Compulsory Excercise 1

Jostein Aasteboel Aanes, Sindre Skaugset Olderkjaer, Markus Traetli

24 februar, 2021

Problem 1

In the task, it is given that $\tilde{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}$ where $\lambda \geq 0$.

Problem 1a)

First, the expected value of $\tilde{\boldsymbol{\beta}}$ is calculated

$$E[\tilde{\boldsymbol{\beta}}] = E[(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}].$$

Using the fact that $\mathbf{Y} = f(\mathbf{X}) + \boldsymbol{\varepsilon}$ and $f(\mathbf{X}) = \mathbf{X}\beta$ the following is obtained

$$E[\tilde{\beta}] = E[(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon})] = E[(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}] + E[(\mathbf{X}^T\mathbf{X}\lambda\mathbf{I})^{-1}\mathbf{X}^T\boldsymbol{\varepsilon}].$$

It is assumed that $E[\boldsymbol{\varepsilon}] = 0$ and since $\boldsymbol{\beta}$ is a constant vector, the expression simplifies to

$$E[\tilde{\boldsymbol{\beta}}] = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}.$$

For the variance-covariance matrix one can use that $\text{Cov}[A\mathbf{X}] = A\text{Cov}[\mathbf{X}]A^T$

$$\text{Cov}[\tilde{\boldsymbol{\beta}}] = \text{Cov}[(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}] = \text{Cov}[W_\lambda\mathbf{Y}] = W_\lambda\text{Cov}[\mathbf{Y}]W_\lambda^T,$$

where $W_\lambda = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T$ and

$$\text{Cov}[\mathbf{Y}] = \text{Cov}[f(\mathbf{X}) + \boldsymbol{\varepsilon}] = \text{Cov}[f(\mathbf{X})] + \text{Cov}[\boldsymbol{\varepsilon}].$$

Since $f(\mathbf{X}) = \mathbf{X}\boldsymbol{\beta}$ and neither $\mathbf{X}$ nor $\boldsymbol{\beta}$ are stochastic, the term vanishes and all that is left is $Cov[\varepsilon] = \sigma^2\mathbf{I}$. Thus

$$\text{Cov}[\tilde{\boldsymbol{\beta}}] = W_\lambda\text{Cov}[\mathbf{Y}]W_\lambda^T = W_\lambda\text{Cov}[\boldsymbol{\varepsilon}]W_\lambda^T = \sigma^2 W_\lambda W_\lambda^T.$$

Problem 1b)

$$E[\tilde{f}(\mathbf{x}_0)] = E[\mathbf{x}_0^T\tilde{\boldsymbol{\beta}}] = \mathbf{x}_0^T E[\tilde{\boldsymbol{\beta}}] = \mathbf{x}_0^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}$$

This can be simplified further by adding and subtracting $\lambda\mathbf{I}$, which is useful in **c)**, as such

$$E[\tilde{f}(\mathbf{x}_0)] = \mathbf{x}_0^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I} - \lambda\mathbf{I})\boldsymbol{\beta} = \mathbf{x}_0^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}((\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}) - \lambda\mathbf{I})\boldsymbol{\beta}.$$

Thus the expected value is

$$E[\tilde{f}(\mathbf{x}_0)] = \mathbf{x}_0^T(\boldsymbol{\beta} - \lambda(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\boldsymbol{\beta}).$$

For calculating the variance one has

$$\text{Var}[\tilde{f}(\mathbf{x}_0)] = \text{Var}[\mathbf{x}_0^T\tilde{\boldsymbol{\beta}}] = \mathbf{x}_0^T\text{Var}[\tilde{\boldsymbol{\beta}}]\mathbf{x}_0 = \sigma^2\mathbf{x}_0^T W_\lambda W_\lambda^T\mathbf{x}_0$$

Problem 1c)

For calculating the expected value of the mean-square error, it is given that

$$E[\text{MSE}] = E[(y_0 - \tilde{f}(\mathbf{x}_0))^2] = (E[\tilde{f}(\mathbf{x}_0) - f(\mathbf{x}_0)])^2 + \text{Var}[\tilde{f}(\mathbf{x}_0)] + \text{Var}[\varepsilon].$$

The results obtained in **b)** yields

$$(E[\tilde{f}(\mathbf{x}_0) - f(\mathbf{x}_0)])^2 = (\mathbf{x}_0^T(\boldsymbol{\beta} - \lambda(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\boldsymbol{\beta}) - \mathbf{x}_0^T\boldsymbol{\beta})^2 = (\lambda\mathbf{x}_0^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\boldsymbol{\beta})^2.$$

Thus, the final expression is

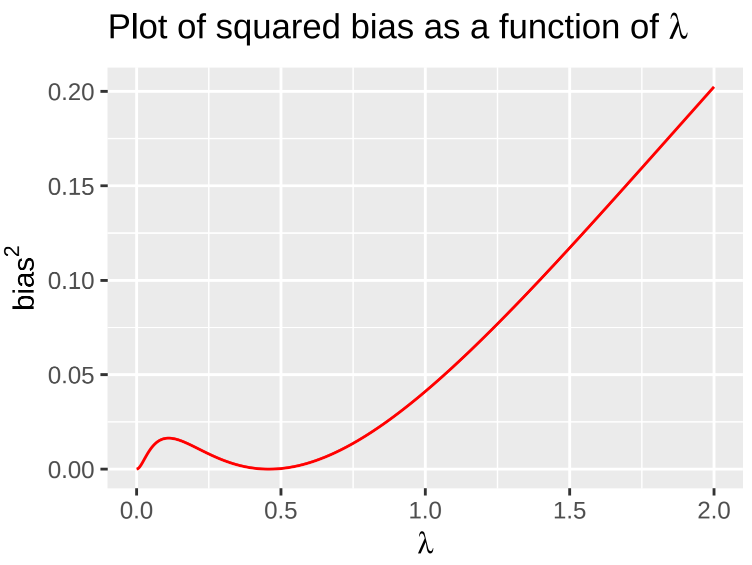
$$E[\text{MSE}] = (\lambda\mathbf{x}_0^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\boldsymbol{\beta})^2 + \sigma^2\mathbf{x}_0^T W_\lambda W_\lambda^T\mathbf{x}_0 + \sigma^2.$$

In this expression, the first term is interpreted as the square bias, the second term is the variance, and the last term is the irreducible error.

Problem 1d)

To calculate and plot the squared bias at $\mathbf{x}_0$ for different values of λ the following code was used

```r
# function for calculating the bias term
bias = function(lambda, X, x0, beta) {
    p = ncol(X)
    value = (lambda * t(x0) %*% solve(t(X) %*%
        X + lambda * diag(p)) %*% beta)^2
    return(value)
}

# Create a linspace for lambda from 0 to
# 2
lambdas = seq(0, 2, length.out = 500)
BIAS = rep(NA, length(lambdas))

# Calculate bias along the linspace of
# lambdas
for (i in 1:length(lambdas)) BIAS[i] = bias(lambdas[i],
    X, x0, beta)

# Make dataframe
dfBias = data.frame(lambdas = lambdas, bias = BIAS)

# Plotting
ggplot(dfBias, aes(x = lambdas, y = bias)) +
    geom_line(color = "red") + xlab(expression(lambda)) +
    ylab(expression(bias^2)) + ggtitle(expression(paste("Plot of squared bias as a function of ",
    lambda)))
```

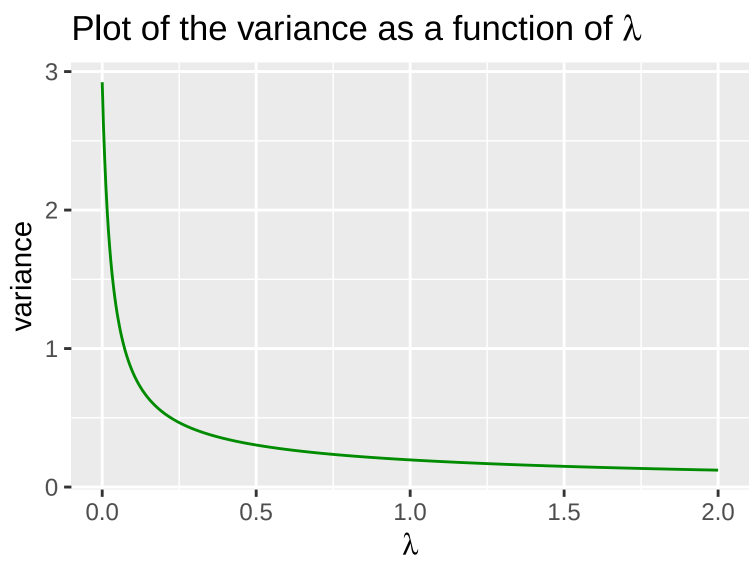


From the plot one can see that initially the bias is small and that it increases slightly, before decreasing until $\lambda = 0.5$. The bias is strictly increasing with values of λ greater than 0.5. The plot indicates that one can potentially obtain an overall smaller MSE with a $\lambda \neq 0$.

Problem 1e)

The following code was used to calculate the variance at $\mathbf{x}_0$ for different values of λ.

```r
# Function for calculating the variance
variance = function(lambda, X, x0, sigma) {
    p = ncol(X)
    W = solve(t(X) %*% X + lambda * diag(p)) %*%
        t(X)
    value = sigma^2 * t(x0) %*% W %*% t(W) %*%
        x0
    return(value)
}

# Linspace the lambdas once again
lambdas = seq(0, 2, length.out = 500)
VAR = rep(NA, length(lambdas))

# Calculate the variance term along the
# lambda linspace
for (i in 1:length(lambdas)) VAR[i] = variance(lambdas[i],
    X, x0, sigma)

# Make dataframe
dfVar = data.frame(lambdas = lambdas, var = VAR)

# Plotting
ggplot(dfVar, aes(x = lambdas, y = var)) +
    geom_line(color = "green4") + xlab(expression(lambda)) +
    ylab("variance") + ggtitle(expression(paste("Plot of the variance as a function of ",
    lambda)))
```



The plot shows that the variance is decreasing with greater values of λ. This, combined with the result from **d)**, further indicates that a $\lambda \neq 0$ can yield a lower MSE than one would obtain using $\lambda = 0$.

Problem 1f)

Using the expression derived in **c)**, one gets:

```
exp_mse = BIAS + VAR + sigma^2
```

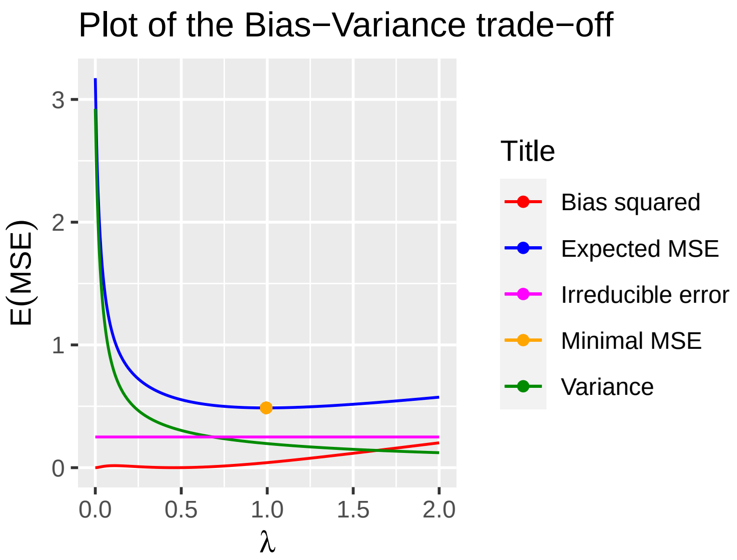
This is now a vector containing the expected value of the mean square error at discrete values of λ between 0 and 2. In order to find the value of lambda that minimizes the mean square error, the following function was used

```
lowest = lambdas[which.min(exp_mse)]  #Finds the lambda which minimizes the function
```

The found value was that $\lambda = 0.993988$ yields the lowest mean square error. For plotting the bias-variance trade-off the following code was used

```
# Create dataframe
dfAll = data.frame(lambda = lambdas, bias = BIAS,
    var = VAR, exp_mse = exp_mse)

# Plotting
plot = ggplot(dfAll) + geom_line(aes(x = lambda,
    y = exp_mse, color = "Expected MSE"))  # Add plot for expected MSE
plot = plot + geom_line(aes(x = lambda, y = bias,
    color = "Bias squared"))  #Add plot for bias
plot = plot + geom_line(aes(x = lambda, y = var,
    color = "Variance"))  #Add plot for variance
```

```
plot = plot + geom_point(aes(y = exp_mse[which.min(exp_mse)],
    lowest, color = "Minimal MSE"))  #Add point at lowest MSE
plot = plot + geom_line(aes(x = lambda, y = sigma^2,
    color = "Irreducible error"))  #Plot irreducible error
plot = plot + labs(x = expression(lambda),
    y = expression(E(MSE)), color = "Title")  #Adjust labels
plot = plot + ggtitle("Plot of the Bias-Variance trade-off")  #Add title
plot = plot + scale_color_manual(values = c(`Bias squared` = "red",
    Variance = "green4", `Expected MSE` = "blue",
    `Minimal MSE` = "orange", `Irreducible error` = "magenta"))
plot  #Show the plot
```



From the graph for the MSE, one sees that it decreases until it reaches $\lambda = 0.993988$. This is expected, as one could see from **d)** that the bias was not strictly increasing with values of $\lambda > 0$, and from **e)** that the variance is strictly decreasing for values of $\lambda > 0$.

Problem 2

Problem 2a)

Here one can see a table with the number of deceased (1) and non-deceased (0) in the whole dataset.

```
table(d.corona[, 1])
```

```
##
##    0    1
## 1905  105
```

Here one can see a table with the number of males and females in each country.

```
table(d.corona[, c(2, 4)])
```

```
##         country
## sex      France indonesia japan Korea
##   female     60        30   120   879
##   male       54        39   174   654
```

Here one can see a table with the number of deceased and non-deceased for each sex.

```
table(d.corona[c(1, 2)])
```

```
##         sex
## deceased female male
##        0   1046  859
##        1     43   62
```

Here one can see a table with the number of deceased and non-deceased for each sex in France.

```
table(subset(d.corona, country == "France")[,
    c(1, 2)])
```

```
##         sex
## deceased female male
##        0     55   43
##        1      5   11
```

Problem 2b)

When choosing a model for this problem, several considerations were made. First, a linear model gave negative probabilities. Second, the questions in this problem would be hard to answer using LDA, QDA or KNN, due to inference being important. Therefore we chose a logistic regression model. The code and model parameters are shown below.

```
deceased.logm <- glm(deceased ~ ., data = d.corona,
    family = "binomial")  #Training a log.regr. model on dataset.
summary(deceased.logm)$coefficients
```

```
##                     Estimate  Std. Error    z value     Pr(>|z|)
## (Intercept)      -3.99348478 0.462190319 -8.6403471 5.604250e-18
## sexmale           0.62606777 0.209044668  2.9948995 2.745353e-03
## age               0.02713421 0.004736262  5.7290352 1.010034e-08
## countryindonesia -0.41185539 0.550050523 -0.7487592 4.540024e-01
## countryjapan     -1.34338289 0.417195836 -3.2200295 1.281774e-03
## countryKorea     -0.77389515 0.307979819 -2.5128112 1.197734e-02
```

i)

```
koreanMan75 <- data.frame(sex = "male", age = 75,
    country = "Korea")  #Creating a 74 yrs old Korean man.
koreanMan75.prediction <- predict.glm(deceased.logm,
    koreanMan75, type = "response")  #Predicting
koreanMan75.prediction
```

```
##         1
## 0.1084912
```

The logistic regression model predicts a 0.108 chance of dying for a 75 year old korean man.

ii) The estimated predictor for the sexmale-variable is 0.626 with p-value 0.00275. Since the p-value is quite small, there is most likely a relation between dying and one's sex, and since the sexmale-predictor is positive, this indicates a higher probability to die for a male than a female.

iii)

```
chis = anova(deceased.logm, test = "Chisq")  #Run a chi-square test
chis$`Pr(>Chi)`[4]  #Prints the p-value from the chi-square test for the country predictor
```

```
## [1] 0.01139039
```

The p-value of the Chi-squared test for the country-variable is 0.0114. This is a pretty small p-value, which indicates a relation between dying and one's country of residence.

iv) Let A be a person and A_{+10} be an identical person but whom is ten years older. Then for a logistic regression model, the following holds:

$$\frac{Odds(A_{+10})}{Odds(A)} = e^{10\beta_{age}}.$$

Hence, the odds to die increase by a factor of $\left(e^{\beta_{age}}\right)^{10} \approx 1.31$ for an identical person but whom is 10 years older.

Problem 2c)

To find out whether two covariates influence each other, one can create a logistic regression model with interaction terms between the two covariates of interest. In order not to waste information, all covariates are included in the models.

i)

```
deceased_ageSex.logm <- lm(deceased ~ country +
    age * sex, data = d.corona)  #Fitting the model
summary(deceased_ageSex.logm)$coefficients  #Prints out the model parameters
```

```
##                       Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept)       0.0592152088 0.0269781549  2.1949317 2.828290e-02
## countryindonesia -0.0530392792 0.0335715970 -1.5798855 1.142910e-01
## countryjapan     -0.0981283873 0.0242629750 -4.0443675 5.445281e-05
## countryKorea     -0.0703969077 0.0215553878 -3.2658613 1.109733e-03
## age               0.0009768171 0.0002989220  3.2677994 1.102203e-03
## sexmale          -0.0034560352 0.0235701992 -0.1466273 8.834409e-01
## age:sexmale       0.0006901775 0.0004307976  1.6020924 1.092928e-01
```

The p-value of the interaction term between age and sex is 0.109, which gives a large probability for the interaction term to not have any effect on the probability to decease. It is not unlikely that age is a greater risk factor for men than for women, but the the p-value is too large to make a conclusion.

ii)

```r
deceased_ageEthnic.logm <- lm(deceased ~
    sex + age * country, data = d.corona)
summary(deceased_ageEthnic.logm)$coefficients
```

```
##                           Estimate   Std. Error   t value     Pr(>|t|)
## (Intercept)           -0.177067516 0.0584394403 -3.029932 2.477409e-03
## sexmale                0.030091584 0.0099039932  3.038328 2.409716e-03
## age                    0.004829619 0.0008709502  5.545229 3.324787e-08
## countryindonesia       0.246837155 0.0934679643  2.640874 8.333559e-03
## countryjapan           0.161257976 0.0684297747  2.356547 1.854164e-02
## countryKorea           0.154892884 0.0598234317  2.589167 9.690561e-03
## age:countryindonesia -0.005118413 0.0016265502 -3.146791 1.675029e-03
## age:countryjapan     -0.004195317 0.0010561854 -3.972140 7.375110e-05
## age:countryKorea     -0.003642209 0.0009056080 -4.021838 5.988977e-05
```

The p-value for the interaction term between age and countryindonesia is 0.00168, which is very low. Since countryFrance is the reference category and the estimate for $\beta_{age:countryindonesia}$ is negative, this p-value indicates that age is a greater risk for the French population than for the Indonesian population.

Problem 2d)

i) True, ii) True, iii) True, iv) False

Problem 3

problem 3a)

i) Using the expression for p_i one obtains the following expression using the log-odds function

$$\text{logit}(p_i) = \log(\frac{p_i}{1-p_i}) = \log(p_i) - \log(1-p_i) = \log(\frac{e^{\beta_0+\mathbf{x}^T\boldsymbol{\beta}}}{1+e^{\beta_0+\mathbf{x}^T\boldsymbol{\beta}}}) - \log(1 - \frac{e^{\beta_0+\mathbf{x}^T\boldsymbol{\beta}}}{1+e^{\beta_0+\mathbf{x}^T\boldsymbol{\beta}}})$$

This expression can be further simplified as

$$\text{logit}(p_i) = \log(e^{\beta_0+\mathbf{x}^T\boldsymbol{\beta}}) - \log(1+e^{\beta_0+\mathbf{x}^T\boldsymbol{\beta}}) - \log(\frac{1+e^{\beta_0+\mathbf{x}^T\boldsymbol{\beta}} - e^{\beta_0+\mathbf{x}^T\boldsymbol{\beta}}}{1+e^{\beta_0+\mathbf{x}^T\boldsymbol{\beta}}}) = \beta_0 + \mathbf{x}^T\boldsymbol{\beta} - \log(1+e^{\beta_0+\mathbf{x}^T\boldsymbol{\beta}}) - \log(\frac{1}{1+e^{\beta_0+\mathbf{x}^T\boldsymbol{\beta}}}).$$

And so, the expression becomes

$$\text{logit}(p_i) = \beta_0 + \mathbf{x}^T\boldsymbol{\beta} - \log(1+e^{\beta_0+\mathbf{x}^T\boldsymbol{\beta}}) - \log(1) + \log(1+e^{\beta_0+\mathbf{x}^T\boldsymbol{\beta}}) = \beta_0 + \mathbf{x}^T\boldsymbol{\beta}.$$

Which indeed shows that the log-odds function is a linear function of the covariates.

ii) To find the parameters for the logistic regression method, the following code was used

```r
# Estimate parameters
diabetes.glm = glm(diabetes ~ ., data = train,
    family = "binomial")
diabetes.glm$coefficients
```

```
##  (Intercept)        npreg          glu           bp         skin          bmi 
## -10.58353812   0.10510941   0.03558603  -0.01465435   0.02037865   0.09468312 
##          ped          age 
##   1.93166638   0.03829093
```

Here, one can see the parameter values for the corresponding predictors. Now that the model has been trained, it can be used to predict the response from the testing set of predictors, and compare it to the actual response in a confusion matrix. To do this, the following code was used

```r
# Logisitc regression, predict response
# using the model.
glm.results = predict(diabetes.glm, newdata = test,
    type = "response")  #Returns probability of having diabetes
# Make a table with predictions and true
# values. Make confusion matrix
response = ifelse(test$diabetes == 1, "Diabetic",
    "Non-diabetic")  #Changes from 1/0 to diabetes/not-biabetes
glm.pred = ifelse(round(glm.results, 0) ==
    1, "Diabetic", "Non-diabetic")  #save the results with new labels. Cuts off at 0.5 with round()

# Make confusion table
tab = table(Predicted = glm.pred, Response = response)  #Using round to simulate 0.5 cutoff
confmat = confusionMatrix(tab, positive = "Diabetic")  #For calculating specificity and sensitivity
confmat$table  #Show the results
```

```
##               Response
## Predicted      Diabetic Non-diabetic
##   Diabetic           48           18
##   Non-diabetic       29          137
```

```r
confmat$byClass[1]  #Sensitivity
```

```
## Sensitivity 
##   0.6233766
```

```r
confmat$byClass[2]  #Specificity
```

```
## Specificity 
##    0.883871
```

Thus, one sees that the sensitivity of the model is 0.6233766, and the specificity is 0.883871.

Problem 3b)

i) π_k is the fraction of a population that belong to class k. It can be interpreted as the probability of being in class k with no knowledge of the predictors. Thus π_0 will be the fraction of people whom does not have diabetes, and π_1 is the fraction of people whom have diabetes. In the model, π_k is estimated by taking the fraction of observations classified to class k divided by the number of observations.

$\boldsymbol{\mu}_k$ is the vector of expected values of the predictors for observations that belong to class k. Thus, $\boldsymbol{\mu}_0$ is the vector containing the expected values for bmi, age, glu etc among the non-diabetic participants. Then, $\boldsymbol{\mu}_1$ is the vector containing the excpected values for diabetic participants. In the model $\boldsymbol{\mu}_k$ is estimated by taking the mean over each value for each class.

$\boldsymbol{\Sigma}$ is the covariance matrix of the predictors. It is assumed, through lda, that $\boldsymbol{\Sigma}_k$ will be the same regardless of the class. The matrix stores the variance for the different predictors and the covariances between them. In this case these will be the variances in bmi, age, glu etc and the covariances between these.

$f_k(x)$ is the probability density function for class k. Since the different classes usually have different expected-values, the response for each class will be distributed differently. This is a key point as it is used in posterior analysis to determine which class the observed set of predictors most likely belong to.

ii) Here the models are fitted and the confusion tables created. The standard cut-off for these functions are at 0.5, so it does not have to be done manually.

```r
# LDA, estimate parameters
diabetes.lda = lda(diabetes ~ ., data = train,
    type = "response")  #train model
lda.results = predict(diabetes.lda, test)  #Predict with 0.5 cutoff
lda.pred = ifelse(lda.results$class == 1,
    "Diabetic", "Non-diabetic")

# Make confusion matrix for LDA
confmatlin = confusionMatrix(table(Predicted = lda.pred,
    Response = response), positive = "Diabetic")
cat("Confusion table for LDA")
```

```
## Confusion table for LDA
```

```r
confmatlin$table  #Display
```

```
##                 Response
## Predicted         Diabetic Non-diabetic
##   Diabetic              47           17
##   Non-diabetic          30          138
```

```r
# QDA, estimate parameters
diabetes.qda = qda(diabetes ~ ., data = train,
    type = "response")  #Train model
qda.results = predict(diabetes.qda, test)  #Predict results
qda.pred = ifelse(qda.results$class == 1,
    "Diabetic", "Non-diabetic")

# Make confusion matrix for QDA
confmatquad = confusionMatrix(table(Predicted = qda.pred,
    Response = response), positive = "Diabetic")
cat("Confusion table for QDA")
```

```
## Confusion table for QDA
```

```r
confmatquad$table  #Show result
```

```
##                 Response
## Predicted      Diabetic Non-diabetic
##   Diabetic           45           24
##   Non-diabetic       32          131
```

The difference between QDA and LDA is that in QDA, the variance-covariance matrix depends on the class, whereas in LDA it does not. This results in different shapes in the decision boundaries. LDA will produce linear decision boundaries and QDA will produce curved boundaries. Since QDA is more flexible it is expected to have higher variance and less bias than LDA.

Problem 3c)

i) The KNN method will classify a new observation by finding the k nearest observations, and classify the new observation as the class that appeared most frequent in those k neighbors.

ii) For choosing the optimal k, one could use cross validation for different values of k, and choose the k that yields the lowest error.

iii) The confusion matrix from the KNN-method is shown below.

```r
# K-nearest-neighbors, make model and
# confusion matrix
diabetes.knn = knn(train, test, train$diabetes,
    k = 25)  #Train model and predict with 0.5 cutoff
# Label the results
knn.pred = ifelse(diabetes.knn == 1, "Diabetic",
    "Non-diabetic")

# Create confusion matrix for KNN
confmatknear = confusionMatrix(table(Predicted = knn.pred,
    Response = response), positive = "Diabetic")
confmatknear$table  #Show the matrix
```

```
##                 Response
## Predicted      Diabetic Non-diabetic
##   Diabetic           41           11
##   Non-diabetic       36          144
```

The function also calculates the specificity to be 0.5324675 and the sensitivity to be 0.9290323, but these results will also be derived. From the confusion table one can read that the model correctly identifies 144 true negatives and 41 true positives. However it also yields 36 false negatives and 11 false positives. Thus there is some error, but this is to be expected of any method. The sensitivity of the model is the fraction of true positives to the sum of true positives and false negatives. Mathematically this means

$$\text{Sensitivity} = \frac{TP}{TP + FN} = \frac{41}{41 + 36} \approx 0.5325$$

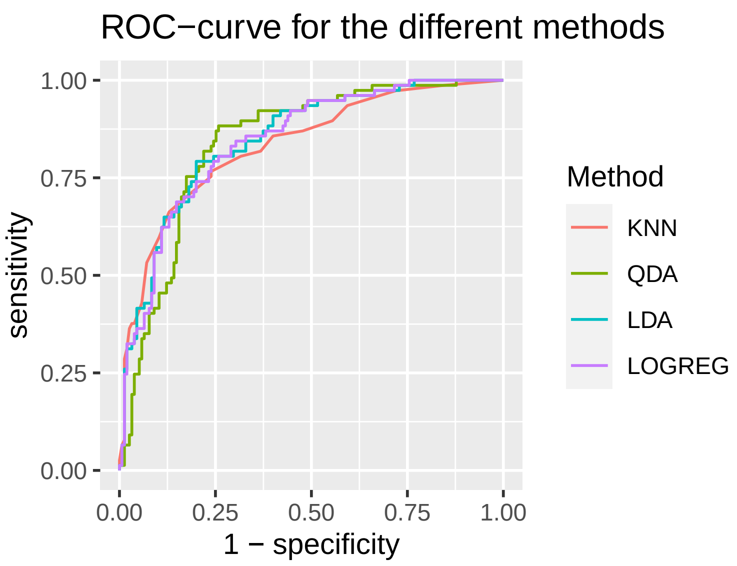
The specificity of the model is the fraction of true negatives to the sum of true negatives and false positives. Mathematically this is

$$\text{Specificity} = \frac{TN}{TN + FP} = \frac{144}{144 + 11} = 0.9290$$

Problem 3d)

Below, there are plots of the ROC-curves for the different methods. This was done using the pROC and ggROC libraries. The code is as follows.

```r
# KNN that returns probabilities instead
# of classifications
knnMod = knn(train, test, train$diabetes,
    k = 25, prob = T)
probKNN = ifelse(knnMod == 0, 1 - attributes(knnMod)$prob,
    attributes(knnMod)$prob)

# Transform the response into a factor
# with levels 1 and 0
response = factor(test$diabetes)

# Create roc-objects for each method
rocknn = roc(response, probKNN)
rocquad = roc(response, qda.results$posterior[,
    2])
roclin = roc(response, lda.results$posterior[,
    2])
roclog = roc(response, glm.results)

# Storing the area under the curve for
# the different methods
list = list(auc(rocknn), auc(rocquad), auc(roclin),
    auc(roclog))

# Plotting the actual curve
ggroc(data = list(KNN = rocknn, QDA = rocquad,
    LDA = roclin, LOGREG = roclog), legacy.axes = TRUE) +
    ggtitle("ROC-curve for the different methods") +
    labs(color = "Method")
```



```
## Area under curve for method KNN is 0.833431085043988
## Area under curve for method QDA is 0.84147465437788
## Area under curve for method LDA is 0.849015500628404
## Area under curve for method LOGREG is 0.845077503142019
```

From the plot one can see that the methods performs very similar, with some fluctuation among them. By looking at the area under the curve, one can see that KNN performs the worst while LDA and logistic regression performs the best. If the goal is to have an interpretable model, then the logistic regression model should be used as it performs well and allows for inference.

Problem 4

Problem 4a)

Hint 1: predicted value of y with the i-th observation left out

$$\hat{y}_{(-i)} = \mathbf{x}_i^T \hat{\boldsymbol{\beta}}_{(-i)}$$

Hint 2:

$$\begin{aligned} X_{(-i)}^T X_{(-i)} &= [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, ..., \mathbf{x}_N] \cdot [x_1^T, x_2^T, ..., \mathbf{x}_{i-1}^T, \mathbf{x}_{i+1}^T, ..., \mathbf{x}_N^T]^T \\ &= \sum_{j=1, j\neq i}^{N} \mathbf{x}_j \mathbf{x}_j^T = \sum_{j=1}^{N} \mathbf{x}_j \mathbf{x}_j^T - \mathbf{x}_i \mathbf{x}_i^T \\ &= X^T X - \mathbf{x}_i \mathbf{x}_i^T \end{aligned}$$

Hint 2.1:

$$\begin{aligned} X_{(-i)}^T \mathbf{y}_{(-i)} &= [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, ..., \mathbf{x}_N] \cdot [y_1, ..., y_{i-1}, y_{i+1}, ..., y_N]^T \\ &= \sum_{j=1, j\neq i}^{N} \mathbf{x}_j y_j = \sum_{j=1}^{N} \mathbf{x}_j y_j - \mathbf{x}_i y_i \\ &= X^T \mathbf{y} - \mathbf{x}_i y_i \end{aligned}$$

Hint 3:
It is plausible to assume that all the P columns in X are linearly independent.

Thus, $X^T X$ is a positive definite square matrix, I.e. $(X^T X)$ is invertible and $v^T (X^T X) u > 0, \forall v, u \in \mathbb{R}^p$.

Hence, one may use the Sherman-Morrison formula.

$$\hat{\boldsymbol{\beta}}_{(-i)} = (X^T X - \mathbf{x}_i \mathbf{x}_i^T)^{-1} (X^T \mathbf{y} - \mathbf{x}_i y_i)$$

Where $(X^T X - \mathbf{x}_i \mathbf{x}_i^T)^{-1} = (X^T X)^{-1} - \frac{(X^T X)^{-1}(-\mathbf{x}_i)\mathbf{x}_i^T (X^T X)^{-1}}{1 + \mathbf{x}_i^T (X^T X)^{-1}(-\mathbf{x}_i)}$ according to the Sherman-Morrison formula. Define $-h_i = \mathbf{x}_i^T (X^T X)^{-1}(-\mathbf{x}_i)$.

As a result, one gets the following expression $(X^T X - \mathbf{x}_i \mathbf{x}_i^T)^{-1} = (X^T X)^{-1}(I + \frac{\mathbf{x}_i \mathbf{x}_i^T (X^T X)^{-1}}{1-h_i})$.

Therefore:

$$\hat{\boldsymbol{\beta}}_{(-i)} = (X^T X)^{-1}(I + \frac{\mathbf{x}_i \mathbf{x}_i^T (X^T X)^{-1}}{1 - h_i})(X^T \mathbf{y} - \mathbf{x}_i y_i)$$

.

From hint 1 it is known that $\hat{y}_{(-i)} = \mathbf{x}_i^T \hat{\boldsymbol{\beta}}_{(-i)}$.

$$\hat{y}_{(-i)} = \mathbf{x}_i^T((X^TX)^{-1} + \frac{(X^TX)^{-1}\mathbf{x}_i\mathbf{x}_i^T(X^TX)^{-1}}{1-h_i})(X^T\mathbf{y} - \mathbf{x}_i y_i)$$

$$= \mathbf{x}_i^T(X^TX)^{-1}X^T\mathbf{y} - \mathbf{x}_i^T(X^TX)^{-1}\mathbf{x}_i y_i + \frac{\mathbf{x}_i^T(X^TX)^{-1}\mathbf{x}_i\mathbf{x}_i^T(X^TX)^{-1}X^T\mathbf{y}}{1-h_i} - \frac{\mathbf{x}_i^T(X^TX)^{-1}\mathbf{x}_i\mathbf{x}_i^T(X^TX)^{-1}\mathbf{x}_i y_i}{1-h_i}.$$

Remember that $h_i = \mathbf{x}_i^T(X^TX)^{-1}\mathbf{x}_i$. Thus the expression becomes

$$\hat{y}_{(-i)} = \mathbf{x}_i^T(X^TX)^{-1}X^T\mathbf{y} - h_i y_i + \frac{h_i\mathbf{x}_i^T(X^TX)^{-1}X^T\mathbf{y}}{1-h_i} - \frac{h_i^2 y_i}{1-h_i},$$

where $\hat{y}_i = x_i^T(X^TX)^{-1}X^T\mathbf{y}$.

This yields

$$\hat{y}_{(-i)} = \hat{y}_i - h_i y_i + \frac{h_i\hat{y}_i}{1-h_i} - \frac{h_i^2 y_i}{1-h_i}$$

$$= \frac{\hat{y}_i - h_i y_i}{1-h_i}.$$

Therefore, the expression for CV becomes as follows $CV = \frac{1}{N}\sum_i MSE_i = \frac{1}{N}\sum(y_i - \hat{y}_{(-i)})^2$ $y_i - \frac{\hat{y}_i - h_i y_i}{1-h_i} = \frac{y_i - h_i y_i + h_i y_i - \hat{y}_i}{1-h_i} = \frac{y_i - \hat{y}_i}{1-h_i}$.

Thus: $CV = \frac{1}{N}\sum_{i=1}^{N}(\frac{y_i - \hat{y}_i}{1-h_i})$

Problem 4b)

i) False, ii) True, iii) False, iv) False

Problem 5

Problem 5a)

```r
# Fit the regression model with bodyfat
# as response, and age, weight and bmi as
# predictors
bodyfat.lm <- lm(bodyfat ~ age + weight +
    bmi, data = d.bodyfat)

summary(bodyfat.lm)$r.square  #Calculates R^2 for the model and prints it
```

```
## [1] 0.5803041
```

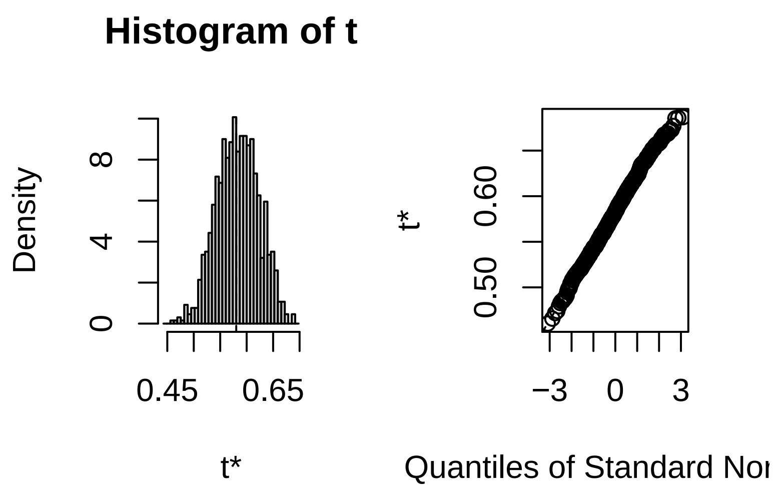
As one can see above, a linear regression model has been used to model bodyfat as the response with age, weight and bmi as predictor variables. R-squared has been calculated to be $R^2 = 0.58$.

Problem 5b)

```r
set.seed(4268)  #For consistent results

# Function fits a linear regression model
# and returns R^2
boot.fnc <- function(formula, data, indices) {
    d <- data[indices, ]
    fitBoot <- lm(formula, data = d)
    return(summary(fitBoot)$r.square)
}

# Generating 1000 bootstrap samples
rsquared.boot <- boot(data = d.bodyfat, statistic = boot.fnc,
    R = 1000, formula = bodyfat ~ age + weight +
        bmi)

# Display the result of the bootstraping
cat(paste("Mean value of R^2:", signif(rsquared.boot$t0,
    3), ", Std. error:", signif(sqrt(var(rsquared.boot$t)),
    3), "\n"))
```

```
## Mean value of R^2: 0.58 , Std. error: 0.0396
```

```r
# Plotting the distribution of R^2
plot(rsquared.boot)
```



```r
# Calculates confidence interval for the
# R^2 estimate
confval = boot.ci(rsquared.boot, type = "bca")  #Create confidence interval for R^2
cat(paste("Lower bound:", signif(confval$bca[4],
    3), ", Upper bound:", signif(confval$bca[5],
    3)))
```

```
## Lower bound: 0.505 , Upper bound: 0.657
```

iii) As one can see in from the “Bootstrap Statistics” above, the standard error for R^2 is 0.0396. A 95% confidence interval for R^2 is
$0.505 < R^2 < 0.657$.

iv) This indicates that what seems like a certain value, actually is a point-estimate and is surprisingly uncertain. One must therefore have this uncertainty in mind when working with the R^2-value.