DATA 558 HW 2 Solutions

Contents

1 Problem 1

1.1 Part 1a)

```
# Load packages
library(ISLR2)

# Load in the data, change origin to a factor - note it is treated as numeric
# otherwise! We also change the labels to be more interpretable.
data(Auto)
Auto$origin = factor(Auto$origin, levels=c(1, 2, 3),
                     labels=c('American', 'European', 'Japanese'))

# Remove the name column, fit linear model on remaining variables.
Auto = subset(Auto, select=-c(name))
lin_mod = lm(mpg ~ ., data=Auto)
results = summary(lin_mod)$coefficients
knitr::kable(round(results, 2), caption='\\label{tbl:1a}Results of linear
            regression of mpg using the Auto data. The variable \\texttt{name}
            has been excluded.')
```

Table 1: Results of linear regression of mpg using the Auto data. The variable `name` has been excluded.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-17.95	4.68	-3.84	0.00
cylinders	-0.49	0.32	-1.52	0.13
displacement	0.02	0.01	3.13	0.00
horsepower	-0.02	0.01	-1.33	0.19
weight	-0.01	0.00	-10.24	0.00
acceleration	0.08	0.10	0.81	0.42
year	0.78	0.05	15.01	0.00
originEuropean	2.63	0.57	4.64	0.00
originJapanese	2.85	0.55	5.16	0.00

The results of the linear regression can be found in Table 1. Note that the predictor `origin` is categorical, so its coefficient names show the corresponding level. That is, the coefficients represent the expected change in `mpg` for cars of that level compared with American cars with all other predictors held constant. Based on p-values, there is insufficient evidence to conclude that `cylinders`, `horsepower`, or `acceleration` are correlated with `mpg`. There is strong evidence that the remaining predictors are associated with the outcome (`mpg`).

1.2 Part 1b)

```
mse = mean(lin_mod$residuals^2)
```

The training set MSE of the model is 10.68.

1.3 Part 1c)

```
# Note that year is coded as only the last two digits!
pred = predict(lin_mod, data.frame(cylinders=3, displacement=100, horsepower=85,
                                   weight=3000, acceleration=20, year=80,
                                   origin='Japanese'))
```

For the given values, the model predictes an `mpg` of 27.89 miles per gallon.

1.4 Part 1d)

```
jp_vs_am = lin_mod$coefficients['originJapanese']
```

The model predicts that holding all other variables constant, a car of Japanese origin will have on average 2.85 higher `mpg` than a car of American origin.

1.5 Part 1e)

```
res_1e = lin_mod$coefficients['horsepower']*10
```

The model predicts that holding all other variables constant, a 10-unit increase in `horsepower` will result on average in a change in `mpg` of -0.18 miles per gallon.

2 Problem 2

2.1 Part 2a)

```
# The base=3 argument makes the third group (Japanese) the baseline. The
# function contr.treatment tells R we want to use an encoding that uses one of
# the groups as a baseline (type ?contr.treatment for other options).
contrasts(Auto$origin) = contr.treatment(n=levels(Auto$origin), base=3)
jp.base = lm(mpg ~ origin, data=Auto)
coefs_2a = coef(jp.base)
res_2a = summary(jp.base)$coefficients
knitr::kable(round(res_2a, 2), caption='\\label{tbl:2a}Results of
             regressing \\texttt{mpg} on \\texttt{origin} using
             \\texttt{Japanese} as the baseline group.')
```

Table 2: Results of regressing `mpg` on `origin` using `Japanese` as the baseline group.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	30.45	0.72	42.31	0.00
originAmerican	-10.42	0.83	-12.59	0.00
originEuropean	-2.85	1.06	-2.69	0.01

```
# Now get fitted values and put in table format.
pred_2a = predict(jp.base, data.frame(origin=levels(Auto$origin)))
pred_2a = matrix(pred_2a, nrow = 1)
colnames(pred_2a) = levels(Auto$origin)
rownames(pred_2a) = c('Predicted \\texttt{mpg}')
knitr::kable(round(pred_2a, 2), caption='\\label{tbl:pred_2a}Predicted values
             for each possible \\texttt{origin}.')
```

Table 3: Predicted values for each possible `origin`.

	American	European	Japanese
Predicted `mpg`	20.03	27.6	30.45

Table 2 shows the results of the regression with `Japanese` as the baseline group. The model asssumes that

$$y_i = \begin{cases} \beta_0 + \epsilon_i & \text{if car } i \text{ is of Japanese origin} \\ \beta_0 + \beta_1 + \epsilon_i & \text{if car } i \text{ is of American origin} \\ \beta_0 + \beta_2 + \epsilon_i & \text{if car } i \text{ is of European origin} \end{cases}, \tag{1}$$

where the coefficient estimates are $\hat{\beta}_0 = 30.45$, $\hat{\beta}_1 = -10.42$, and $\hat{\beta}_2 = -2.85$. The predicted values $\hat{y}$ are shown in Table 3 for each possible level of `origin`.

2.2 Part 2b)

```r
# The base=1 argument makes the first group (American) the baseline.
contrasts(Auto$origin) = contr.treatment(n=levels(Auto$origin), base=1)
am.base = lm(mpg ~ origin, data=Auto)
coefs_2b = coef(am.base)
res_2b = summary(am.base)$coefficients
knitr::kable(round(res_2b, 2), caption='\\label{tbl:2b}Results of
             regressing \\texttt{mpg} on \\texttt{origin} using
             \\texttt{American} as the baseline group.')
```

Table 4: Results of regressing `mpg` on `origin` using `American` as the baseline group.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	20.03	0.41	49.02	0
originEuropean	7.57	0.88	8.63	0
originJapanese	10.42	0.83	12.59	0

```r
pred_2b = predict(am.base, data.frame(origin=levels(Auto$origin)))
pred_2b = matrix(pred_2b, nrow = 1)
colnames(pred_2b) = levels(Auto$origin)
rownames(pred_2b) = c('Predicted \\texttt{mpg}')
knitr::kable(round(pred_2b, 2), caption='\\label{tbl:pred_2b}Predicted values
             for each possible \\texttt{origin}.')
```

Table 5: Predicted values for each possible `origin`.

	American	European	Japanese
Predicted `mpg`	20.03	27.6	30.45

Table 4 shows the results of the regression with `American` as the baseline group. The model assumes that

$$y_i = \begin{cases} \beta_0 + \epsilon_i & \text{if car } i \text{ is of American origin} \\ \beta_0 + \beta_1 + \epsilon_i & \text{if car } i \text{ is of European origin} \\ \beta_0 + \beta_2 + \epsilon_i & \text{if car } i \text{ is of Japanese origin} \end{cases}, \tag{2}$$

where the coefficient estimates are $\hat{\beta}_0 = 20.03$, $\hat{\beta}_1 = 7.57$, and $\hat{\beta}_2 = 10.42$. The predicted values for each level are shown in Table 5.

2.3 Part 2c)

```r
# Here, we specify the contrasts manually.
contrasts(Auto$origin) = matrix(c(-1, 1, -1, -1, -1, 1), ncol=2)
plus.minus = lm(mpg ~ origin, data=Auto)
coefs_2c = coef(plus.minus)
res_2c = summary(plus.minus)$coefficients
knitr::kable(round(res_2c, 2), caption='\\label{tbl:2c}Results of
            regressing \\texttt{mpg} on \\texttt{origin} using +/- 1 encoding.')
```

Table 6: Results of regressing `mpg` on `origin` using +/- 1 encoding.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	29.03	0.53	54.87	0
origin1	3.78	0.44	8.63	0
origin2	5.21	0.41	12.59	0

```r
pred_2c = predict(plus.minus, data.frame(origin=levels(Auto$origin)))
pred_2c = matrix(pred_2c, nrow = 1)
colnames(pred_2c) = levels(Auto$origin)
rownames(pred_2c) = c('Predicted \\texttt{mpg}')
knitr::kable(round(pred_2c, 2), caption='\\label{tbl:pred_2c}Predicted values
            for each possible \\texttt{origin}.')
```

Table 7: Predicted values for each possible `origin`.

	American	European	Japanese
Predicted `mpg`	20.03	27.6	30.45

Table 6 shows the results of the regression with +/- 1 variable encoding. The model assumes that

$$y_i = \begin{cases} \beta_0 - \beta_1 - \beta_2 + \epsilon_i & \text{if car } i \text{ is of American origin} \\ \beta_0 + \beta_1 - \beta_2 + \epsilon_i & \text{if car } i \text{ is of European origin} \\ \beta_0 - \beta_1 + \beta_2 + \epsilon_i & \text{if car } i \text{ is of Japanese origin} \end{cases}, \tag{3}$$

where the coefficient estimates are $\hat{\beta}_0 = 29.03$, $\hat{\beta}_1 = 3.78$, and $\hat{\beta}_2 = 5.21$. The predicted values are shown in Table 7.

2.4 Part 2d)

```r
# Here, we specify the contrasts manually. How.many=1 constrains the contrasts
# to use a single variable.
contrasts(Auto$origin, how.many=1) = matrix(c(1, 2, 0), ncol=1)
one.var = lm(mpg ~ origin, data=Auto)
coefs_2d = coef(one.var)
res_2d = summary(one.var)$coefficients
knitr::kable(round(res_2d, 2), caption='\\label{tbl:2d}Results of
             regressing \\texttt{mpg} on \\texttt{origin} using one variable.')
```

Table 8: Results of regressing `mpg` on `origin` using one variable.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	25.24	0.73	34.42	0
origin1	-1.85	0.64	-2.89	0

```r
pred_2d = predict(one.var, data.frame(origin=c('American', 'European', 'Japanese')))
pred_2d = matrix(pred_2d, nrow = 1)
colnames(pred_2d) = c('American', 'European', 'Japanese')
rownames(pred_2d) = c('Predicted \\texttt{mpg}')
knitr::kable(round(pred_2d, 2), caption='\\label{tbl:pred_2d}Predicted values
             for each possible \\texttt{origin}.')
```

Table 9: Predicted values for each possible `origin`.

	American	European	Japanese
Predicted `mpg`	23.39	21.55	25.24

Table 8 shows the results of the regression with one variable used to encode the categories. The model assumes that

$$y_i = \begin{cases} \beta_0 + \epsilon_i & \text{if car } i \text{ is of Japanese origin} \\ \beta_0 + \beta_1 + \epsilon_i & \text{if car } i \text{ is of American origin} \\ \beta_0 + 2\beta_1 + \epsilon_i & \text{if car } i \text{ is of European origin} \end{cases}, \tag{4}$$

where the coefficient estimates are $\hat{\beta}_0 = 25.24$ and $\hat{\beta}_1 = -1.85$. The predicted values are shown in Table 9.

2.5 Part 2e)

We note that although the coefficient estimates change with each of the encodings, the predicted values in parts (a)-(c) are all the same. This shows that regardless of the variable encoding (contrasts) or baseline group used, our results are the same, so long as we encode the categories with two dummy variables (in general, for K categories, we need $K - 1$ dummy variables). However, the predicted values in part (d) are different. This is because here we have used a dummy encoding that utilizes only a single variable. As a result, we make the implicit assumption that the difference in `mpg` between American and Japanese cars is the same as the difference in mpg between European and American cars. This model predicts European cars to have lower `mpg` on average than American cars, but all of the other models predict the opposite! This highlights the problematic issues that can arise when treating categorical variables as quantitative.

3 Problem 3

```r
# First, let's get our contrasts back to default. We'll use American as the
# baseline group.
contrasts(Auto$origin) = contr.treatment(n=levels(Auto$origin), base = 1)

# Now, let's fit a linear model with an interaction term using * in the formula.
lin.mod.p3 = lm(mpg ~ origin*horsepower, data=Auto)
res.p3 = summary(lin.mod.p3)$coefficients
coefs.p3 = lin.mod.p3$coefficients
knitr::kable(round(res.p3, 2), caption='\\label{tbl:p3}Results of fitting a
             regression of mpg on origin and horsepower with an interaction
             term.')
```

Table 10: Results of fitting a regression of mpg on origin and horsepower with an interaction term.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	34.48	0.89	38.71	0
originEuropean	11.00	2.40	4.59	0
originJapanese	14.34	2.46	5.82	0
horsepower	-0.12	0.01	-17.10	0
originEuropean:horsepower	-0.10	0.03	-3.63	0
originJapanese:horsepower	-0.11	0.03	-3.75	0

Table 10 shows the results of the regression. The model assumes that

$$y_i = \begin{cases} \beta_0 + \beta_3 \times \texttt{horsepower}_i + \epsilon_i & \text{if car } i \text{ is of American origin} \\ (\beta_0 + \beta_1) + (\beta_3 + \beta_4) \times \texttt{horsepower}_i + \epsilon_i & \text{if car } i \text{ is of European origin ,} \\ (\beta_0 + \beta_2) + (\beta_3 + \beta_5) \times \texttt{horsepower}_i + \epsilon_i & \text{if car } i \text{ is of Japanese origin} \end{cases} \tag{5}$$

where the coefficient estimates are $\hat{\beta}_0 = 34.48$, $\hat{\beta}_1 = 11$, $\hat{\beta}_2 = 14.34$, $\hat{\beta}_3 = -0.12$, $\hat{\beta}_4 = -0.1$, and $\hat{\beta}_5 = -0.11$. The `origin` variable allows each category to have a separate intercept, while the `origin:horsepower` interaction terms allow each category to have a separate slope for `horsepower` as well.

For a Japanese car, an increase in 1 units of `horsepower` is estimated to result in an average change in `mpg` of $\beta_3 + \beta_5 = -0.23$. For American cars, this value is $\beta_3 = -0.12$. For European cars, this value is $\beta_3 + \beta_4 = -0.22$.

4 Problem 4

4.1 Part 4a)

```r
beta0.4a = -165.1
beta1.4a = 4.8
height.4a = 64
pred.4a = beta0.4a + beta1.4a * height.4a
```

We will predict $\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 \times X_1 = -165.1 + 4.8 \times 64 = 142.1$.

4.2 Part 4b)

```
beta0.4b = beta0.4a
beta1.4b = beta1.4a * 12
height.4b = height.4a / 12
pred.4b = beta0.4b + beta1.4b * height.4b
```

Using equation (3.4) from the textbook, we know that

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_{i1} - \bar{x}_1)(y_i - \bar{y})}{\sum_{i=1}^n (x_{i1} - \bar{x}_1)^2}. \tag{6}$$

Thus, for $\hat{\beta}_1^\star$, using the fact that $x_{i1} = 12x_{i2} \Rightarrow x_{i2} = \frac{1}{12}x_{i1}$ for each i, we have that

$$\begin{aligned}
\hat{\beta}_1^\star &= \frac{\sum_{i=1}^n (x_{i2} - \bar{x}_2)(y_i - \bar{y})}{\sum_{i=1}^n (x_{i2} - \bar{x}_2)^2} && (7)\\
&= \frac{\sum_{i=1}^n (\frac{1}{12}x_{i1} - \frac{1}{12}\bar{x}_1)(y_i - \bar{y})}{\sum_{i=1}^n (\frac{1}{12}x_{i1} - \frac{1}{12}\bar{x}_1)^2} && (8)\\
&= \frac{\frac{1}{12}\sum_{i=1}^n (x_{i1} - \bar{x}_1)(y_i - \bar{y})}{\left(\frac{1}{12}\right)^2 \sum_{i=1}^n (x_{i1} - \bar{x}_1)^2} && (9)\\
&= 12 \times \frac{\sum_{i=1}^n (x_{i1} - \bar{x}_1)(y_i - \bar{y})}{\sum_{i=1}^n (x_{i1} - \bar{x}_1)^2} && (10)\\
&= 12\hat{\beta}_1. && (11)
\end{aligned}$$

So, the slope is twelve times larger, which makes sense because our new units make the predictor 12 times smaller.

Meanwhile, for the intercept we have

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1\bar{x}_1. \tag{12}$$

Thus, for $\hat{\beta}_0^\star$, we have

$$\begin{aligned}
\hat{\beta}_0^\star &= \bar{y} - \hat{\beta}_1^\star \bar{x}_2 && (13)\\
&= \bar{y} - (12\hat{\beta}_1) \times \left(\frac{1}{12}\bar{x}_1\right) && (14)\\
&= \bar{y} - \hat{\beta}_1\bar{x}_1 && (15)\\
&= \hat{\beta}_0. && (16)
\end{aligned}$$

We see that the intercept is unchanged.

We will predict $\hat{Y} = \hat{\beta}_0^\star + \hat{\beta}_1^\star \times X_2 = -165.1 + 57.6 \times 5.33 = 142.1$.

4.3 Part 4c)

In order to set this model apart from the one in part (a), we will use the notation

$$Y = \beta_0' + \beta_1' X_1 + \beta_2' X_2 + \epsilon. \tag{17}$$

Note that because $X_1 = 12X_2$, this is equivalent to

$$\begin{aligned}
Y &= \beta_0' + \beta_1' X_1 + \frac{1}{12}\beta_2' X_1 + \epsilon && (18)\\
&= \beta_0' + \left(\beta_1' + \frac{1}{12}\beta_2'\right) X_1 + \epsilon && (19)\\
&:= \beta_0' + \beta_3' X_1 + \epsilon, && (20)
\end{aligned}$$

where we have defined $\beta_3' = \beta_1' + \frac{1}{12}\beta_2'$. From part (a), we already know that the estimates of (β_0', β_3') that minimize the sum of the squared residuals are $\hat{\beta}_0' = \hat{\beta}_0 = -165.1$ and $\hat{\beta}_3{}' = \hat{\beta}_1 = 4.8$. So, any $\hat{\beta}_0'$, $\hat{\beta}_1'$, and $\hat{\beta}_2'$ that satisfy the equations

$$\hat{\beta}_0' = \hat{\beta}_0 = -165.1 \tag{21}$$

$$\hat{\beta}_1' + \frac{1}{12}\hat{\beta}_2' = \hat{\beta}_1 = 4.8, \tag{22}$$

will be least squares estimators of the coefficients β_0', β_1', β_2'. The predicted values $\hat{Y}$ will be the same as in part (a).

4.4 Part 4d)

All three models (a)-(c) will have the same fitted values, hence they will have the same MSE.

5 Problem 5

5.1 Part 5a)

By Bayes' rule and the fact that $X|Y=1 \sim \mathrm{N}(\mu, \sigma^2)$, we have

$$\mathbb{P}(Y=1|X=x) = \frac{f_1(x|Y=1)\mathbb{P}(Y=1)}{f(x)} \tag{23}$$

$$= \frac{\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\pi_1}{f(x)}, \tag{24}$$

where f_1 is the probability density function (pdf) of $X|Y=1$, π_1 denotes $\mathbb{P}(Y=1)$, and f is the marginal pdf of X. Similarly, using the fact that $X|Y=2 \sim \mathrm{Unif}(-2,2)$, we have

$$\mathbb{P}(Y=2|X=x) = \frac{f_2(x|Y=2)\mathbb{P}(Y=2)}{f(x)} \tag{25}$$

$$= \frac{\frac{1}{4}\mathbb{1}_{\{-2\le x\le 2\}}\pi_2}{f(x)}, \tag{26}$$

where f_2 is the pdf of $X|Y=2$, $\pi_2 = 1-\pi_1$ denotes $\mathbb{P}(Y=2)$, and f is again the marginal pdf of X. Here, $\mathbb{1}$ denotes the *indicator function*, that is,

$$\mathbb{1}_{\{-2\le x\le 2\}} = \begin{cases} 1 & -2 \le x \le 2 \\ 0 & \text{otherwise} \end{cases}. \tag{27}$$

This comes from the fact that $X|Y=2$ is constrained to be between -2 and 2, so if this condition is not met, the probability density of $X|Y=2$ is 0.

First, we assume for simplicity that $-2 \le x \le 2$. Note that if this is not true, the Bayes classifier will always assign the label $Y=1$, because $f_2(x|Y=2)=0$ in this case. Then, combining these results, we have the decision boundary

$$\mathbb{P}(Y=1|X=x) = \mathbb{P}(Y=2|X=x) \tag{28}$$

$$\frac{\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\pi_1}{f(x)} = \frac{\frac{1}{4}\pi_2}{f(x)} \tag{29}$$

$$\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = \frac{\sqrt{2\pi}\sigma}{4}\times\frac{1-\pi_1}{\pi_1} \tag{30}$$

$$-\frac{(x-\mu)^2}{2\sigma^2} = \log\left(\frac{\sqrt{\pi}\sigma}{2\sqrt{2}}\right) + \log(1-\pi_1) - \log(\pi_1) \tag{31}$$

$$(x-\mu)^2 = 2\sigma^2\left\{[\log(\pi_1) - \log(1-\pi_1)] + \log\left(\frac{2\sqrt{2}}{\sqrt{\pi}\sigma}\right)\right\} \tag{32}$$

$$x = \mu \pm \sqrt{2}\sigma\sqrt{[\log(\pi_1) - \log(1-\pi_1)] - \log(\sigma) + \log\left(\frac{2\sqrt{2}}{\sqrt{\pi}}\right)}. \tag{33}$$

Here, by taking the square root, we have implicitly assumed that $[\log(\pi_1) - \log(1-\pi_1)] + \log\left(\frac{2\sqrt{2}}{\sqrt{\pi}\sigma}\right) \geq 0$. If this is the case, then in total we have four decision boundaries: $x = 2$, $x = -2$, $x = \mu + \sqrt{2}\sigma\sqrt{[\log(\pi_1) - \log(1-\pi_1)] - \log(\sigma) + \log\left(\frac{2\sqrt{2}}{\sqrt{\pi}}\right)}$, and $x = \mu - \sqrt{2}\sigma\sqrt{[\log(\pi_1) - \log(1-\pi_1)] - \log(\sigma) + \log\left(\frac{2\sqrt{2}}{\sqrt{\pi}}\right)}$, unless one or both of the corresponding solutions from (33) are outside the range $(-2,2)$. If this is not the case, then equation (28) has no solution, and the only Bayes decision boundaries are $x = \pm 2$, which correspond to the limits of the uniform distribution.

5.2 Part 5b)

```
mu = 0
sigma = 1
pi.1 = 0.45
bound = sqrt(2*(log(pi.1) - log(1-pi.1) - log(sigma) + log(2*sqrt(2)/sqrt(pi))))
density.center = dnorm(x=0, mean=mu, sd=sigma)
density.outer = dnorm(x=1, mean=mu, sd=sigma)
```

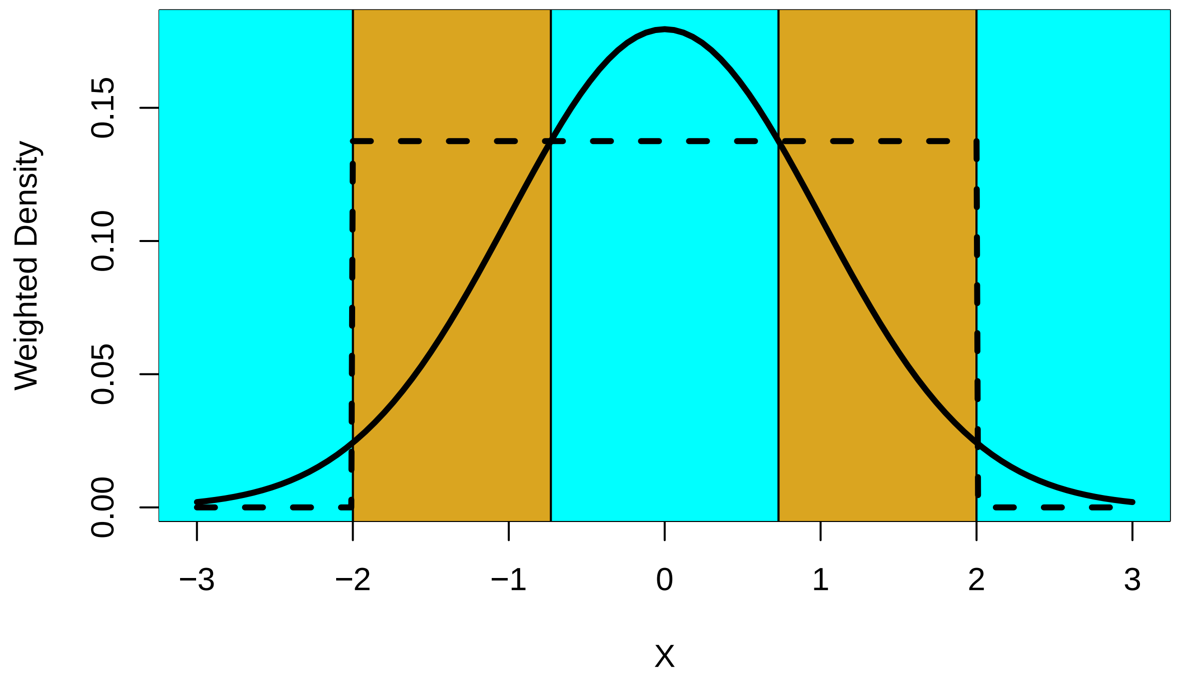
With the given values, the bounds in (33) become

$$x = \pm\sqrt{2[\log(0.45) - \log(0.55)] - 2\log(1) + 2\log\left(\frac{2\sqrt{2}}{\sqrt{\pi}}\right)} \tag{34}$$

$$= \pm 0.73. \tag{35}$$

Now, we can draw a simple plot to determine the decision regions:

```
curve(dnorm(x)*pi.1, from=-3, to=3, xlab='X', ylab='Weighted Density')
rect(-bound, -1, bound, 0.5, col='cyan', xpd=FALSE)
rect(-2, -1, -bound, 0.5, col='goldenrod', xpd=FALSE)
rect(-4, -1, -2, 0.5, col='cyan', xpd=FALSE)
rect(bound, -1, 2, 0.5, col='goldenrod', xpd=FALSE)
rect(2, -1, 4, 0.5, col='cyan', xpd=FALSE)
curve(dnorm(x, mean=mu, sd=sigma)*pi.1, from=-3, to=3, lty=1, lwd=3, add=TRUE)
curve(dunif(x, min=-2, max=2)*(1-pi.1), from=-2, to=2, lty=2, lwd=3, add=TRUE)
curve(dunif(x, min=-2, max=2)*(1-pi.1), from=-3, to=-2, lty=2, lwd=3, add=TRUE)
curve(dunif(x, min=-2, max=2)*(1-pi.1), from=2, to=3, lty=2, lwd=3, add=TRUE)
```



Here, the solid curve indicates the density $f_1(x|Y = 1)$ weighted by the prior probability π_1, which is proportional to the posterior probability $\mathbb{P}(Y = 1|X = x)$. Similarly, the dashed curve represents the density $f_2(x|Y = 2)$ weighted by the prior probability $\pi_2 = 1 - \pi_1$, which is proportional to the posterior probability $\mathbb{P}(Y = 2|X = x)$. Note that

$$\begin{aligned} \mathbb{P}(Y = 1|X = x) > \mathbb{P}(Y = 2|X = x) &\Leftrightarrow \frac{f_1(x|Y = 1)\pi_1}{f(x)} > \frac{f_2(x|Y = 2)\pi_2}{f(x)} && (36) \\ &\Leftrightarrow f_1(x|Y = 1)\pi_1 > f_2(x|Y = 2)(1 - \pi_1), && (37) \end{aligned}$$

so it suffices to consider these weighted densities. The cyan regions show the areas where $\mathbb{P}(Y = 1|X = x) > \mathbb{P}(Y = 2|X = x)$, while the orange regions show where $\mathbb{P}(Y = 2|X = x) > \mathbb{P}(Y = 1|X = x)$. Thus, if X falls into one of the cyan regions, the Bayes classifier assigns $\hat{Y} = 1$, while if it falls into one of the orange regions, the Bayes classifier assigns it $\hat{Y} = 2$. So, we have

$$\hat{Y} = \begin{cases} 1 & -0.73 < x < 0.73,\ x > 2,\ x < -2 \\ 2 & -2 < x < -0.73,\ 0.73 < x < 2 \end{cases}. \tag{38}$$

5.3 Part 5c)

A natural estimator for μ is the mean of the observations we know belong to class 1. Similarly, we can estimate σ using the standard deviation of observations in class 1. Finally, we will use the overall proportion of observations that are in class 1 to estimate π_1. This leads to the estimators

$$\hat{\mu} = \frac{1}{n_1} \sum_{i:y_i=1} x_i \tag{39}$$

$$\hat{\sigma} = \sqrt{\frac{1}{n_1 - 1} \sum_{i:y_i=1} (x_i - \hat{\mu})^2} \tag{40}$$

$$\hat{\pi}_1 = \frac{n_1}{n}, \tag{41}$$

where n_1 is the number of observations with label $y_i = 1$.

5.4 Part 5d)

From (a), we have

$$\mathbb{P}(Y=1|X=x_0) = \frac{\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x_0-\mu)^2}{2\sigma^2}\right)\pi_1}{f(x_0)}. \tag{42}$$

Note that $f(x) = f_1(x|Y=1)\pi_1 + f_2(x|Y=2)(1-\pi_1)$. First, we will again assume for simplicity that $-2 \le x_0 \le 2$. Then, we can plug in our estimators from part (c) to obtain the plug-in estimator

$$\begin{aligned}
\hat{p}(x_0) &= \hat{\mathbb{P}}(Y=1|X=x_0) & (43)\\
&= \frac{\frac{1}{\sqrt{2\pi}\hat\sigma}\exp\left(-\frac{(x_0-\hat\mu)^2}{2\hat\sigma^2}\right)\hat\pi_1}{\frac{1}{\sqrt{2\pi}\hat\sigma}\exp\left(-\frac{(x_0-\hat\mu)^2}{2\hat\sigma^2}\right)\hat\pi_1 + \frac{1}{4}(1-\hat\pi_1)} & (44)\\
&= \frac{4\exp\left(-\frac{(x_0-\hat\mu)^2}{2\hat\sigma^2}\right)\hat\pi_1}{4\exp\left(-\frac{(x_0-\hat\mu)^2}{2\hat\sigma^2}\right)\hat\pi_1 + \sqrt{2\pi}\hat\sigma(1-\hat\pi_1)} & (45)\\
&= \frac{4\exp\left(-\frac{\left(x_0 - \frac{1}{n_1}\sum_{i:y_i=1}x_i\right)^2}{\frac{2}{n_1-1}\sum_{i:y_i=1}(x_i - \frac{1}{n_1}\sum_{i:y_i=1}x_i)^2}\right)\left(\frac{n_1}{n}\right)}{4\exp\left(-\frac{\left(x_0 - \frac{1}{n_1}\sum_{i:y_i=1}x_i\right)^2}{\frac{2}{n_1-1}\sum_{i:y_i=1}(x_i - \frac{1}{n_1}\sum_{i:y_i=1}x_i)^2}\right)\frac{n_1}{n} + \sqrt{\frac{2\pi}{n_1-1}\sum_{i:y_i=1}(x_i-\frac{1}{n_1}\sum_{i:y_i=1}x_i)^2}\left(1-\frac{n_1}{n}\right)}. & (46)
\end{aligned}$$

If $x_0 > 2$ or $x_0 < -2$, then $\hat{P}(Y=1|X=x_0) = 1$, because in this case we know that x_0 cannot belong to class 2 because the uniform distribution is bounded between -2 and 2.

6 Problem 6

6.1 Part 6a)

```
log.odds = 0.7
posterior.prob = exp(0.7) / (1 + exp(0.7))
```

Because Y is binary, we have that $\hat{\mathbb{P}}(Y=0|X=x) = 1 - \hat{\mathbb{P}}(Y=1|X=x)$. Thus, we have

$$\begin{aligned}
\log\left(\frac{\hat{\mathbb{P}}(Y=1|X=x)}{\hat{\mathbb{P}}(Y=0|X=x)}\right) &= \log\left(\frac{\hat{\mathbb{P}}(Y=1|X=x)}{1-\hat{\mathbb{P}}(Y=1|X=x)}\right) & (47)\\
&= 0.7 & (48)
\end{aligned}$$

Rearranging the above yields

$$\begin{aligned}
\frac{\hat{\mathbb{P}}(Y=1|X=x)}{1-\hat{\mathbb{P}}(Y=1|X=x)} &= \exp(0.7) & (49)\\
\hat{\mathbb{P}}(Y=1|X=x) &= \exp(0.7)[1-\hat{\mathbb{P}}(Y=1|X=x)] & (50)\\
[1+\exp(0.7)]\hat{\mathbb{P}}(Y=1|X=x) &= \exp(0.7) & (51)\\
\hat{\mathbb{P}}(Y=1|X=x) &= \frac{\exp(0.7)}{1+\exp(0.7)} & (52)\\
&= 0.67 & (53)
\end{aligned}$$

6.2 Part 6b)

From equation (4.6) in the textbook, we know that our estimate of the log odds is

$$\log\left(\frac{\hat{\mathbb{P}}(Y=1|X=x)}{1-\hat{\mathbb{P}}(Y=1|X=x)}\right) = \hat{\beta}_0 + \sum_{i=1}^{p} \hat{\beta}_i x_i = 0.7 \tag{54}$$

Now, for the observation $x^\star$, we have

$$\begin{aligned}
\log\left(\frac{\hat{\mathbb{P}}(Y=1|X=x^\star)}{1-\hat{\mathbb{P}}(Y=1|X=x^\star)}\right) &= \hat{\beta}_0 + \sum_{i=1}^{p} \hat{\beta}_i x_i^\star && (55)\\
&= \hat{\beta}_0 + \hat{\beta}_1(x_1+1) + \hat{\beta}_2(x_2-1) + \hat{\beta}_3(x_3+2) + \sum_{i=4}^{p} \hat{\beta}_i x_i && (56)\\
&= \hat{\beta}_0 + \sum_{i=1}^{p} \hat{\beta}_i x_i + \hat{\beta}_1 - \hat{\beta}_2 + 2\hat{\beta}_3 && (57)\\
&= 0.7 + \hat{\beta}_1 - \hat{\beta}_2 + 2\hat{\beta}_3. && (58)
\end{aligned}$$

Rearranging as in part (a), we have

$$\hat{\mathbb{P}}(Y=1|X=x^\star) = \frac{\exp(0.7 + \hat{\beta}_1 - \hat{\beta}_2 + 2\hat{\beta}_3)}{1+\exp(0.7 + \hat{\beta}_1 - \hat{\beta}_2 + 2\hat{\beta}_3)}. \tag{59}$$

7 Problem 7

7.1 Part 7a)

```r
library(MASS)
```

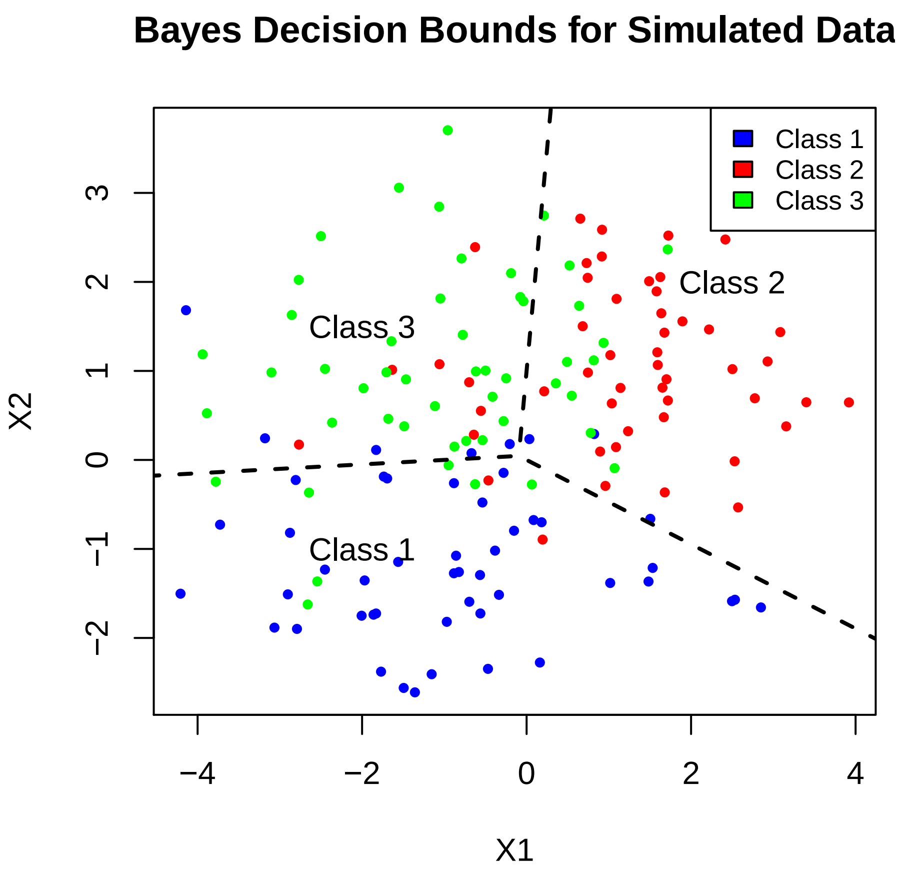
```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:ISLR2':
##
##     Boston
```

```r
# Write a function that generates the data
make.data.p7 = function(n, mus, Sigma){
  # We assume that mus is a list containing the means for each class
  K = length(mus)   # number of classes K
  p = length(mus[[1]])  # number of predictors p

  # Initialize vector of class labels and a matrix where we will store our
  # generated feature vectors
  y = c()
  x = matrix(nrow=0, ncol=2)

  # Simulate the data for each class
  for(ii in 1:K){
    y = c(y, rep(ii, n))
    x = rbind(x, mvrnorm(n=n, mu=mus[[ii]], Sigma=Sigma))
  }

  # Combine into a data frame
  data = data.frame(x1 = x[, 1], x2 = x[, 2], y = y)
  return(data)
}

# Initialize values for the data
mus = list(c(-1, -1), c(1, 1), c(-1, 1))
Sigma = matrix(c(2, 0.1, 0.1, 1), nrow=2)
n = 50
K = length(mus)

set.seed(42)
training.data.p7 = make.data.p7(n=n, mus=mus, Sigma=Sigma)
```

Here, we have chosen $\mu_1 = (-1, -1)^T$, $\mu_2 = (1, 1)^T$, $\mu_3 = (-1, 1)^T$, and $\Sigma = \begin{pmatrix} 2 & 0.1 \\ 0.1 & 1 \end{pmatrix}$.

7.2 Part 7b)

```r
# Compute Bayes decision boundaries, see derivation below.
pis = c(1/3, 1/3, 1/3)  # in this case we have equal frequencies of all classes
bayes.slopes = c()
bayes.intercepts = c()
for(ii in 1:(K-1)){
  for(jj in (ii+1):K){
    v = solve(Sigma) %*% (mus[[ii]] - mus[[jj]])
    c = ((mus[[ii]] + mus[[jj]]) %*% solve(Sigma) %*% (mus[[ii]] - mus[[jj]]) / 2
         + log(pis[jj]) - log(pis[jj]))

    bayes.slopes = c(bayes.slopes, -v[1]/v[2])
    bayes.intercepts = c(bayes.intercepts, c/v[2])
  }
}

# Make class 1 blue, class 2 red, and class 3 green
y = training.data.p7$y
colors = rep(NA, length(y))
colors[y==1] = 'blue'
colors[y==2] = 'red'
colors[y==3] = 'green'

# Plot the data
plot(training.data.p7$x1, training.data.p7$x2,
     col=colors, pch=16, cex=0.7, xlab='X1', ylab='X2',
     main='Bayes Decision Bounds for Simulated Data')
legend('topright', legend=c('Class 1', 'Class 2', 'Class 3'),
       fill=c('blue', 'red', 'green'), cex=0.8)

# Plot the decision boundaries
x.int = ((bayes.intercepts[2] - bayes.intercepts[1])
         / (bayes.slopes[1] - bayes.slopes[2]))  # boundary intersection point
x.starts = c(5, -5, 1)  # x values to start each line segment
for(ii in 1:K){
  x0 = x.starts[ii]
  y0 = x.starts[ii]*bayes.slopes[ii] + bayes.intercepts[ii]
  x1 = x.int
  y1 = x.int*bayes.slopes[ii] + bayes.intercepts[ii]
  segments(x0, y0, x1, y1, lty=2, lwd=2)
}

# Label regions
text(x=c(-2, 2.5, -2), y=c(-1, 2, 1.5), labels=c('Class 1', 'Class 2', 'Class 3'))
```



Here, we see a plot of the simulated training data colored by their class. The dashed lines represent the Bayes decision boundaries, the calculation of which we describe below.

Note that the data in this problem exactly meets the assumptions of linear discriminant analysis (LDA). Thus, the Bayes discriminant function is

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log\pi_k, \tag{60}$$

and the Bayes classifier assigns each point x to the class k for which this discriminant function is largest. Note that in this problem, $\pi_1 = \pi_2 = \pi_3 = \frac{1}{3}$. To compute the decision boundaries, we set the discriminant functions for each pair of classes equal to each other:

$$\begin{aligned}
\delta_k(x) &= \delta_\ell(x) && (61)\\
x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log\pi_k &= x^T\Sigma^{-1}\mu_\ell - \frac{1}{2}\mu_\ell^T\Sigma^{-1}\mu_\ell + \log\pi_\ell && (62)\\
x^T\Sigma^{-1}(\mu_k - \mu_\ell) &= \frac{1}{2}(\mu_k^T\Sigma^{-1}\mu_k - \mu_\ell^T\Sigma^{-1}\mu_\ell) + (\log\pi_\ell - \log\pi_k) && (63)\\
&= \frac{1}{2}(\mu_k + \mu_\ell)^T\Sigma^{-1}(\mu_k - \mu_\ell) + (\log\pi_\ell - \log\pi_k). && (64)
\end{aligned}$$

Now, to simplify notation, we set $v := \Sigma^{-1}(\mu_k - \mu_\ell)$ and $c := \frac{1}{2}(\mu_k + \mu_\ell)^T\Sigma^{-1}(\mu_k - \mu_\ell) + (\log\pi_\ell - \log\pi_k)$. Then, the above equation simplifies to

$$x^T v = c \tag{65}$$

$$x_1 v_1 + x_2 v_2 = c \tag{66}$$

$$x_2 = -\frac{v_1}{v_2}x_1 + \frac{c}{v_2}, \tag{67}$$

so the decision boundary is a line with slope m defined by

$$m = -\frac{v_1}{v_2} \tag{68}$$

$$= -\frac{[\Sigma^{-1}(\mu_k - \mu_\ell)]_1}{[\Sigma^{-1}(\mu_k - \mu_\ell)]_2}, \tag{69}$$

and intercept b defined by

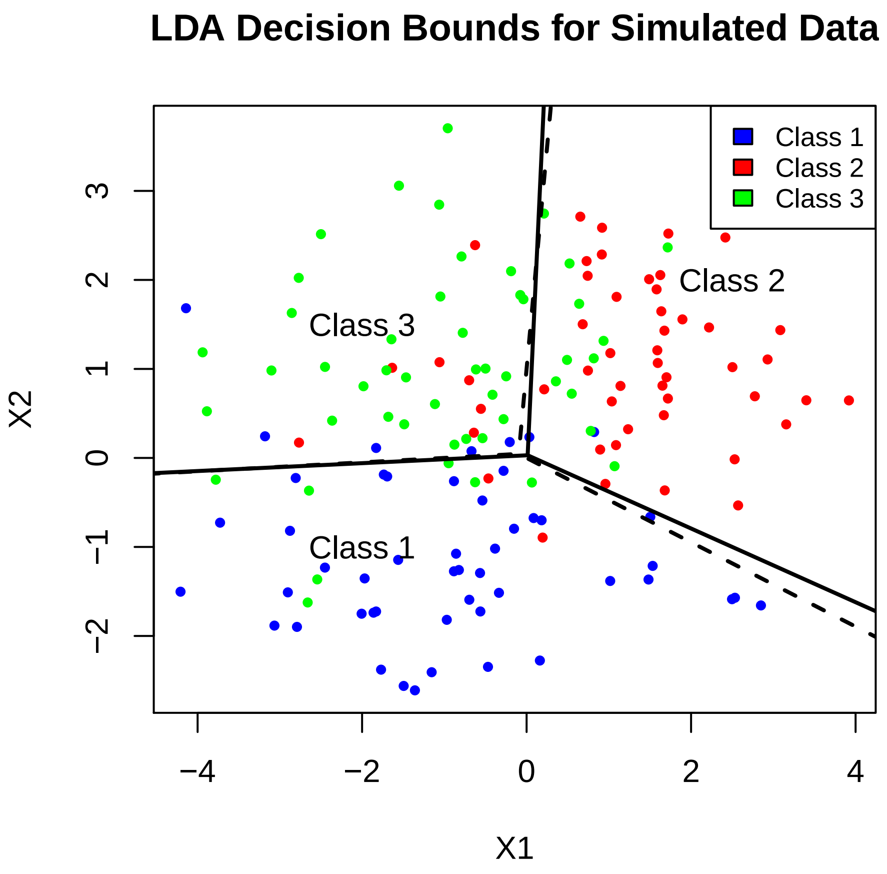
$$b = \frac{c}{v_2} \tag{70}$$

$$= \frac{(\mu_k + \mu_\ell)^T\Sigma^{-1}(\mu_k - \mu_\ell) + (\log\pi_\ell - \log\pi_k)}{2[\Sigma^{-1}(\mu_k - \mu_\ell)]_2}. \tag{71}$$

7.3 Part 7c)

```r
# Fit LDA
lda.fit = lda(y ~ ., data=training.data.p7)

# Get estimated prior probability, means, covariance
pi.hats = lda.fit$prior
mu.hats = lda.fit$means
coefs = coef(lda.fit)
Sigma.hat = solve(coefs %*% t(coefs))

# Compute LDA decision boundary Slopes/Intercepts
lda.slopes = c()
lda.intercepts = c()
for(ii in 1:(K-1)){
  for(jj in (ii+1):K){
    v = solve(Sigma.hat) %*% (mu.hats[ii,] - mu.hats[jj,])
    c = ((mu.hats[ii,] + mu.hats[jj,]) %*% solve(Sigma)
         %*% (mu.hats[ii,] - mu.hats[jj,]) / 2 + log(pi.hats[jj]) - log(pi.hats[ii]))

    lda.slopes = c(lda.slopes, -v[1]/v[2])
    lda.intercepts = c(lda.intercepts, c/v[2])
  }
}

# Plot data
plot(training.data.p7$x1, training.data.p7$x2,
     col=colors, pch=16, cex=0.7, xlab='X1', ylab='X2',
     main='LDA Decision Bounds for Simulated Data')
legend('topright', legend=c('Class 1', 'Class 2', 'Class 3'),
       fill=c('blue', 'red', 'green'), cex=0.8)

# Plot LDA decision boundaries
```

```
x.int = ((lda.intercepts[2] - lda.intercepts[1])
         / (lda.slopes[1] - lda.slopes[2]))  # boundary intersection point
x.starts = c(5, -5, 1)  # x values to start each line segment
for(ii in 1:K){
  x0 = x.starts[ii]
  y0 = x.starts[ii]*lda.slopes[ii] + lda.intercepts[ii]
  x1 = x.int
  y1 = x.int*lda.slopes[ii] + lda.intercepts[ii]
  segments(x0, y0, x1, y1, lty=1, lwd=2)
}

# Plot Bayes Decision Boundaries for reference
x.int = ((bayes.intercepts[2] - bayes.intercepts[1])
         / (bayes.slopes[1] - bayes.slopes[2]))  # boundary intersection point
x.starts = c(5, -5, 1)  # x values to start each line segments
for(ii in 1:K){
  x0 = x.starts[ii]
  y0 = x.starts[ii]*bayes.slopes[ii] + bayes.intercepts[ii]
  x1 = x.int
  y1 = x.int*bayes.slopes[ii] + bayes.intercepts[ii]
  segments(x0, y0, x1, y1, lty=2, lwd=2)
}

text(x=c(-2, 2.5, -2), y=c(-1, 2, 1.5), labels=c('Class 1', 'Class 2', 'Class 3'))
```



The LDA decision boundaries are shown as solid black lines. The Bayes decision boundaries are shown as dashed black lines. We note that the two are very similar, which is to be expected because the data follows the LDA assumptions exactly.

7.4 Part 7d)

```
fitted = predict(lda.fit)$class
confusion.matrix = table(fitted, y)
train.error = 1 - sum(diag(confusion.matrix)) / sum(confusion.matrix)
```

The confusion matrix is $\begin{pmatrix} 43 & 2 & 7 \\ 1 & 41 & 11 \\ 6 & 7 & 32 \end{pmatrix}$. The training error is 0.23.

7.5 Part 7e)

```
set.seed(8675309)
test.data.p7 = make.data.p7(n=n, mus=mus, Sigma=Sigma)
y.test = test.data.p7$y

test.fitted = predict(lda.fit, test.data.p7[, 1:2])$class
test.confusion.matrix = table(test.fitted, y.test)
test.error = 1 - sum(diag(test.confusion.matrix)) / sum(test.confusion.matrix)
```

The confusion matrix on the test set is $\begin{pmatrix} 42 & 5 & 7 \\ 1 & 32 & 10 \\ 7 & 13 & 33 \end{pmatrix}$. The test error is 0.29.

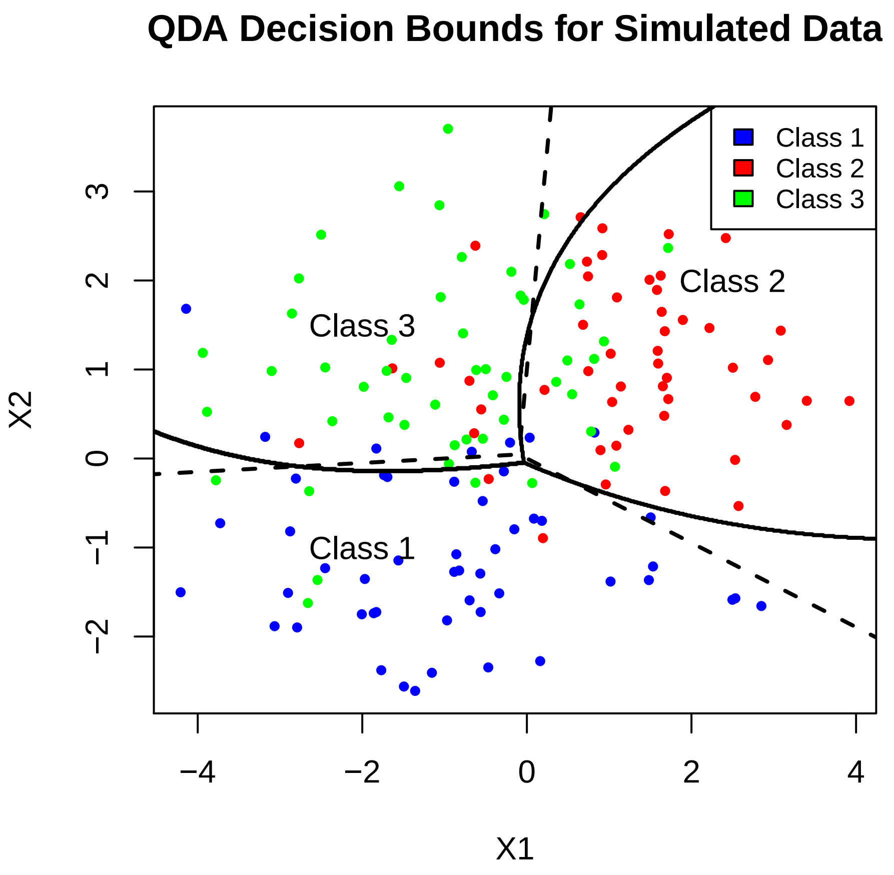
7.6 Part 7f)

We see that the test set error is higher than the training error, which we expect. In general, it seems that classes 2 and 3 were the most difficult to separate, while separating class 1 from the other classes was an easier task.

8 Problem 8

8.1 Part 8a)

```
# Fit QDA
qda.fit = qda(y ~ ., data=training.data.p7)

# Plot data
plot(training.data.p7$x1, training.data.p7$x2,
     col=colors, pch=16, cex=0.7, xlab='X1', ylab='X2',
     main='QDA Decision Bounds for Simulated Data')
legend('topright', legend=c('Class 1', 'Class 2', 'Class 3'),
       fill=c('blue', 'red', 'green'), cex=0.8)

# Plot Bayes Decision Boundaries for reference
x.int = ((bayes.intercepts[2] - bayes.intercepts[1])
         / (bayes.slopes[1] - bayes.slopes[2]))  # boundary intersection point
x.starts = c(5, -5, 1)  # x values to start each line segments
for(ii in 1:K){
  x0 = x.starts[ii]
  y0 = x.starts[ii]*bayes.slopes[ii] + bayes.intercepts[ii]
  x1 = x.int
  y1 = x.int*bayes.slopes[ii] + bayes.intercepts[ii]
  segments(x0, y0, x1, y1, lty=2, lwd=2)
}

# Plot QDA Decision Boundaries
num.points = 1000
grid.spec = seq(from=-5, to=5, length.out=num.points)
grid.points = expand.grid(grid.spec, grid.spec, stringsAsFactors = FALSE)
colnames(grid.points) = c('x1', 'x2')
grid.points = grid.points[order(grid.points$x1, grid.points$x2),]

grid.labels = predict(qda.fit, grid.points)$class
grid.z = matrix(as.numeric(grid.labels), nrow=num.points, byrow=TRUE)
contour(grid.spec, grid.spec, grid.z, add = TRUE, lwd = 2,
        levels = (1:(K-1))+.5, drawlabels = FALSE)


text(x=c(-2, 2.5, -2), y=c(-1, 2, 1.5), labels=c('Class 1', 'Class 2', 'Class 3'))
```



Here, we have plotted the Bayes decision boundaries as dashed lines and the QDA decision boundaries as solid lines. We can see that as opposed to LDA, the QDA decision boundaries deviate significantly from the Bayes decision boundaries as we move away from the center of the plot.

To calculate the QDA decision boundary, first note that from equation (4.28) in the textbook, the discriminant function for QDA is

$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\Sigma_k| + \log\pi_k &(72)\\ &= -\frac{1}{2}x^T\Sigma_k^{-1}x + x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\log|\Sigma_k| + \log\pi_k. &(73)\end{aligned}$$

Now, to get the decision boundary for a given pair of groups, we set their discriminant functions equal to each other as in problem 7:

$$\delta_k(x) = \delta_\ell(x) \quad (74)$$

$$-\frac{1}{2}x^T\Sigma_k^{-1}x + x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\log|\Sigma_k| + \log\pi_k = -\frac{1}{2}x^T\Sigma_\ell^{-1}x + x^T\Sigma_\ell^{-1}\mu_\ell - \frac{1}{2}\mu_\ell^T\Sigma_\ell^{-1}\mu_\ell - \frac{1}{2}\log|\Sigma_\ell| + \log\pi_\ell \quad (75)$$

$$x^T(\Sigma_k^{-1} - \Sigma_\ell^{-1})x + 2x^T(\Sigma_\ell^{-1}\mu_\ell - \Sigma_k^{-1}\mu_k) = \mu_\ell\Sigma_\ell^{-1}\mu_\ell - \mu_k^T\Sigma_k^{-1}\mu_k + \log|\Sigma_\ell| - \log|\Sigma_k| + \log\pi_k - \log\pi_\ell \quad (76)$$

$$x^TAx + x^Tb + c = 0, \quad (77)$$

where

$$\begin{aligned}A &= (\Sigma_k^{-1} - \Sigma_\ell^{-1}) &(78)\\ b &= 2(\Sigma_\ell^{-1}\mu_\ell - \Sigma_k^{-1}\mu_k) &(79)\\ c &= \mu_k\Sigma_k^{-1}\mu_k - \mu_\ell^T\Sigma_\ell^{-1}\mu_\ell + \log|\Sigma_k| - \log|\Sigma_\ell| + \log\pi_\ell - \log\pi_k. &(80)\end{aligned}$$

Then, we can plug in our estimates for μ_k, μ_ℓ, Σ_k, Σ_ℓ, π_k, and π_ℓ and plot the resulting boundaries for each pair of classes. Alternatively, we can use the `predict` and `contour` functions in `R`, see code above. Another option is to use the `R` package `klaR`.

8.2 Part 8b)

```r
fitted = predict(qda.fit)$class
confusion.matrix = table(fitted, y)
train.error = 1 - sum(diag(confusion.matrix)) / sum(confusion.matrix)
```

The confusion matrix is $\begin{pmatrix}43 & 2 & 6\\ 2 & 40 & 10\\ 5 & 8 & 34\end{pmatrix}$. The training error is 0.22.

8.3 Part 8c)

```r
test.fitted = predict(qda.fit, test.data.p7[, 1:2])$class
test.confusion.matrix = table(test.fitted, y.test)
test.error = 1 - sum(diag(test.confusion.matrix)) / sum(test.confusion.matrix)
```

The confusion matrix on the test set is $\begin{pmatrix}40 & 5 & 7\\ 1 & 30 & 12\\ 9 & 15 & 31\end{pmatrix}$. The test error is 0.33.

8.4 Part 8d)

The test error is about 50% higher than training error for QDA. It's expected that the test error would be higher than the training error, but the gap is bigger here than in was for LDA in problem (7). Thus, QDA may not generalize as well for this particular problem setup.

8.5 Part 8e)

QDA had slightly lower training error with this setup. This is not too surprising, as QDA is much more flexible than LDA. However, the difference is quite small, so it is unclear how consistent this trend would we if we repeated the experiment many times.

8.6 Part 8f)

In this case, LDA outperformed QDA in terms of test error. This is somewhat expected, as we saw that the fitted decision boundaries for LDA were much closer to the Bayes boundaries than the QDA boundaries were. Coupled with the lower training error for QDA, this may suggest overfitting; the QDA boundaries may be tuned too closely to the particular training data set. However, as mentioned in (e), the differences were not egregiously large. This is consistent with the experiments in section 4.5 of the textbook.

9 Problem 9 (Extra Credit)

First, we define

$$\beta \coloneqq (\beta_1, \beta_2, \dots, \beta_p)^T \in \mathbb{R}^p \tag{81}$$

$$y \coloneqq (y_1, y_2, \dots, y_n)^T \in \mathbb{R}^n \tag{82}$$

$$X \coloneqq \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix} \in \mathbb{R}^{n \times p}. \tag{83}$$

Our goal is to minimize

$$\ell(\beta) = \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{84}$$

$$= (y - X\beta)^T (y - X\beta) + \lambda \beta^T \beta, \tag{85}$$

where we assume $\beta_0 = 0$ for simplicity. Any local optimum $\beta^\star$ must satisfy $\ell'(\beta^\star) = 0$. Differentiating $\ell(\beta)$ with respect to β yields

$$\frac{d}{d\beta}\ell(\beta) = -2X^T(y - X\beta) + 2\lambda\beta. \tag{86}$$

Setting this equal to zero gives

$$-2X^T(y - X\beta^\star) + 2\lambda\beta^\star = 0 \tag{87}$$

$$(\lambda I_p + X^T X)\beta^\star = X^T y \tag{88}$$

$$\beta^\star = (\lambda I_p + X^T X)^{-1} X^T y, \tag{89}$$

where I_p denotes the $p \times p$ identity matrix. Note that $(\lambda I_p + X^T X)^{-1}$ exists because $\lambda > 0$, I_p is positive definite, and $X^T X$ is positive semidefinite for any X; thus, $\lambda I_p + X^T X$ is positive definite, hence invertible.

We have shown that $\beta^\star$ is a critical point of the loss function $\ell(\beta)$. It remains to show that this corresponds to the global minimum of $\ell(\beta)$. If we can show that $\ell(\beta)$ is a convex function of β, then it follows that $\beta^\star$ corresponds to the global minimum. To do so, we can calculate the *Hessian*: the matrix of second derivates of $\ell(\beta)$. If the Hessian is positive definite, then $\ell(\beta)$ is a convex function of β. We have

$$\frac{d^2}{d\beta^2}\ell(\beta) = \frac{d}{d\beta}[-2X^T(y - X\beta) + 2\lambda\beta] \tag{90}$$

$$= 2X^T X + 2\lambda I_p \tag{91}$$

$$= 2(X^T X + \lambda I_p), \tag{92}$$

which we have argued above is positive definite. Hence, $\beta^\star$ is the global minimizer of $\ell(\beta)$.

Thus, the least squares ridge regression estimator is

$$\hat{\beta} = (\lambda I_p + X^T X)^{-1} X^T y. \tag{93}$$