  3 3 1 4 4 2 1 2 3 2 ...
data$admit <- as.factor(data$admit)
summary(data)
##  admit        gre             gpa             rank
##  0:273   Min.   :220.0   Min.   :2.260   Min.   :1.000
##  1:127   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:2.000
##          Median :580.0   Median :3.395   Median :2.000
##          Mean   :587.7   Mean   :3.390   Mean   :2.485
##          3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000
##          Max.   :800.0   Max.   :4.000   Max.   :4.000
prop.table(table(data$admit)) # Here we numerically saw the 0 class is higher
than 1 hence there class imbalance
##
##      0      1
## 0.6825 0.3175
barplot(prop.table(table(data$admit)),col=rainbow(2),ylim = c(0,0.7),main = "
Class Distribution") # Here We saw visually
```



Split Data

```
set.seed(123)
ind <- sample(2,nrow(data),replace = T,prob = c(0.7,0.3))
train <- data[ind == 1,]
test <- data[ind == 2,]
```

Check count and Probability of train dataset

```
table(train$admit)
##
##   0   1
## 188  97
prop.table(table(train$admit))
##
##         0         1
## 0.6596491 0.3403509
```

Build a Model

```
library(randomForest)
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
## The following object is masked from 'package:psych':
##
##     outlier
## The following object is masked from 'package:dplyr':
##
##     combine
rftrain <- randomForest(admit ~., data = train)
```

Confusion Matrix

```
library(caret)
confusionMatrix(predict(rftrain,test),test$admit, positive = '1')
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 70 22
##          1 15  8
##
##                Accuracy : 0.6783
##                  95% CI : (0.5847, 0.7623)
##     No Information Rate : 0.7391
##     P-Value [Acc > NIR] : 0.9418
##
##                   Kappa : 0.0976
##
##  Mcnemar's Test P-Value : 0.3239
```

```
##
##             Sensitivity : 0.26667
##             Specificity : 0.82353
##          Pos Pred Value : 0.34783
##          Neg Pred Value : 0.76087
##              Prevalence : 0.26087
##          Detection Rate : 0.06957
##    Detection Prevalence : 0.20000
##       Balanced Accuracy : 0.54510
##
##        'Positive' Class : 1
##
# Confusion matrix shows low Accuracy and
# it shows high differnce in Sensitivity and Specificity
# so we can conclude there is Class imbalance problem in this Model
```

HANDLING CLASS IMBALANCE

Class Imbalance – Use

Under Sampling

Over Sampling

Both under and Over sampling

generate Synthetic Data

check which model is performing best and keep that for model building

Over Sampling for best sensitivity

```
library(ROSE)
## Loaded ROSE 0.0-3
over <- ovun.sample(admit ~.,data=train,method = "over",N=376)$data # -- Over sampling
table(over$admit)
##
##   0   1
## 188 188
rfover <- randomForest(admit~.,data = over)
confusionMatrix(predict(rfover,test),test$admit, positive = '1')
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 60 15
##          1 25 15
##
##                Accuracy : 0.6522
##                  95% CI : (0.5577, 0.7386)
##     No Information Rate : 0.7391
```

```
##     P-Value [Acc > NIR] : 0.9852
##
##                   Kappa : 0.1858
##
##  Mcnemar's Test P-Value : 0.1547
##
##             Sensitivity : 0.5000
##             Specificity : 0.7059
##          Pos Pred Value : 0.3750
##          Neg Pred Value : 0.8000
##              Prevalence : 0.2609
##          Detection Rate : 0.1304
##    Detection Prevalence : 0.3478
##       Balanced Accuracy : 0.6029
##
##        'Positive' Class : 1
##
```

Under Sampling

```
under <- ovun.sample(admit~.,data=train,method = "under",N=194)$data #-- Unde
r sampling
table(under$admit)
##
##  0  1
## 97 97
rfunder <- randomForest(admit ~.,data = under)
confusionMatrix(predict(rfunder,test),test$admit, positive = '1')
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 52 12
##          1 33 18
##
##                Accuracy : 0.6087
##                  95% CI : (0.5133, 0.6984)
##     No Information Rate : 0.7391
##     P-Value [Acc > NIR] : 0.999220
##
##                   Kappa : 0.1727
##
##  Mcnemar's Test P-Value : 0.002869
##
##             Sensitivity : 0.6000
##             Specificity : 0.6118
##          Pos Pred Value : 0.3529
##          Neg Pred Value : 0.8125
##              Prevalence : 0.2609
##          Detection Rate : 0.1565
##    Detection Prevalence : 0.4435
```

```
##       Balanced Accuracy : 0.6059
##
##        'Positive' Class : 1
##
```

Both Sampling

```
both <- ovun.sample(admit~.,data=train,method = "both",p=0.5,seed = 222,N=285
)$data # both under and over
table(both$admit)
##
##   0   1
## 134 151
rfboth <- randomForest(admit ~.,data = both)
confusionMatrix(predict(rfboth,test),test$admit, positive = '1')
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 40  9
##          1 45 21
##
##                Accuracy : 0.5304
##                  95% CI : (0.4351, 0.6241)
##     No Information Rate : 0.7391
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1229
##
##  Mcnemar's Test P-Value : 1.908e-06
##
##             Sensitivity : 0.7000
##             Specificity : 0.4706
##          Pos Pred Value : 0.3182
##          Neg Pred Value : 0.8163
##              Prevalence : 0.2609
##          Detection Rate : 0.1826
##    Detection Prevalence : 0.5739
##       Balanced Accuracy : 0.5853
##
##        'Positive' Class : 1
##
```

Synthetic Data Generation

```
rose <- ROSE(admit ~., data=train,N=500,seed = 111)$data
table(rose$admit)
##
##   0   1
## 234 266
rfrose <- randomForest(admit~.,data = rose)
confusionMatrix(predict(rfrose,test),test$admit, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 36 12
##          1 49 18
##
##                Accuracy : 0.4696
##                  95% CI : (0.3759, 0.5649)
##     No Information Rate : 0.7391
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0168
##
##  Mcnemar's Test P-Value : 4.04e-06
##
##             Sensitivity : 0.6000
##             Specificity : 0.4235
##          Pos Pred Value : 0.2687
##          Neg Pred Value : 0.7500
##              Prevalence : 0.2609
##          Detection Rate : 0.1565
##    Detection Prevalence : 0.5826
##       Balanced Accuracy : 0.5118
##
##        'Positive' Class : 1
##
```

PROBLEM 3 : MULTICOLLINARITY

DETECT MULTICOLLINARITY

```
library(faraway)
## Registered S3 methods overwritten by 'lme4':
##   method                          from
##   cooks.distance.influence.merMod car
##   influence.merMod                car
##   dfbeta.influence.merMod         car
##   dfbetas.influence.merMod        car
##
## Attaching package: 'faraway'
## The following object is masked from 'package:lattice':
##
##     melanoma
## The following object is masked from 'package:psych':
##
##     logit
## The following object is masked from 'package:VIM':
##
##     diabetes
## The following object is masked from 'package:mice':
##
##     mammalsleep
data("divusa")
head(divusa)
##   year divorce unemployed femlab marriage birth military
## 1 1920     8.0        5.2  22.70     92.0 117.9   3.2247
## 2 1921     7.2       11.7  22.79     83.0 119.8   3.5614
## 3 1922     6.6        6.7  22.88     79.7 111.2   2.4553
## 4 1923     7.1        2.4  22.97     85.2 110.5   2.2065
## 5 1924     7.2        5.0  23.06     80.3 110.9   2.2889
## 6 1925     7.2        3.2  23.15     79.2 106.6   2.1735
```

Example 1:

```
mydata <- data.frame(divusa[,-1]) # Remvoing year column
head(mydata)
##   divorce unemployed femlab marriage birth military
## 1     8.0        5.2  22.70     92.0 117.9   3.2247
## 2     7.2       11.7  22.79     83.0 119.8   3.5614
## 3     6.6        6.7  22.88     79.7 111.2   2.4553
## 4     7.1        2.4  22.97     85.2 110.5   2.2065
## 5     7.2        5.0  23.06     80.3 110.9   2.2889
## 6     7.2        3.2  23.15     79.2 106.6   2.1735
round(cor(mydata),2)
##             divorce unemployed femlab marriage birth military
## divorce        1.00      -0.21   0.91    -0.53 -0.72     0.02
## unemployed    -0.21       1.00  -0.26    -0.27 -0.31    -0.40
## femlab         0.91      -0.26   1.00    -0.65 -0.60     0.05
## marriage      -0.53      -0.27  -0.65     1.00  0.67     0.26
```

```
## birth        -0.72      -0.31  -0.60     0.67  1.00     0.14
## military      0.02      -0.40   0.05     0.26  0.14     1.00
mymodel <- lm(divorce ~., mydata)
class(mymodel)
## [1] "lm"
summary(mymodel)
##
## Call:
## lm(formula = divorce ~ ., data = mydata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8611 -0.8916 -0.0496  0.8650  3.8300
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.48784    3.39378   0.733   0.4659
## unemployed  -0.11125    0.05592  -1.989   0.0505 .
## femlab       0.38365    0.03059  12.543  < 2e-16 ***
## marriage     0.11867    0.02441   4.861 6.77e-06 ***
## birth       -0.12996    0.01560  -8.333 4.03e-12 ***
## military    -0.02673    0.01425  -1.876   0.0647 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.65 on 71 degrees of freedom
## Multiple R-squared:  0.9208, Adjusted R-squared:  0.9152
## F-statistic: 165.1 on 5 and 71 DF,  p-value: < 2.2e-16
vif(mymodel) ## VIF should be less than 10 if it is greater than 10 then we c
an conclude there is multicollinarity problem
## unemployed     femlab   marriage      birth   military
##   2.252888   3.613276   2.864864   2.585485   1.249596
```

FEATURE SELECTION

```
library(Boruta)
## Loading required package: ranger
##
## Attaching package: 'ranger'
## The following object is masked from 'package:randomForest':
##
##     importance
library(mlbench)
library(caret)
library(randomForest)
data("Sonar")
str(Sonar)
## 'data.frame':    208 obs. of  61 variables:
##  $ V1   : num  0.02 0.0453 0.0262 0.01 0.0762 0.0286 0.0317 0.0519 0.0223
0.0164 ...
##  $ V2   : num  0.0371 0.0523 0.0582 0.0171 0.0666 0.0453 0.0956 0.0548 0.0
```

```
375 0.0173 ...
##  $ V3   : num  0.0428 0.0843 0.1099 0.0623 0.0481 ...
##  $ V4   : num  0.0207 0.0689 0.1083 0.0205 0.0394 ...
##  $ V5   : num  0.0954 0.1183 0.0974 0.0205 0.059 ...
##  $ V6   : num  0.0986 0.2583 0.228 0.0368 0.0649 ...
##  $ V7   : num  0.154 0.216 0.243 0.11 0.121 ...
##  $ V8   : num  0.16 0.348 0.377 0.128 0.247 ...
##  $ V9   : num  0.3109 0.3337 0.5598 0.0598 0.3564 ...
##  $ V10  : num  0.211 0.287 0.619 0.126 0.446 ...
##  $ V11  : num  0.1609 0.4918 0.6333 0.0881 0.4152 ...
##  $ V12  : num  0.158 0.655 0.706 0.199 0.395 ...
##  $ V13  : num  0.2238 0.6919 0.5544 0.0184 0.4256 ...
##  $ V14  : num  0.0645 0.7797 0.532 0.2261 0.4135 ...
##  $ V15  : num  0.066 0.746 0.648 0.173 0.453 ...
##  $ V16  : num  0.227 0.944 0.693 0.213 0.533 ...
##  $ V17  : num  0.31 1 0.6759 0.0693 0.7306 ...
##  $ V18  : num  0.3 0.887 0.755 0.228 0.619 ...
##  $ V19  : num  0.508 0.802 0.893 0.406 0.203 ...
##  $ V20  : num  0.48 0.782 0.862 0.397 0.464 ...
##  $ V21  : num  0.578 0.521 0.797 0.274 0.415 ...
##  $ V22  : num  0.507 0.405 0.674 0.369 0.429 ...
##  $ V23  : num  0.433 0.396 0.429 0.556 0.573 ...
##  $ V24  : num  0.555 0.391 0.365 0.485 0.54 ...
##  $ V25  : num  0.671 0.325 0.533 0.314 0.316 ...
##  $ V26  : num  0.641 0.32 0.241 0.533 0.229 ...
##  $ V27  : num  0.71 0.327 0.507 0.526 0.7 ...
##  $ V28  : num  0.808 0.277 0.853 0.252 1 ...
##  $ V29  : num  0.679 0.442 0.604 0.209 0.726 ...
##  $ V30  : num  0.386 0.203 0.851 0.356 0.472 ...
##  $ V31  : num  0.131 0.379 0.851 0.626 0.51 ...
##  $ V32  : num  0.26 0.295 0.504 0.734 0.546 ...
##  $ V33  : num  0.512 0.198 0.186 0.612 0.288 ...
##  $ V34  : num  0.7547 0.2341 0.2709 0.3497 0.0981 ...
##  $ V35  : num  0.854 0.131 0.423 0.395 0.195 ...
##  $ V36  : num  0.851 0.418 0.304 0.301 0.418 ...
##  $ V37  : num  0.669 0.384 0.612 0.541 0.46 ...
##  $ V38  : num  0.61 0.106 0.676 0.881 0.322 ...
##  $ V39  : num  0.494 0.184 0.537 0.986 0.283 ...
##  $ V40  : num  0.274 0.197 0.472 0.917 0.243 ...
##  $ V41  : num  0.051 0.167 0.465 0.612 0.198 ...
##  $ V42  : num  0.2834 0.0583 0.2587 0.5006 0.2444 ...
##  $ V43  : num  0.282 0.14 0.213 0.321 0.185 ...
##  $ V44  : num  0.4256 0.1628 0.2222 0.3202 0.0841 ...
##  $ V45  : num  0.2641 0.0621 0.2111 0.4295 0.0692 ...
##  $ V46  : num  0.1386 0.0203 0.0176 0.3654 0.0528 ...
##  $ V47  : num  0.1051 0.053 0.1348 0.2655 0.0357 ...
##  $ V48  : num  0.1343 0.0742 0.0744 0.1576 0.0085 ...
##  $ V49  : num  0.0383 0.0409 0.013 0.0681 0.023 0.0264 0.0507 0.0285 0.077
7 0.0092 ...
##  $ V50  : num  0.0324 0.0061 0.0106 0.0294 0.0046 0.0081 0.0159 0.0178 0.0
```

```
439 0.0198 ...
##  $ V51  : num  0.0232 0.0125 0.0033 0.0241 0.0156 0.0104 0.0195 0.0052 0.0
061 0.0118 ...
##  $ V52  : num  0.0027 0.0084 0.0232 0.0121 0.0031 0.0045 0.0201 0.0081 0.0
145 0.009 ...
##  $ V53  : num  0.0065 0.0089 0.0166 0.0036 0.0054 0.0014 0.0248 0.012 0.01
28 0.0223 ...
##  $ V54  : num  0.0159 0.0048 0.0095 0.015 0.0105 0.0038 0.0131 0.0045 0.01
45 0.0179 ...
##  $ V55  : num  0.0072 0.0094 0.018 0.0085 0.011 0.0013 0.007 0.0121 0.0058
0.0084 ...
##  $ V56  : num  0.0167 0.0191 0.0244 0.0073 0.0015 0.0089 0.0138 0.0097 0.0
049 0.0068 ...
##  $ V57  : num  0.018 0.014 0.0316 0.005 0.0072 0.0057 0.0092 0.0085 0.0065
0.0032 ...
##  $ V58  : num  0.0084 0.0049 0.0164 0.0044 0.0048 0.0027 0.0143 0.0047 0.0
093 0.0035 ...
##  $ V59  : num  0.009 0.0052 0.0095 0.004 0.0107 0.0051 0.0036 0.0048 0.005
9 0.0056 ...
##  $ V60  : num  0.0032 0.0044 0.0078 0.0117 0.0094 0.0062 0.0103 0.0053 0.0
022 0.004 ...
##  $ Class: Factor w/ 2 levels "M","R": 2 2 2 2 2 2 2 2 2 ...
dim(Sonar)
## [1] 208  61
```

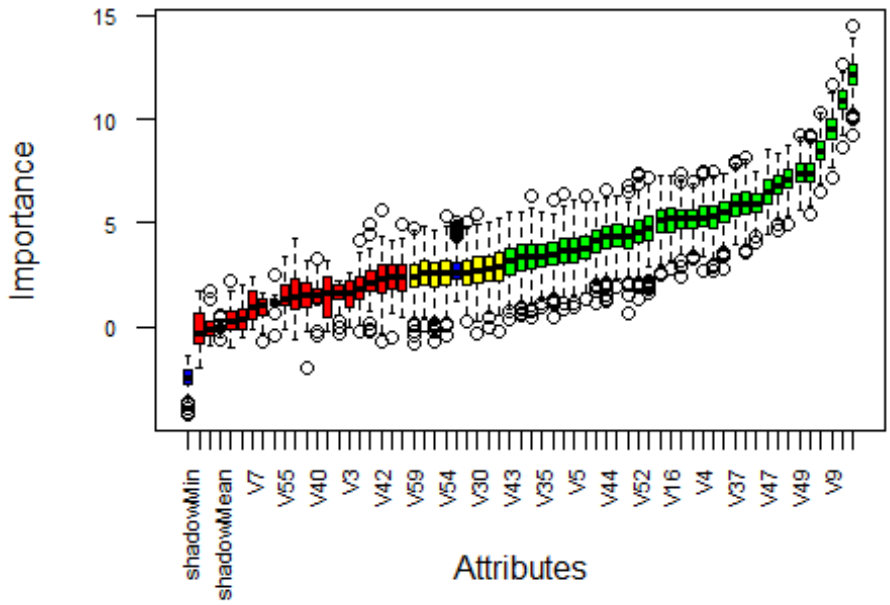
FEATURE SELECTION USING BORUTA

```
set.seed(111)
boruta <- Boruta(Class ~., data = Sonar,doTrace = 2,maxRuns = 500)
##  1. run of importance source...
##  2. run of importance source...
##  3. run of importance source...
##  4. run of importance source...
## After 13 iterations, +3.1 secs:
##  confirmed 18 attributes: V10, V11, V12, V13, V16 and 13 more;
##  rejected 9 attributes: V3, V33, V41, V50, V53 and 4 more;
##  still have 33 attributes left.
##  14. run of importance source...
##  15. run of importance source...
##  16. run of importance source...
##  17. run of importance source...
## After 17 iterations, +3.9 secs:
##  confirmed 1 attribute: V52;
##  rejected 1 attribute: V38;
##  still have 31 attributes left.
##  18. run of importance source...
##  19. run of importance source...
##  20. run of importance source...
##  21. run of importance source...
## After 21 iterations, +4.6 secs:
##  confirmed 4 attributes: V17, V23, V44, V51;
```

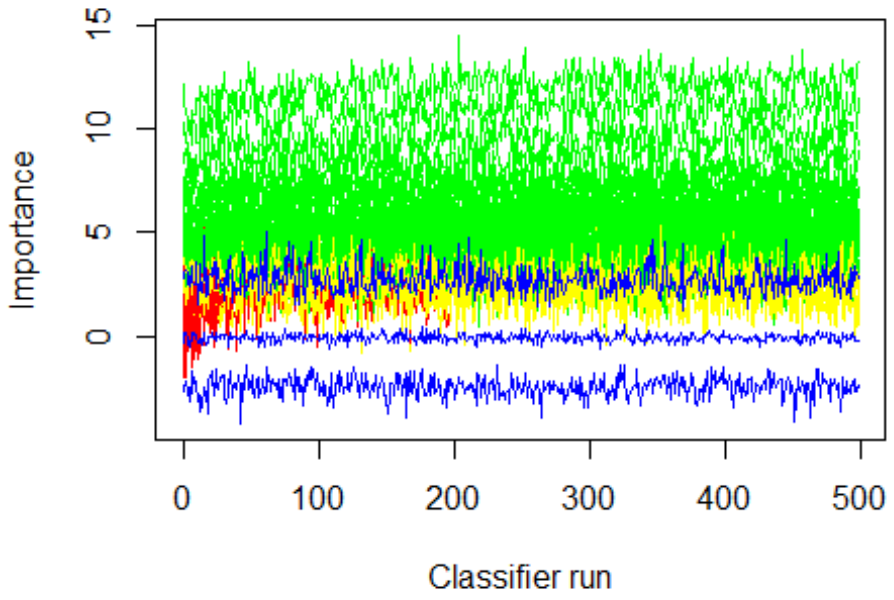
```
print(boruta)
## Boruta performed 499 iterations in 1.447117 mins.
##  33 attributes confirmed important: V1, V10, V11, V12, V13 and 28 more;
##  19 attributes confirmed unimportant: V14, V24, V25, V29, V3 and 14
## more;
##  8 tentative attributes left: V2, V30, V32, V34, V39 and 3 more;
```

PLOT

```
plot(boruta,las=2,cex.axis = 0.7)
```




```
plotImpHistory(boruta)
```



Tentative Fix

```
bor <- TentativeRoughFix(boruta)
print(bor)
## Boruta performed 499 iterations in 1.447117 mins.
## Tentatives roughfixed over the last 499 iterations.
##  35 attributes confirmed important: V1, V10, V11, V12, V13 and 30 more;
##  25 attributes confirmed unimportant: V14, V2, V24, V25, V29 and 20
## more;
attStats(boruta)
##         meanImp  medianImp      minImp    maxImp    normHits  decision
## V1   3.64522003  3.7069599  0.85491112  6.448672 0.789579158 Confirmed
## V2   2.56018821  2.5626953  0.09577678  4.832342 0.466933868 Tentative
## V3   1.53640604  1.6545624 -0.02771654  2.603076 0.000000000  Rejected
## V4   5.30944437  5.2845461  2.71035669  7.509522 0.997995992 Confirmed
## V5   3.69000042  3.7252982  0.92142097  6.127346 0.799599198 Confirmed
## V6   2.18830603  2.1252480 -0.17730273  4.904838 0.042084168  Rejected
## V7   1.03898921  0.9892355 -0.13815483  2.362981 0.000000000  Rejected
## V8   2.66000552  2.6414343  0.22717957  5.007238 0.486973948 Tentative
## V9   9.51173729  9.5357971  7.22261200 11.711460 1.000000000 Confirmed
## V10  8.51515555  8.4883443  6.47416983 10.335991 1.000000000 Confirmed
## V11 12.13764515 12.1775562  9.21761906 14.505315 1.000000000 Confirmed
## V12 10.89425776 10.9111204  8.65645614 12.667007 1.000000000 Confirmed
## V13  5.50798754  5.4933987  2.78879046  7.420437 1.000000000 Confirmed
## V14  2.41023091  2.3999910 -0.50695307  4.212125 0.132264529  Rejected
## V15  4.34261293  4.3930849  0.61592128  6.753799 0.933867735 Confirmed
## V16  5.17396457  5.2035382  2.81557876  7.253278 0.989979960 Confirmed
## V17  5.05871802  5.1037869  2.46848997  7.243639 0.975951904 Confirmed
```

```
## V18  4.36590047  4.3918431  1.93050330  6.293713 0.955911824 Confirmed
## V19  3.34609225  3.3693378  0.52844387  5.570725 0.757515030 Confirmed
## V20  5.19118713  5.2292685  3.25603946  7.035820 0.993987976 Confirmed
## V21  5.96180245  5.9419988  4.04078228  7.483560 1.000000000 Confirmed
## V22  3.50041802  3.5212700  0.41925281  6.156342 0.803607214 Confirmed
## V23  4.10471949  4.1205249  1.12774891  6.026547 0.905811623 Confirmed
## V24  1.86463918  1.8598165 -0.24316160  4.159954 0.008016032  Rejected
## V25  1.57307466  1.4761946 -0.64936962  4.278884 0.016032064  Rejected
## V26  3.36722318  3.3863951  0.49718680  6.290979 0.725450902 Confirmed
## V27  5.19330498  5.2199392  2.45062446  7.395997 0.989979960 Confirmed
## V28  5.92003198  5.9215241  3.55000381  8.177161 0.997995992 Confirmed
## V29  2.36380059  2.4331130  0.41855434  4.924354 0.146292585  Rejected
## V30  2.72433806  2.7302000 -0.31134803  5.463489 0.531062124 Tentative
## V31  3.79173559  3.7805256  1.29042012  6.273788 0.835671343 Confirmed
## V32  2.79951948  2.8016323 -0.03446566  5.008672 0.547094188 Tentative
## V33  1.12225354  1.2055141 -0.41222226  2.554510 0.000000000  Rejected
## V34  2.45599135  2.5641232 -0.75556745  4.685484 0.448897796 Tentative
## V35  3.41122437  3.4095102  0.99417292  5.552401 0.743486974 Confirmed
## V36  7.12145569  7.1137963  4.94001158  8.775913 1.000000000 Confirmed
## V37  5.87362312  5.8958438  4.01343638  8.011247 0.995991984 Confirmed
## V38  1.38115887  1.6142404 -0.28281227  2.207593 0.002004008  Rejected
## V39  2.87672468  2.9165136 -0.23381634  5.236905 0.573146293 Tentative
## V40  1.50602681  1.5564009 -0.40502675  3.333382 0.004008016  Rejected
## V41  0.86652433  1.0532712 -0.75195407  1.592248 0.000000000  Rejected
## V42  2.28692462  2.3312930 -0.71096381  5.584076 0.132264529  Rejected
## V43  3.14236669  3.1836053  0.36356323  5.493385 0.667334669 Confirmed
## V44  4.28166655  4.3592863  1.26698494  6.581545 0.921843687 Confirmed
## V45  6.80681837  6.8227098  4.63217297  8.399992 1.000000000 Confirmed
## V46  5.26449127  5.3204731  2.73211690  7.480674 0.993987976 Confirmed
## V47  6.50308156  6.5252337  4.50034343  8.545674 1.000000000 Confirmed
## V48  7.42278058  7.4090418  5.41780480  9.279586 1.000000000 Confirmed
## V49  7.38630014  7.3813674  5.65293243  9.258094 1.000000000 Confirmed
## V50  1.27572757  1.5538894 -1.97956728  3.161847 0.000000000  Rejected
## V51  4.73282122  4.7433019  1.62275314  7.182236 0.957915832 Confirmed
## V52  4.55828189  4.5996913  1.35163088  7.405190 0.951903808 Confirmed
## V53  0.43347134  0.2650497 -1.01651240  2.167386 0.000000000  Rejected
## V54  2.56394927  2.5872541 -0.43400500  5.286786 0.470941884 Tentative
## V55  1.57808488  1.3809339 -0.08305523  3.380146 0.010020040  Rejected
## V56  0.05343385 -0.1672761 -0.90862287  1.717345 0.000000000  Rejected
## V57 -0.14278800 -0.3478498 -2.00639915  1.707670 0.000000000  Rejected
## V58  1.41303985  1.5999681 -0.20246595  3.222278 0.008016032  Rejected
## V59  2.43538028  2.4571110 -0.76676381  4.762017 0.438877756 Tentative
## V60  0.54378749  0.3639745 -0.51079512  2.059714 0.000000000  Rejected
```

Data Partition

```
set.seed(222)
ind <- sample(2,nrow(Sonar),replace = T,prob = c(0.6,0.4))
train <- Sonar[ind == 1,]
test <- Sonar[ind == 2,]
```

Build RandomForest model on entire dataset

```
set.seed(333)
rf60 <- randomForest(Class ~., data = train)
rf60
##
## Call:
##  randomForest(formula = Class ~ ., data = train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 23.08%
## Confusion matrix:
##    M  R class.error
## M 51 10   0.1639344
## R 17 39   0.3035714
```

Predict Test Data

```
p <- predict(rf60,test)
confusionMatrix(p,test$Class)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  M  R
##          M 46 17
##          R  4 24
##
##                Accuracy : 0.7692
##                  95% CI : (0.6691, 0.8511)
##     No Information Rate : 0.5495
##     P-Value [Acc > NIR] : 1.134e-05
##
##                   Kappa : 0.5202
##
##  Mcnemar's Test P-Value : 0.008829
##
##             Sensitivity : 0.9200
##             Specificity : 0.5854
##          Pos Pred Value : 0.7302
##          Neg Pred Value : 0.8571
```

```
##              Prevalence : 0.5495
##          Detection Rate : 0.5055
##    Detection Prevalence : 0.6923
##       Balanced Accuracy : 0.7527
##
##        'Positive' Class : M
##
```

FROM BORUTA

GET NON REJECTED VARIABLES

```
getNonRejectedFormula(boruta)
## Class ~ V1 + V2 + V4 + V5 + V8 + V9 + V10 + V11 + V12 + V13 +
##     V15 + V16 + V17 + V18 + V19 + V20 + V21 + V22 + V23 + V26 +
##     V27 + V28 + V30 + V31 + V32 + V34 + V35 + V36 + V37 + V39 +
##     V43 + V44 + V45 + V46 + V47 + V48 + V49 + V51 + V52 + V54 +
##     V59
## <environment: 0x000000001585d938>
```

BUILD A MODEL ON THEM

```
rf41 <- randomForest(Class ~ V1 + V2 + V4 + V5 + V8 + V9 + V10 + V11 + V12 +
V13 +
                       V14 + V15 + V16 + V17 + V18 + V19 + V20 + V21 + V22 +
V23 +
                       V26 + V27 + V28 + V29 + V30 + V31 + V32 + V35 + V36 +
V37 +
                       V39 + V43 + V44 + V45 + V46 + V47 + V48 + V49 + V51 +
V52 +
                       V54 + V59,data=train)
```

PREDICTION

```
p <- predict(rf41,test)
confusionMatrix(p,test$Class)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  M  R
##          M 44 16
##          R  6 25
##
##                Accuracy : 0.7582
##                  95% CI : (0.6572, 0.8419)
##     No Information Rate : 0.5495
##     P-Value [Acc > NIR] : 3.058e-05
##
##                   Kappa : 0.5007
##
##  Mcnemar's Test P-Value : 0.05501
##
##             Sensitivity : 0.8800
##             Specificity : 0.6098
##          Pos Pred Value : 0.7333
```

```
##          Neg Pred Value : 0.8065
##              Prevalence : 0.5495
##          Detection Rate : 0.4835
##    Detection Prevalence : 0.6593
##       Balanced Accuracy : 0.7449
##
##        'Positive' Class : M
##
```

GET CONFIRMED VARIABLES and

```
getNonRejectedFormula(boruta)
## Class ~ V1 + V2 + V4 + V5 + V8 + V9 + V10 + V11 + V12 + V13 +
##     V15 + V16 + V17 + V18 + V19 + V20 + V21 + V22 + V23 + V26 +
##     V27 + V28 + V30 + V31 + V32 + V34 + V35 + V36 + V37 + V39 +
##     V43 + V44 + V45 + V46 + V47 + V48 + V49 + V51 + V52 + V54 +
##     V59
## <environment: 0x0000000018763b98>
```

BUILD MODLE

```
rf33 <- randomForest(Class ~ V1 + V4 + V5 + V9 + V10 + V11 + V12 + V13 + V15
+ V16 +
                       V17 + V18 + V19 + V20 + V21 + V22 + V23 + V26 + V27 +
V28 +
                       V31 + V35 + V36 + V37 + V43 + V44 + V45 + V46 + V47 +
V48 +
                       V49 + V51 + V52,data = train)
```

PREDICT

```
p <- predict(rf33,test)
confusionMatrix(p,test$Class)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  M  R
##          M 44 16
##          R  6 25
##
##                Accuracy : 0.7582
##                  95% CI : (0.6572, 0.8419)
##     No Information Rate : 0.5495
##     P-Value [Acc > NIR] : 3.058e-05
##
##                   Kappa : 0.5007
##
##  Mcnemar's Test P-Value : 0.05501
##
##             Sensitivity : 0.8800
##             Specificity : 0.6098
##          Pos Pred Value : 0.7333
##          Neg Pred Value : 0.8065
##              Prevalence : 0.5495
```

```
##          Detection Rate : 0.4835
##    Detection Prevalence : 0.6593
##       Balanced Accuracy : 0.7449
##
##        'Positive' Class : M
##
```

PRINCIPAL COMPONENT ANALYSIS

```
data("iris")
str(iris)
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1
1 1 1 1 ...
summary(iris)
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

Data partition

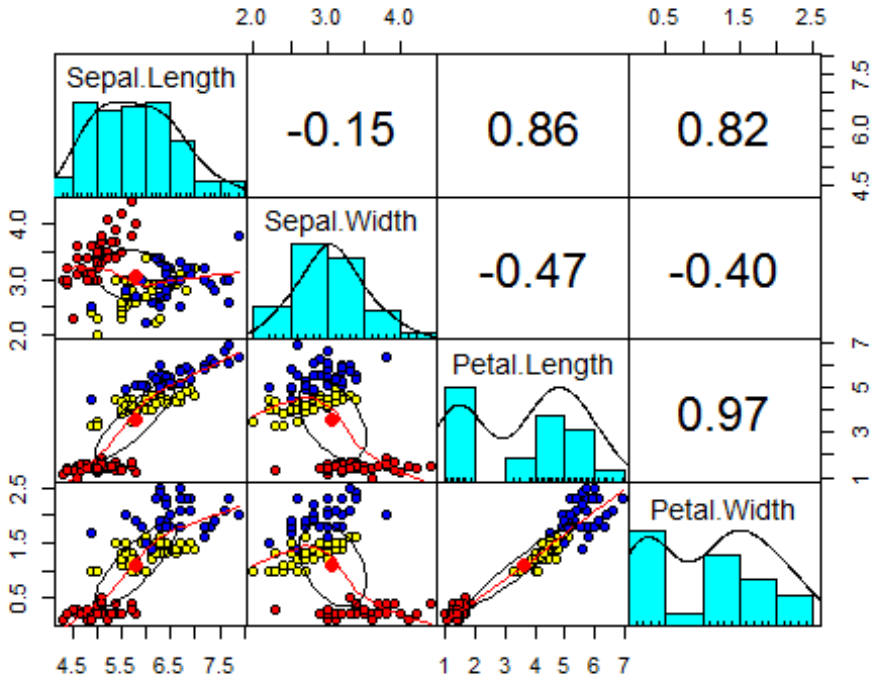
```
set.seed(111)
ind <- sample(2,nrow(iris),replace = T,prob = c(0.8,0.2))
training <- iris[ind == 1,]
testing <- iris[ind ==2,]
```

if independent variables are highly correlated then it creates multicollinearity problem.

so, we need to bring them on to common ground

PCA

```
library(psych)
pairs.panels(training[,-5],gap=0,bg=c('red','yellow','blue')[training$Species
],pch=21)
```

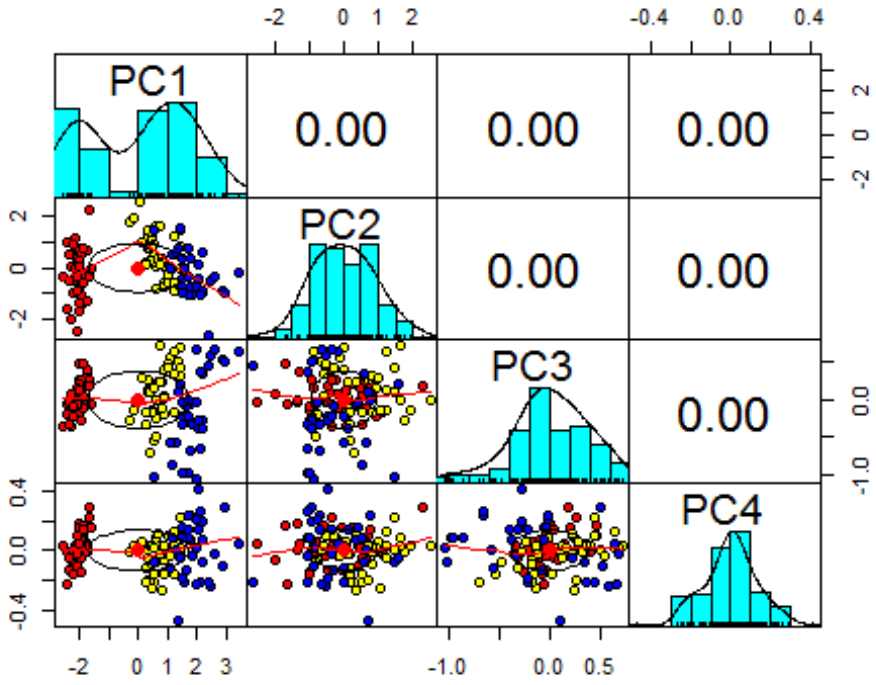



```
pc <- prcomp(training[,-5],center = TRUE,scale. = TRUE)
attributes(pc)
## $names
## [1] "sdev"     "rotation" "center"   "scale"    "x"
##
## $class
## [1] "prcomp"
pc$center  # -- center is mean of each varaibles
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.790000     3.069167     3.597500     1.111667
mean(training$Sepal.Length)
## [1] 5.79
pc$scale  # -- scale is standard deviation for each variables
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##    0.8234382    0.4588615    1.7872782    0.7556158
sd(training$Sepal.Length)
## [1] 0.8234382
print(pc)
## Standard deviations (1, .., p=4):
## [1] 1.7173318 0.9403519 0.3843232 0.1371332
##
## Rotation (n x k) = (4 x 4):
##                      PC1         PC2        PC3        PC4
## Sepal.Length  0.5147163 -0.39817685  0.7242679  0.2279438
## Sepal.Width  -0.2926048 -0.91328503 -0.2557463 -0.1220110
## Petal.Length  0.5772530 -0.02932037 -0.1755427 -0.7969342
## Petal.Width   0.5623421 -0.08065952 -0.6158040  0.5459403
```

```
summary(pc)
## Importance of components:
##                           PC1    PC2     PC3    PC4
## Standard deviation     1.7173 0.9404 0.38432 0.1371
## Proportion of Variance 0.7373 0.2211 0.03693 0.0047
## Cumulative Proportion  0.7373 0.9584 0.99530 1.0000
```

Orthogonality of PCs

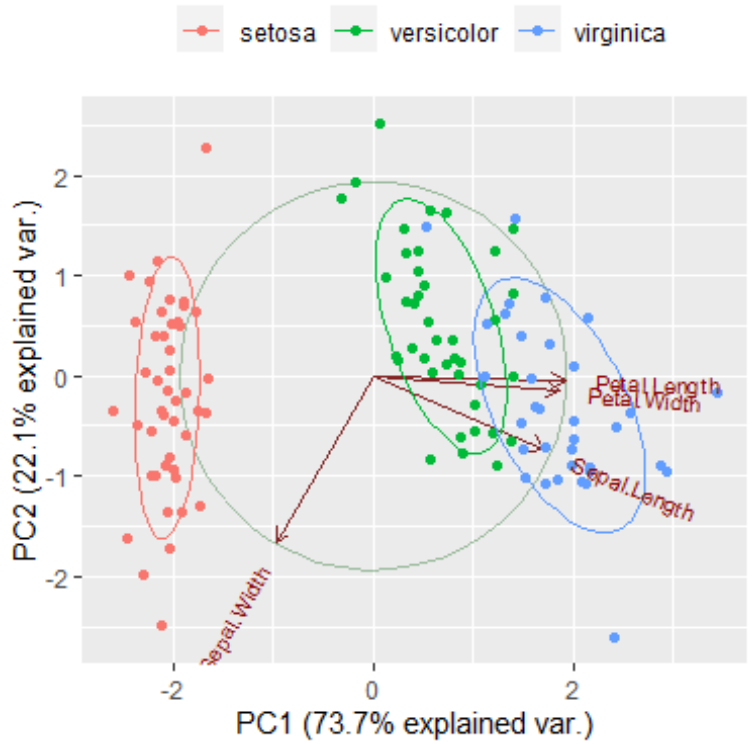
```
pairs.panels(pc$x,gap=0,bg=c('red','yellow','blue')[training$Species],pch=21)
```



Bi-plot

```
library(devtools)
## Loading required package: usethis
library(ggplot2)
#install_github("vqv/ggbiplot")
library(ggbiplot)
## Loading required package: plyr
## ------------------------------------------------------------------------------
----
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, th
en dplyr:
## library(plyr); library(dplyr)
## ------------------------------------------------------------------------------
----
##
## Attaching package: 'plyr'
```

```
## The following object is masked from 'package:faraway':
##
##     ozone
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
## Loading required package: scales
##
## Attaching package: 'scales'
## The following objects are masked from 'package:psych':
##
##     alpha, rescale
g <- ggbiplot(pc,obs.scale = 1,var.scale = 1,groups = training$Species,
              ellipse = TRUE,
              circle = TRUE,
              ellipse.prob = 0.68)
g <- g + scale_color_discrete(name= '')
g <- g + theme(legend.direction = 'horizontal',legend.position = 'top')
print(g)
```



Prediction with PC

```
trg <- predict(pc,training)
trg <- data.frame(trg,training[5])

tst <- predict(pc,testing)
tst <- data.frame(tst,testing$Species)
```

Multinomial Logistic Regression with First Two PC's

```
library(nnet)
trg$Species <- relevel(trg$Species,ref="setosa")
mymodel <- multinom(Species ~PC1 + PC2,data = trg)
## # weights:  12 (6 variable)
## initial  value 131.833475
## iter  10 value 20.607042
## iter  20 value 18.331120
## iter  30 value 18.204474
## iter  40 value 18.199783
## iter  50 value 18.199009
## iter  60 value 18.198506
## final  value 18.198269
## converged
summary(mymodel)
## Call:
## multinom(formula = Species ~ PC1 + PC2, data = trg)
##
## Coefficients:
##            (Intercept)      PC1      PC2
## versicolor   7.2345029 14.05161 3.167254
## virginica   -0.5757544 20.12094 3.625377
##
## Std. Errors:
##            (Intercept)      PC1      PC2
## versicolor    187.5986 106.3766 127.8815
## virginica     187.6093 106.3872 127.8829
##
## Residual Deviance: 36.39654
## AIC: 48.39654
```

Confusion matrix - training

```
p <- predict(mymodel,trg)
tab <- table(p,trg$Species)
tab
##
## p            setosa versicolor virginica
##   setosa         45          0         0
##   versicolor      0         35         3
##   virginica       0          5        32
1 - sum(diag(tab))/sum(tab)
## [1] 0.06666667
```

PROBLEM NO 4: HETEROSCADASTICITY

DETECT HETEROSCADASTICITY

```
library(AER)
## Loading required package: car
## Loading required package: carData
##
## Attaching package: 'car'
## The following objects are masked from 'package:faraway':
##
##     logit, vif
## The following object is masked from 'package:psych':
##
##     logit
## The following object is masked from 'package:dplyr':
##
##     recode
## Loading required package: lmtest
## Loading required package: zoo
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
## Loading required package: sandwich
## Loading required package: survival
##
## Attaching package: 'survival'
## The following objects are masked from 'package:faraway':
##
##     rats, solder
## The following object is masked from 'package:caret':
##
##     cluster
```

Read Data

```
hp_data <- read.csv("houseprices.csv",header = TRUE,sep = ",")
summary(hp_data)
##      price            lotsize          bdrms            age
##  Min.   : 550000   Min.   : 1028   Min.   :1.000   Min.   : 2.00
##  1st Qu.:6424985   1st Qu.: 3067   1st Qu.:2.000   1st Qu.: 7.00
##  Median :7328278   Median : 5667   Median :4.000   Median :12.00
##  Mean   :7107384   Mean   : 6597   Mean   :3.514   Mean   :11.75
##  3rd Qu.:8784329   3rd Qu.: 9888   3rd Qu.:5.000   3rd Qu.:16.25
##  Max.   :9911910   Max.   :14573   Max.   :6.000   Max.   :30.00
##      sqrft
```

```
##  Min.   : 302.0
##  1st Qu.: 609.5
##  Median :1135.0
##  Mean   :1108.0
##  3rd Qu.:1562.0
##  Max.   :1994.0
```

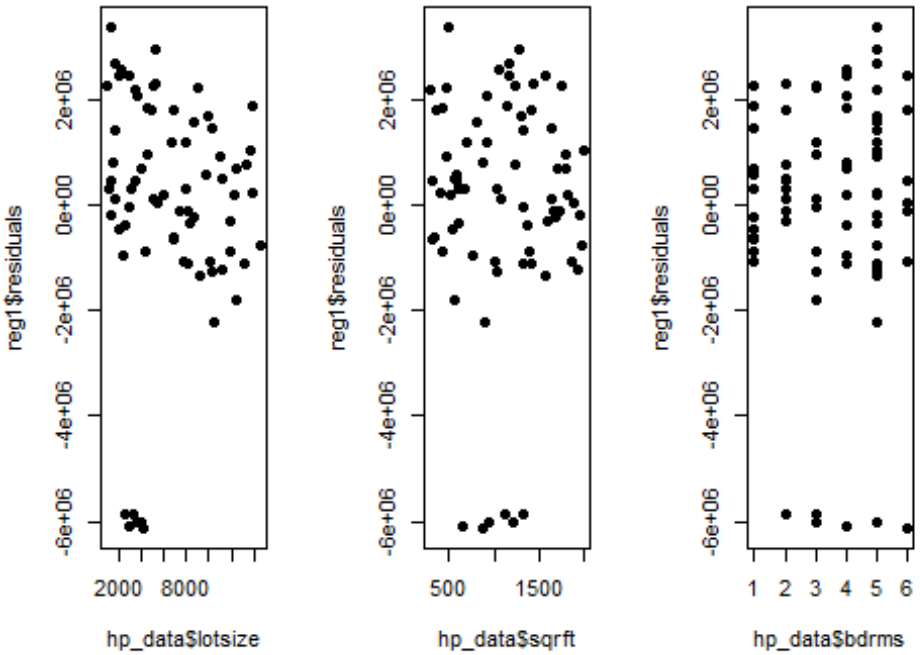
Build Regression Model

```
reg1 <- lm(price~lotsize+sqrft+bdrms,data = hp_data)
reg1_sm <- summary(reg1)
print(reg1_sm)
##
## Call:
## lm(formula = price ~ lotsize + sqrft + bdrms, data = hp_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -6126934  -792477   251333  1488902  3350020
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5992715.05  909530.26   6.589 7.75e-09 ***
## lotsize         157.87      67.42   2.342   0.0221 *
## sqrft          -169.29     532.38  -0.318   0.7515
## bdrms         74211.50  162078.34   0.458   0.6485
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2252000 on 68 degrees of freedom
## Multiple R-squared:  0.07671,    Adjusted R-squared:  0.03598
## F-statistic: 1.883 on 3 and 68 DF,  p-value: 0.1406
```

How woud you investigate whether heteroscedasticity is a problem in this regression.

Residual Plots

```
par(mfrow = c(1,3))
plot(reg1$residuals~hp_data$lotsize,pch=16)
plot(reg1$residuals~hp_data$sqrft,pch=16)
plot(reg1$residuals~hp_data$bdrms,pch=16)
```




```
par(mfrow=c(1,1))
```

1. hpi = alpha + alpha1.Xi+alpha2.X2+alpha3.X3+Ui –> Esitmated by OLS

2. Predicted_Ui^2 = coef0+coef1.X1+coef2.X2+coef3.X2+Vi –> R-square

3. LM=R-Square = 88.R-square

H0 : no heteroscadacity

H1 : coefi not equal to 0, i=1,2 or 3 the heteroscadicity

Reject H0 p-value < alpha =0.01

White test

```
bptest(reg1)
##
##  studentized Breusch-Pagan test
##
## data:  reg1
## BP = 7.2054, df = 3, p-value = 0.06563
bptest(reg1,~lotsize+I(lotsize^2)+sqrft+I(sqrft^2)+bdrms+I(bdrms^2),data = hp
_data)
##
##  studentized Breusch-Pagan test
##
## data:  reg1
## BP = 11.693, df = 6, p-value = 0.06918
```

White standard Errors

```
coeftest(reg1,vcov=vcovHC(reg1,type = 'HC1'))
##
## t test of coefficients:
##
##                Estimate  Std. Error t value  Pr(>|t|)
## (Intercept) 5992715.045  727228.167  8.2405 8.016e-12 ***
## lotsize         157.865      58.766  2.6863  0.009071 **
## sqrft          -169.290     358.602 -0.4721  0.638379
## bdrms         74211.499  144785.783  0.5126  0.609920
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
library(lmtest)
```

Step 1: Install package lmtest

Step 2 : Run a suitable regression model

step 3: bptest(model1)

H0: There is constant variance or homoscedasticity in residual.

(P-value is greater than 0.05: Accept H0) it means there is no Heteroscedasticity problem in this dataset

```
cars
##    speed dist
## 1      4    2
## 2      4   10
## 3      7    4
## 4      7   22
## 5      8   16
## 6      9   10
## 7     10   18
## 8     10   26
## 9     10   34
## 10    11   17
## 11    11   28
## 12    12   14
## 13    12   20
## 14    12   24
## 15    12   28
## 16    13   26
## 17    13   34
## 18    13   34
## 19    13   46
## 20    14   26
## 21    14   36
## 22    14   60
## 23    14   80
## 24    15   20
## 25    15   26
```

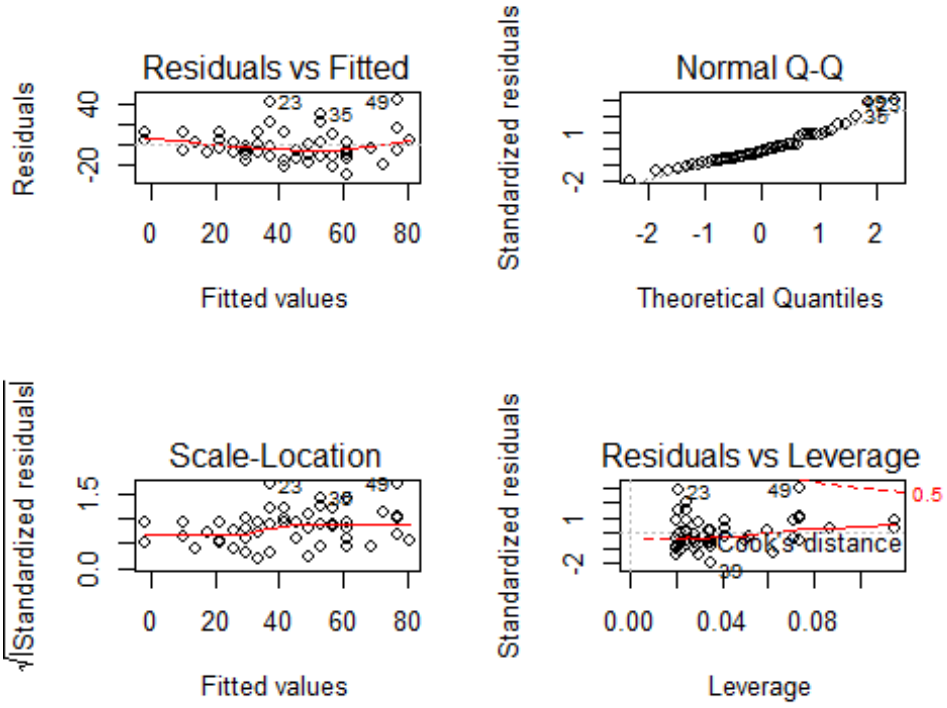
```
## 26    15   54
## 27    16   32
## 28    16   40
## 29    17   32
## 30    17   40
## 31    17   50
## 32    18   42
## 33    18   56
## 34    18   76
## 35    18   84
## 36    19   36
## 37    19   46
## 38    19   68
## 39    20   32
## 40    20   48
## 41    20   52
## 42    20   56
## 43    20   64
## 44    22   66
## 45    23   54
## 46    24   70
## 47    24   92
## 48    24   93
## 49    24  120
## 50    25   85
```

```
lmMod <- lm(dist ~ speed, data=cars) # initial model
summary(lmMod)
```

```
##
## Call:
## lm(formula = dist ~ speed, data = cars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -29.069  -9.525  -2.272   9.215  43.201
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -17.5791     6.7584  -2.601   0.0123 *
## speed         3.9324     0.4155   9.464 1.49e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.38 on 48 degrees of freedom
## Multiple R-squared:  0.6511, Adjusted R-squared:  0.6438
## F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

Graphical method

```
par(mfrow=c(2,2)) # init 4 charts in 1 panel
plot(lmMod)
```



here use observed red line is not straight and residuals seems to increase as the fitted Y values increase.

so the inference here is heteroscidasticity exits.

Statistical test.

Breush Pagan Test

```
lmtest::bptest(lmMod)  # Breusch-Pagan test
##
##  studentized Breusch-Pagan test
##
## data:  lmMod
## BP = 3.2149, df = 1, p-value = 0.07297
```

NCV Test

```
car::ncvTest(lmMod)
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 4.650233, Df = 1, p = 0.031049
```

Both these test have a p-value less that a significance level of 0.05, therefore we can reject the null hypothesis

that the variance of the residuals is constant and infer that heteroscedasticity is indeed present, thereby confirming our graphical inference.

```
ifelse(0.07297 < 0.05,TRUE,FALSE)
```

```
## [1] FALSE
ifelse(0.031049 > 0.05,TRUE,FALSE)
## [1] FALSE
```

HOW TO RECTIFY

Re-build the model with new predictors.

Variable transformation such as Box-Cox transformation.

```
distBCMod <- caret::BoxCoxTrans(cars$dist)
print(distBCMod)
## Box-Cox Transformation
##
## 50 data points used to estimate Lambda
##
## Input data summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00   26.00   36.00   42.98   56.00  120.00
##
## Largest/Smallest: 60
## Sample Skewness: 0.759
##
## Estimated Lambda: 0.5
```

append the transformed variable to cars

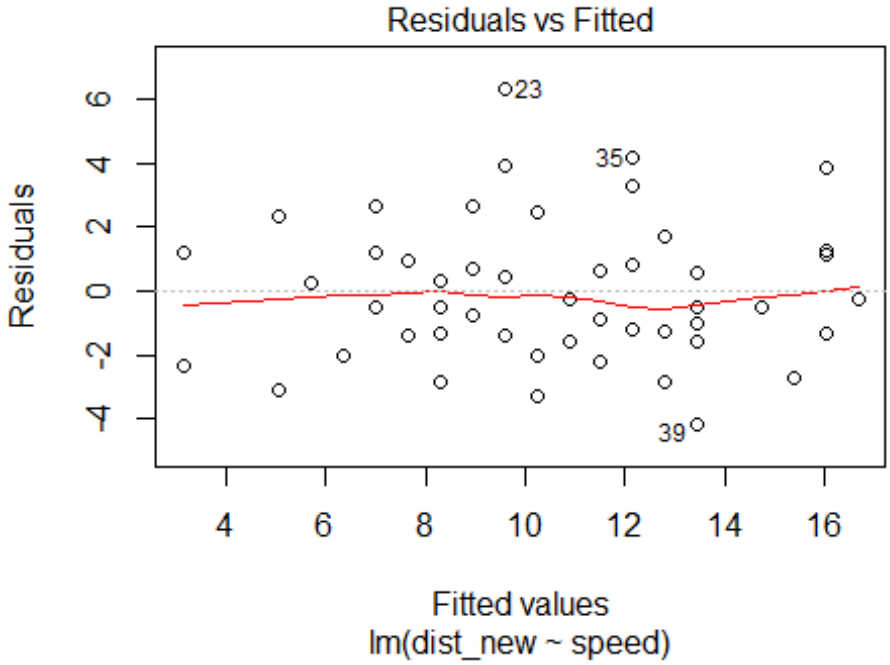
```
cars <- cbind(cars, dist_new=predict(distBCMod, cars$dist))
```

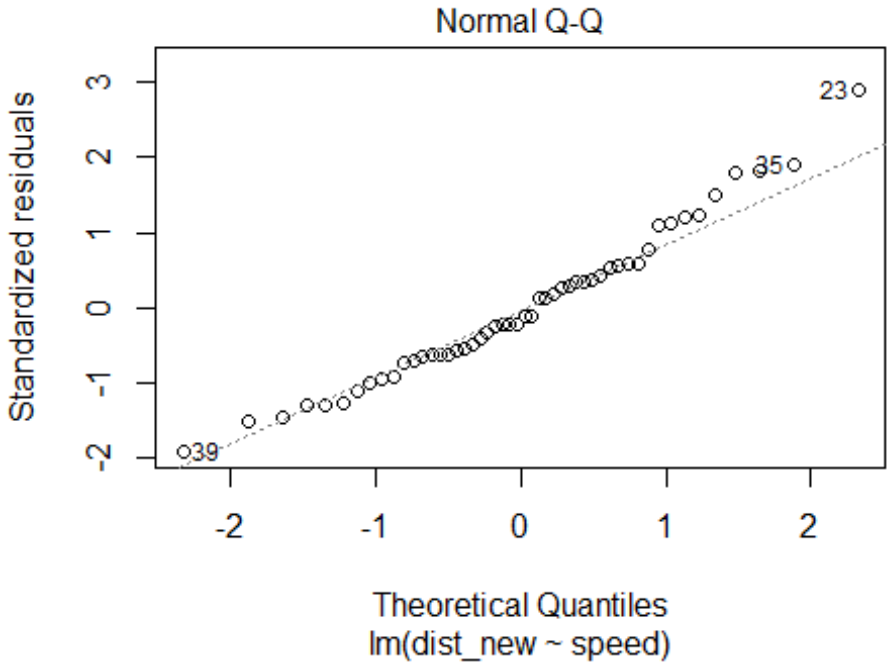
view the top 6 rows

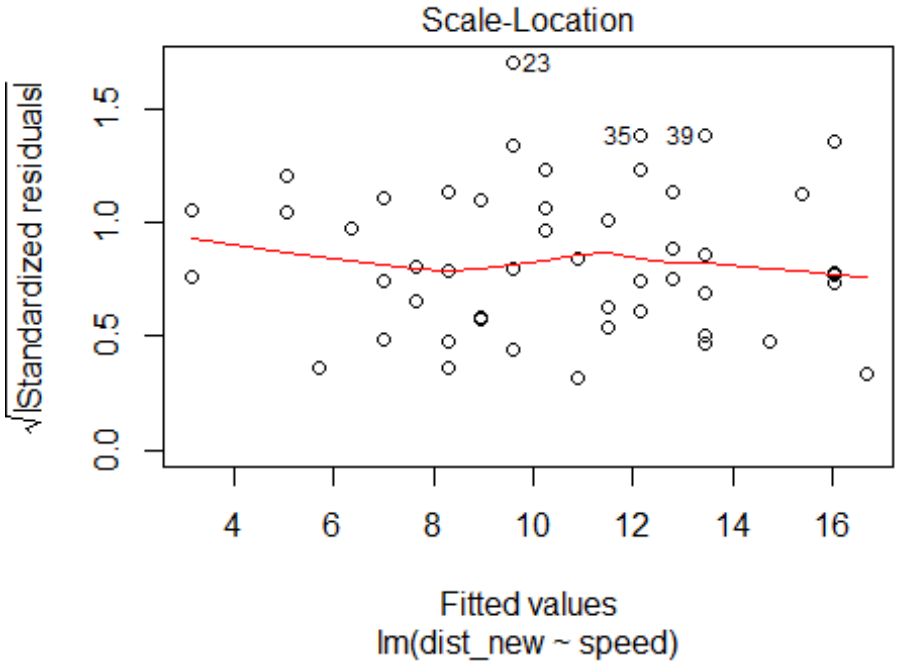
```
head(cars)
##   speed dist  dist_new
## 1     4    2 0.8284271
## 2     4   10 4.3245553
## 3     7    4 2.0000000
## 4     7   22 7.3808315
## 5     8   16 6.0000000
## 6     9   10 4.3245553
```

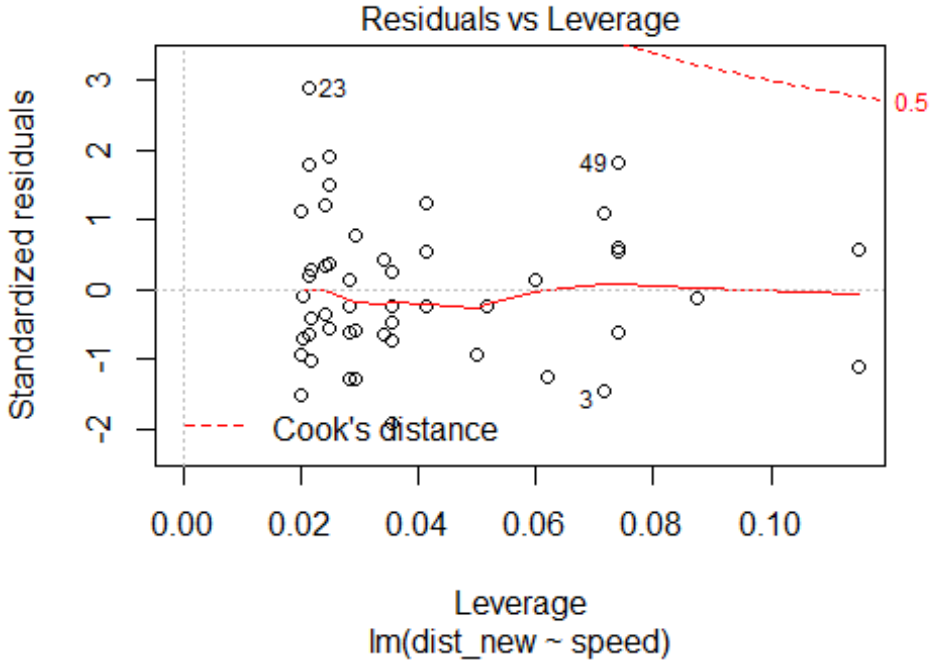
Build a new model

```
lmMod_bc <- lm(dist_new ~ speed, data=cars)
bptest(lmMod_bc)
##
##  studentized Breusch-Pagan test
##
## data:  lmMod_bc
## BP = 0.011192, df = 1, p-value = 0.9157
plot(lmMod_bc)
```


Residuals vs Fitted
Residuals
Fitted values
lm(dist_new ~ speed)
23
35
39
-4
-2
0
2
4
6
4
6
8
10
12
14
16


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(dist_new ~ speed)
23
35
39
-2
-1
0
1
2
3
-2
-1
0
1
2





Breush Pagan Test

```
lmtest::bptest(lmMod_bc)
##
##  studentized Breusch-Pagan test
```

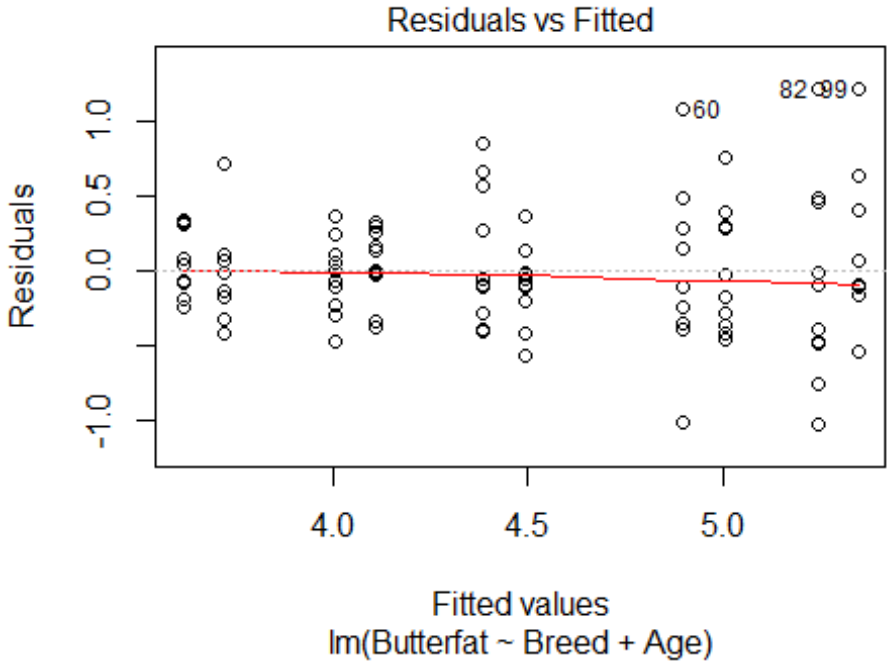
```
##
## data:  lmMod_bc
## BP = 0.011192, df = 1, p-value = 0.9157
```

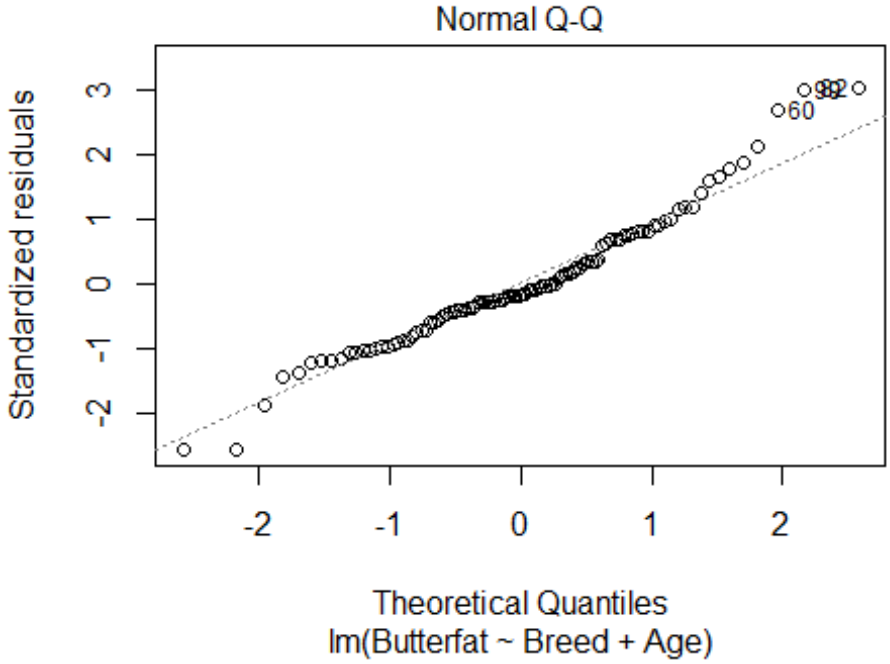
NCV Test

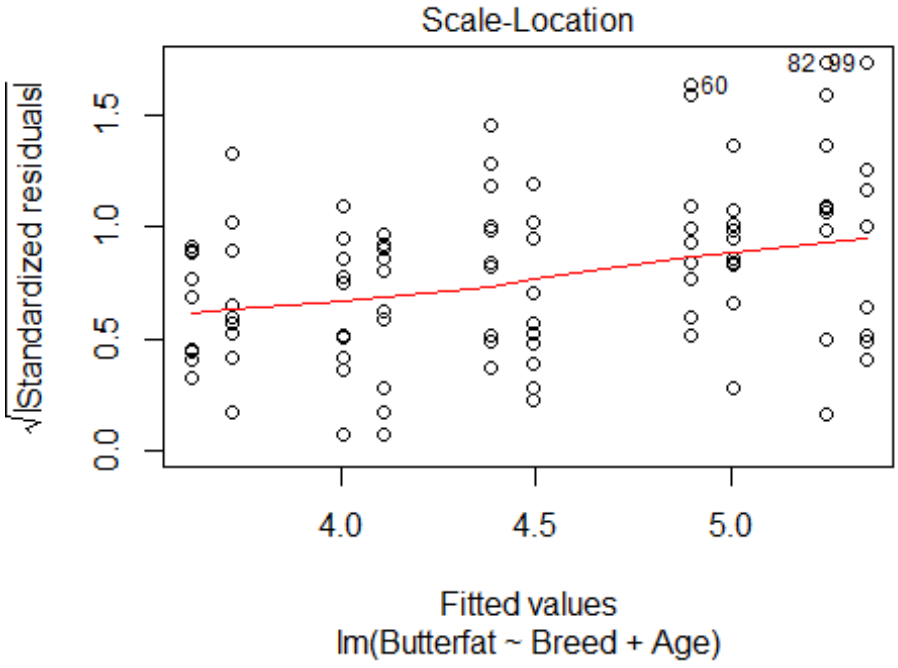
```
car::ncvTest(lmMod_bc)
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 0.01205185, Df = 1, p = 0.91258
```

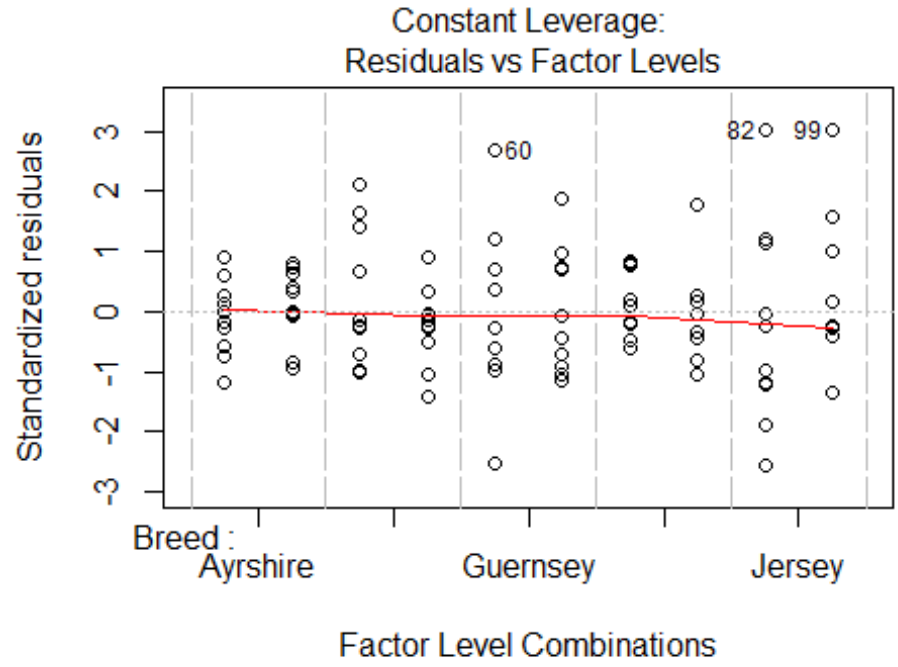
Example 2 :

```
library(faraway)
data("butterfat")
str(butterfat)
## 'data.frame':    100 obs. of  3 variables:
##  $ Butterfat: num  3.74 4.01 3.77 3.78 4.1 4.06 4.27 3.94 4.11 4.25 ...
##  $ Breed    : Factor w/ 5 levels "Ayrshire","Canadian",..: 1 1 1 1 1 1 1 1
1 1 ...
##  $ Age      : Factor w/ 2 levels "2year","Mature": 2 1 2 1 2 1 2 1 2 1 ...
fullmodel <- lm(Butterfat ~ Breed  + Age ,data = butterfat)
plot(fullmodel)
```


Residuals vs Fitted
Residuals
1.0
0.5
0.0
-1.0
82
99
60
4.0
4.5
5.0
Fitted values
lm(Butterfat ~ Breed + Age)

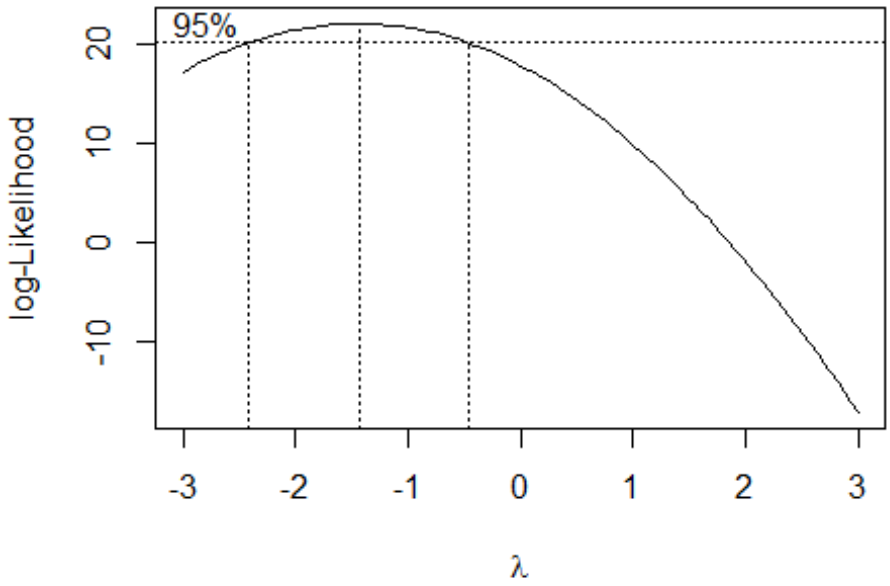

Normal Q-Q
Standardized residuals
3
2
1
0
-1
-2
60
-2
-1
0
1
2
Theoretical Quantiles
lm(Butterfat ~ Breed + Age)






```
bptest(fullmodel)
##
##  studentized Breusch-Pagan test
##
```

```
## data:  fullmodel
## BP = 14.739, df = 5, p-value = 0.01154
ncvTest(fullmodel)
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 16.75812, Df = 1, p = 4.246e-05
library(MASS)
##
## Attaching package: 'MASS'
## The following object is masked from 'package:dplyr':
##
##     select
bc <- boxcox(fullmodel,lambda = seq(-3,3))
```

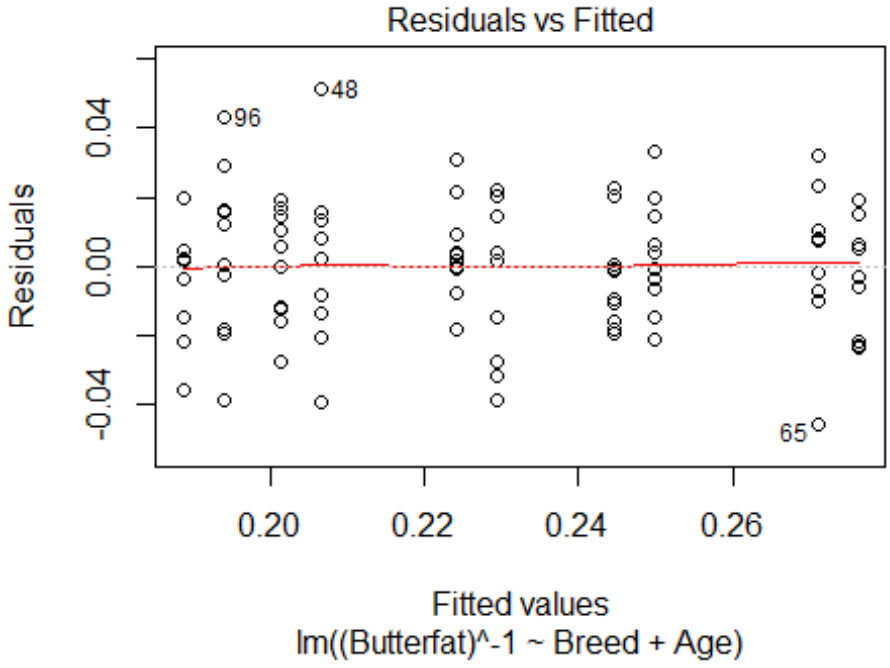


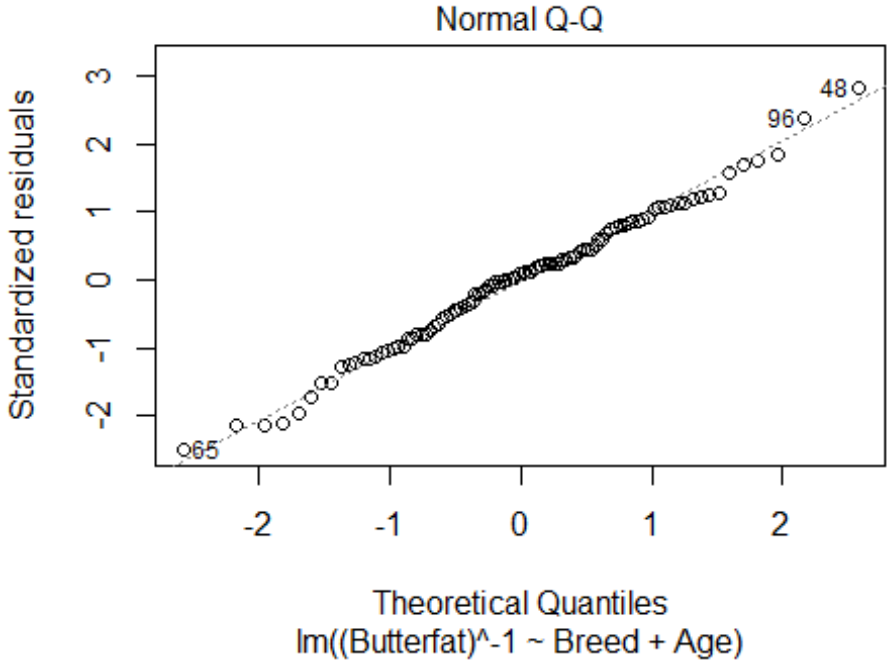
Extract best lambda

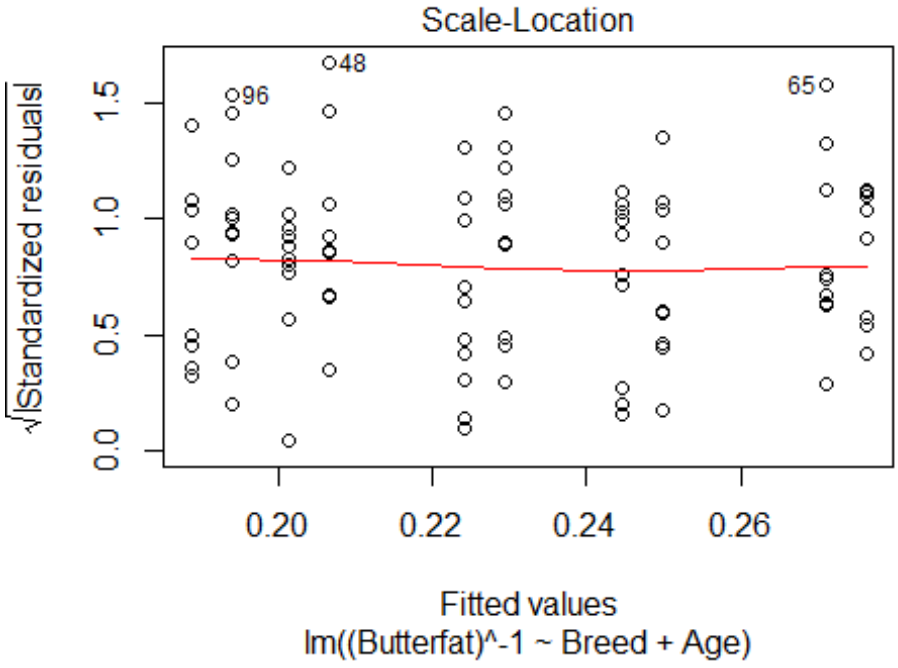
```
best.lam <- bc$x[which(bc$y==max(bc$y))]
```

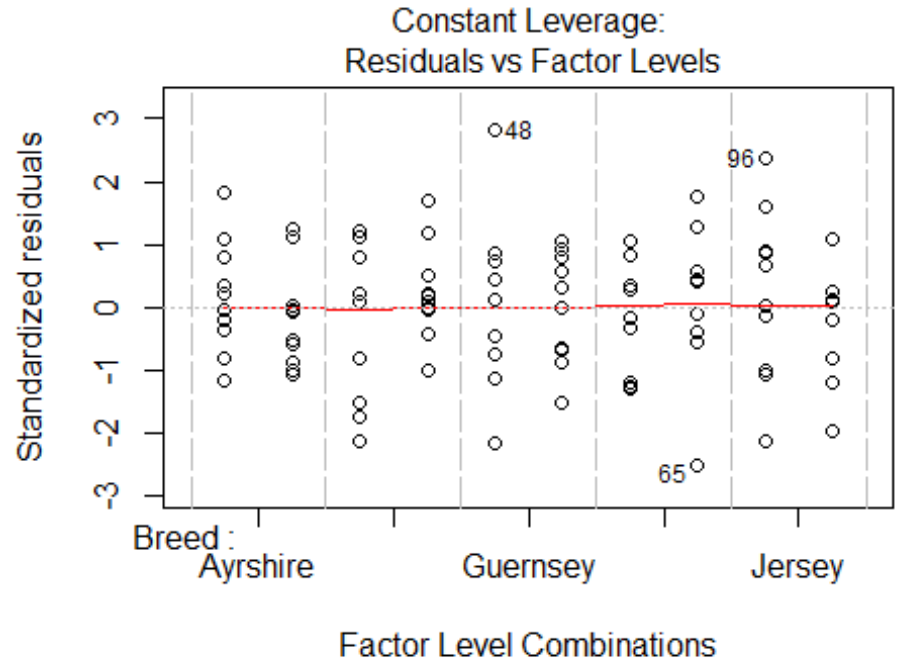
Transform the data using inverse

```
fullmodel.inv <- lm((Butterfat)^-1 ~ Breed + Age,data = butterfat)
plot(fullmodel.inv)
```


Residuals vs Fitted
Residuals
0.04
0.00
-0.04
48
96
65
0.20
0.22
0.24
0.26
Fitted values
lm((Butterfat)^-1 ~ Breed + Age)


Normal Q-Q
Standardized residuals
3
2
1
0
-1
-2
48
96
65
-2
-1
0
1
2
Theoretical Quantiles
lm((Butterfat)^-1 ~ Breed + Age)


Scale-Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
48
96
65
0.20
0.22
0.24
0.26
Fitted values
lm((Butterfat)^-1 ~ Breed + Age)


Constant Leverage:
Residuals vs Factor Levels
Standardized residuals
-3
-2
-1
0
1
2
3
48
96
65
Breed :
Ayrshire
Guernsey
Jersey
Factor Level Combinations

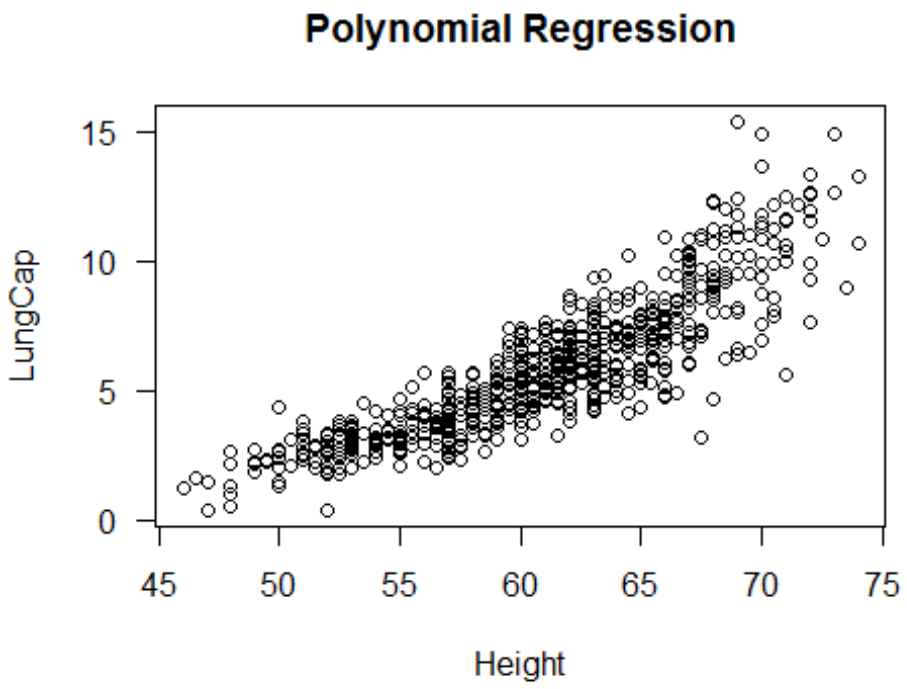
PROBLEM NO 5: Non Linearity

DETECT NON-LINEARITY

```
LungCapData <- read.csv("LungCapData2.csv",header = T,sep = ",")
attach(LungCapData)
summary(LungCapData)
##       Age            LungCap           Height         Gender    Smoke
##  Min.   : 3.000   Min.   : 0.373   Min.   :46.00   female:318   no :589
##  1st Qu.: 8.000   1st Qu.: 3.943   1st Qu.:57.00   male  :336   yes: 65
##  Median :10.000   Median : 5.643   Median :61.50
##  Mean   : 9.931   Mean   : 5.910   Mean   :61.14
##  3rd Qu.:12.000   3rd Qu.: 7.356   3rd Qu.:65.50
##  Max.   :19.000   Max.   :15.379   Max.   :74.00
```

make a plot of LungCap vs. Height

```
plot(Height,LungCap,main = "Polynomial Regression",las=1)
```

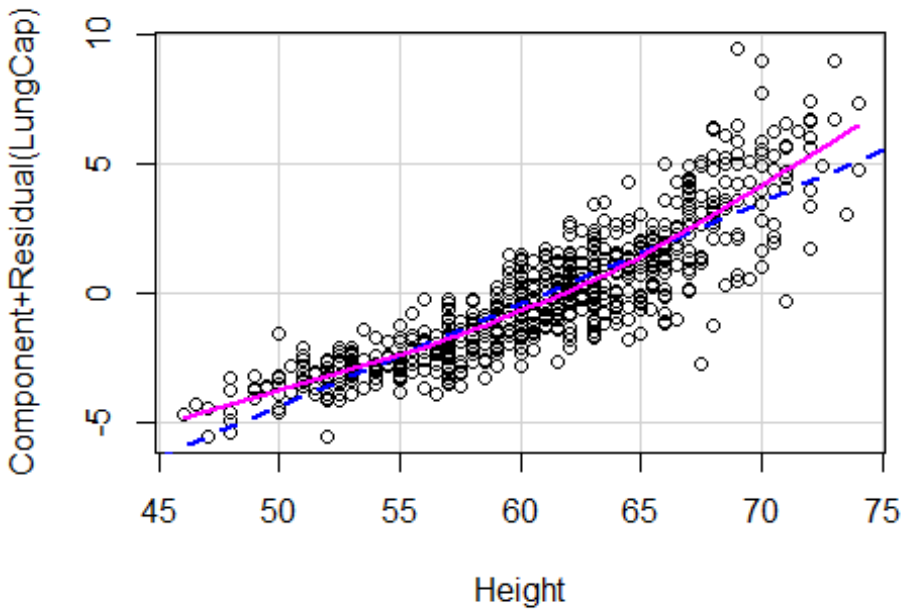


now, let's fit a linear Regression

```
model1 <- lm(LungCap ~ Height)
```

DETECT THE NON LINEARITY

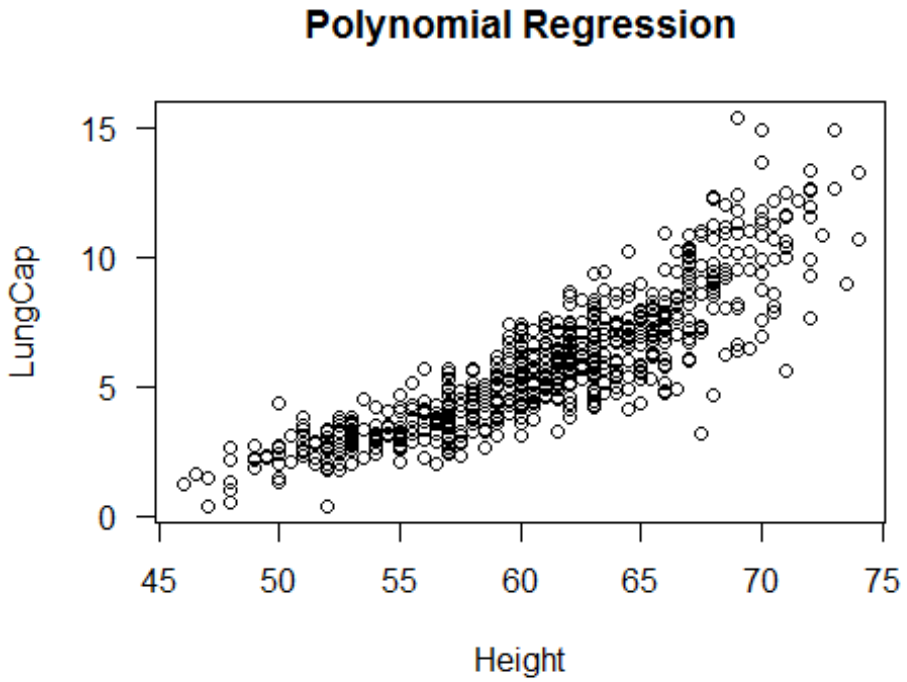
```
library(corrplot)
crPlots(model1)
```



HANDLING NON-LINEARITY PROBLEM

and add the line to the plot… make it thick and red…

```
plot(Height,LungCap,main = "Polynomial Regression",las=1)
```



```r
model1 <- lm(LungCap ~ Height)
# abline(model1,lwd=3,col="red")
```

We can also use Residual plots to help with assessing linearity and checking other model assumptions

first, the WRONG WAY ….

```r
model2 <- lm(LungCap~Height+Height^2)
summary(model2)
```

```
##
## Call:
## lm(formula = LungCap ~ Height + Height^2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.2550 -0.7986 -0.0120  0.7342  6.3581
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -18.298036   0.544380  -33.61   <2e-16 ***
## Height        0.395927   0.008865   44.66   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.292 on 652 degrees of freedom
## Multiple R-squared:  0.7537, Adjusted R-squared:  0.7533
## F-statistic:  1995 on 1 and 652 DF,  p-value: < 2.2e-16
```

first, the RIGTH WAY

```r
model3 <- lm(LungCap~Height+I(Height^2))
summary(model3)
```

```
##
## Call:
## lm(formula = LungCap ~ Height + I(Height^2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4031 -0.6878 -0.0076  0.6577  5.9910
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 16.080634   4.509553   3.566 0.000389 ***
## Height      -0.750147   0.149566  -5.015 6.83e-07 ***
## I(Height^2)  0.009466   0.001233   7.675 6.07e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.238 on 651 degrees of freedom
```

```
## Multiple R-squared:  0.7741, Adjusted R-squared:  0.7734
## F-statistic:  1115 on 2 and 651 DF,  p-value: < 2.2e-16
```

Alternative ways to include Height:

1: Create a new variable called Height^2 and include this variable into the model

Or, Create Height^2, and then include this in model…it's the same!

```
HeightSquare <- Height^2
model3again <- lm(LungCap~Height+HeightSquare)
summary(model3again)
##
## Call:
## lm(formula = LungCap ~ Height + HeightSquare)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4031 -0.6878 -0.0076  0.6577  5.9910
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  16.080634   4.509553   3.566 0.000389 ***
## Height       -0.750147   0.149566  -5.015 6.83e-07 ***
## HeightSquare  0.009466   0.001233   7.675 6.07e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.238 on 651 degrees of freedom
## Multiple R-squared:  0.7741, Adjusted R-squared:  0.7734
## F-statistic:  1115 on 2 and 651 DF,  p-value: < 2.2e-16
```

Or, use the "poly" command…it's the same!

```
model3againagain <- lm(LungCap~poly(Height,degree = 2,raw = T))
summary(model3againagain)
##
## Call:
## lm(formula = LungCap ~ poly(Height, degree = 2, raw = T))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4031 -0.6878 -0.0076  0.6577  5.9910
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     16.080634   4.509553   3.566 0.000389 *
**
## poly(Height, degree = 2, raw = T)1 -0.750147   0.149566  -5.015 6.83e-07 *
**
## poly(Height, degree = 2, raw = T)2  0.009466   0.001233   7.675 6.07e-14 *
**
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.238 on 651 degrees of freedom
## Multiple R-squared:  0.7741, Adjusted R-squared:  0.7734
## F-statistic:  1115 on 2 and 651 DF,  p-value: < 2.2e-16
```

now,let's add this to the plot, using a thick blue line

```
#lines(smooth.spline(Height,predict(model3)),col='blue',lwd=3)
```

test if the model including Height^2 i signif. better than one without using the partial F-Test

```
anova(model2,model3)
## Analysis of Variance Table
##
## Model 1: LungCap ~ Height + Height^2
## Model 2: LungCap ~ Height + I(Height^2)
##   Res.Df     RSS Df Sum of Sq      F    Pr(>F)
## 1    652 1088.41
## 2    651  998.09  1    90.314 58.907 6.069e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

from the F-test we can see such a small p-value we will reject the null hypothesis and conclude that we have evidence to believe the model including height squared provides a Statistically significantly better fit than the model without Height squared.

try fitting a model that includes Height^3 as well

```
model4 <- lm(LungCap ~Height+I(Height^2)+I(Height^3))
summary(model4)
##
## Call:
## lm(formula = LungCap ~ Height + I(Height^2) + I(Height^3))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.3885 -0.6900  0.0069  0.6511  5.9936
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.293e-01  3.803e+01  -0.017    0.987
## Height       9.179e-02  1.908e+00   0.048    0.962
## I(Height^2) -4.567e-03  3.173e-02  -0.144    0.886
## I(Height^3)  7.739e-05  1.749e-04   0.443    0.658
##
## Residual standard error: 1.239 on 650 degrees of freedom
## Multiple R-squared:  0.7742, Adjusted R-squared:  0.7731
## F-statistic: 742.7 on 3 and 650 DF,  p-value: < 2.2e-16
```

Now,let's add this model to the plot, using a thick dashed green line

```
#lines(smooth.spline(Height,predict(model4)),col="green",lwd=3,lty=3)
```

and, let's add a lenged to clarify the lines

```
#legend(46,15,legend = c("model1:linear","model3:poly x^2","model:poly"),
#       col=c("red","blue","green"),lty=c(1,1,3),lwd=3,bty="n",cex=0.9)
```

PROBLEM NO 6: AUTO- CORRELATION

What is AUTO CORRELATION?

Autocorrelation is a type of serial dependence. Specifically, autocorrelation is when a time series is linearly related to a lagged version of itself. By contrast, correlation is simply when two independent variables
are linearly related.

Detect Auto Correlation

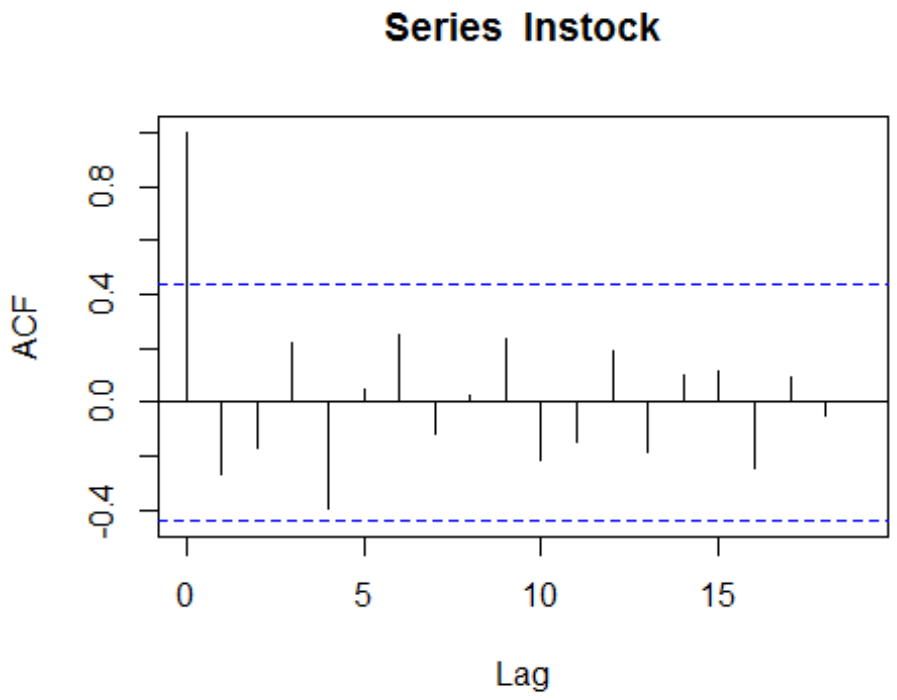
```
library(MASS)
library(tseries)
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
library(forecast)
data <- read.csv("autocor.csv",header = T,sep = ",")
```

Plot and Convert to ln format

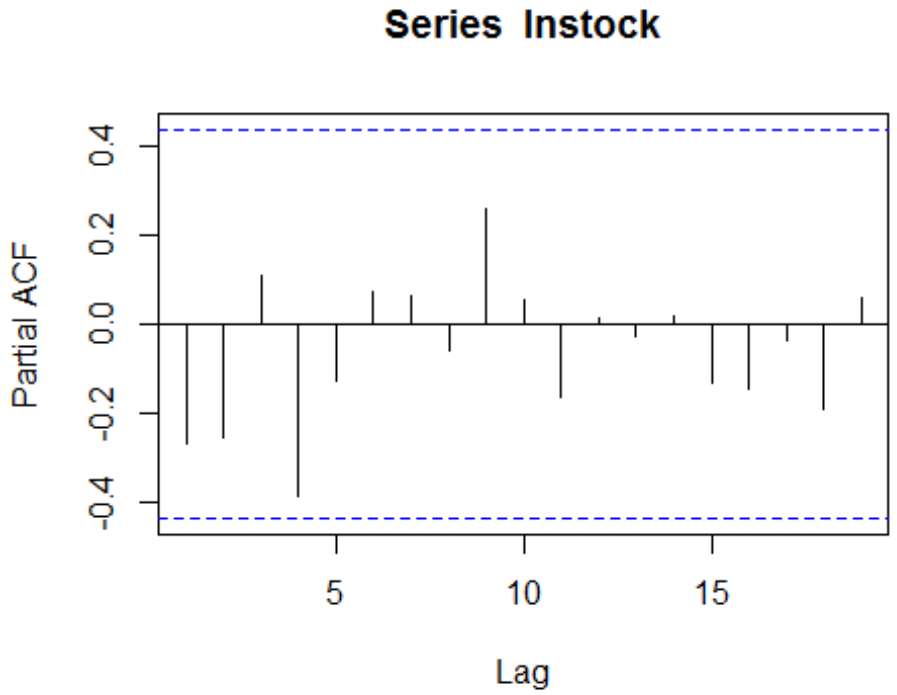
```
lnstock <- log(data$Y[1:20])
lnstock
##  [1] 6.013715 5.666427 6.232448 5.869297 6.001415 6.098074 5.736572 5.9188
94
##  [9] 6.086775 6.025866 5.940171 6.393591 6.035481 5.883322 6.068426 5.5294
29
## [17] 6.100319 6.624065 5.820083 6.169611
```

ACF, PACF and Dickey-Fuller Test

```
acf(lnstock,lag.max = 20)
```




```
pacf(lnstock,lag.max = 20)
```

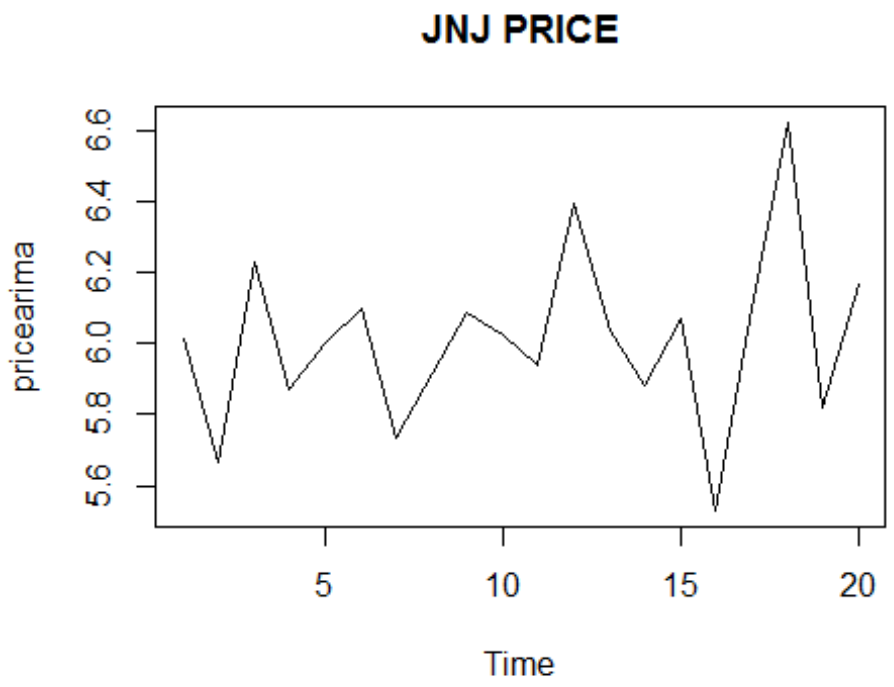



```
difflnstock <- diff(lnstock,1)
difflnstock
##  [1] -0.34728847  0.56602133 -0.36315110  0.13211796  0.09665940 -0.361501
98
```

```
##  [7]  0.18232156  0.16788087 -0.06090875 -0.08569472  0.45341950 -0.358109
32
## [13] -0.15215904  0.18510320 -0.53899650  0.57088986  0.52374628 -0.803982
30
## [19]  0.34952780
adf.test(lnstock)
##
##  Augmented Dickey-Fuller Test
##
## data:  lnstock
## Dickey-Fuller = -2.0513, Lag order = 2, p-value = 0.5528
## alternative hypothesis: stationary
adf.test(difflnstock)
##
##  Augmented Dickey-Fuller Test
##
## data:  difflnstock
## Dickey-Fuller = -2.5652, Lag order = 2, p-value = 0.3571
## alternative hypothesis: stationary
```

Time Series and auto.arima

```
pricearima <- ts(lnstock)
fitlnstock <- auto.arima(pricearima)
fitlnstock
## Series: pricearima
## ARIMA(0,0,0) with non-zero mean
##
## Coefficients:
##         mean
##       6.0107
## s.e.  0.0533
##
## sigma^2 estimated as 0.0597:  log likelihood=0.32
## AIC=3.36   AICc=4.07   BIC=5.35
plot(pricearima,type='l',main='JNJ PRICE')
```

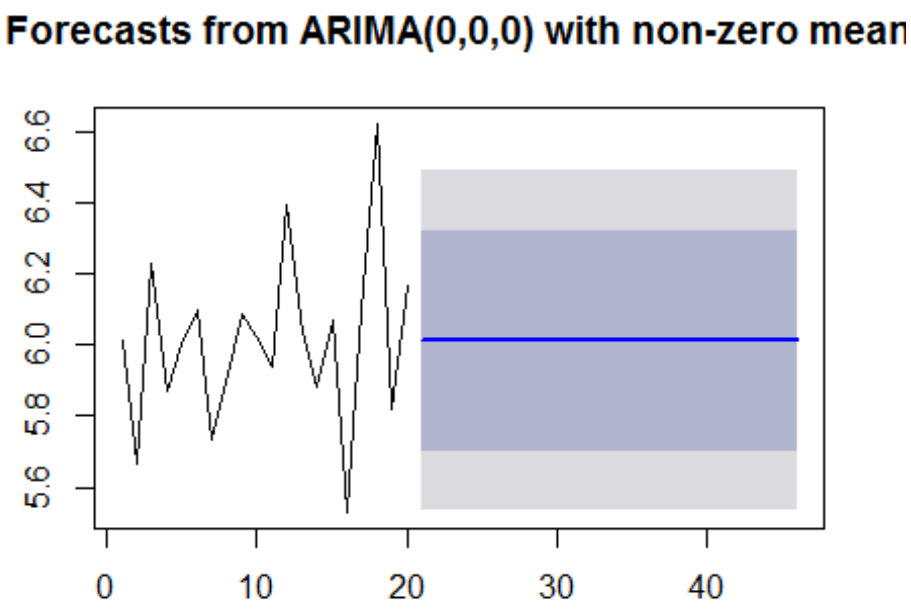



```
exp(lnstock)
##  [1] 409 289 509 354 404 445 310 372 440 414 380 598 418 359 432 252 446 7
53 337
## [20] 478
```

Forecaseted values from ARIMA

```
forecastedvalues_ln <- forecast(fitlnstock,h=26)
forecastedvalues_ln
##    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 21       6.010699 5.697581 6.323817 5.531826 6.489572
## 22       6.010699 5.697581 6.323817 5.531826 6.489572
## 23       6.010699 5.697581 6.323817 5.531826 6.489572
## 24       6.010699 5.697581 6.323817 5.531826 6.489572
## 25       6.010699 5.697581 6.323817 5.531826 6.489572
## 26       6.010699 5.697581 6.323817 5.531826 6.489572
## 27       6.010699 5.697581 6.323817 5.531826 6.489572
## 28       6.010699 5.697581 6.323817 5.531826 6.489572
## 29       6.010699 5.697581 6.323817 5.531826 6.489572
## 30       6.010699 5.697581 6.323817 5.531826 6.489572
## 31       6.010699 5.697581 6.323817 5.531826 6.489572
## 32       6.010699 5.697581 6.323817 5.531826 6.489572
## 33       6.010699 5.697581 6.323817 5.531826 6.489572
## 34       6.010699 5.697581 6.323817 5.531826 6.489572
## 35       6.010699 5.697581 6.323817 5.531826 6.489572
## 36       6.010699 5.697581 6.323817 5.531826 6.489572
## 37       6.010699 5.697581 6.323817 5.531826 6.489572
## 38       6.010699 5.697581 6.323817 5.531826 6.489572
## 39       6.010699 5.697581 6.323817 5.531826 6.489572
```

```
## 40       6.010699 5.697581 6.323817 5.531826 6.489572
## 41       6.010699 5.697581 6.323817 5.531826 6.489572
## 42       6.010699 5.697581 6.323817 5.531826 6.489572
## 43       6.010699 5.697581 6.323817 5.531826 6.489572
## 44       6.010699 5.697581 6.323817 5.531826 6.489572
## 45       6.010699 5.697581 6.323817 5.531826 6.489572
## 46       6.010699 5.697581 6.323817 5.531826 6.489572
plot(forecastedvalues_ln)
```




```
forecastedvaluesextracted <- as.numeric(forecastedvalues_ln$mean)
finalforecastedvalues <- exp(forecastedvaluesextracted)
finalforecastedvalues
##  [1] 407.7683 407.7683 407.7683 407.7683 407.7683 407.7683 407.7683 407.76
83
##  [9] 407.7683 407.7683 407.7683 407.7683 407.7683 407.7683 407.7683 407.76
83
## [17] 407.7683 407.7683 407.7683 407.7683 407.7683 407.7683 407.7683 407.76
83
## [25] 407.7683 407.7683
```

Percentage Error

```
# df <- data.frame(data$Y[5:10],finalforecastedvalues)
# col_heding <- c('Actual Price', "forecasted Price")
# names(df) <- col_heading
# attach(df)
# percentage_error = ((df$'Acutal Price' - df$'Forecased Price')/(df$'Acutal
Price'))
# Percentage_error
# mean(percentage_error)
```