Demonstrate-the-use-ridge-regression-to-predict-the-mileage-of-the-car-using-mtcars-dataset.R

Harini G

```
#Name: Harini G

#1.Load mtcars dataset
data(mtcars)
#?mtcars

#2.install ridge and glmnet packages
#install.packages("glmnet")
#install.packages("ridge")

#3.Perform the exploratory data analysis
nrow(mtcars)
```

```
## [1] 32
```

```
ncol(mtcars)
```

```
## [1] 11
```

```
#inference:the dataset contains 32 rows and 11 coulmns
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

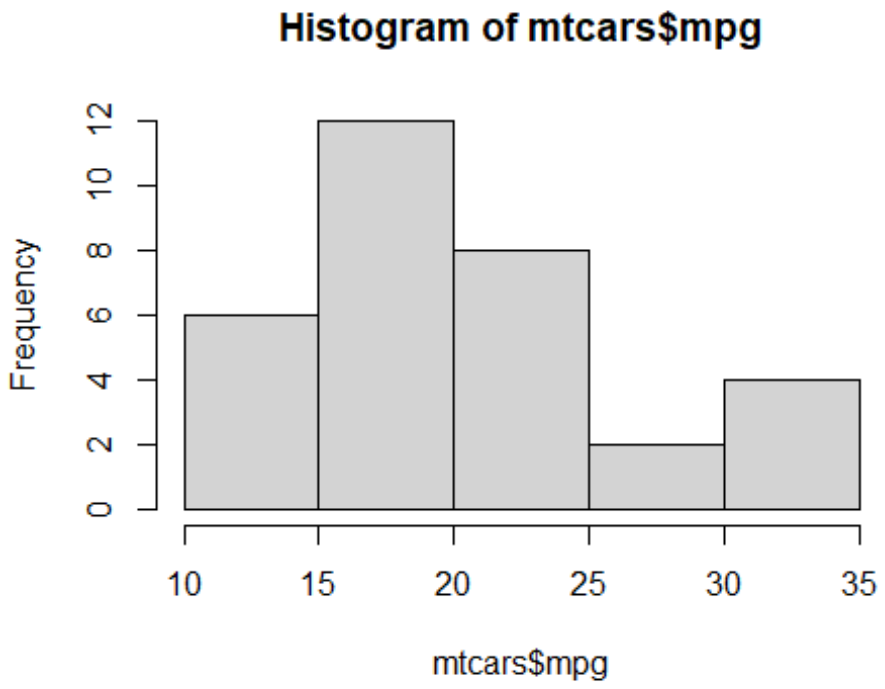
```
tail(mtcars)
```

```
##                 mpg cyl  disp  hp drat    wt qsec vs am gear carb
## Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.7  0  1    5    2
## Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
## Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.5  0  1    5    4
## Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.5  0  1    5    6
## Maserati Bora  15.0   8 301.0 335 3.54 3.570 14.6  0  1    5    8
## Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.6  1  1    4    2
```

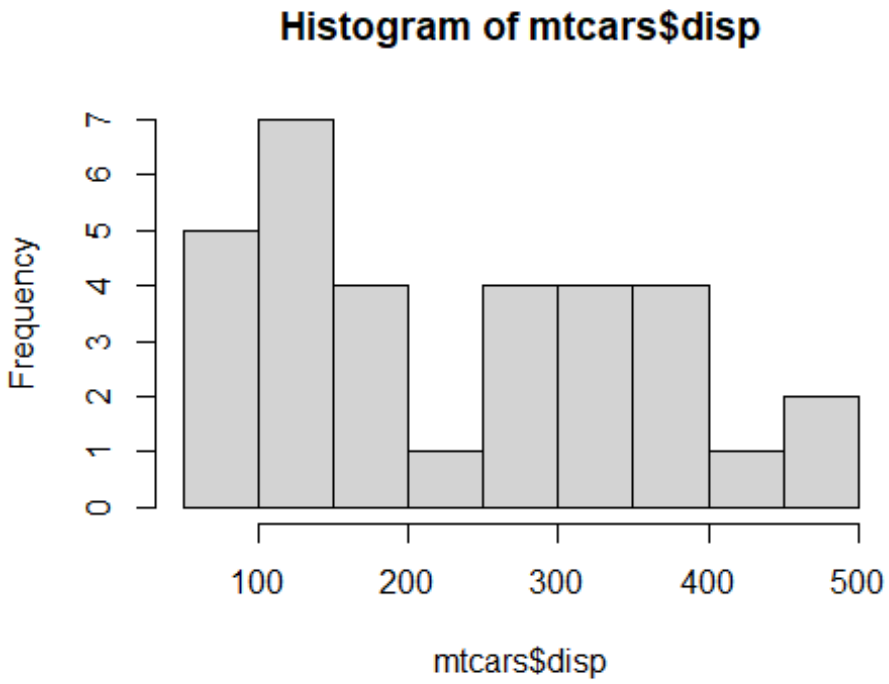
```
summary(mtcars)
```

```
##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
##       drat             wt             qsec             vs        
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
##        am              gear            carb      
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
##  Median :0.0000   Median :4.000   Median :2.000  
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812  
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

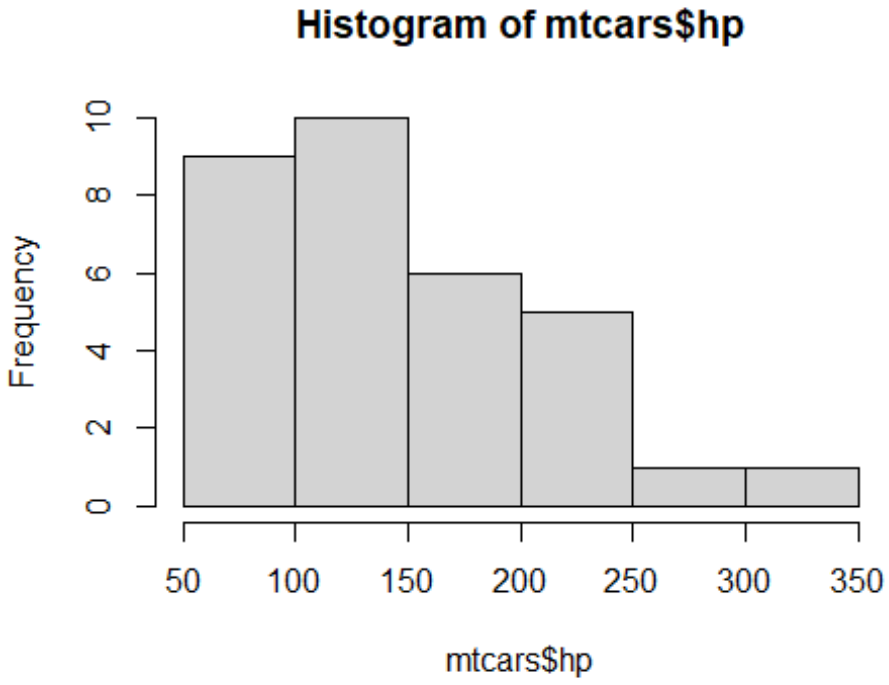
```
#Historgram
hist(mtcars$mpg)
```



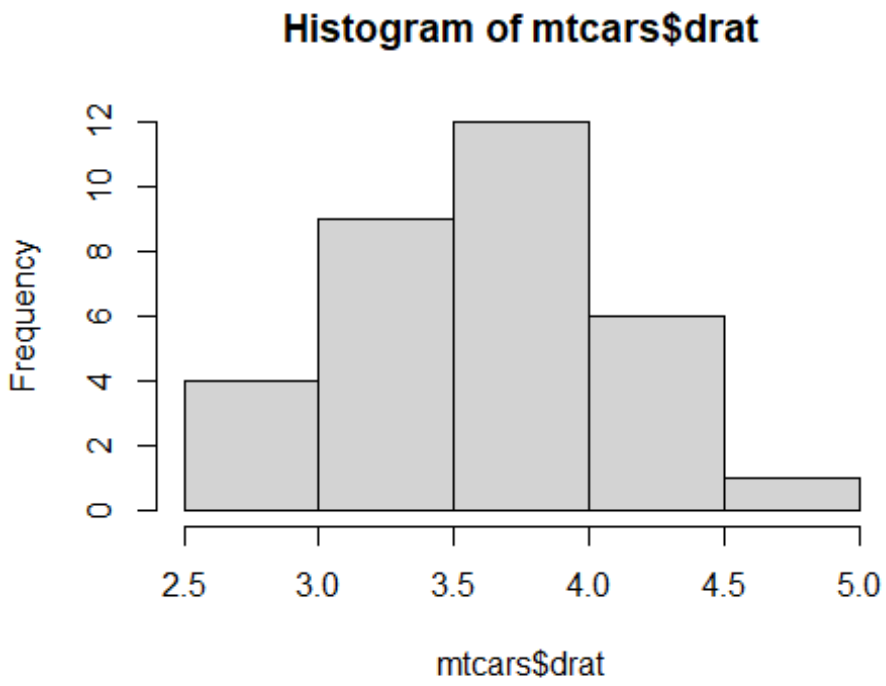

```
#inference: miles/ gallon is left skewed
hist(mtcars$disp)
```



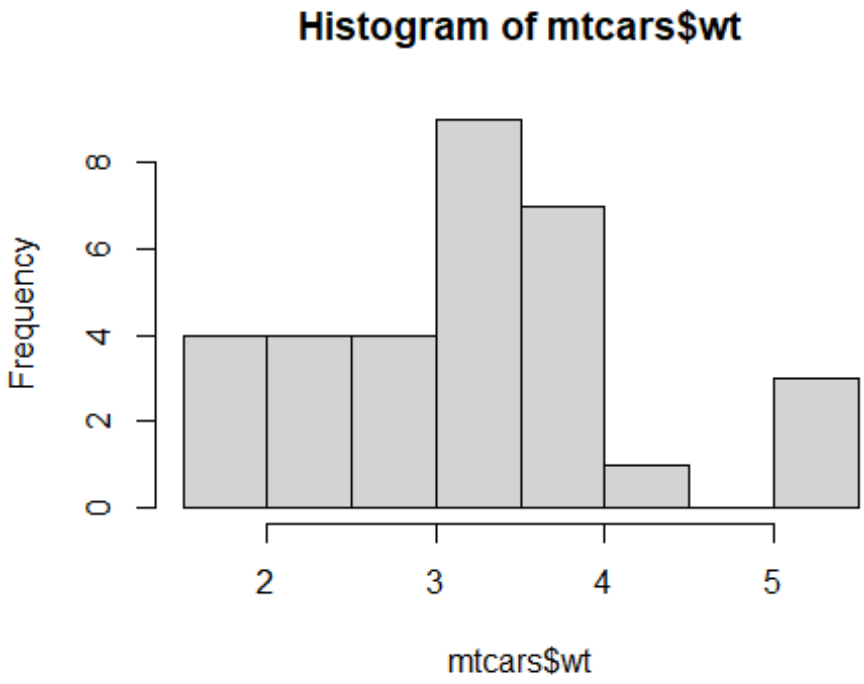

```
#inference: the displacement is having Multi-Modal Distribution
hist(mtcars$hp)
```



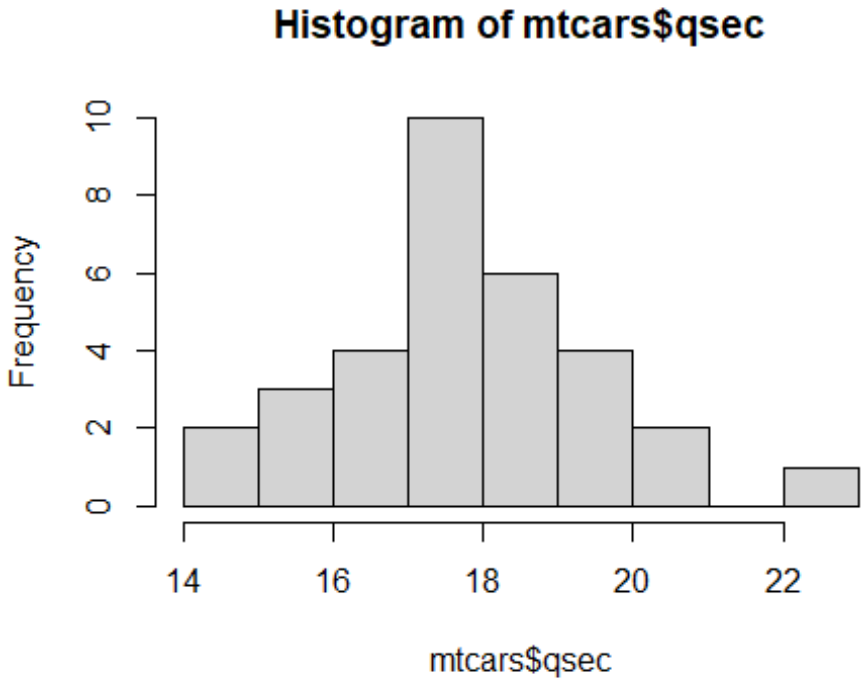
```r
#inference: gross horsepower is left skewed
hist(mtcars$drat)
```



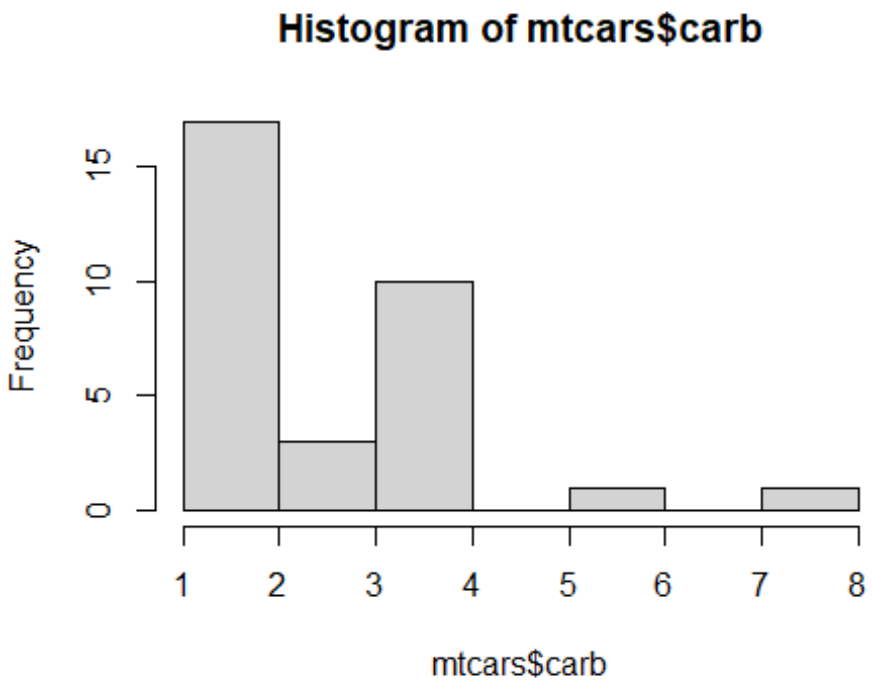

```r
#inference: Rear axle ratio is following normal distribution
hist(mtcars$wt)
```



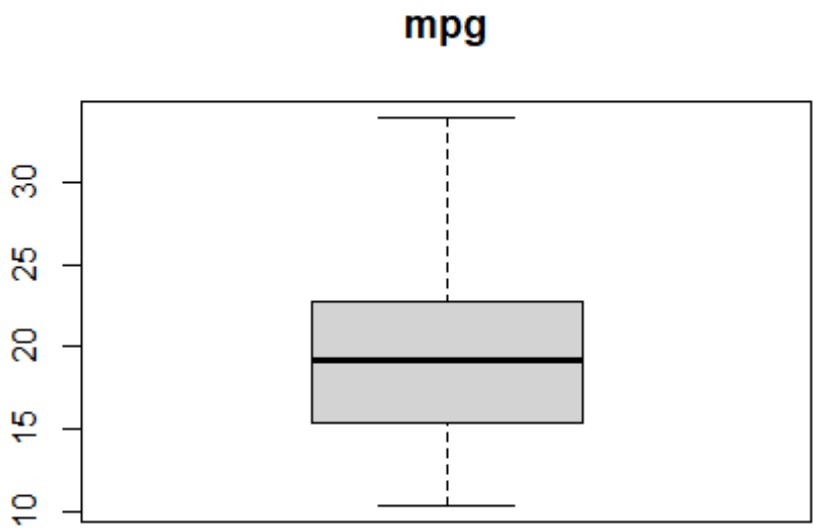

```
#inference: weight is having The Bi-Modal Distribution
hist(mtcars$qsec)
```



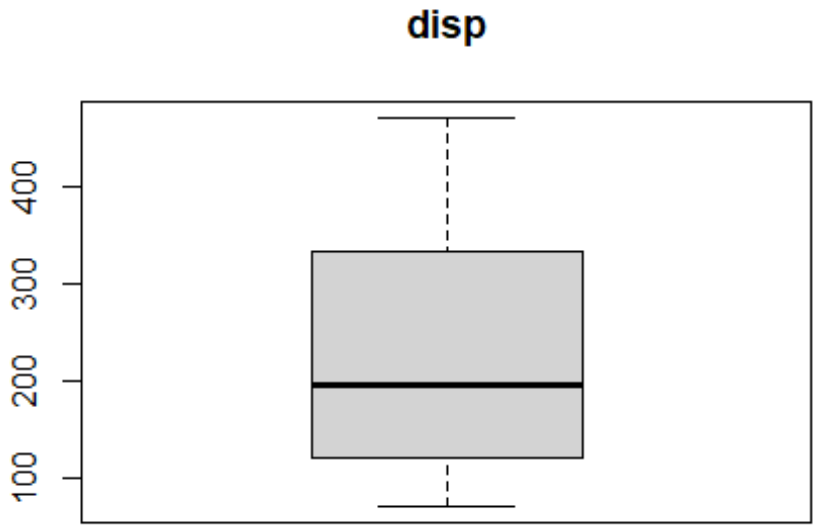
```r
#inference: mile time is following normal distribution
hist(mtcars$carb)
```



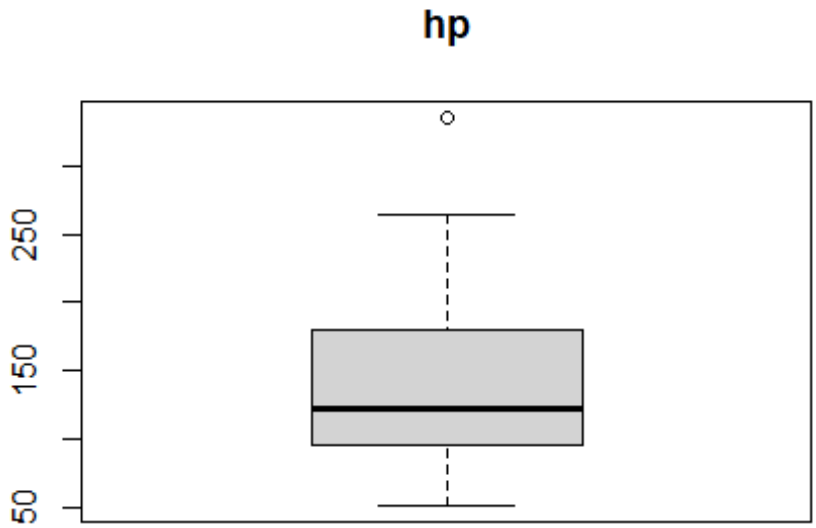

```r
#inference: carburetors is having The Bi-Modal Distribution

#boxplot
boxplot(mtcars$mpg, main="mpg")
```



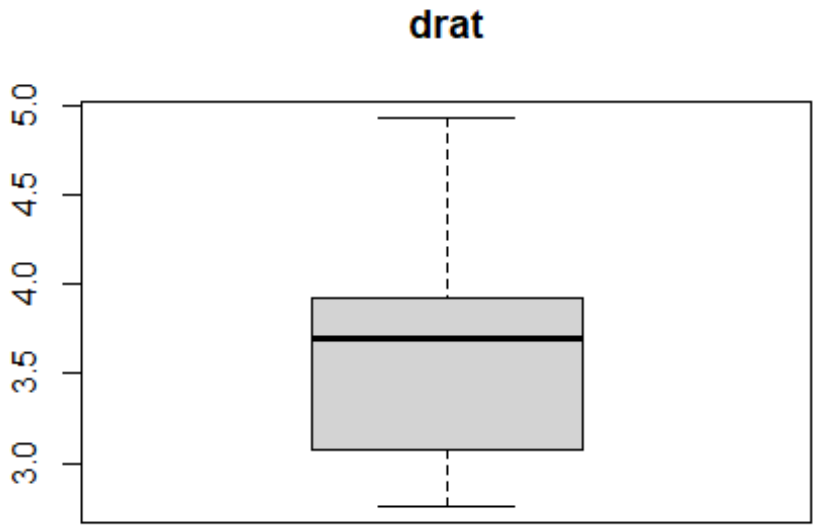

```
boxplot(mtcars$disp, main="disp")
```



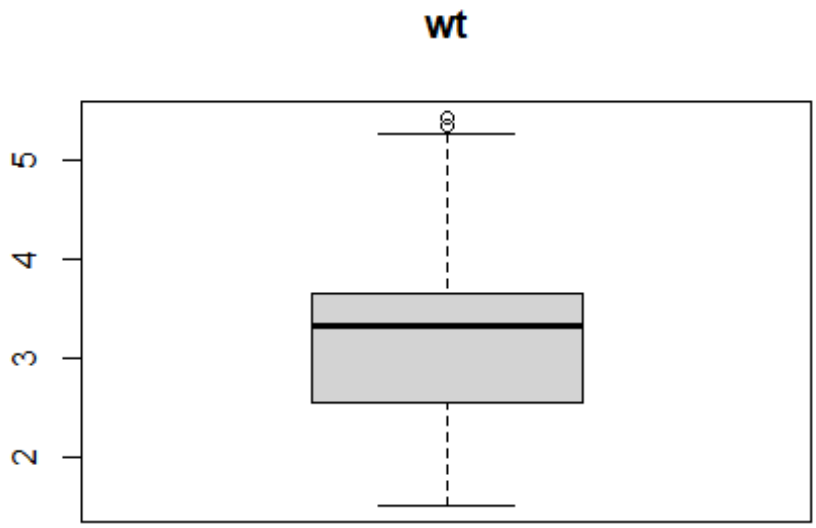

```
boxplot(mtcars$hp, main="hp")
```



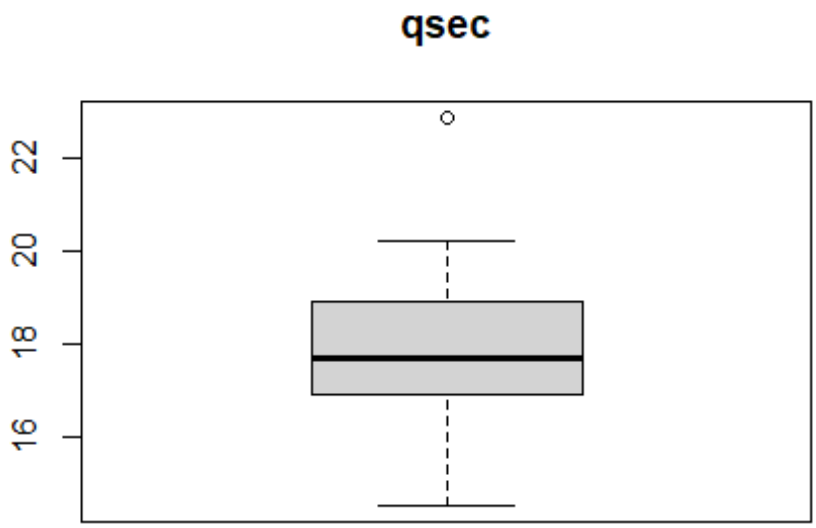

```
boxplot(mtcars$drat, main="drat")
```




```
boxplot(mtcars$wt, main="wt")
```




```
boxplot(mtcars$qsec, main="qsec")


#4.Choose optimum lamba value
x_var=model.matrix(mpg~.,mtcars)[,-1]
y_var=mtcars$mpg

lambda_seq =10^seq(2, -2, by = -.1)
set.seed(86)
train = sort(sample(1:nrow(x_var),0.8*nrow(x_var)))
x_test = (-train)
y_test = y_var[x_test]


#5.Extract the model using k-cross validation
library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1
```




```
cv_output = cv.glmnet(x_var[train,], y_var[train], alpha = 1, lambda =
lambda_seq)

## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3
observations per
## fold

best_lambda= cv_output$lambda.min
best_lambda

## [1] 0.3981072

best_fit=cv_output$glmnet.fit
summary(best_fit)

##           Length Class     Mode
## a0         41    -none-    numeric
## beta      410    dgCMatrix S4
## df         41    -none-    numeric
## dim         2    -none-    numeric
## lambda     41    -none-    numeric
## dev.ratio  41    -none-    numeric
## nulldev     1    -none-    numeric
## npasses     1    -none-    numeric
## jerr        1    -none-    numeric
## offset      1    -none-    logical
## call        5    -none-    call
## nobs        1    -none-    numeric
```

```r
#6.Build the final model and interpret
library(ridge)
inputData = data.frame (mtcars)
trainingData=inputData[train, ]
testData = inputData[-train, ]
linRidgeMod =linearRidge(y_var ~ x_var, data = trainingData)
linRidgeMod
```

```
##
## Call:
## linearRidge(formula = y_var ~ x_var, data = trainingData)
##
##  (Intercept)     x_varcyl    x_vardisp      x_varhp    x_vardrat
x_varwt
## 19.228871873 -0.244604033 -0.001631983 -0.013126081  0.971434008 -
1.935291447
##    x_varqsec      x_varvs      x_varam    x_vargear    x_varcarb
##  0.327054947  0.469354207  2.131844699  0.639243640 -0.660066576
```

```r
predicted=predict(linRidgeMod, testData) # predict on test data
```

```
## Warning: 'newdata' had 7 rows but variables found have 32 rows
```

```r
compare = cbind (actual=testData$response, predicted) # combine
compare
```

```
##    predicted
## 1   22.20627
## 2   21.89592
## 3   26.68822
## 4   20.75135
## 5   16.95415
## 6   20.34093
## 7   14.13596
## 8   22.85553
## 9   23.68130
## 10  19.39501
## 11  19.59124
## 12  15.19320
## 13  15.91661
## 14  15.95067
## 15  11.65483
## 16  11.22209
## 17  11.30333
## 18  27.82737
## 19  29.00313
## 20  28.83592
## 21  23.87643
## 22  16.76810
## 23  17.47494
## 24  13.99427
```

```
## 25  16.04689
## 26  28.15331
## 27  26.49133
## 28  27.35073
## 29  18.62860
## 30  19.82039
## 31  13.73599
## 32  25.15599
```

```
mean (apply(compare, 1, min)/apply(compare, 1, max))
```

```
## [1] 1
```

```
#accuracy is 1
```