Solution to Series 7

1. **a)** Maximizing $\hat{\pi}_j$ (see (6.4)) over j for fixed x is equivalent to the maximization of $\log(\hat{f}_{X|Y=j}(x)p_j)$ (the denominator does not depend on j).
The normal density is given by

$$\hat{f}_{X|Y=j}(x) = \frac{1}{(2\pi)^{p/2}\det(\hat{\Sigma}_j)^{1/2}} \exp(-\frac{1}{2}(x-\hat{\mu}_j)^\intercal \hat{\Sigma}_j^{-1}(x-\hat{\mu}_j)).$$

We get

$$\log(\hat{f}_{X|Y=j}(x)p_j) = -\frac{p}{2}\log(2\pi) - \log(\det(\hat{\Sigma}_j))/2 - (x-\hat{\mu}_j)^\intercal \hat{\Sigma}_j^{-1}(x-\hat{\mu}_j)/2 + \log(\hat{p}_j).$$

Ignoring the leading constant term gives the solution $\hat{\delta}_j^{QDA}$.

b) We can directly start at the solution of a) and replace $\hat{\Sigma}_j$ by $\hat{\Sigma}$:

$$\hat{\delta}_j^{pre}(x) = -\log(\det(\hat{\Sigma}))/2 - x^\intercal\hat{\Sigma}^{-1}x/2 + x^\intercal\hat{\Sigma}^{-1}\hat{\mu}_j - \hat{\mu}_j{}^\intercal\hat{\Sigma}^{-1}\hat{\mu}_j/2 + \log(\hat{p}_j)$$

Again, by ignoring the two leading terms that are constant with respect to the maximization over j, we get

$$\hat{\delta}_j^{\mathrm{LDA}}(x) = (x - \frac{\hat{\mu}_j}{2})^\intercal\hat{\Sigma}^{-1}\hat{\mu}_j + \log(\hat{p}_j). \tag{1}$$

c) The boundary between the two groups is the set $B = \{x \,|\, \hat{\delta}_0(x) = \hat{\delta}_1(x)\}$. We get

$$\begin{aligned} B &= \{x \,|\, x^\intercal b_0 + c_0 = x^\intercal b_1 + c_1\} \\ &= \{x \,|\, x^\intercal(b_0 - b_1) + c_0 - c_1 = 0\} \end{aligned} \tag{2}$$

which is a $p-1$ dimensional subspace of $\mathbb{R}^p$, e.g. a straight line in $\mathbb{R}^2$ or a plane in $\mathbb{R}^3$.
$B_0 = \{x \,|\, \hat{\delta}_0(x) > \hat{\delta}_1(x)\}$ is the set of points which are assigned to class 0. We can write $B_0 = \{x \,|\, x^\intercal(b_0 - b_1) + c_0 - c_1 > 0\}$. This is the set of points x which lie on "one side" of the boundary B. From equation (1), we find

$$b_j = \hat{\Sigma}^{-1}\hat{\mu}_j, \tag{3}$$

$$c_j = \log\hat{p}_j - \frac{1}{2}\hat{\mu}_j{}^\intercal\hat{\Sigma}^{-1}\hat{\mu}_j = \log\hat{p}_j - \frac{1}{2}\hat{\mu}_j{}^\intercal b_j. \tag{4}$$

Hence the decision boundary (2) is

$$\begin{aligned} B &= \{x \,|\, x^\intercal \underbrace{\hat{\Sigma}^{-1}(\hat{\mu}_0 - \hat{\mu}_1)}_{=:w} + \underbrace{\log\frac{\hat{p}_0}{\hat{p}_1} - \frac{1}{2}\hat{\mu}_0{}^\intercal\hat{\Sigma}^{-1}\hat{\mu}_0 + \frac{1}{2}\hat{\mu}_1{}^\intercal\hat{\Sigma}^{-1}\hat{\mu}_1}_{=:-a} = 0\} \\ &= \{x \,|\, x^\intercal w = a\}. \end{aligned}$$

If we define

$$z = \frac{1}{2}(\hat{\mu}_0 + \hat{\mu}_1) + \frac{\hat{\mu}_0 - \hat{\mu}_1}{(\hat{\mu}_0 - \hat{\mu}_1)^\intercal\hat{\Sigma}^{-1}(\hat{\mu}_0 - \hat{\mu}_1)}\log\frac{\hat{p}_1}{\hat{p}_0}$$

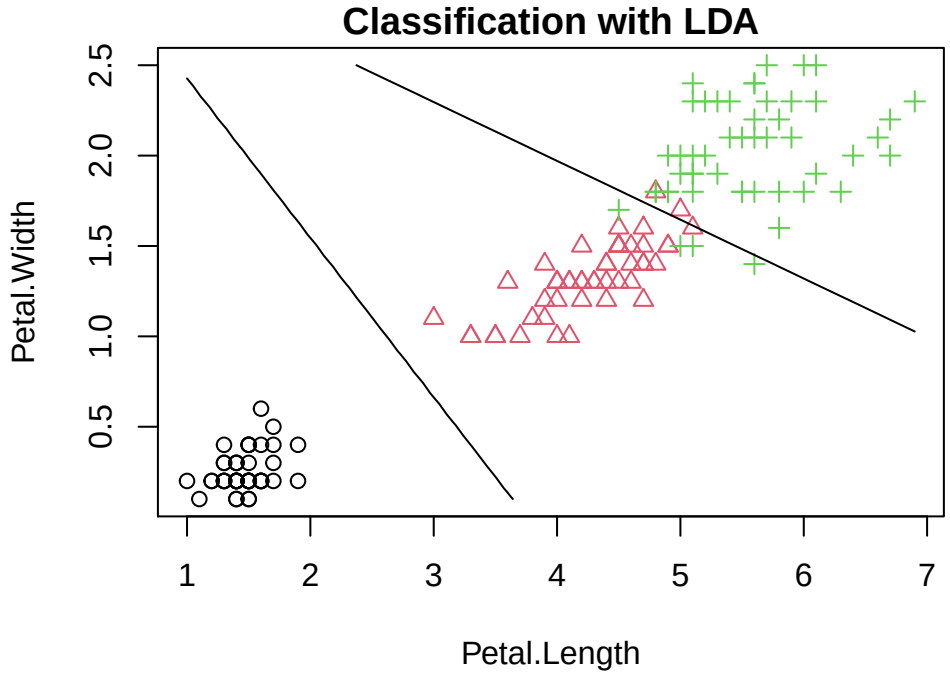
and write $B = \{x|(x-z)^\intercal w = 0\}$, one can show that the decision boundary shifts towards $\hat{\mu}_0$ (and away from $\hat{\mu}_1$) as the fraction of the estimated prior probabilities $\hat{p}_1/\hat{p}_0$ increases, and vice versa. Moreover, if the covariance matrix $\hat{\Sigma}$ is diagonal and has constant components, then w is parallel to the straight line connecting the means $\hat{\mu}_0$ and $\hat{\mu}_1$, and the decision boundary B is orthogonal to it.

2. Preparations for the exercise: set the seed, read in the data, parameters and functions:

```
> ## Read in dataset, set seed, load package
> Iris <- iris[,c("Petal.Length", "Petal.Width", "Species")]
> grIris <- as.integer(Iris[, "Species"])
> set.seed(16)
> library(MASS)
> ## Read n
> n <- nrow(Iris)
> ## Utility functiom for plotting boundaries
> predplot <- function(object, x, gr = grIris, main = "", lines.only = FALSE,
                       len = 42, colcont = "black", ...)
 {
   ##  gr : the true grouping/class vector
   stopifnot(length(gr) == nrow(x))
   xp <- seq(min(x[, 1]), max(x[, 1]), length = len)
   yp <- seq(min(x[, 2]), max(x[, 2]), length = len)
   grid <- expand.grid(xp, yp)
   colnames(grid) <- colnames(x)[-3]
   Z <- predict(object, grid, ...)
   zp <- as.numeric(Z$class)
   zp <- Z$post[, 3] - pmax(Z$post[, 2], Z$post[, 1])
   if(!lines.only)
     plot(x[,1], x[,2], col = gr, pch = gr,
          main = main, xlab = colnames(x)[1], ylab = colnames(x)[2])
   contour(xp, yp, matrix(zp, len),
           add = TRUE, levels = 0, drawlabels = FALSE, col = colcont)
   zp <- Z$post[, 1] - pmax(Z$post[, 2], Z$post[, 3])
   contour(xp, yp, matrix(zp, len),
           add = TRUE, levels = 0, drawlabels = FALSE, col = colcont)
 }
> ## Bootstrap size
> B <- 1000
```

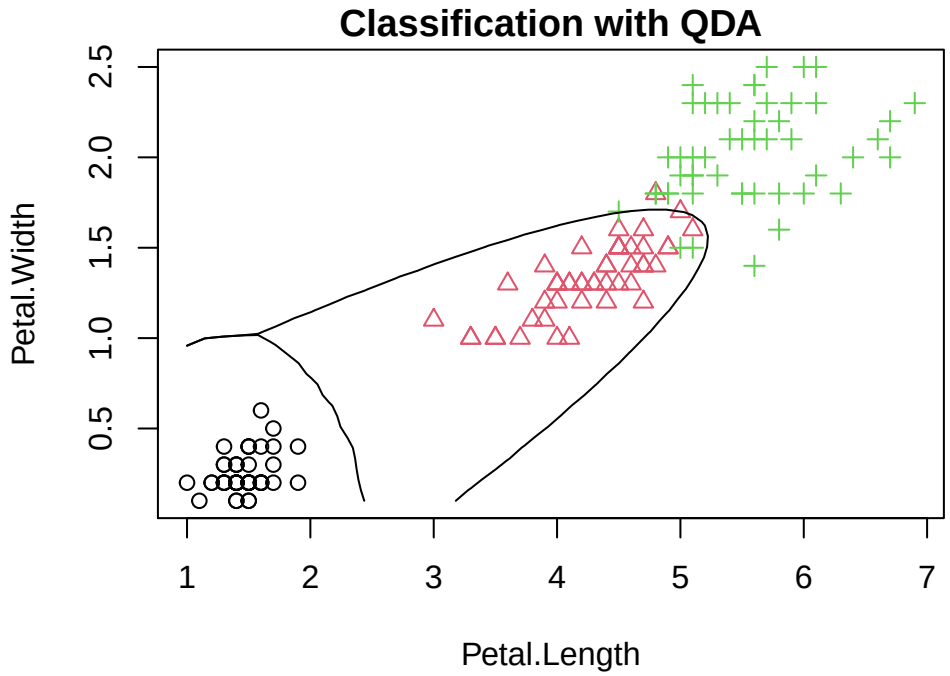
a) Here we fit the data with the LDA and QDA methods and plot the resulting classifications. We begin with the LDA method.

```
> class_lda <- lda(x = Iris[, c("Petal.Length", "Petal.Width")],
                   grouping = Iris[, "Species"])
> predplot(class_lda, Iris, main = "Classification with LDA")
```



Now we have the QDA method.

```
> class_qda <- qda(x = Iris[ ,c("Petal.Length", "Petal.Width")],
                   grouping = Iris[, "Species"])
> predplot(class_qda, Iris, main = "Classification with QDA")
```



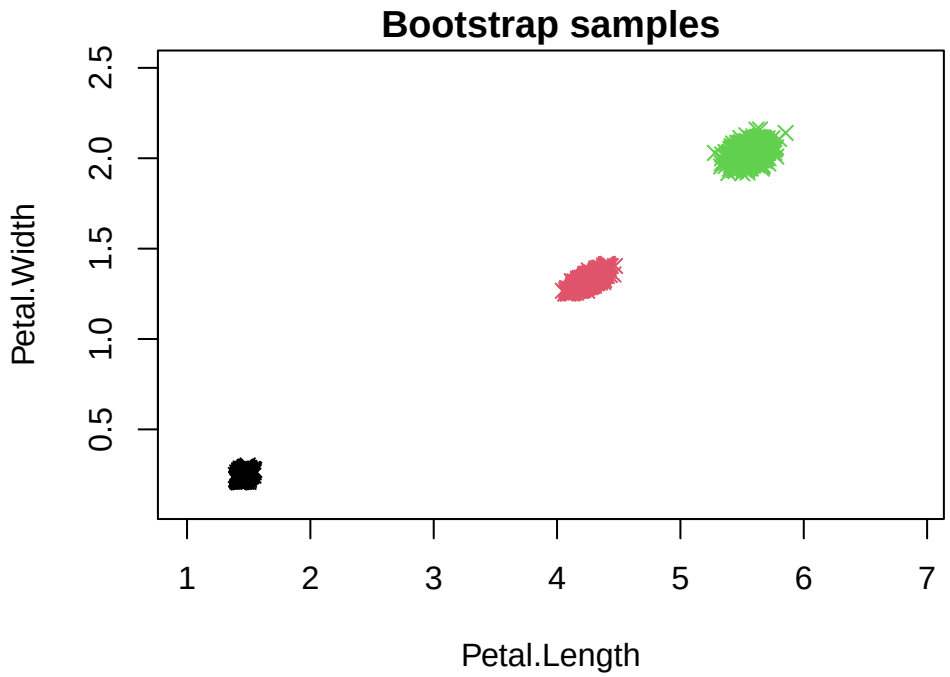
b) We begin by generating an indexes matrix for the bootstrap samples. Then we save the bootstrapped fits and resulting means.

```
> #random index to generate bootstrap
> index <- matrix(sample.int(n, n*B, replace = TRUE), nrow = n, ncol = B)
> #initial list for LDA and QDA fits
> fit_lda <- vector("list", B)
> fit_qda <- vector("list", B)
> #use both methods on the bootstrap samples
> for(i in 1:B) {
   ind <- index[, i]
   fit_lda[[i]] <- lda(x = Iris[ind , c("Petal.Length", "Petal.Width")],
                       grouping = Iris[ind , "Species"])
   fit_qda[[i]] <- qda(x = Iris[ind , c("Petal.Length", "Petal.Width")],
                       grouping = Iris[ind , "Species"])
 }
> #initialize the mu_hat bootstrap estimates
> mu_hat_1 <- mu_hat_2 <- mu_hat_3 <- matrix(0, ncol = B, nrow = 2)
> #determine the mu_hat bootstrap estimates
>
> for(i in 1:B){
   mu_hat_all <- fit_lda[[i]]$means
   mu_hat_1[, i] <- mu_hat_all[1,]
   mu_hat_2[, i] <- mu_hat_all[2,]
   mu_hat_3[, i] <- mu_hat_all[3,]
 }
```

Finally we plot the bootstrapped means in a single plot.

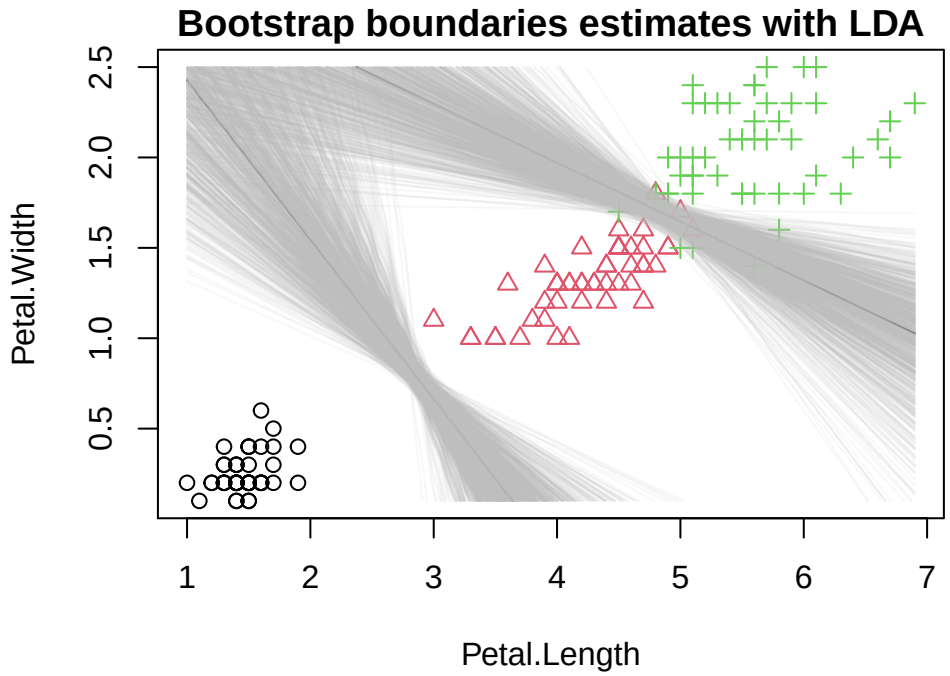
```
> xmin <- min(Iris[, "Petal.Length"])
> xmax <- max(Iris[, "Petal.Length"])
> ymin <- min(Iris[, "Petal.Width"])
> ymax <- max(Iris[, "Petal.Width"])
> plot(mu_hat_1[1, ], mu_hat_1[2, ], xlim = c(xmin,
      xmax), ylim = c(ymin, ymax), xlab = colnames(Iris)[1],
      ylab = colnames(Iris)[2], pch = 4, main = "Bootstrap samples")
```

```
> points(mu_hat_2[1, ], mu_hat_2[2, ], col = 2,
     pch = 4)
> points(mu_hat_3[1, ], mu_hat_3[2, ], col = 3,
     pch = 4)
```



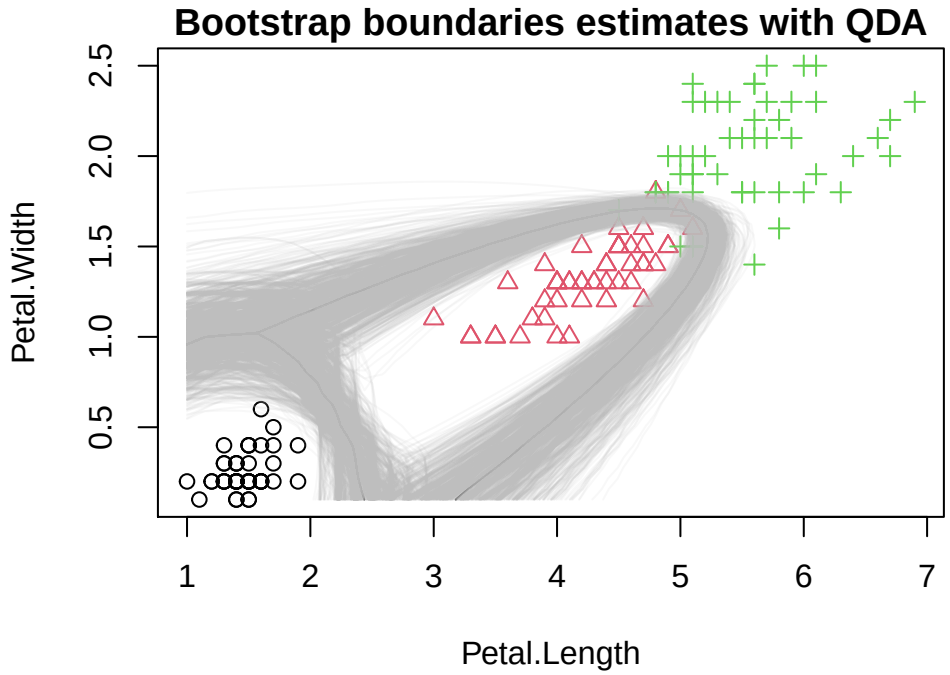
c) Here we plot the classification boundaries of the bootstrap samples. We begin with the LDA method.

```
> predplot(class_lda, Iris, main = "Bootstrap boundaries estimates with LDA")
> for(i in 1:B){
   fit <- fit_lda[[i]]
   predplot(fit, Iris, lines.only = TRUE, colcont = adjustcolor("gray", 0.1))}
```



Now we plot the boundaries resulting from the QDA method.

```
> predplot(class_qda, Iris, main = "Bootstrap boundaries estimates with QDA")
> for(i in 1:B){
   fit <- fit_qda[[i]]
   predplot(fit, Iris,lines.only = TRUE,colcont = adjustcolor("gray", 0.1))}
```



d) We calculate the LOOCV estimate of the generalization error for both methods.

```
> pred_lda <- Iris[,"Species"]
> pred_qda <- Iris[,"Species"]
> for (i in 1:n){
   fiti_lda <- lda(x = Iris[-i , c("Petal.Length", "Petal.Width")],
                   grouping = Iris[-i , "Species"])
   pred_lda[i] <- (predict(fiti_lda, Iris[i, c("Petal.Length",
                                              "Petal.Width")])$class )
   fiti_qda <- qda(x = Iris[-i , c("Petal.Length", "Petal.Width")],
                   grouping = Iris[-i , "Species"])
   pred_qda[i] <- (predict(fiti_qda, Iris[i, c("Petal.Length",
                                              "Petal.Width")])$class )
 }
> ##print the error
> cat("CV error for LDA:", format(mean(pred_lda != Iris[,"Species"]), digits = 3))
CV error for LDA: 0.04
> cat("CV error for QDA:", format(mean(pred_qda != Iris[,"Species"]), digits = 3))
CV error for QDA: 0.0333
```

We see that QDA slightly outperforms LDA in this metric. More precisely, LDA missclassifies 6 out of 150 samples while as for QDA we only encounter 5 missclassifications.

3. **a)** The likelihood reads

$$L(\beta;(x_1,m_1,N_1),\ldots,(x_n,m_n,N_n)) = \prod_{i=1}^{n}\binom{m_i}{N_i}\pi(x_i)^{N_i}(1-\pi(x_i))^{m_i-N_i} \ ;$$

taking the logarithm gives

$$\ell(\beta;(x_1,m_1,N_1),\ldots,(x_1,m_1,N_1)) = \sum_{i=1}^{n}\left[\log\binom{m_i}{N_i} + N_i\log(\pi(x_i)) + (m_i-N_i)\log\left(1-\pi(x_i)\right)\right] \ . \quad (5)$$

From the logistic transform,

$$g(x_i) = \log\left(\frac{\pi(x_i)}{1-\pi(x_i)}\right) = \log(\pi(x_i)) - \log(1-\pi(x_i)) \ ,$$

where we leave out the parameter β of g for better readability, we find $\log(\pi(x_i)) = g(\beta; x_i) + \log(1 - \pi(x_i))$. Solving for $\pi(x_i)$ yields

$$\pi(x_i) = \frac{e^{g(x_i)}}{1 + e^{g(x_i)}} \quad \text{and} \quad 1 - \pi(x_i) = \frac{1}{1 + e^{g(x_i)}} \ ,$$

and hence $\log(1 - \pi(x_i)) = -\log\left(1 + e^{g(x_i)}\right)$. Plugging those identities for $\log(\pi(x_i))$ and $\log(1 - \pi(x_i))$ into Equation (5) yields the claimed result.

b) We write a simple auxillary function that calculates $g(\beta; x)$:

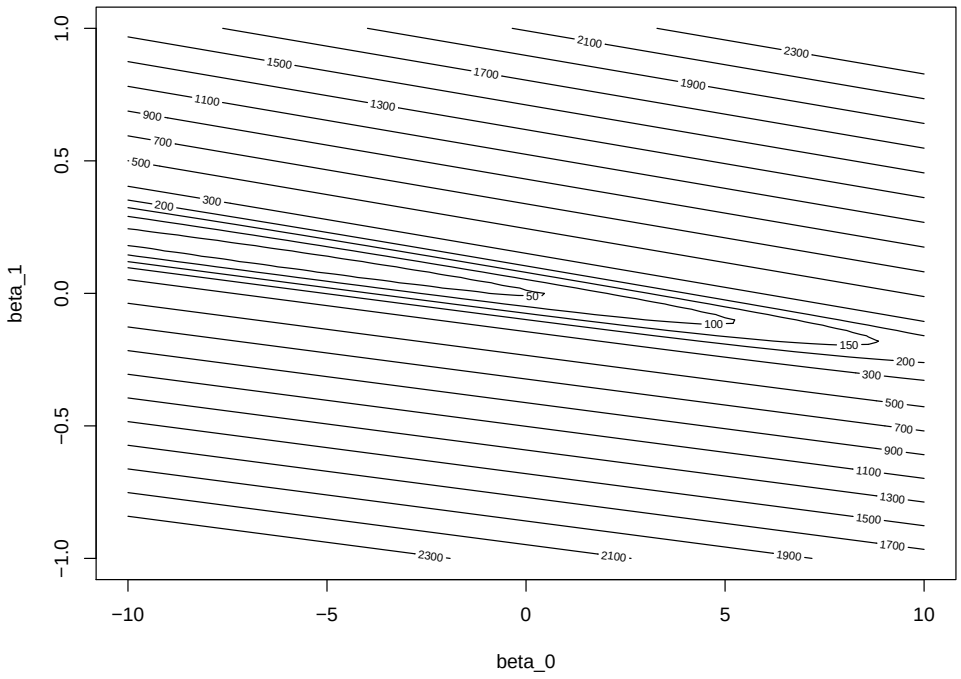
```
> g <- function(beta, x) beta[1] + beta[2]*x
```

Next, we implement the negative log-likelihood calculated in task a):

```
> neg.ll <- function(beta, data){
   - sum(log(choose(data$m, data$N)) +
         data$N * g(beta,data$age) -
         data$m * log( 1 + exp(g(beta,data$age))))
 }
```

We then generate the contour plot:

```
> heart <- read.table("http://stat.ethz.ch/Teaching/Datasets/heart.dat",
                      header = TRUE)
> beta0.grid <- seq(-10, 10, length = 101)
> beta1.grid <- seq(-1, 1, length = 101)
> ## initialize grid:
> neg.ll.values <- matrix(0, nrow = length(beta0.grid), ncol = length(beta1.grid))
> ## evaluate negative log-likelihood on every combination of beta0 and beta1
> for (i in 1:length(beta0.grid))
   for (j in 1:length(beta1.grid))
     neg.ll.values[i, j] <- neg.ll(c(beta0.grid[i], beta1.grid[j]), heart)
> contour(beta0.grid, beta1.grid, neg.ll.values, xlab = "beta_0",
         ylab = "beta_1",levels=c(seq(50,200,50),seq(300,2300,200)))
```



c)

```
> fit <- glm(cbind(N, m - N) ~ age, family = binomial, data = heart)
> summary(fit)

Call:
glm(formula = cbind(N, m - N) ~ age, family = binomial, data = heart)

Deviance Residuals:
```

```
     Min        1Q    Median        3Q       Max
-1.36404  -0.54656   0.02468   0.55254   1.53533

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -5.0993     1.1090  -4.598 4.26e-06 ***
age           0.1084     0.0238   4.555 5.24e-06 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 53.466  on 42  degrees of freedom
Residual deviance: 25.153  on 41  degrees of freedom
AIC: 63.888

Number of Fisher Scoring iterations: 4
```

We obtain $\hat{\beta}_0 = -5.1$ for the intercept and $\hat{\beta}_1 = 0.11$ for the coefficient of age. The influence of age is significant (p-value < 0.001). The positive sign of $\hat{\beta}_1$ means that the logit increases with age, and since the logit is an increasing function of the probability of having symptoms, this probability increases with age as well. The absolute value of $\widehat{\beta}$ is harder to interpret due to the logit-scale.
By optimizing our own log-likelihood function, we get the same parameters:

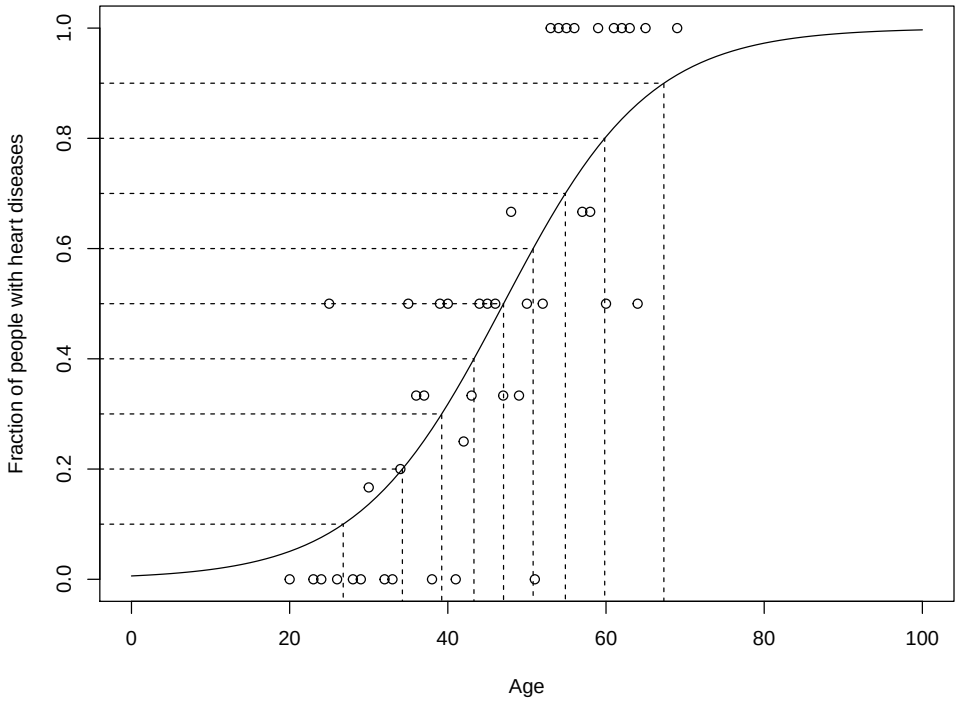
```
> optim(c(0, 0), neg.ll, data = heart)$par
[1] -5.0990932  0.1083889
```

d) From the model equation, we infer that

$$x_i = \frac{\log\left(\frac{\pi_i}{1-\pi_i}\right) - \beta_0}{\beta_1}.$$

Setting $\pi_i = 0.1, 0.2, \ldots, 0.9$, and plugging in the estimated values for β_0 and β_1, we obtain the age at which we expect $10\%, 20\%, \ldots, 90\%$ of people to have symptoms of heart disease.

```
> new.age <- 0:100
> heart.pred <- predict(fit, new = data.frame(age = new.age), type = "response")
> plot(heart$age, heart$N/heart$m, xlim = c(0, 100), ylim = c(0,1),
      xlab = "Age", ylab = "Fraction of people with heart diseases")
> lines(new.age, heart.pred)
> perc         <- (1:9)/10
> x.age        <- (log(perc/(1-perc)) - coef(fit)[1])/coef(fit)[2]
> names(x.age) <- perc
> for(n in 1:9)
     lines(c(-4, x.age[n], x.age[n]), c(perc[n], perc[n], -0.04), lty = 2)
```



symptoms	10%	20%	30%	40%	50%	60%	70%	80%	90%
age	27	34	39	43	47	51	55	60	67

Between ages 39 and 55, the predicted probability of having symptoms increases linearly ($+10\%$ every 4 years). Out of this range, the probability increases less fast.