chap4_exercise_labs

```r
library(ISLR2)
library(ggplot2)
library(MASS)
```

```

Attaching package: 'MASS'

The following object is masked from 'package:ISLR2':

    Boston
```

```r
library(e1071)
library(class)
library(dplyr)
```

```

Attaching package: 'dplyr'

The following object is masked from 'package:MASS':

    select

The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

#######labs

```r
head(Smarket)
```

```
  Year   Lag1   Lag2   Lag3   Lag4   Lag5 Volume  Today Direction
1 2001  0.381 -0.192 -2.624 -1.055  5.010 1.1913  0.959        Up
2 2001  0.959  0.381 -0.192 -2.624 -1.055 1.2965  1.032        Up
3 2001  1.032  0.959  0.381 -0.192 -2.624 1.4112 -0.623      Down
4 2001 -0.623  1.032  0.959  0.381 -0.192 1.2760  0.614        Up
5 2001  0.614 -0.623  1.032  0.959  0.381 1.2057  0.213        Up
6 2001  0.213  0.614 -0.623  1.032  0.959 1.3491  1.392        Up
```

```r
names(Smarket)
```

```
[1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
[7] "Volume"    "Today"     "Direction"
```

```r
summary(Smarket)
```

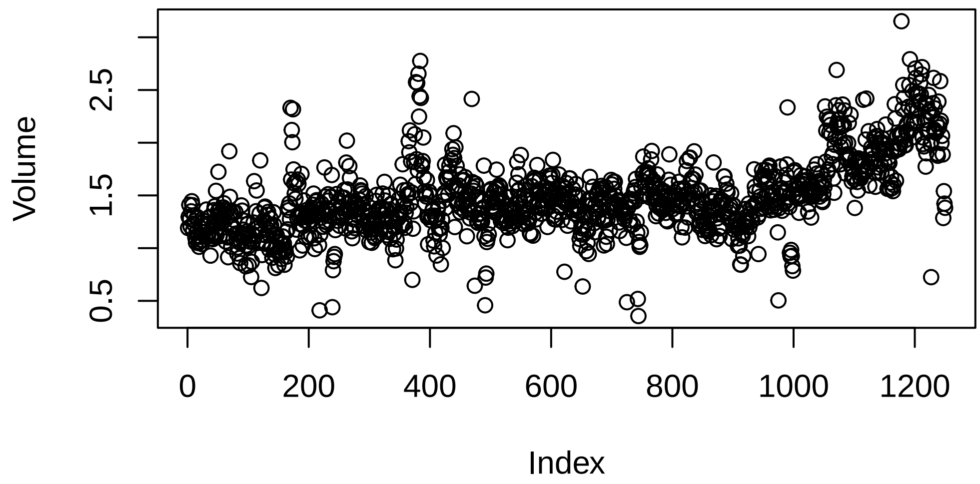
```
      Year           Lag1                Lag2                Lag3          
 Min.   :2001   Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.922000  
 1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500   1st Qu.:-0.640000  
 Median :2003   Median : 0.039000   Median : 0.039000   Median : 0.038500  
 Mean   :2003   Mean   : 0.003834   Mean   : 0.003919   Mean   : 0.001716  
 3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.596750  
 Max.   :2005   Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.733000  
      Lag4                Lag5              Volume           Today          
 Min.   :-4.922000   Min.   :-4.92200   Min.   :0.3561   Min.   :-4.922000  
 1st Qu.:-0.640000   1st Qu.:-0.64000   1st Qu.:1.2574   1st Qu.:-0.639500  
 Median : 0.038500   Median : 0.03850   Median :1.4229   Median : 0.038500  
 Mean   : 0.001636   Mean   : 0.00561   Mean   :1.4783   Mean   : 0.003138  
 3rd Qu.: 0.596750   3rd Qu.: 0.59700   3rd Qu.:1.6417   3rd Qu.: 0.596750  
 Max.   : 5.733000   Max.   : 5.73300   Max.   :3.1525   Max.   : 5.733000  
 Direction 
 Down:602  
```

```
 Up  :648
```

```r
cor(Smarket[, -9])
```

```
             Year         Lag1         Lag2         Lag3         Lag4
Year   1.00000000  0.029699649  0.030596422  0.033194581  0.035688718
Lag1   0.02969965  1.000000000 -0.026294328 -0.010803402 -0.002985911
Lag2   0.03059642 -0.026294328  1.000000000 -0.025896670 -0.010853533
Lag3   0.03319458 -0.010803402 -0.025896670  1.000000000 -0.024051036
Lag4   0.03568872 -0.002985911 -0.010853533 -0.024051036  1.000000000
Lag5   0.02978799 -0.005674606 -0.003557949 -0.018808338 -0.027083641
Volume 0.53900647  0.040909908 -0.043383215 -0.041823686 -0.048414246
Today  0.03009523 -0.026155045 -0.010250033 -0.002447647 -0.006899527
               Lag5      Volume        Today
Year    0.029787995  0.53900647  0.030095229
Lag1   -0.005674606  0.04090991 -0.026155045
Lag2   -0.003557949 -0.04338321 -0.010250033
Lag3   -0.018808338 -0.04182369 -0.002447647
Lag4   -0.027083641 -0.04841425 -0.006899527
Lag5    1.000000000 -0.02200231 -0.034860083
Volume -0.022002315  1.00000000  0.014591823
Today  -0.034860083  0.01459182  1.000000000
```

```r
attach(Smarket)
plot(Volume)
```



#Logistic Regression

```
glm.fits <- glm(
  Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
  data = Smarket,
  family = binomial
)
summary(glm.fits)
```

```
Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
    Volume, family = binomial, data = Smarket)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.126000   0.240736  -0.523    0.601
Lag1        -0.073074   0.050167  -1.457    0.145
Lag2        -0.042301   0.050086  -0.845    0.398
Lag3         0.011085   0.049939   0.222    0.824
Lag4         0.009359   0.049974   0.187    0.851
Lag5         0.010313   0.049511   0.208    0.835
```

```
Volume       0.135441   0.158360   0.855    0.392

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1731.2  on 1249  degrees of freedom
Residual deviance: 1727.6  on 1243  degrees of freedom
AIC: 1741.6

Number of Fisher Scoring iterations: 3
```

smallest p-value here is associated with Lag1

```r
coef(glm.fits)
```

```
 (Intercept)         Lag1         Lag2         Lag3         Lag4         Lag5 
-0.126000257 -0.073073746 -0.042301344  0.011085108  0.009358938  0.010313068 
      Volume 
 0.135440659 
```

```r
summary(glm.fits)$coef
```

```
                Estimate Std. Error    z value  Pr(>|z|)
(Intercept) -0.126000257 0.24073574 -0.5233966 0.6006983
Lag1        -0.073073746 0.05016739 -1.4565986 0.1452272
Lag2        -0.042301344 0.05008605 -0.8445733 0.3983491
Lag3         0.011085108 0.04993854  0.2219750 0.8243333
Lag4         0.009358938 0.04997413  0.1872757 0.8514445
Lag5         0.010313068 0.04951146  0.2082966 0.8349974
Volume       0.135440659 0.15835970  0.8552723 0.3924004
```

```r
summary(glm.fits)$coef[, 4]###col 5 the p-value
```

```
(Intercept)        Lag1        Lag2        Lag3        Lag4        Lag5 
  0.6006983   0.1452272   0.3983491   0.8243333   0.8514445   0.8349974 
     Volume 
  0.3924004 
```

```r
glm.probs <- predict(glm.fits, type = "response")
glm.probs[1:10]
```

```
        1         2         3         4         5         6         7         8
0.5070841 0.4814679 0.4811388 0.5152224 0.5107812 0.5069565 0.4926509 0.5092292
        9        10
0.5176135 0.4888378
```

```r
contrasts(Direction)
```

```
     Up
Down  0
Up    1
```

```r
glm.pred <- rep("Down", 1250)
glm.pred[glm.probs > .5] = "Up"
```

```r
table(glm.pred, Direction)
```

```
        Direction
glm.pred Down  Up
    Down  145 141
    Up    457 507
```

```r
mean(glm.pred == Direction)
```

```
[1] 0.5216
```

```r
train <- (Year < 2005)
Smarket.2005 <- Smarket[!train, ]#test df
dim(Smarket.2005)
```

```
[1] 252   9
```

```r
Direction.2005 <- Direction[!train]
```

```r
glm.fits <- glm(
  Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
  data = Smarket, family = binomial, subset = train
)
```

#use the test_x to predict

```r
glm.probs <- predict(glm.fits, Smarket.2005,
                     type = "response")
```

```r
glm.pred <- rep("Down", 252)
glm.pred[glm.probs > .5] <- "Up"
table(glm.pred, Direction.2005)#Direction.2005 truth/test y
```

```
        Direction.2005
glm.pred Down Up
    Down   77 97
    Up     34 44
```

```r
mean(glm.pred == Direction.2005)
```

```
[1] 0.4801587
```

#refit use Lag1 and Lag2

```r
glm.fits <- glm(
  Direction ~ Lag1 + Lag2 ,
  data = Smarket, family = binomial, subset = train
)
```

```r
glm.probs <- predict(glm.fits, Smarket.2005,
                     type = "response")

glm.pred <- rep("Down", 252)
glm.pred[glm.probs > .5] <- "Up"
table(glm.pred, Direction.2005)
```

```
        Direction.2005
glm.pred Down  Up
    Down   35  35
    Up     76 106
```

```r
mean(glm.pred == Direction.2005)
```

```
[1] 0.5595238
```

```r
predict(glm.fits,
        newdata = data.frame(Lag1 = c(1.2, 1.5), Lag2 = c(1.1, -0.8)),
        type = "response" )
```

```
        1         2 
0.4791462 0.4960939 
```

#LDA

```r
lda.fit <- lda(Direction ~ Lag1 + Lag2,
               data = Smarket,
               subset = train)
lda.fit
```

```
Call:
lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)

Prior probabilities of groups:
    Down       Up 
0.491984 0.508016 

Group means:
            Lag1        Lag2
Down  0.04279022  0.03389409
Up   -0.03954635 -0.03132544

Coefficients of linear discriminants:
            LD1
Lag1 -0.6420190
Lag2 -0.5135293
```

```r
lda.pred <- predict(lda.fit, Smarket.2005)
lda.class <- lda.pred$class
table(lda.class, Direction.2005)#LDA,logistic almost identical
```

```
         Direction.2005
lda.class Down  Up
     Down   35  35
     Up     76 106
```

```r
mean(lda.class == Direction.2005)
```

```
[1] 0.5595238
```

```r
sum(lda.pred$posterior[, 1] >= .5)
```

```
[1] 70
```

```r
sum(lda.pred$posterior[, 1] < .5)#the col2 is p(Down)
```

```
[1] 182
```

```r
head(lda.pred$posterior)
```

```
          Down        Up
999  0.4901792 0.5098208
1000 0.4792185 0.5207815
1001 0.4668185 0.5331815
1002 0.4740011 0.5259989
1003 0.4927877 0.5072123
1004 0.4938562 0.5061438
```

```r
lda.class[1:20]
```

```
 [1] Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Down Up   Up   Up
[16] Up   Up   Down Up   Up
Levels: Down Up
```

```r
sum(lda.pred$posterior[, 1] > .9)
```

```
[1] 0
```

```r
qda.fit <- qda(Direction ~ Lag1 + Lag2, data = Smarket,
               subset = train)
qda.fit
```

```
Call:
qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)

Prior probabilities of groups:
    Down       Up 
0.491984 0.508016 

Group means:
            Lag1        Lag2
Down  0.04279022  0.03389409
Up   -0.03954635 -0.03132544
```

```r
qda.class <- predict(qda.fit, Smarket.2005)$class
table(qda.class, Direction.2005)
```

```
         Direction.2005
qda.class Down  Up
     Down   30  20
     Up     81 121
```

```r
mean(qda.class == Direction.2005)
```

```
[1] 0.5992063
```

#Naive Bayes

```r
nb.fit <- naiveBayes(Direction ~ Lag1 + Lag2, data = Smarket,
                     subset = train)
```

```r
nb.fit
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
    Down       Up 
0.491984 0.508016 

Conditional probabilities:
      Lag1
Y             [,1]     [,2]
  Down  0.04279022 1.227446
  Up   -0.03954635 1.231668

      Lag2
Y             [,1]     [,2]
  Down  0.03389409 1.239191
  Up   -0.03132544 1.220765
```

```r
mean(Lag1[train][Direction[train] == "Down"])
```

```
[1] 0.04279022
```

```r
nb.class <- predict(nb.fit, Smarket.2005)
table(nb.class, Direction.2005)
```

```
        Direction.2005
nb.class Down  Up
    Down   28  20
    Up     83 121
```

```r
mean(nb.class == Direction.2005)
```

```
[1] 0.5912698
```

```r
nb.preds <- predict(nb.fit, Smarket.2005, type = "raw")
nb.pred <- rep("Down", 252)
nb.pred[nb.preds[,2] > .5] <- "Up"
table(nb.pred, Direction.2005)
```

```
       Direction.2005
nb.pred Down  Up
   Down   28  20
   Up     83 121
```

```r
#K-Nearest Neighbors
train.X <- cbind(Lag1, Lag2)[train, ]
test.X <- cbind(Lag1, Lag2)[!train, ]
train.Direction <- Direction[train]
```

```r
set.seed(1)
knn.pred <- knn(train.X, test.X, train.Direction, k = 1)
table(knn.pred, Direction.2005)
```

```
        Direction.2005
knn.pred Down Up
    Down   43 58
    Up     68 83
```

```r
knn.pred <- knn(train.X, test.X, train.Direction, k = 3)
table(knn.pred, Direction.2005)
```

```
        Direction.2005
knn.pred Down Up
    Down   48 54
    Up     63 87
```

```r
mean(knn.pred == Direction.2005)
```

```
[1] 0.5357143
```

```r
dim(Caravan)
```

```
[1] 5822   86
```

```r
attach(Caravan)
summary(Purchase)
```

```
  No  Yes
5474  348
```

```r
standardized.X <- scale(Caravan[, -86])#only drop Purchase
```

```r
test <- 1:1000
train.X <- standardized.X[-test, ]
test.X <- standardized.X[test, ]
train.Y <- Purchase[-test]
test.Y <- Purchase[test]
```

```r
set.seed(1)
knn.pred <- knn(train.X, test.X, train.Y, k = 1)
mean(test.Y != knn.pred)
```

```
[1] 0.118
```

```r
mean(test.Y != "No")
```

```
[1] 0.059
```

```r
table(knn.pred, test.Y)
```

```
        test.Y
knn.pred  No Yes
     No  873  50
     Yes  68   9
```

```
knn.pred <- knn(train.X, test.X, train.Y, k = 3)
table(knn.pred, test.Y)
```

```
        test.Y
knn.pred  No Yes
     No  920  54
     Yes  21   5
```

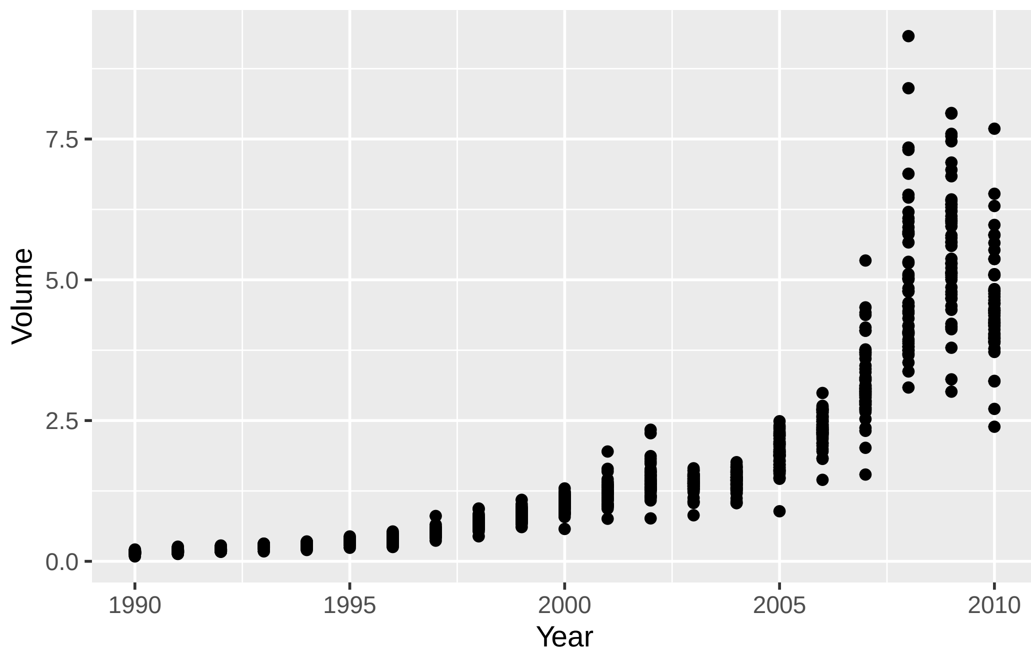
```
knn.pred <- knn(train.X, test.X, train.Y, k = 5)
table(knn.pred, test.Y)
```

```
        test.Y
knn.pred  No Yes
     No  930  55
     Yes  11   4
```

#######exercise

13.

```
Weekly %>%
  ggplot(aes(x = Year, y = Volume)) +
  geom_point()
```




```r
glm.fits <- glm(
  Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
  data = Weekly,
  family = binomial
)
summary(glm.fits) #lag2
```

```
Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
    Volume, family = binomial, data = Weekly)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.26686    0.08593   3.106   0.0019 **
Lag1        -0.04127    0.02641  -1.563   0.1181
Lag2         0.05844    0.02686   2.175   0.0296 *
Lag3        -0.01606    0.02666  -0.602   0.5469
Lag4        -0.02779    0.02646  -1.050   0.2937
Lag5        -0.01447    0.02638  -0.549   0.5833
Volume      -0.02274    0.03690  -0.616   0.5377
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1496.2  on 1088  degrees of freedom
Residual deviance: 1486.4  on 1082  degrees of freedom
AIC: 1500.4

Number of Fisher Scoring iterations: 4
```

```
glm.probs <- predict(glm.fits, Weekly,
                     type = "response")
glm.pred <- rep("Down", 1089)
glm.pred[glm.probs > .5] = "Up"
table(glm.pred, Weekly$Direction)
```

```
glm.pred Down  Up
    Down   54  48
    Up    430 557
```

```
attach(Weekly)
```

```
The following objects are masked from Smarket:

    Direction, Lag1, Lag2, Lag3, Lag4, Lag5, Today, Volume, Year
```

```
train <- (Year < 2009)
Weekly.test <- Weekly[!train, ]#test df
Direction.test <- Direction[!train]
```

```
glm.fits <- glm(
  Direction ~ Lag2,
  data = Weekly, family = binomial, subset = train
)
glm.probs <- predict(glm.fits, Weekly.test, type = "response")
```

#logistic

```
glm.pred <- rep("Down", 104)
glm.pred[glm.probs > .5] <- "Up"
table(glm.pred, Direction.test)
```

```
        Direction.test
glm.pred Down Up
    Down    9  5
    Up     34 56
```

```
mean(glm.pred == Direction.test)
```

```
[1] 0.625
```

#LDA

```
lda.fit <- lda(Direction ~ Lag2,
               data = Weekly,
               subset = train)

lda.pred <- predict(lda.fit, Weekly.test)
lda.class <- lda.pred$class
table(lda.class, Direction.test)
```

```
         Direction.test
lda.class Down Up
     Down    9  5
     Up     34 56
```

#QDA

```
qda.fit <- qda(Direction ~ Lag2,
               data = Weekly,
               subset = train)

qda.pred <- predict(qda.fit, Weekly.test)
qda.class <- qda.pred$class
table(qda.class, Direction.test)
```

```
         Direction.test
qda.class Down Up
     Down    0  0
     Up     43 61
```

```r
mean(qda.class == Direction.test)
```

```
[1] 0.5865385
```

###KNN with K = 1

```r
train.X <- matrix(Weekly$Lag2[train], ncol = 1)
test.X <- matrix(Weekly$Lag2[!train], ncol = 1)
train.Direction <- Direction[train]

set.seed(1)
knn.pred <- knn(train.X, test.X, train.Direction, k = 1)
table(knn.pred, Direction.test)
```

```
        Direction.test
knn.pred Down Up
    Down   21 30
    Up     22 31
```

#h

```r
nb.fit <- naiveBayes(Direction ~ Lag2, data = Weekly,
                     subset = train)
nb.class <- predict(nb.fit, Weekly.test)
table(nb.class, Direction.test)
```

```
        Direction.test
nb.class Down Up
    Down    0  0
    Up     43 61
```

```r
mean(nb.class == Direction.test)
```

```
[1] 0.5865385
```

#j Experiment

```r
for (K in 1:5) {
  train.X <- matrix(Weekly$Lag2[train], ncol = 1)
  test.X <- matrix(Weekly$Lag2[!train], ncol = 1)
  train.Direction <- Direction[train]
  set.seed(1)
  knn.pred <- knn(train.X, test.X, train.Direction, k = K)
  C <- table(knn.pred, Direction.test)

  if ("Up" %in% rownames(C)) {
    pred <- sum(C["Up",])
    did_increase <- C["Up", "Up"]
    accuracy <- did_increase / pred  # Ensure 'pred' is not zero before division

    # Printing results
    cat(sprintf("K=%d: # predicted to be up: %2d, # who did increase %d, accuracy %.1f%%\n
                K, pred, did_increase, accuracy * 100))
  }
}
```

```
K=1: # predicted to be up: 53, # who did increase 31, accuracy 58.5%
K=2: # predicted to be up: 58, # who did increase 34, accuracy 58.6%
K=3: # predicted to be up: 68, # who did increase 41, accuracy 60.3%
K=4: # predicted to be up: 67, # who did increase 44, accuracy 65.7%
K=5: # predicted to be up: 67, # who did increase 40, accuracy 59.7%
```

#14

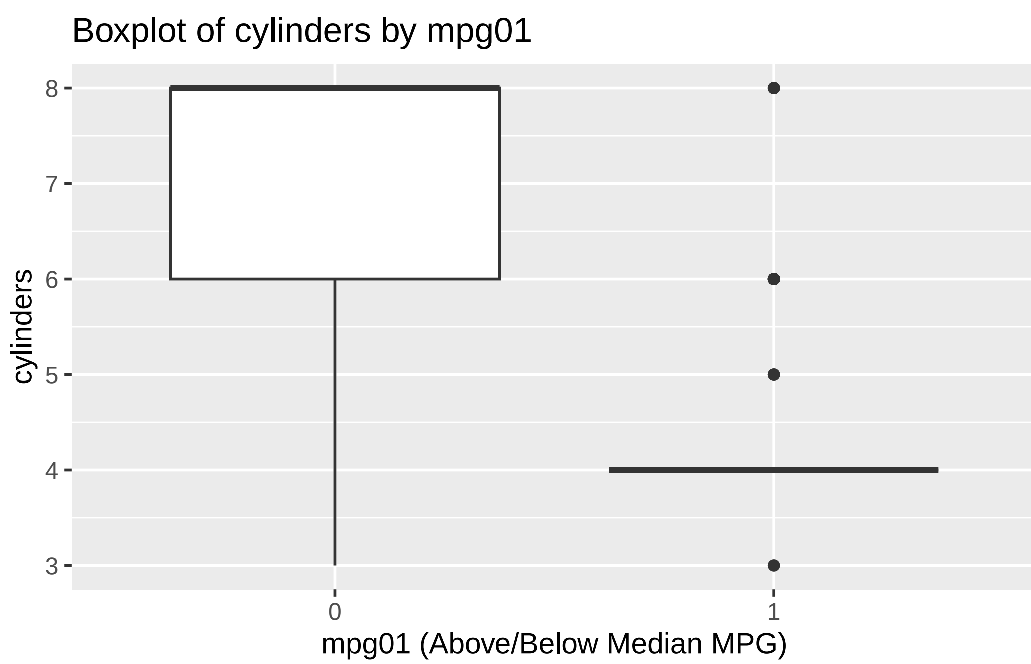
```r
Auto <- read.csv("Auto.csv")
Auto$horsepower <- as.numeric(Auto$horsepower)
```

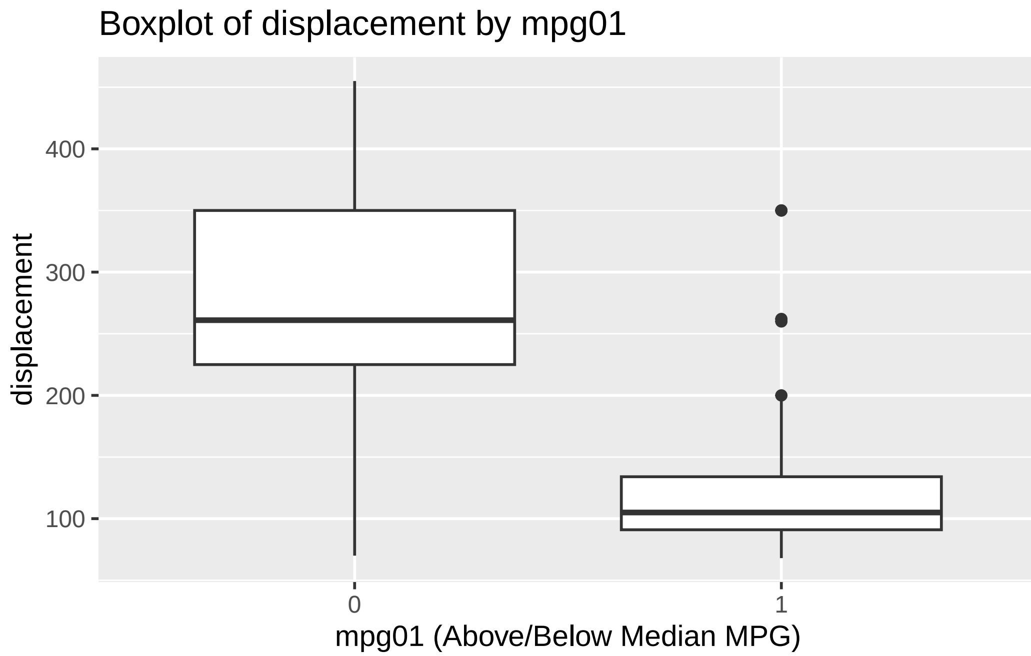
```
Warning: NAs introduced by coercion
```

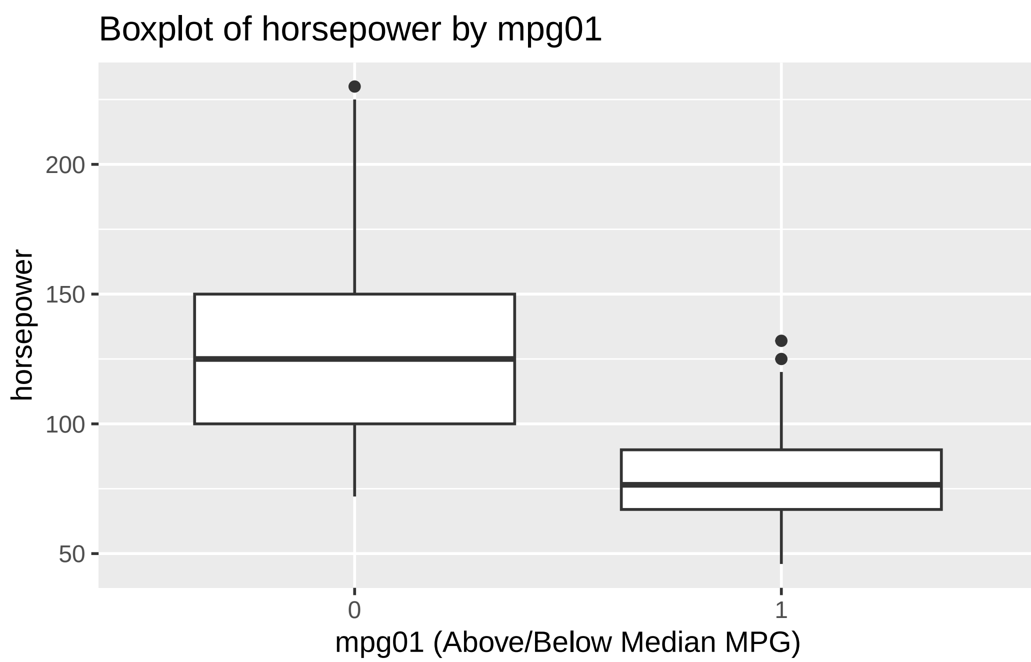
```r
Auto <- Auto[!is.na(Auto$horsepower), ]
```

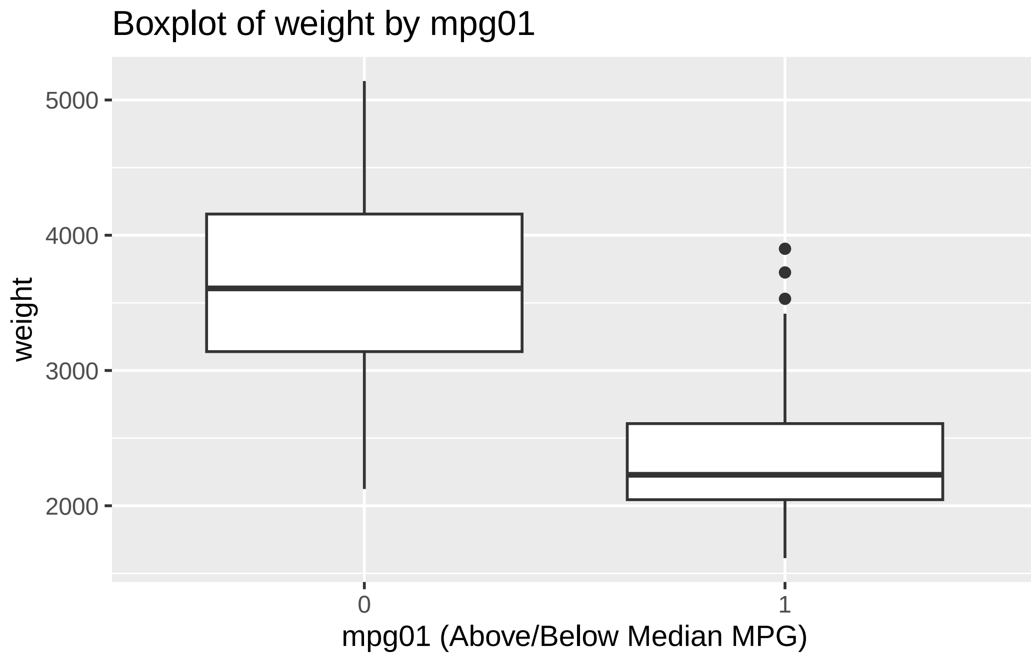
```r
Auto$mpg01 <- if_else(Auto$mpg > median(Auto$mpg), 1, 0)
```

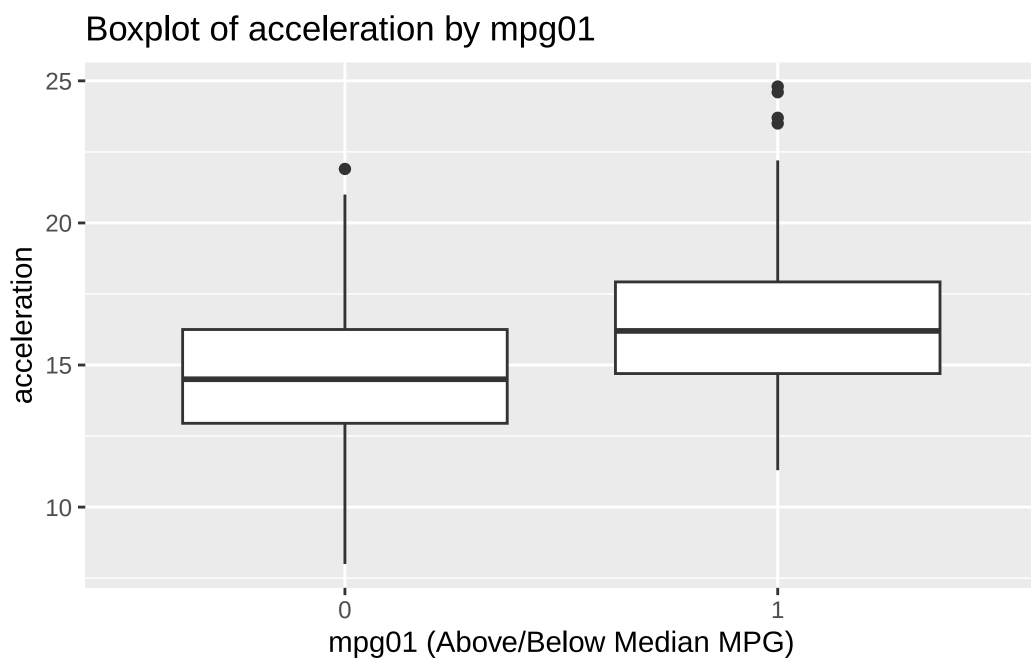
```r
vars <- c('cylinders', 'displacement', 'horsepower', 'weight', 'acceleration')

# Loop through the variable names
for (i in vars) {
    # Use ggplot to create a box plot
    p <- ggplot(Auto, aes(x = factor(mpg01), y = get(i))) +
        geom_boxplot() +
        labs(title = paste('Boxplot of', i, 'by mpg01'),
             x = 'mpg01 (Above/Below Median MPG)',
             y = i)

    # Print the plot
    print(p)
}
```




Boxplot of displacement by mpg01
400
300
200
100
displacement
0
1
mpg01 (Above/Below Median MPG)


Boxplot of horsepower by mpg01
200
150
100
50
horsepower
0
1
mpg01 (Above/Below Median MPG)


Boxplot of weight by mpg01
weight
5000
4000
3000
2000
0
1
mpg01 (Above/Below Median MPG)


Boxplot of acceleration by mpg01
acceleration
25
20
15
10
0
1
mpg01 (Above/Below Median MPG)

```r
#'displacement' 'weight'
set.seed(1)
Auto_df <- Auto[, c('displacement', 'weight', 'mpg01')]
test <- sample(nrow(Auto_df), 100)

train.X <- Auto_df[-test, -3]
test.X <- Auto_df[test, -3]
train.df <- Auto_df[-test, ]

train.Y <- Auto_df[-test, 'mpg01']
test.Y <- Auto_df[test, 'mpg01']
```

#d LDA

```r
lda.fit <- lda(mpg01 ~ displacement + weight,
               data = train.df)

lda.pred <- predict(lda.fit, test.X)
lda.class <- lda.pred$class
table(lda.class, test.Y)
```

```
         test.Y
lda.class  0  1
        0 43  4
        1  7 46
```

```r
mean(lda.class == test.Y)
```

```
[1] 0.89
```

#QDA

```r
qda.fit <- qda(mpg01 ~ displacement + weight,
               data = train.df)

qda.pred <- predict(qda.fit, test.X)
qda.class <- qda.pred$class
table(qda.class, test.Y)
```

```
          test.Y
qda.class  0  1
        0 47  5
        1  3 45
```

```r
mean(qda.class == test.Y)
```

```
[1] 0.92
```

#logistic regression

```r
glm.fits <- glm(
  mpg01 ~ displacement + weight,
  data = train.df, family = binomial
)

glm.probs <- predict(glm.fits, test.X,
                     type = "response")
```

```r
glm.pred <- rep("0", 100)
glm.pred[glm.probs > .5] <- "1"
table(glm.pred, test.Y)
```

```
        test.Y
glm.pred  0  1
       0 46  5
       1  4 45
```

```r
mean(glm.pred == test.Y)
```

```
[1] 0.91
```

naive Bayes

```r
nb.fit <- naiveBayes(mpg01 ~ displacement + weight,
                     data = train.df)
nb.class <- predict(nb.fit, test.X)
table(nb.class, test.Y)
```

```
        test.Y
nb.class  0  1
       0 45  4
       1  5 46
```

```r
mean(nb.class == test.Y)
```

```
[1] 0.91
```

```r
knn.pred <- knn(train.X, test.X, train.Y, k = 1)
C <- table(Predicted = knn.pred, Actual = test.Y)
print(C)
```

```
         Actual
Predicted  0  1
        0 43  8
        1  7 42
```

```r
for (K in 1:5) {
  set.seed(1)
  knn.pred <- knn(train.X, test.X, train.Y, k = K)
  C <- table(Predicted = knn.pred, Actual = test.Y)


  if ("1" %in% rownames(C)) {
    pred <- sum(C["1",])
    did_higher <- C["1", "1"]
    if (pred > 0) {
      accuracy <- did_higher / pred

      # Printing results
```

```
      cat(sprintf("K=%d: # predicted to be higher than median: %2d, # who did higher than
                  K, pred, did_higher, accuracy * 100))
    }
  }
}
```

```
K=1: # predicted to be higher than median: 49, # who did higher than median 42, accuracy 85.
K=2: # predicted to be higher than median: 50, # who did higher than median 44, accuracy 88.
K=3: # predicted to be higher than median: 50, # who did higher than median 44, accuracy 88.
K=4: # predicted to be higher than median: 50, # who did higher than median 45, accuracy 90.
K=5: # predicted to be higher than median: 52, # who did higher than median 46, accuracy 88.
```

#15

```
Power <- function() {
    result <- 2^3
    print(result)
}
Power()
```

```
[1] 8
```

```
Power2 <- function(x, a) {
    result <- x^a
    print(result)
}
Power2(3, 8)
```

```
[1] 6561
```

```
Power2(10, 3)
```

```
[1] 1000
```

```
Power2(8, 17)
```

```
[1] 2.2518e+15
```

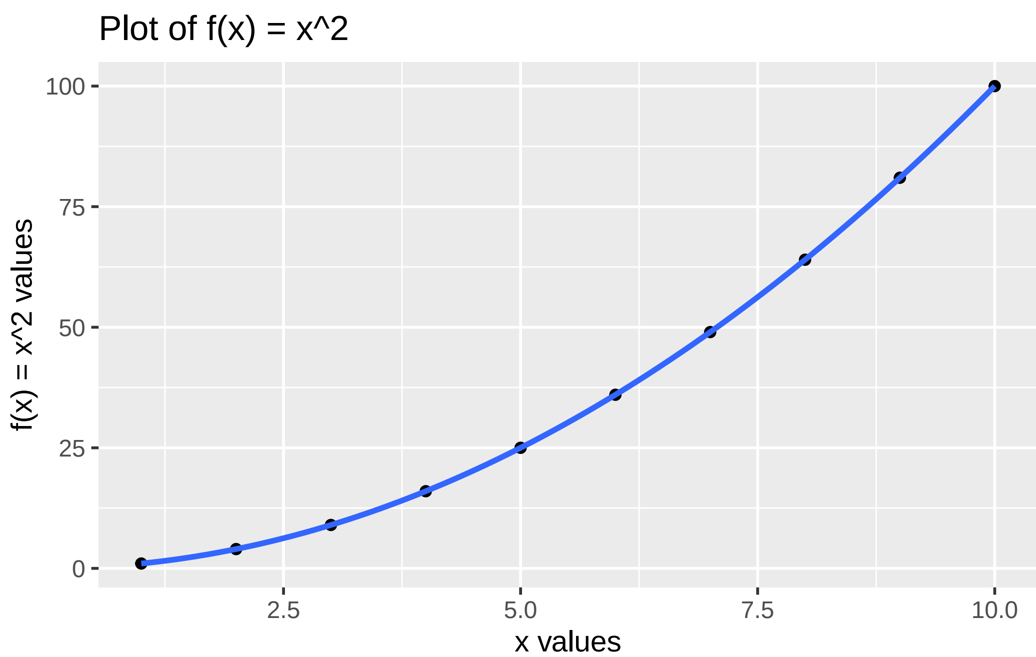
```r
Power2(13, 13)
```

```
[1] 3.028751e+14
```

```r
Power3 <- function(x, a) {
    result <- x^a
    return(result)
}


value1 <- Power3(2, 3)
```

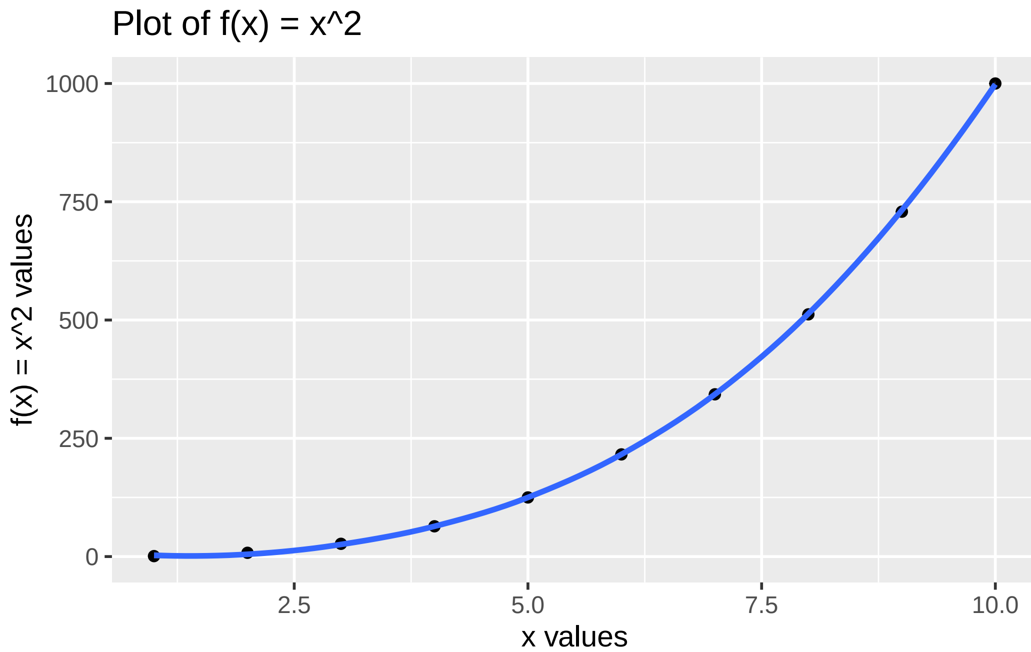
```r
x_values <- 1:10
y_values <- x_values^2

df <- data.frame(x_values, y_values)

df %>%
  ggplot( aes(x = x_values, y = y_values)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Plot of f(x) = x^2") +
  xlab("x values") +
  ylab("f(x) = x^2 values")
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```r
PlotPower <- function(x_values, a) {
  y_values <- x_values^a

  df <- data.frame(x_values, y_values)

  df %>%
  ggplot( aes(x = x_values, y = y_values)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Plot of f(x) = x^2") +
  xlab("x values") +
  ylab("f(x) = x^2 values")
}

PlotPower(1:10, 3)
```

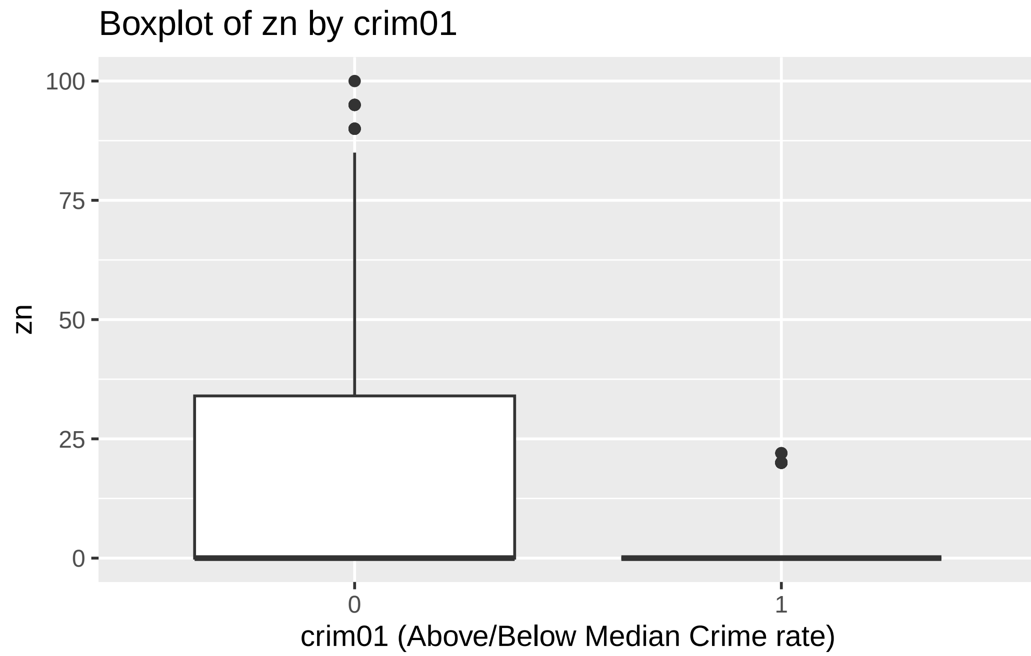
```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

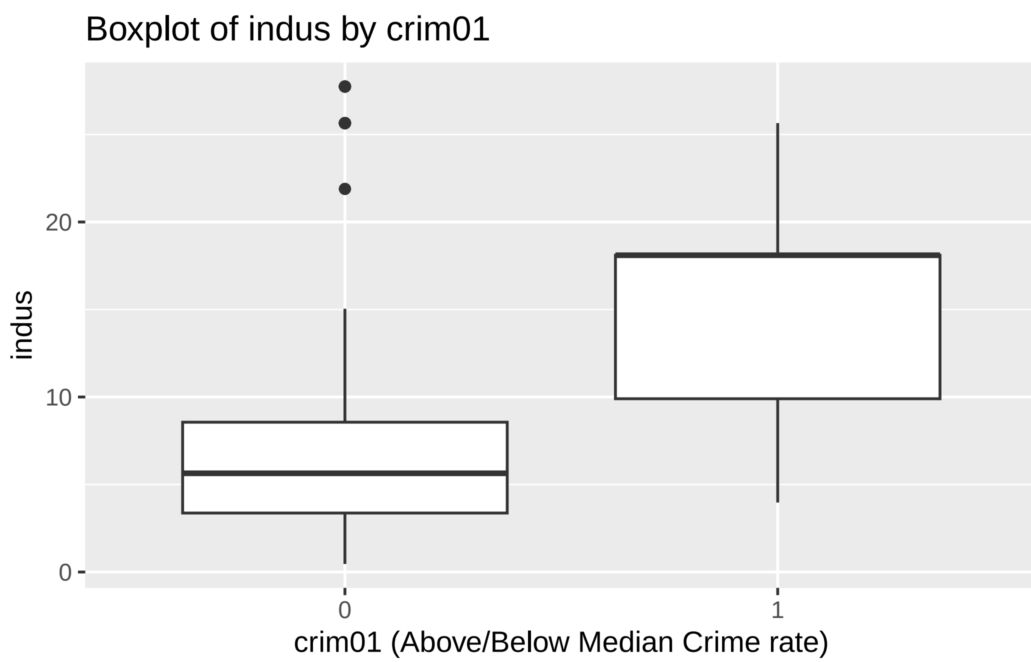


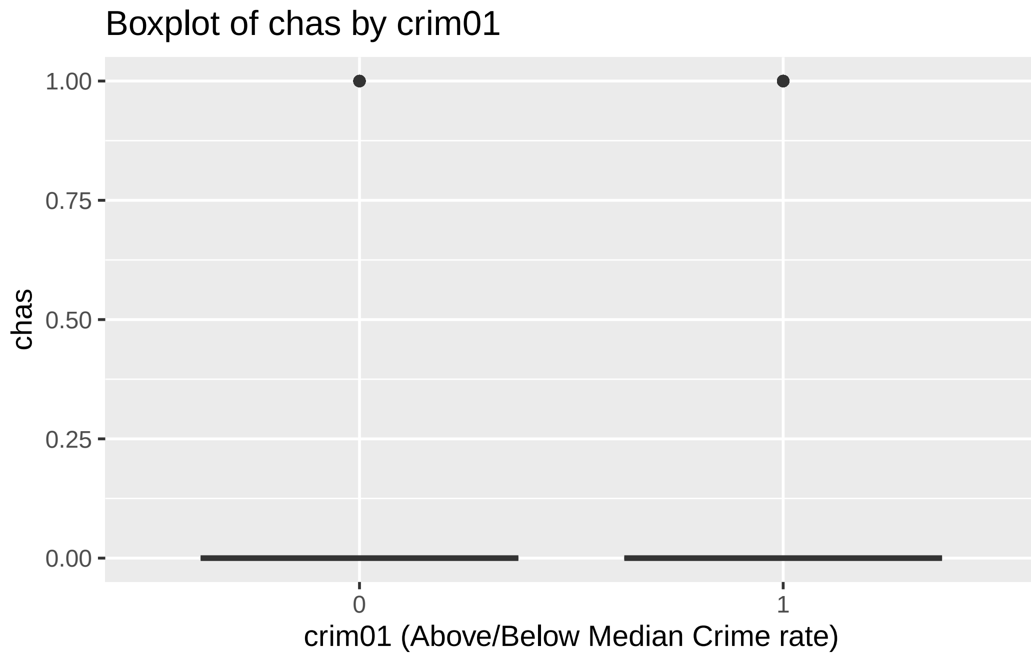
#16.

```
Boston$crim01 <- if_else(Boston$crim > median(Boston$crim), 1, 0)
```

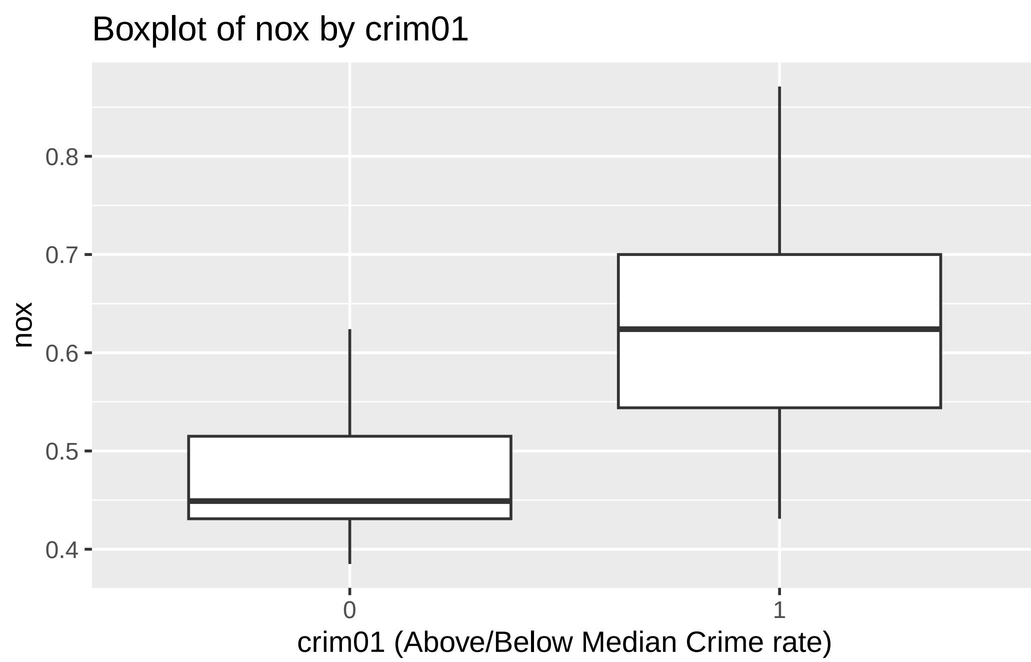
```
vars <- names(Boston)[which(names(Boston) == "zn"):which(names(Boston) == "medv")]


# Loop through the variable names
for (i in vars) {
    # Use ggplot to create a box plot
    p <- ggplot(Boston, aes(x = factor(crim01), y = get(i))) +
        geom_boxplot() +
        labs(title = paste('Boxplot of', i, 'by crim01'),
             x = 'crim01 (Above/Below Median Crime rate)',
             y = i)

    # Print the plot
    print(p)
}
```


Boxplot of zn by crim01
zn
100
75
50
25
0
0
1
crim01 (Above/Below Median Crime rate)

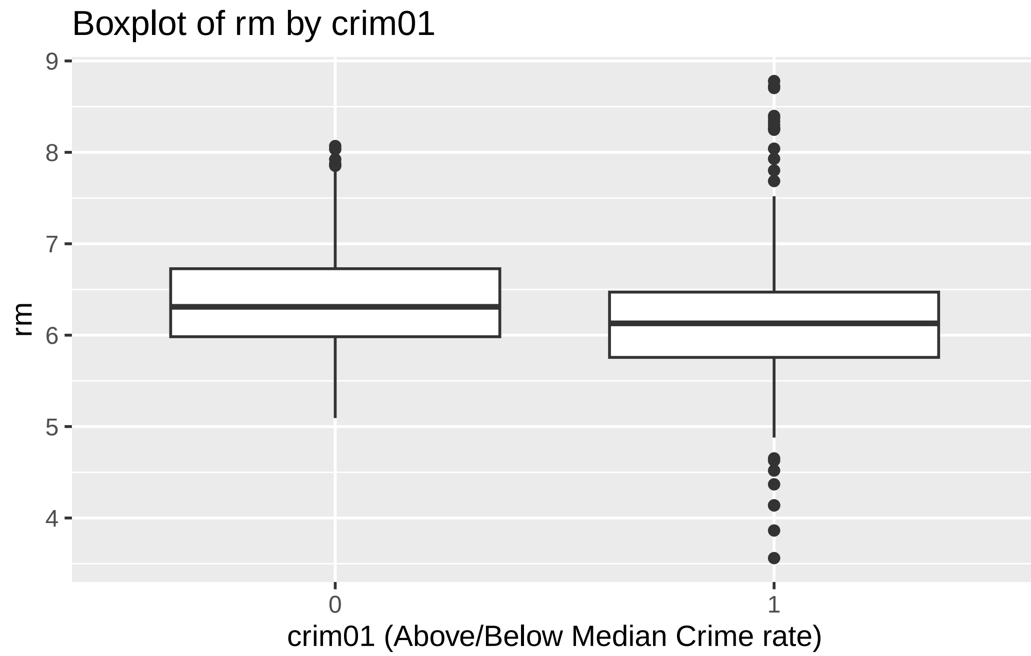

Boxplot of indus by crim01
indus
20
10
0
0
1
crim01 (Above/Below Median Crime rate)

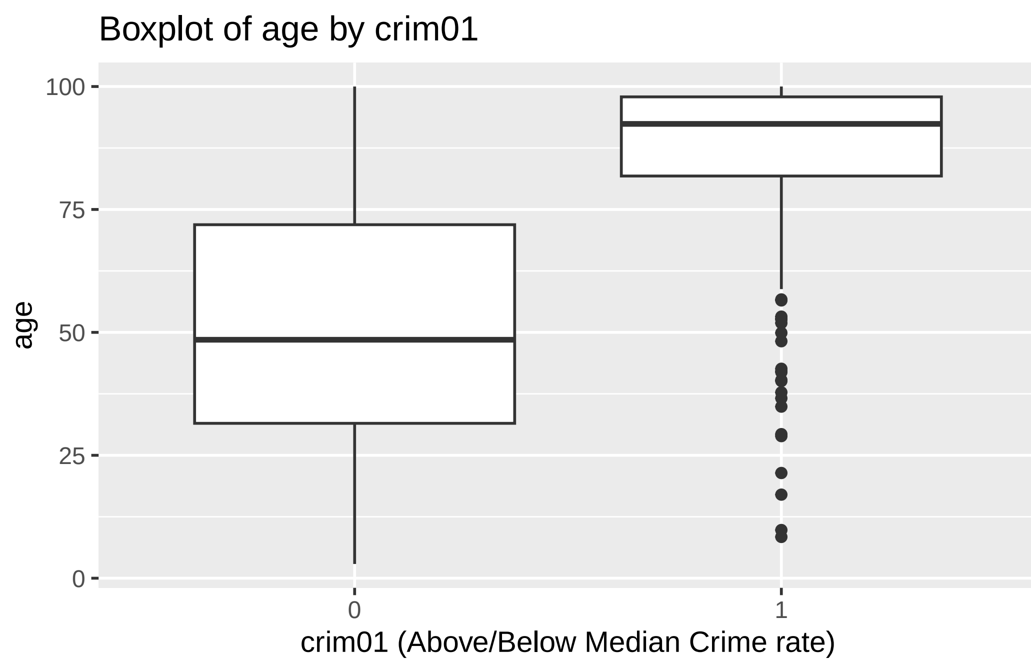


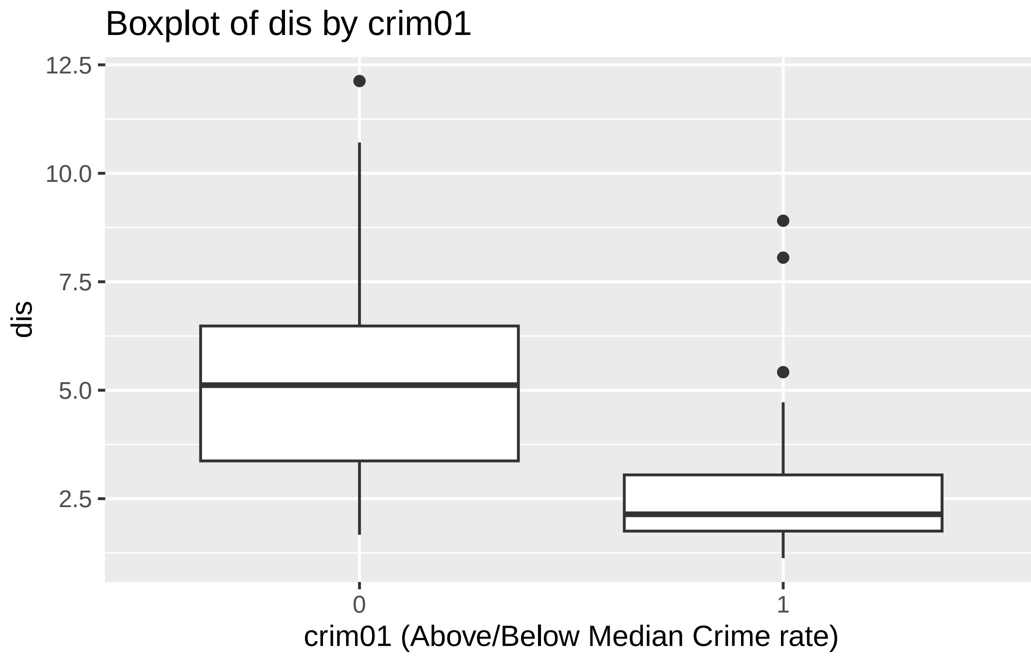
Boxplot of chas by crim01
1.00
0.75
0.50
0.25
0.00
chas
0
1
crim01 (Above/Below Median Crime rate)

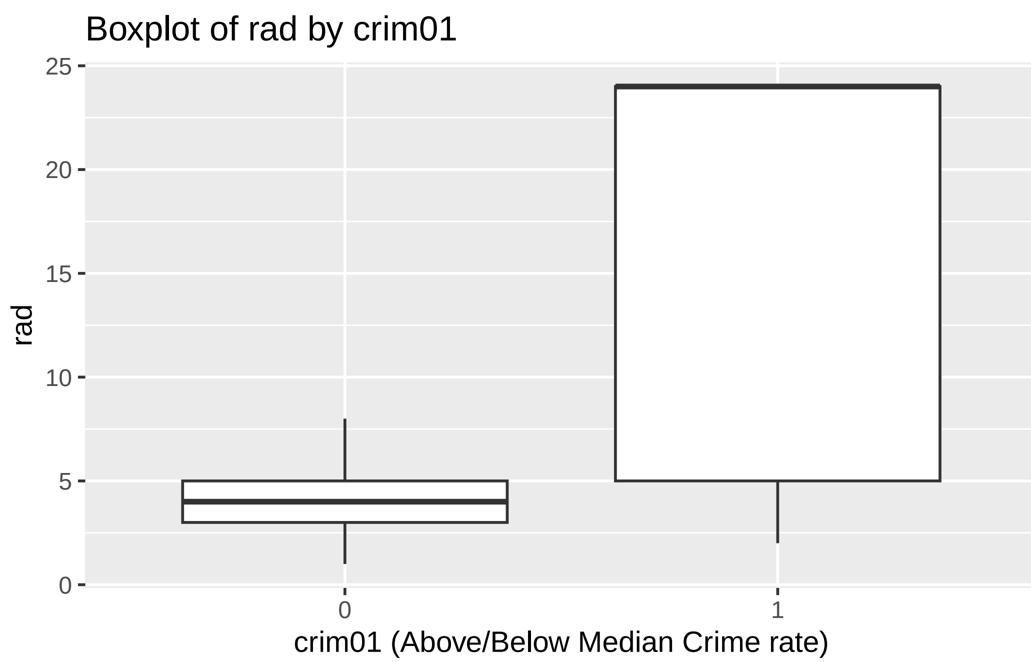


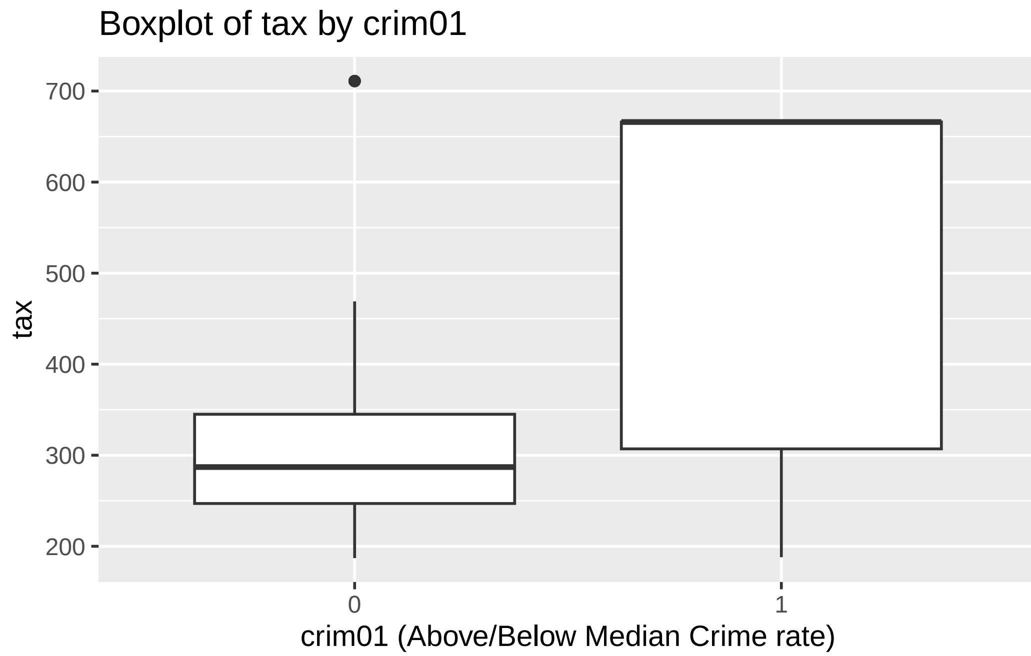
Boxplot of nox by crim01
0.8
0.7
0.6
0.5
0.4
nox
0
1
crim01 (Above/Below Median Crime rate)

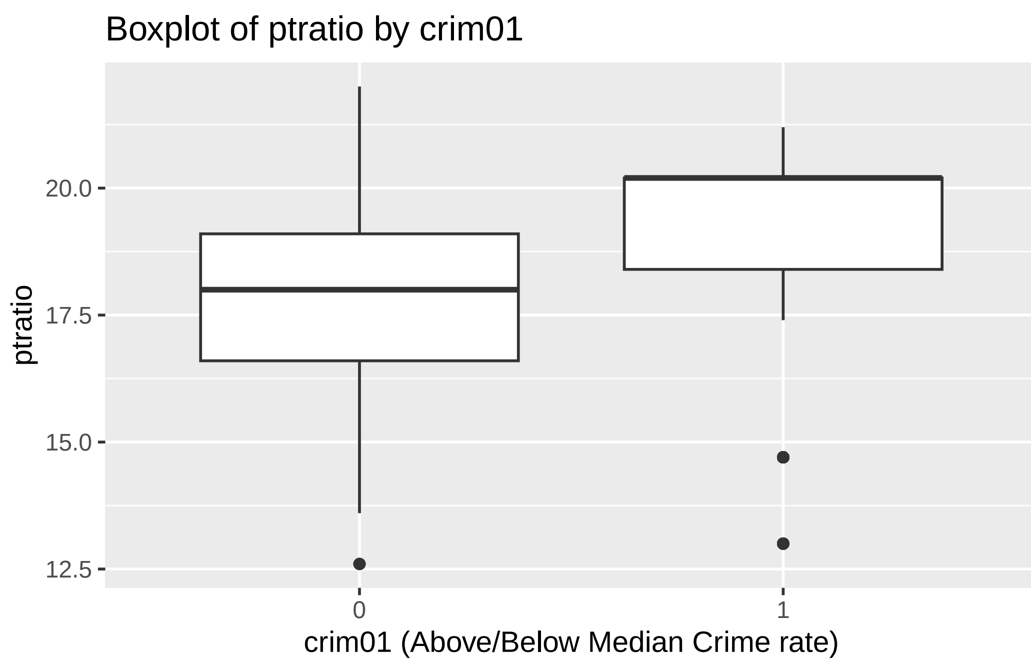

Boxplot of rm by crim01
rm
9
8
7
6
5
4
0
1
crim01 (Above/Below Median Crime rate)

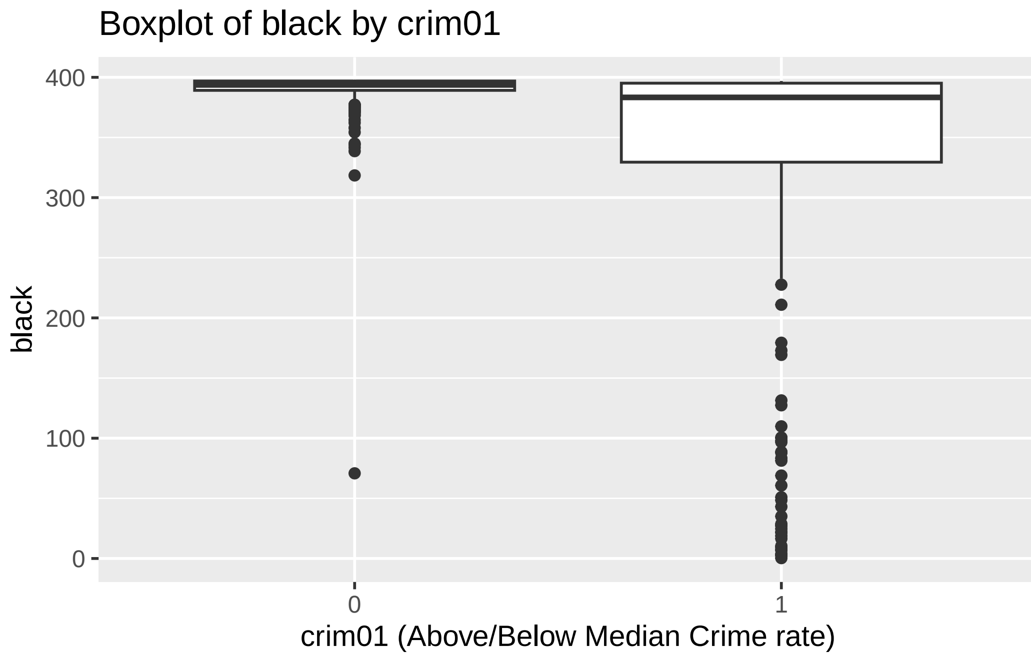

Boxplot of age by crim01
age
100
75
50
25
0
0
1
crim01 (Above/Below Median Crime rate)

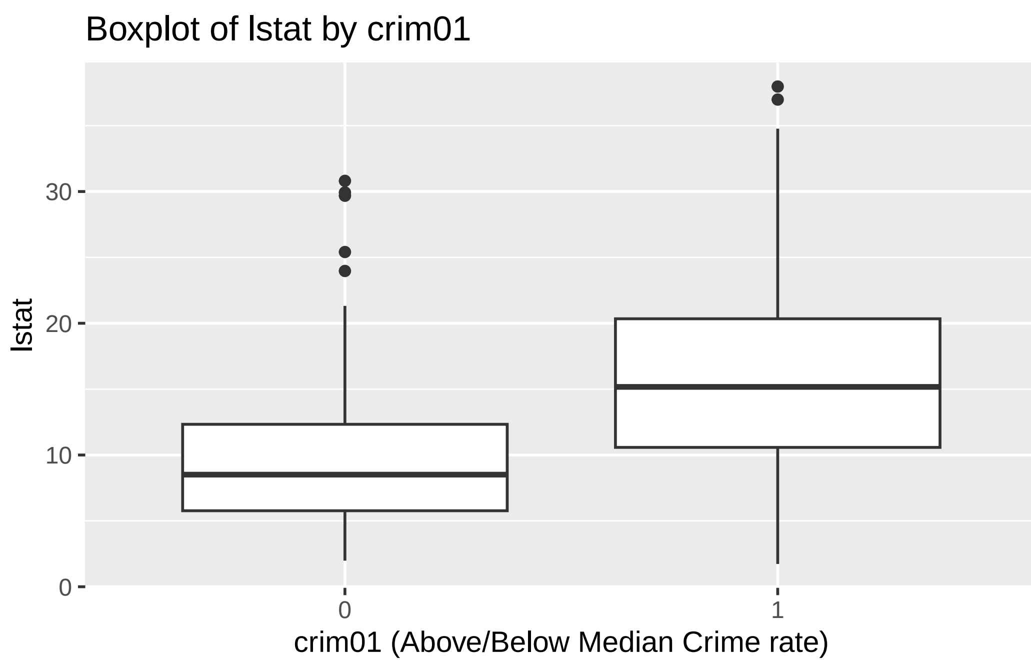

Boxplot of dis by crim01
12.5
10.0
7.5
5.0
2.5
dis
0
1
crim01 (Above/Below Median Crime rate)

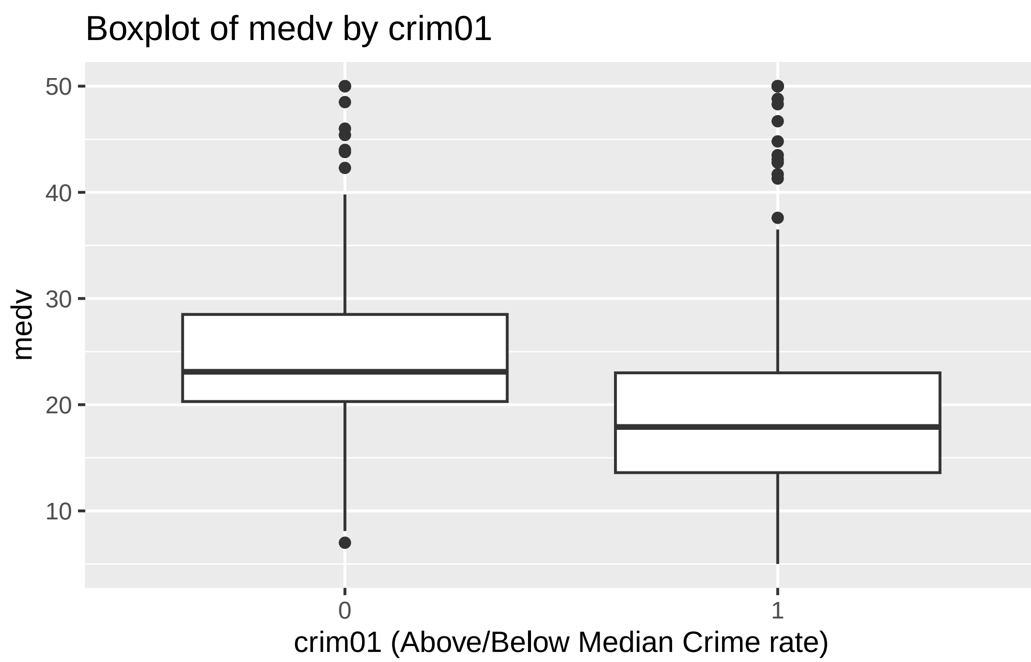

Boxplot of rad by crim01
25
20
15
10
5
0
rad
0
1
crim01 (Above/Below Median Crime rate)


Boxplot of tax by crim01
700
600
500
400
300
200
tax
0
1
crim01 (Above/Below Median Crime rate)


Boxplot of ptratio by crim01
20.0
17.5
15.0
12.5
ptratio
0
1
crim01 (Above/Below Median Crime rate)


Boxplot of black by crim01
400
300
200
100
0
black
0
1
crim01 (Above/Below Median Crime rate)


Boxplot of lstat by crim01
30
20
10
0
lstat
0
1
crim01 (Above/Below Median Crime rate)




```
#indus,nox,age,dis,rad
```

```
set.seed(1)
Boston_df <- Boston[, c('indus', 'nox', 'age','dis','rad','crim01')]
test <- sample(nrow(Boston_df), 200)

train.X <- Boston_df[-test, -6]
test.X <- Boston_df[test, -6]
train.df <- Boston_df[-test, ]

train.Y <- Boston_df[-test, 'crim01']
test.Y <- Boston_df[test, 'crim01']
```

#logistic

```
glm.fits <- glm(
    crim01 ~ indus + nox + age +dis + rad,
    data = train.df, family = binomial
)

glm.probs <- predict(glm.fits, test.X,
```

```r
                        type = "response")
```

```r
glm.pred <- rep("0", 200)
glm.pred[glm.probs > .5] <- "1"
table(glm.pred, test.Y)
```

```
        test.Y
glm.pred  0  1
       0 97 16
       1  9 78
```

```r
mean(glm.pred == test.Y)
```

```
[1] 0.875
```

#LDA

```r
lda.fit <- lda(crim01 ~ indus + nox + age +dis + rad,
               data = train.df)

lda.pred <- predict(lda.fit, test.X)
lda.class <- lda.pred$class
table(lda.class, test.Y)
```

```
         test.Y
lda.class  0  1
        0 95 23
        1 11 71
```

```r
mean(lda.class == test.Y)
```

```
[1] 0.83
```

#qda

```r
qda.fit <- qda(crim01 ~ indus + nox + age +dis + rad,
               data = train.df)

qda.pred <- predict(qda.fit, test.X)
qda.class <- qda.pred$class
table(qda.class, test.Y)
```

```
         test.Y
qda.class  0  1
        0 99 18
        1  7 76
```

```r
mean(qda.class == test.Y)
```

```
[1] 0.875
```

#naiveBayes

```r
nb.fit <- naiveBayes(crim01 ~ indus + nox + age +dis + rad,
                     data = train.df)
nb.fit
```

```

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        0         1
0.4803922 0.5196078

Conditional probabilities:
   indus
Y        [,1]     [,2]
  0  6.681088 5.289341
  1 15.132893 5.550083
```

```
   nox
Y         [,1]       [,2]
  0 0.4681619 0.05532716
  1 0.6397610 0.09878089

   age
Y        [,1]     [,2]
  0 51.71361 25.69922
  1 86.19434 17.05572

   dis
Y        [,1]     [,2]
  0 5.188369 2.099384
  1 2.456703 1.048184

   rad
Y         [,1]     [,2]
  0  4.163265 1.613474
  1 14.572327 9.565536
```

```
nb.class <- predict(nb.fit, test.X)
table(nb.class, test.Y)
```

```
        test.Y
nb.class  0  1
       0 92 20
       1 14 74
```