E1. Build a logistic regression model to predict the probability that a student will be in the honors class, based on information we know about the student: male, math = 65, reading = 70. What is the probability?

log(p/1-p) = Beta0 + Beta1*math + Beta2*read

log(p/1-p)=0.85949

p/1-p= exp(0.85949)

p = exp(0.85949)/(1+exp(0.85949)) = 0.7025541

Result: 0.7025541

E2. There are two students, A and B. A's math score is 10 points higher than B's. Build an appropriate model to answer: The odds of A getting in the honors class is _____ times more than the odds of B getting in the honors class.

A. 2.775

B. 3.775

C. 4.775

D. Insufficient information to answer.

E3. There are two students, Mary and John (note: gender is implied).

(a) If they have the same math score and we have no information about their reading and writing scores, who do you think has a higher chance of being in the honors class?

Result: Females has a higher chance than males.

(b) Suppose John's math score is 7 points higher than Mary's, which statement is right about their odds of being in the honors class?

A. The odds of John is about 20% higher than the odds of Mary.

B. The odds of Mary is about 20% higher than the odds of John.

C. There is insufficient information to tell.

---

5)

a) When the Bayes decision boundary is linear, QDA performs worse on the training set than LDA, since it fits a more flexible classifier than necessary and LDA performs better on both the train and test sets.

(b) This depends on the nature of the nonlinearity of the Bayes decision boundary. If the nonlinearity is quadratic (or close to that), then we would expect QDA to perform significantly better both on the training data and test data.

(c) As n increases, the flexibility of a model increases, too. So, QDA will perform better to fit and predict test data.

(d) False. This statement depends on the sample size. If the sample size is small, QDA as a more flexible method will overfit the data and the test error rate will be inferior.

---

6)

Y = Beta0 + Beta1 * x1 + Beta2 * x2

Y = log(p/1-p)

a) If x1 = 40h, x2 = 3.5 then the odds of getting A in the class will be =
-6 + 0.05*40 + 1*3.5 = -0.5
p = exp(-0.5)/(1+exp(-0.5)) = 0.3775407

b)If p = 0.5, then p/(1-p)=1, so: Beta0 + Beta1 * x1 + Beta2 * x2 = 0,

-6 + 0.05*x1 + 1*3.5=0, x1 = 50

---

8)
Logistic regression method will be preferred. This is because KNN method with k=1 has a high flexibility, high variance and low Bias. It means it has higher test error rates. As KNN's average error rate is 18%, its test error rate is 36% and higher than logistic regression method. So, we prefer logistic regression method in this case.

---

9)
a) p/1-p = 0.37, so p = 0.270073
b) p=0.16, so the individual's odds will be= p/1-p = 0.1904762

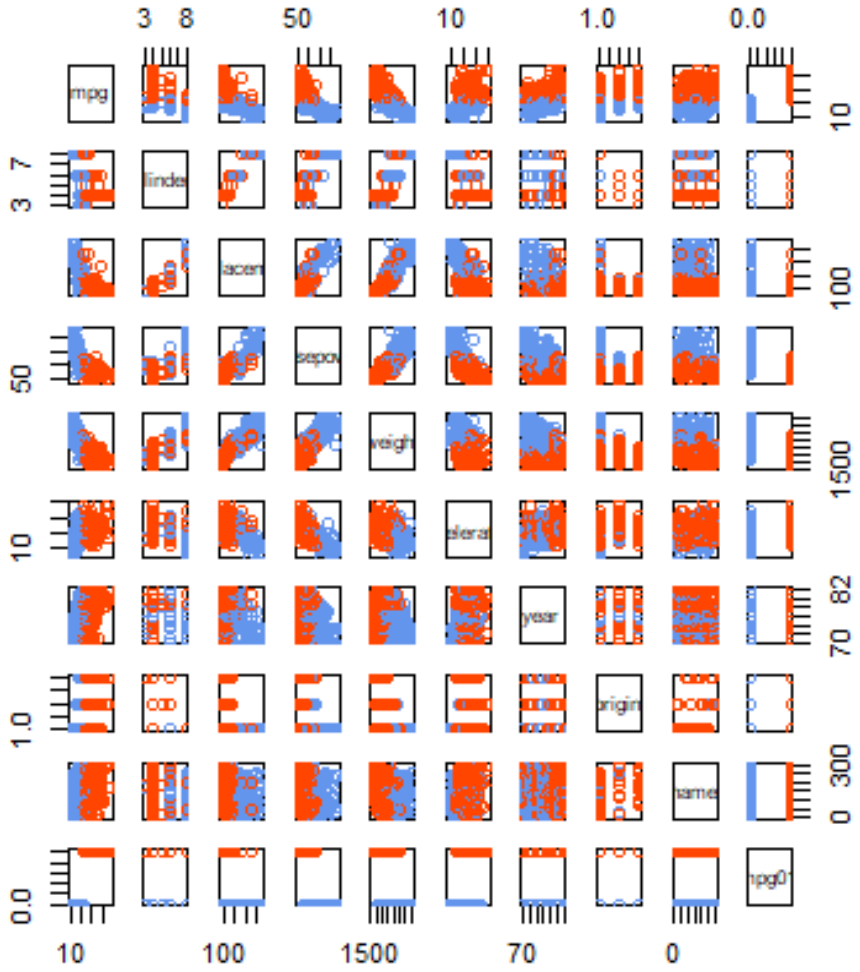
---

11)
a)

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

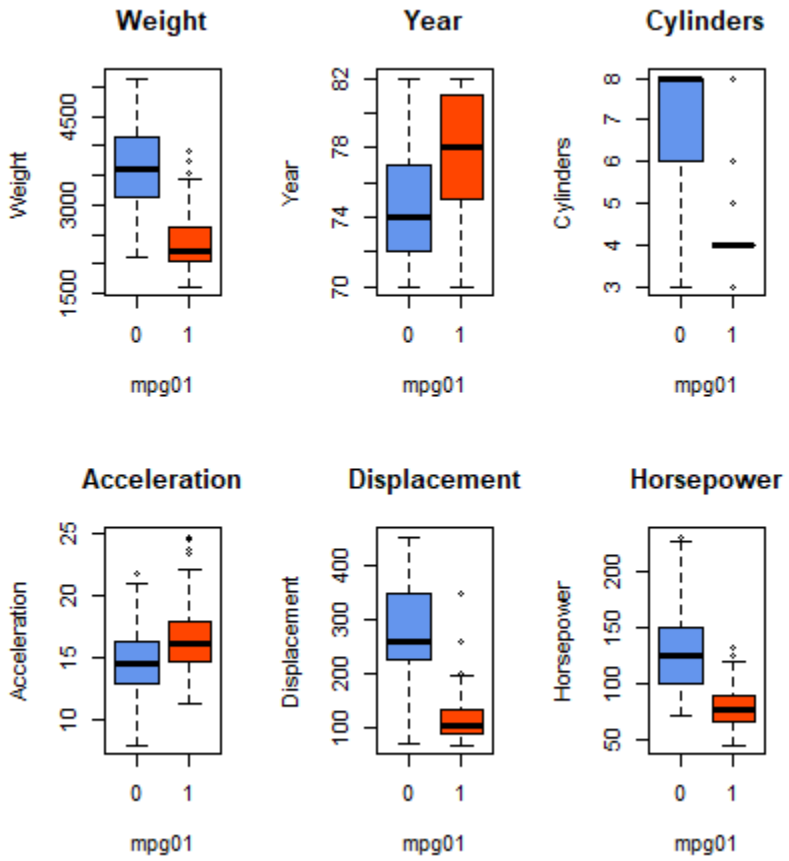
```
df <- data.frame(Auto)
colnames(df)
```
Output:
```
> colnames(df)
 [1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"
 [6] "acceleration" "year"         "origin"       "name"         "mpg01"
```

b)
```
>cor(df[,-9])
> cols <- character(nrow(df))
> cols[] <- "black"
> cols[df$mpg01 == 1] <- "orangered"
> cols[df$mpg01 == 0] <- "cornflowerblue"
```




```
>boxplot(weight ~ mpg01, data = df, main = "Weight",
        xlab = "mpg01", ylab = "Weight",
        col = c("cornflowerblue", "orangered"))

>boxplot(year ~ mpg01, data = df, main = "Year",
        xlab = "mpg01", ylab = "Year",
        col = c("cornflowerblue", "orangered"))
```

```
>boxplot(cylinders ~ mpg01, data = df, main = "Cylinders",
        xlab = "mpg01", ylab = "Cylinders",
        col = c("cornflowerblue", "orangered"))

>boxplot(acceleration ~ mpg01, data = df, main = "Acceleration",
        xlab = "mpg01", ylab = "Acceleration",
        col = c("cornflowerblue", "orangered"))

>boxplot(displacement ~ mpg01, data = df, main = "Displacement",
        xlab = "mpg01", ylab = "Displacement",
        col = c("cornflowerblue", "orangered"))

>boxplot(horsepower ~ mpg01, data = df, main = "Horsepower",
        xlab = "mpg01", ylab = "Horsepower",
        col = c("cornflowerblue", "orangered"))
```



Based on the above plots, `mpg01` has a negative association with `Weight`, `Cylinders`, `Displacement`, and `Horsepower`.

c)

```
n <- nrows(df)

train_indx <- sample(1:n, 0.8*n)

test_indx <- setdiff(1:n, train_indx)
```

d)

```
library(MASS)

ml <- lda(mpg01 ~ weight, data = df, subset = train_indx,
type="response")
```

summary(ml)

pred <- predict(ml, data = df[test_indx,])

output: the test error rate is 12.63736%

e)

mq <- qda(mpg01 ~ weight + horsepower + displacement + cylinders, data = df, subset = train_indx)

pred <- predict(mq, data = df[test_indx,])

summary(mq)

output: the test error rate is 13.18681%

f)

mg <- glm(mpg01 ~ weight + horsepower + displacement + cylinders, data = df, subset = train_indx)

summary(mg)

pred <- predict(mg, data = df[test_indx,])

output: the test error rate is 12.08791%

g)

library(class)

train.X <- cbind(cylinders, weight, displacement, horsepower)[train,]

test.X <- cbind(cylinders, weight, displacement, horsepower)[test,]

train.mpg01 <- mpg01[train]

set.seed(1)

```
# KNN (k=1)
knn.pred <-  knn(train.X, test.X, train.mpg01, k = 1)
mean(knn.pred != mpg01.test)
output: [1] 0.1538462
# KNN (k=10)
knn.pred <-  knn(train.X, test.X, train.mpg01, k = 10)
mean(knn.pred != mpg01.test)
output: [1] 0.1648352
# KNN (k=100)
knn.pred <-  knn(train.X, test.X, train.mpg01, k = 100)
mean(knn.pred != mpg01.test)
output: [1] 0.1428571
```

As we see in the above results, when k=100, the error rate is smaller than k=1 and k=10. So, the best choice is 100 nearest neighbors.