HW2_435

```
# Part 1a
(fit <- lm(mpg ~ cylinders + displacement + horsepower + weight + acceleration
          + year + origin, data = Auto))
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + year + origin, data = Auto)
##
## Coefficients:
##  (Intercept)     cylinders  displacement    horsepower        weight
##   -17.218435     -0.493376      0.019896     -0.016951     -0.006474
## acceleration          year        origin
##     0.080576      0.750773      1.426140
```

```
  # From the table, we can see the coefficients representing each predictors


# Part 1b
data <- data.frame(pred = predict(fit), actual = Auto$mpg)
mean((data$actual - data$pred)^2) # calculate MSE
```

```
## [1] 10.84748
```

```
# Part 1c
(mpg_jpn <- -17.218435 + (-0.493376*3) + (0.019896*100) + (-0.016951*85) +
          (-0.006474*3000) + (0.080576*20) + (0.750773*(1980%%100)))
```

```
## [1] 24.10156
```

```
# Part 1d
Auto %>% group_by(origin) %>% summarise(mean = mean(mpg))
```

```
## # A tibble: 3 x 2
##   origin  mean
##    <int> <dbl>
## 1      1  20.0
## 2      2  27.6
## 3      3  30.5
```

```
(diff <- 30.45063 - 20.03347)
```

```
## [1] 10.41716
```

```r
# Part 1e
(fit1 <- lm(mpg ~ horsepower, data = Auto))
```

```
##
## Call:
## lm(formula = mpg ~ horsepower, data = Auto)
##
## Coefficients:
## (Intercept)   horsepower
##     39.9359      -0.1578
```

```r
(change <- (-0.1578 * 10))
```

```
## [1] -1.578
```

1a. We can think of this model as each origin getting assigned a value that will impact the overall mpg. Therefore, we can also see that the main qualitative variable is origin.

1c. We predict a gas mileage of 24.10156.

1d. The difference between the mpg of a Japanese car and the mpg of an American car is 10.41716.

1e. holding all other covariates fixed, the change in mpg associated with a 10-unit change in horsepower is -1.578.

```r
# Part 2a (equation see picture)
Auto$origin_A <- ifelse(Auto$origin == "1", 1, 0)
Auto$origin_E <- ifelse(Auto$origin == "2", 1, 0)
Auto$origin_J <- ifelse(Auto$origin == "3", 1, 0)

(fit2 <- lm(mpg ~ origin_A + origin_E, data = Auto))
```

```
##
## Call:
## lm(formula = mpg ~ origin_A + origin_E, data = Auto)
##
## Coefficients:
## (Intercept)     origin_A     origin_E
##      30.451      -10.417       -2.848
```

```r
(mpg_J <- fit2$coefficients[1])
```

```
## (Intercept)
##    30.45063
```

```r
(mpg_A <- fit2$coefficients[1] + fit2$coefficients[2])
```

```
## (Intercept)
##    20.03347
```

```r
(mpg_E <- fit2$coefficients[1] + fit2$coefficients[3])
```

```
## (Intercept)
##    27.60294
```

```r
# Part 2b (equation see picture)
(fit3 <- lm(mpg ~ origin_E + origin_J, data = Auto))
```

```
##
## Call:
## lm(formula = mpg ~ origin_E + origin_J, data = Auto)
##
## Coefficients:
## (Intercept)     origin_E     origin_J
##      20.033        7.569       10.417
```

```r
(mpg_J1 <- fit3$coefficients[1] + fit3$coefficients[3])
```

```
## (Intercept)
##    30.45063
```

```r
(mpg_A1 <- fit3$coefficients[1])
```

```
## (Intercept)
##    20.03347
```

```r
(mpg_E1 <- fit3$coefficients[1] + fit3$coefficients[2])
```

```
## (Intercept)
##    27.60294
```

```r
# Part 2c (equation see picture)
Auto$origin_A <- ifelse(Auto$origin == "1", 1, -1)
Auto$origin_E <- ifelse(Auto$origin == "2", 1, -1)

(fit4 <- lm(mpg ~ origin_A + origin_E, data = Auto))
```

```
##
## Call:
## lm(formula = mpg ~ origin_A + origin_E, data = Auto)
##
## Coefficients:
## (Intercept)     origin_A     origin_E
##      23.818       -5.209       -1.424
```

```r
(mpg_J <- fit4$coefficients[1] - fit4$coefficients[2] - fit4$coefficients[3])
```

```
## (Intercept)
##    30.45063
```

```r
(mpg_A <- fit4$coefficients[1] + fit4$coefficients[2] - fit4$coefficients[3])
```

```
## (Intercept)
##    20.03347
```

```r
(mpg_E <- fit4$coefficients[1] - fit4$coefficients[2] + fit4$coefficients[3])
```

```
## (Intercept)
##    27.60294
```

```r
# Part 2d (equation see picture)
Auto$originD <- ifelse(Auto$origin == "1", 1, ifelse(Auto$origin == "2", 2, 0))

(fit5 <- lm(mpg ~ originD, data = Auto))
```

```
##
## Call:
## lm(formula = mpg ~ originD, data = Auto)
##
## Coefficients:
## (Intercept)      originD
##      25.239       -1.845
```

```r
(mpg_J <- fit5$coefficients[1])
```

```
## (Intercept)
##    25.23947
```

```r
(mpg_A <- fit5$coefficients[1] + fit5$coefficients[2])
```

```
## (Intercept)
##    23.39414
```

```r
(mpg_E <- fit5$coefficients[1] + (2*fit5$coefficients[2]))
```

```
## (Intercept)
##     21.5488
```

2a. The predicted mpg for a Japanese car is 30.45063. The predicted mpg for an American car is 20.03347. The predicted mpg for a European car is 27.60294

2b. The predicted mpg for a Japanese car is 30.45063. The predicted mpg for an American car is 20.03347. The predicted mpg for a European car is 27.60294

2c. The predicted mpg for a Japanese car is 30.45063. The predicted mpg for an American car is 20.03347. The predicted mpg for a European car is 27.60294

2d. The predicted mpg for a Japanese car is 25.23947. The predicted mpg for an American car is 23.39414. The predicted mpg for a European car is 21.5488

2e. As we can see the predicted mpg for the Japanese, American, and European cars are all the same for 2a-2c because the order does not matter and that using the value -1 or 1 also does not matter as it generates the same result. However, for 2d, we can see that the prediction changes because we are using a single variable, and also using 2 as a value is not suggested because it implies a worth, which then change the prediction.

2a) $y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i2} + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if } i^{th} \text{ car is American} \\ \beta_0 + \beta_2 + \epsilon_i & \text{if } i^{th} \text{ car is European} \\ \beta_0 + \epsilon_i & \text{if } i^{th} \text{ car is Japanese} \end{cases}$

2b) $y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i2} + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if } i^{th} \text{ car is American} \\ \beta_0 + \beta_2 + \epsilon_i & \text{if } i^{th} \text{ car is Japanese} \\ \beta_0 + \epsilon_i & \text{if } i^{th} \text{ car is American} \end{cases}$

2c) $y_i = \beta_0 + \beta_1 x_{i_1} + \beta_2 x_{i2} + \epsilon_i = \begin{cases} \beta_0 + \beta_1 - \beta_2 + \epsilon_i & \text{if } i^{th} \text{ car is American} \\ \beta_0 - \beta_1 + \beta_2 + \epsilon_i & \text{if } i^{th} \text{ car is European} \\ \beta_0 - \beta_1 - \beta_2 + \epsilon_i & \text{if } i^{th} \text{ car is Japanese} \end{cases}$

2d) $y_i = \beta_0 + \beta_1 x_{i_1} + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if } i^{th} \text{ car is American} \\ \beta_0 + 2\beta_1 + \epsilon_i & \text{if } i^{th} \text{ car is European} \\ \beta_0 + \epsilon_i & \text{if } i^{th} \text{ car is Japanese} \end{cases}$

```
# Part 3 (equation see picture)
(fit6 <- lm(mpg ~ origin * horsepower, data=Auto))
```

```
##
## Call:
## lm(formula = mpg ~ origin * horsepower, data = Auto)
##
## Coefficients:
##       (Intercept)             origin         horsepower  origin:horsepower
##          26.79098            7.87119            -0.05942            -0.06338
```

The mpg will change by 0.2496 for Japanese cars, 0.1228 for American and 0.18618 for European cars.

3) $y_i = \beta_0 + \beta_1 \times \text{horsepower}_i + \begin{cases} \beta_2 + \beta_3 \times \text{horsepower}_i & \text{if } i^{th} \text{ car is American} \\ 2\beta_2 + 2\beta_3 \times \text{horsepower}_i & \text{if } i^{th} \text{ car is European} \\ 3\beta_2 + 3\beta_3 \times \text{horsepower}_i & \text{if } i^{th} \text{ car is Japanese} \end{cases}$

```
# Part 4a
(predicted_weight <- -165.1 + (4.8*64))
```

```
## [1] 142.1
```

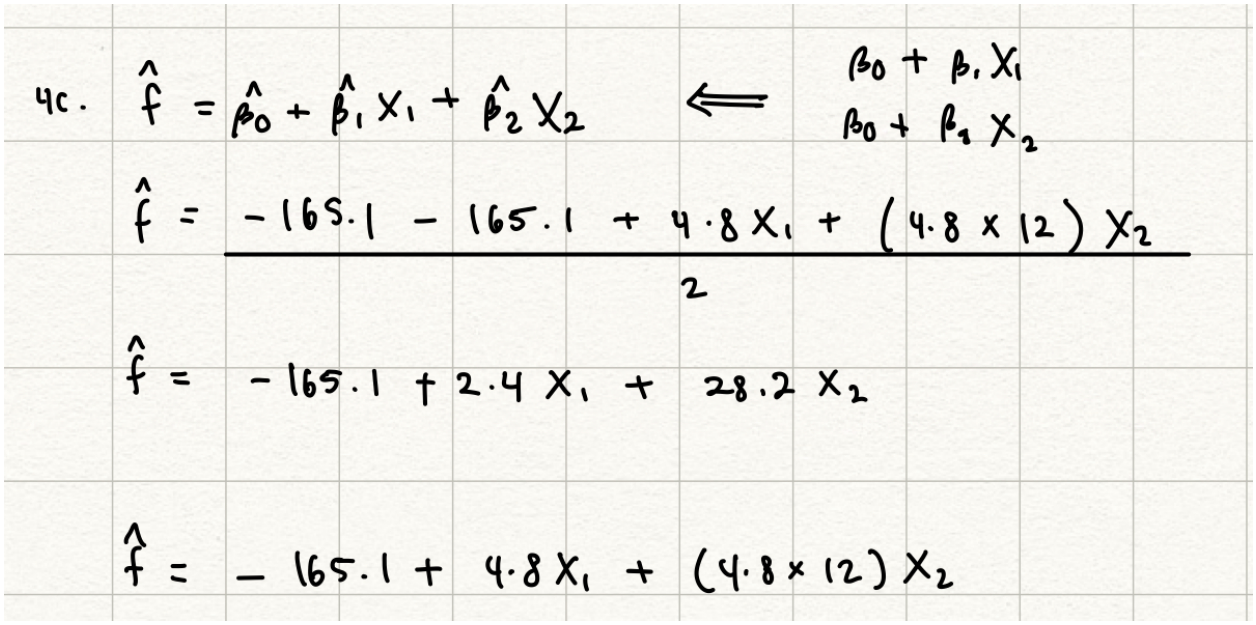
```
# Part 4b
(predicted_weight_ft <- -165.1 + (4.8*12*5.333333))
```

```
## [1] 142.1
```

4a. The weight we will predict is 142.1

4b. The coefficients 165.1 for Beta0, and 57.6 for Beta1. The weight we will predict is 142.1

4c.



4d. The mean squared errors for the three models should be the same because these three models will generate the same value because unit (feet/inches) does not matter.

5a.

Bayes:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B|A)\,P(A) + P(B|A^c)\,P(A^c)}$$

$$P(Y=1 \mid X=x) = \frac{P(X=x \mid Y=1)\cdot P(Y=1)}{P(X=x|Y=1)\cdot P(Y=1) + P(X=x \mid Y=2)\cdot P(Y=2)}$$

$$P(Y=2 \mid X=x) = \frac{P(X=x \mid Y=2)\cdot P(Y=2)}{P(X=x|Y=2)\cdot P(Y=2) + P(X=x \mid Y=1)\cdot P(Y=1)}$$

$$\frac{P(X=x \mid Y=1)\cdot P(Y=1)}{P(X=x|Y=1)\cdot P(Y=1) + P(X=x \mid Y=2)\cdot P(Y=2)} = \frac{P(X=x \mid Y=2)\cdot P(Y=2)}{P(X=x|Y=2)\cdot P(Y=2) + P(X=x \mid Y=1)\cdot P(Y=1)}$$

$$P(X=x \mid Y=1)\cdot P(Y=1) = P(X=x \mid Y=2)\cdot P(Y=2)$$

5b.

5b) $\mu = 0 \quad \sigma = 1 \quad \pi_1 = 0.45$ ← prior probability

uniform pdf = $\frac{1}{b-a}$

normal pdf = $\frac{1}{\sqrt{2\pi\sigma}} \cdot e^{-(x-\mu)^2/2\sigma^2}$

$$P(X=x \mid Y=1) \cdot P(Y=1) = P(X=x \mid Y=2) \cdot P(Y=2)$$

$$\frac{1}{\sqrt{2\pi\sigma}} \cdot e^{-(x-\mu)^2/2\sigma^2} \cdot (0.45) > \frac{1}{4} \cdot (1-0.45)$$

$$\frac{0.45}{\sqrt{2\pi}} \cdot e^{-x^2/2} > \frac{1}{4} \cdot 0.55$$

$$e^{-\frac{x^2}{2}} > \frac{0.55}{4} \cdot \frac{\sqrt{2\pi}}{0.45}$$

$$-\frac{x^2}{2} > \ln\left(\frac{0.55 \cdot \sqrt{2\pi}}{4 \cdot 0.45}\right)$$

$$\frac{x^2}{2} < -\ln\left(\frac{0.55 \cdot \sqrt{2\pi}}{4 \cdot 0.45}\right)$$

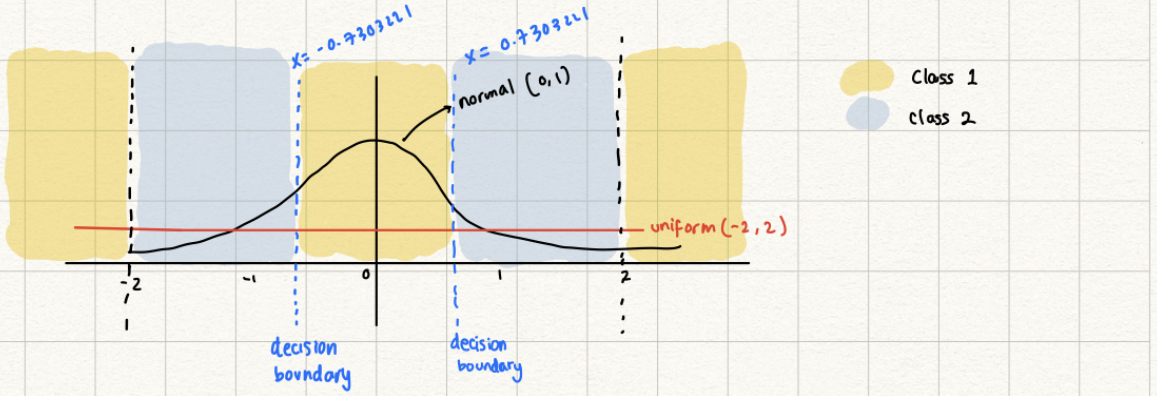
$$x^2 < -2\ln\left(\frac{0.55 \cdot \sqrt{2\pi}}{4 \cdot 0.45}\right)$$

$$x^2 < 0.5333703$$

$$x < -0.7303221 \text{ & } x > 0.7303221$$

$[-2, -0.7303221) \cup (0.7303221, 2] \Rightarrow$ assigned to class 2

$(-\infty, -2] \cup [-0.7303221, 0.7303221] \cup [2, \infty) \Rightarrow$ assigned to class 1



5c.

5c. n training observations $(x_1, y_1), \ldots, (x_n, y_n)$

explain how you use these observations to estimate μ, σ, π_1

estimate for μ : To estimate μ, we can filter out the observations for class 1, and then we can take the average of all the x's

estimate for σ : To estimate σ, we can filter out the observations for class 2, and we take the SD of the x's

estimate for π_1 : Since we want $P(y=1)$ and we know how many $y=1$ are in our observations, we can count them up and divide them by the total number of observations.

5d.

5d. Given test observation $X = x_0$. Provide an estimate $P(Y=1 \mid X = x_0)$

$$P(Y=1 \mid X=x_0) = \frac{P(X=x_0 \mid Y=1) \cdot P(Y=1)}{P(X=x_0 \mid Y=1) \cdot P(Y=1) + P(X=x_0 \mid Y=2) \cdot P(Y=2)}$$

$$= \frac{dnorm(\hat{\mu}, \hat{\sigma}) \cdot \hat{\pi}_1}{dnorm(\hat{\mu}, \hat{\sigma}) \cdot \hat{\pi}_1 + dunif \cdot (1 - \hat{\pi}_1)}$$

6a.

6a) observation $x = (x_1, \ldots, x_p)^T$

$$\log\left[\frac{p(x)}{1-p(x)}\right] = 0.7$$

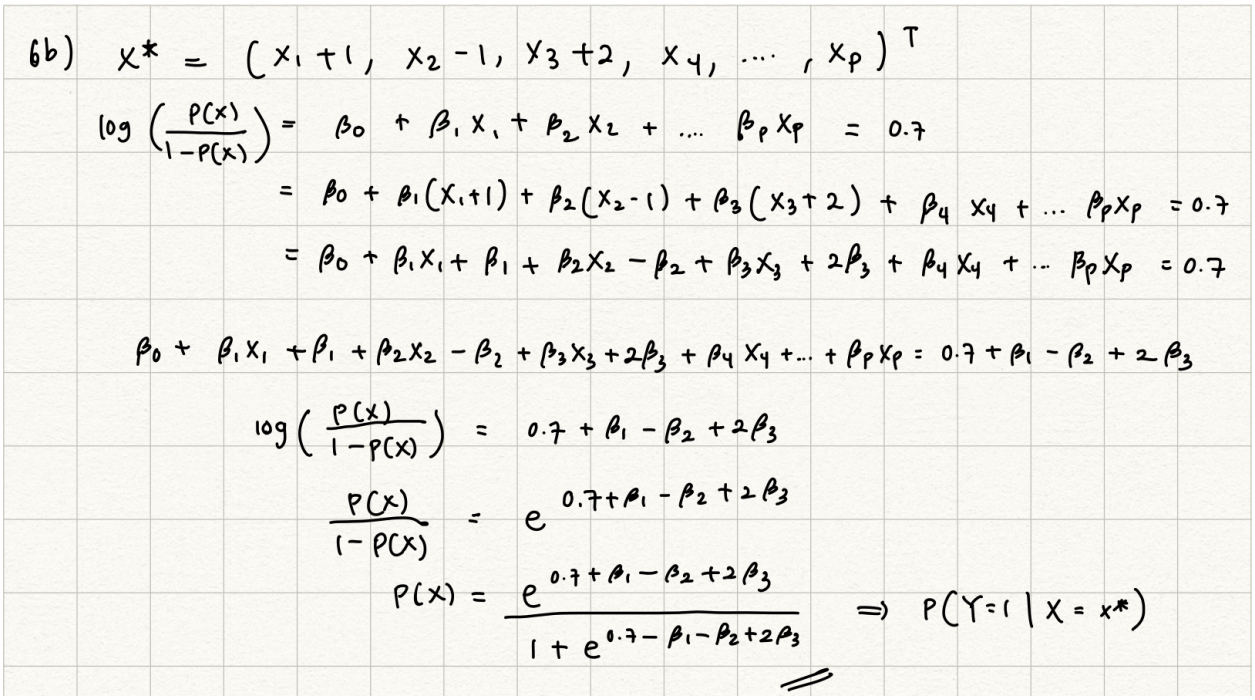
$$\frac{p(x)}{1-p(x)} = e^{0.7}$$

$$p(x) = e^{0.7} - e^{0.7} p(x)$$

$$p(x) = \frac{e^{0.7}}{1+e^{0.7}}$$

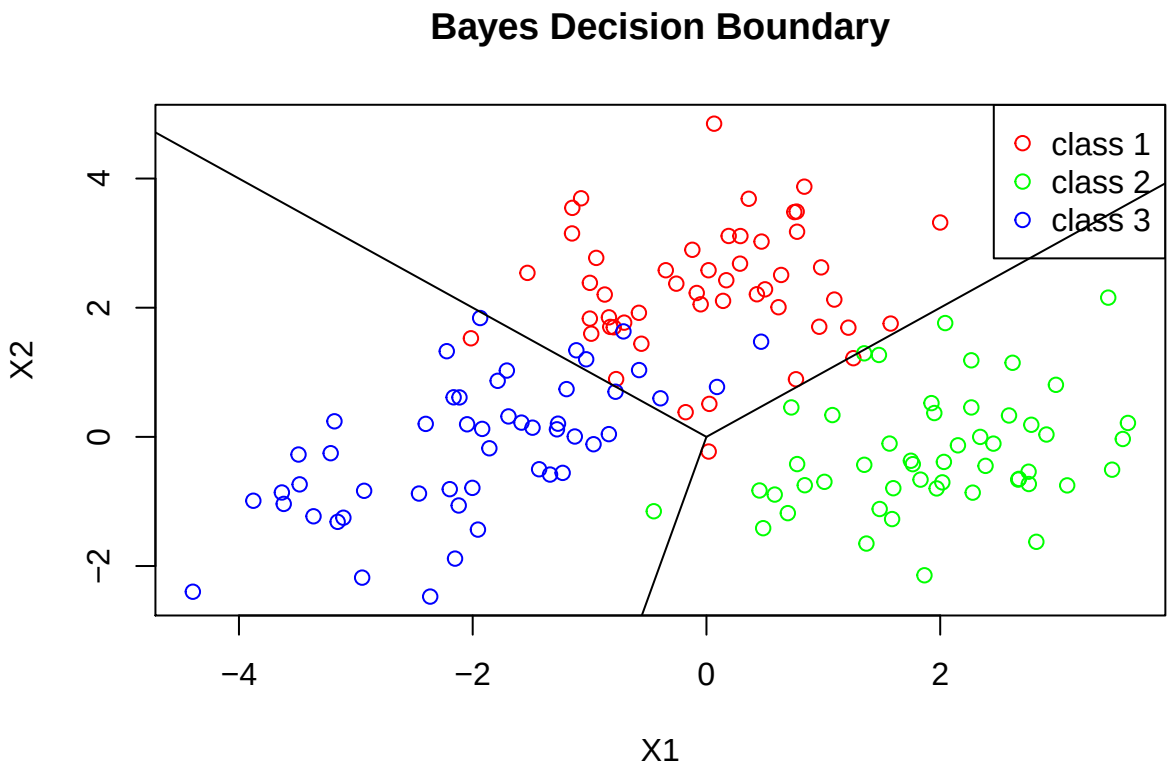
$$= 0.668778 \Rightarrow P(Y=1 \mid X=x)$$

6b.



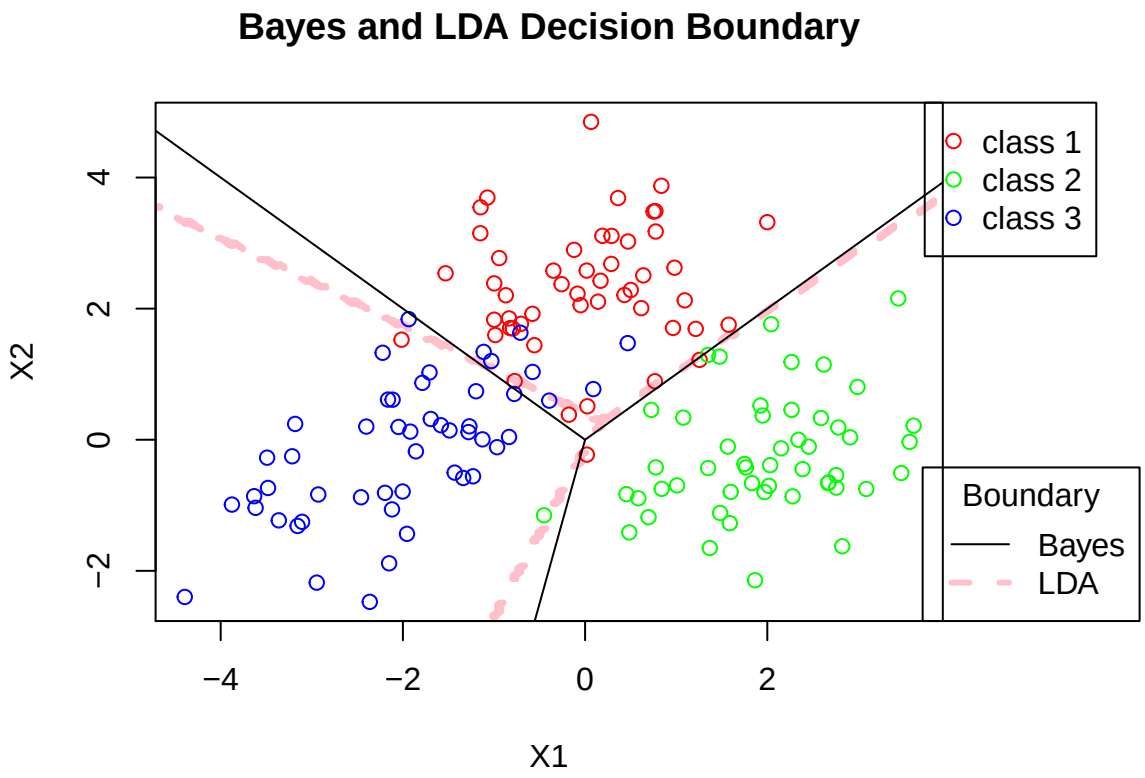

```r
# Part 7a
set.seed(435)
mu1 <- c(0, 2)
mu2 <- c(2, 0)
mu3 <- c(-2, 0)
cov <- matrix(c(1,0.2,0.2,1), nrow = 2, ncol = 2)
data1 <- mvrnorm(n = 50, mu = mu1, Sigma = cov)
data2 <- mvrnorm(n = 50, mu = mu2, Sigma = cov)
data3 <- mvrnorm(n = 50, mu = mu3, Sigma = cov)

train <- data.frame(x1 = c(data1[,1],data2[,1],data3[,1]),
                    x2 = c(data1[,2],data2[,2],data3[,2]),
                    class = c(rep(1,50),rep(2,50),rep(3,50)))

# Part 7b (see calculation in picture)
sigma_inv <- solve(cov)
mu1_trans <- t(mu1)
mu2_trans <- t(mu2)
mu3_trans <- t(mu3)

calc_1 <- mu1_trans %*% sigma_inv %*% mu1
calc_2 <- mu2_trans %*% sigma_inv %*% mu2
calc_3 <- mu3_trans %*% sigma_inv %*% mu3

calc_4 <- sigma_inv %*% mu1
calc_5 <- sigma_inv %*% mu2
calc_6 <- sigma_inv %*% mu3

plot(train$x1, train$x2, xlim = range(train$x1), ylim = range(train$x2),
     main="Bayes Decision Boundary",
     xlab = "X1", ylab ="X2", col = ifelse(train$class==1, "red",
```

```
                  ifelse(train$class==2, "green", "blue")))
lines(x=c(0,5),y=c(0,5))
lines(x=c(0,-5),y=c(0,5))
lines(x=c(0,-2),y=c(0,-10))

legend("topright", legend=c("class 1", "class 2", "class 3"),
       col=c("red","green","blue"), pch=c(1,1,1))
```




```
# Part 7c
grid <- seq(-5, 6, length=200)
grid_2d <- expand.grid(x1=grid, x2=grid)
lda.fit = lda(class ~ x1 + x2, data=train)

grid_pred_lda <- as.numeric(predict(lda.fit, newdata=grid_2d)$class)
grid_pred_lda <- matrix(grid_pred_lda, ncol=200)

par(mar=c(5, 4, 4, 8))
contour(grid, grid, grid_pred_lda,
       xlim=range(train$x1), ylim=range(train$x2),
       drawlabels=F, xlab="X1", ylab="X2",
       main="Bayes and LDA Decision Boundary", lty=2, lwd=3, col="pink")
points(train$x1, train$x2, xlim = range(train$x1), ylim = range(train$x2),
     col = ifelse(train$class==1, "red",
                  ifelse(train$class==2, "green", "blue")))
lines(x=c(0,5),y=c(0,5))
```

```
lines(x=c(0,-5),y=c(0,5))
lines(x=c(0,-2),y=c(0,-10))
par(xpd=TRUE)
legend("topright", legend=c("class 1", "class 2", "class 3"), inset=c(-0.195, 0),
       col=c("red","green","blue"), pch=c(1,1,1))
legend("bottomright", title="Boundary", legend=c("Bayes","LDA"),
       inset=c(-0.25, 0), col=c("black","pink"), lty=c(1,2), lwd=c(1,3))
```




```
# Part 7d
train_pred_lda <- as.numeric(predict(lda.fit, newdata=train)$class)
as.table(confusionMatrix(data = factor(train_pred_lda), reference =
                                 factor(train$class)))
```

```
##            Reference
## Prediction  1  2  3
##          1 45  0  7
##          2  2 49  0
##          3  3  1 43
```

```
(training_error <- mean(train$class != train_pred_lda))
```

```
## [1] 0.08666667
```

```r
# Part 7e
set.seed(342)
data1 <- mvrnorm(n = 50, mu = mu1, Sigma = cov)
data2 <- mvrnorm(n = 50, mu = mu2, Sigma = cov)
data3 <- mvrnorm(n = 50, mu = mu3, Sigma = cov)

test <- data.frame(x1 = c(data1[,1],data2[,1],data3[,1]),
                   x2 = c(data1[,2],data2[,2],data3[,2]),
                   class = c(rep(1,50),rep(2,50),rep(3,50)))

test_pred_lda <- as.numeric(predict(lda.fit, newdata=test)$class)
as.table(confusionMatrix(data = factor(test_pred_lda), reference =
                                 factor(test$class)))
```

```
##           Reference
## Prediction  1  2  3
##          1 39  1  5
##          2  5 48  2
##          3  6  1 43
```

```r
(test_error <- mean(test$class != test_pred_lda))
```

```
## [1] 0.1333333
```

7b.

7b. $x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k = x^T \Sigma^{-1} \mu_l - \frac{1}{2} \mu_l^T \Sigma^{-1} \mu_l$

① $x^T \Sigma^{-1} \mu_1 - \frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1 = x^T \Sigma^{-1} \mu_2 - \frac{1}{2} \mu_2^T \Sigma^{-1} \mu_2$

$x^T \Sigma^{-1} \mu_1 - \frac{1}{2} \cdot (4.166667) = x^T \Sigma^{-1} \mu_2 - \frac{1}{2} \cdot (4.166667)$

$-0.416667 x_1 + 2.08333 x_2 = 2.08333 x_1 - 0.416667 x_2$

$x_2 (2.08333 + 0.416667) = x_1 (2.08333 + 0.41667)$

$x_1 = x_2$

② $x^T \Sigma^{-1} \mu_1 - \frac{1}{2} \cdot \mu_1^T \Sigma^{-1} \mu_1 = x^T \Sigma^{-1} \mu_3 - \frac{1}{2} \mu_3^T \Sigma^{-1} \mu_3$

$x^T \Sigma^{-1} \mu_1 - \frac{1}{2} \cdot (4.166667) = x^T \Sigma^{-1} \mu_3 - \frac{1}{2} \cdot (4.1666 7)$

$-0.416667 x_1 + 2.08333 x_2 = -2.08333 x_1 + 0.4166667 x_2$

$(-0.4166667 + 2.08333) x_1 = (0.4166667 - 2.08333) x_2$

$x_1 = -x_2 \Rightarrow -x_1 = x_2$

③ $x^T \Sigma^{-1} \mu_2 - \frac{1}{2} \cdot (4.1667) = x^T \Sigma^{-1} \mu_3 - \frac{1}{2} \cdot (4.166667)$

$2.08333 x_1 - 0.4166667 x_2 = -2.08333 x_1 + 0.4166667 x_2$

$(2.08333 + 2.08333) x_1 = (0.4166667 + 0.4166667) x_2$

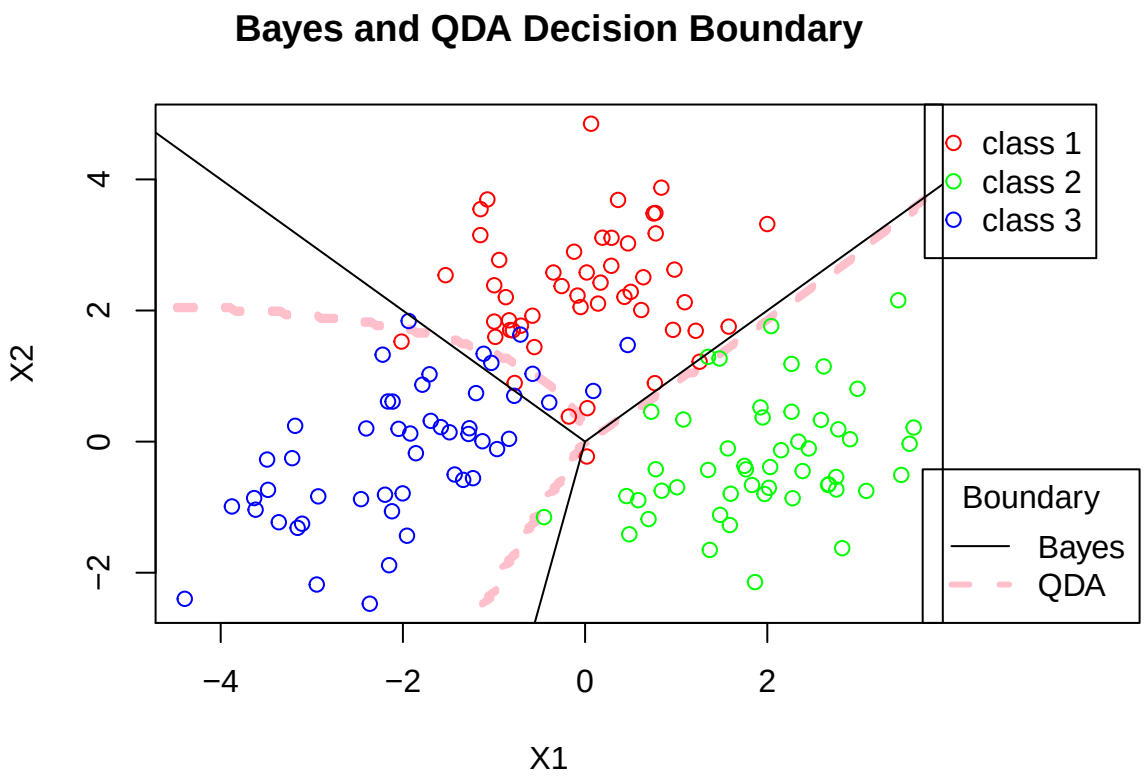
$5 x_1 = x_2$

7c. The Bayes Boundary and the LDA decision boundary is similar to each other. However, the Bayes Boundary follows the true distribution for each class, while the LDA decision boundary follows the training data.

7f. As we can see our training error is lower than our test error. This is because our model is based on training data instead of the test data. Therefore, it makes sense that the training error is lower.

```
# Part 8a
grid <- seq(-5, 6, length=200)
grid_2d <- expand.grid(x1=grid, x2=grid)
qda.fit = qda(class ~ x1 + x2, data=train)

grid_pred_qda <- as.numeric(predict(qda.fit, newdata=grid_2d)$class)
grid_pred_qda <- matrix(grid_pred_qda, ncol=200)

par(mar=c(5, 4, 4, 8))
contour(grid, grid, grid_pred_qda,
       xlim=range(train$x1), ylim=range(train$x2),
       drawlabels=F, xlab="X1", ylab="X2",
       main="Bayes and QDA Decision Boundary", lty=2, lwd=3, col="pink")
points(train$x1, train$x2, xlim = range(train$x1), ylim = range(train$x2),
     col = ifelse(train$class==1, "red",
                  ifelse(train$class==2, "green", "blue")))
```

```
lines(x=c(0,5),y=c(0,5))
lines(x=c(0,-5),y=c(0,5))
lines(x=c(0,-2),y=c(0,-10))
par(xpd=TRUE)
legend("topright", legend=c("class 1", "class 2", "class 3"), inset=c(-0.195, 0),
       col=c("red","green","blue"), pch=c(1,1,1))
legend("bottomright", title="Boundary", legend=c("Bayes","QDA"),
       inset=c(-0.25, 0), col=c("black","pink"), lty=c(1,2), lwd=c(1,3))
```




```
# Part 8b
train_pred_qda <- as.numeric(predict(qda.fit, newdata=train)$class)
as.table(confusionMatrix(data = factor(train_pred_qda), reference =
                                  factor(train$class)))
```

```
##            Reference
## Prediction  1  2  3
##          1 46  1  4
##          2  1 49  0
##          3  3  0 46
```

```
(training_error <- mean(train$class != train_pred_qda))
```

```
## [1] 0.06
```

```r
# Part 8c
test_pred_qda <- as.numeric(predict(qda.fit, newdata=test)$class)
as.table(confusionMatrix(data = factor(test_pred_qda), reference =
                                  factor(test$class)))
```

```
##            Reference
## Prediction  1  2  3
##          1 42  1  4
##          2  2 48  2
##          3  6  1 44
```

```r
(test_error <- mean(test$class != test_pred_qda))
```

```
## [1] 0.1066667
```

8a. As we can see the Bayes and the QDA boundary is also similar to each other. However, the QDA curves more making it able to create a non-linear boundary. On the other hand, the Bayes decision boundary just follows the true distribution of each classes.

8d. As we can see our training error is lower than our test error. This is because our model is based on training data instead of the test data. Therefore, it makes sense that the training error is lower.

8e. QDA has a smaller training error than LDA as it curves more, which makes it more flexible. LDA is a much less flexible classifier than QDA, and so has substantially lower variance.

8f. QDA also has a smaller test error because it is more flexible than LDA. This can potentially lead to improved prediction performance. Therefore, QDA has a better fit for the boundary.

9.