Compulsory exercise 1: Group 2

TMA4268 Statistical Learning V2019

Anders Thallaug Fagerli, Rebecca Sandstø

22 februar, 2019

Problem 1: Multiple linear regression

```r
library(GLMsData)
data("lungcap")
lungcap$Htcm=lungcap$Ht*2.54
modelA = lm(log(FEV) ~ Age + Htcm + Gender + Smoke, data=lungcap)
summary(modelA)
```

```
##
## Call:
## lm(formula = log(FEV) ~ Age + Htcm + Gender + Smoke, data = lungcap)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.63278 -0.08657  0.01146  0.09540  0.40701
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.943998   0.078639 -24.721  < 2e-16 ***
## Age          0.023387   0.003348   6.984  7.1e-12 ***
## Htcm         0.016849   0.000661  25.489  < 2e-16 ***
## GenderM      0.029319   0.011719   2.502   0.0126 *
## Smoke       -0.046067   0.020910  -2.203   0.0279 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1455 on 649 degrees of freedom
## Multiple R-squared:  0.8106, Adjusted R-squared:  0.8095
## F-statistic: 694.6 on 4 and 649 DF,  p-value: < 2.2e-16
```

Q1:

In multiple linear regression, we have a reponse variable $\boldsymbol{Y}$ written as a function of the observations $\boldsymbol{X}$, regression parameters $\boldsymbol{\beta}$ and random errors $\boldsymbol{\epsilon}$, where $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$. Taking the logarithm of our response, `modelA` can then be written as:

$$\log(\text{FEV}) = \beta_0 + \beta_1\text{Age} + \beta_2\text{Htcm} + \beta_3\text{GenderM} + \beta_4\text{Smoke} + \epsilon$$

We can then fit this model by estimating the parameters $\boldsymbol{\beta}$, so that $\hat{\boldsymbol{Y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$. Using the estimated parameters from the `summary`, the fitted `modelA` can be written as:

$$\widehat{\log(\text{FEV})} = -1.944 + 0.0234 \cdot \text{Age} + 0.0169 \cdot \text{Htcm} + 0.0293 \cdot \text{GenderM} + 0.0461 \cdot \text{Smoke}$$

Q2:

- `Estimate`: The estimated regression coefficients, $\hat{\boldsymbol{\beta}}$. Assuming a normal linear regression model, these coefficients are found by either minimizing the residual sum of squares RSS or maximizing the likelihood function with respect to $\boldsymbol{\beta}$, resulting in the estimated vector of coefficients $\hat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{\mathbf{-1}}\mathbf{X}^{\mathbf{T}}\mathbf{Y}$. The regression coefficients explain how the different covariates affect the response, as the response is linearly proportional to all coefficients. Ex. an increase in `Age` will tend to an increase in the response `FEV`, as the associated coefficient to `Age` is positive. In contrast, a person who smokes will tend to a decrease in the response `FEV`, as the associated coefficient to `Smoke` is negative. As the response is linearly proportional to a coeffiecient, an increase in a covariate from x_j to $x_j + 1$ will result in an increase in the response by $\hat{\beta}_j$ when all other covariates are held constant. Ex. an increase in `Age` from 15 to 16 will lead to an increase in log(FEV) by 0.0234. The intercept $\hat{\beta}_0$ is not proportional to any covariate, meaning it is the response value when all covariates are set to zero. The intercept alone does not give any meaningful information in our case, as it reports a negative forced expiratory volume in litres. This means the regression model is only valid within an intervall of values for the covariates. Ex. `Age` should be within 3 and 19, as the data is collected from people aged 3 - 19.

- `Std.Error`: The standard error of each estimated coefficient, $\text{SD}(\hat{\beta}_j) = \sqrt{\text{Var}(\hat{\beta}_j)}$. This value quantifies the amount of variation in the estimated coefficients from the estimated values, and is found in the diagonal of the covariance matrix for $\hat{\beta}$, given by $\text{Cov}(\hat{\beta}) = \sigma^2(\mathbf{X}^{\mathbf{T}}\mathbf{X})^{\mathbf{-1}}$. Unless the true value of the variance σ^2 in the random errors ϵ_i is known, this must be estimated. An unbiased estimator for σ^2 is $\hat{\sigma}^2 = RSS/(n-p-1)$, where n is the number of observations and p is the number of covariates.

- `Residual standard error`: The standard deviation of the residuals d, given by $RSE = \sqrt{RSS/(n-p-1)}$. A residual d_i is an estimate of a random error ϵ_i, by taking $d_i = Y_i - \hat{Y}_i$ for some observation i.

- `F-statistic`: Used to test if the regression is significant, meaning to test if any of the coefficients are different from zero. This can be expressed as the hypothesis

  $$H_0 : \beta_1 = \beta_2 = \cdots = \beta_p = 0 \quad vs. \quad H_1 : \text{at least one different from zero}$$

  F-tests are used to perform the hypothesis test, using the `F-statistic`

  $$F = \frac{(TSS - RSS)/p}{RSS/(n-p-1)} \sim F_{p,n-p-1}$$

  If F is greater than 1, we expect H_0 to be false, and the regression to be significant. The p-value can be calculated with the `F-statistic` by using the upper tail of the F-distribution, and the hypothesis can be concluded by setting an appropriate significance level α. The small p-value calculated from our sample implies that a linear regression model fits the data well.

Q3:

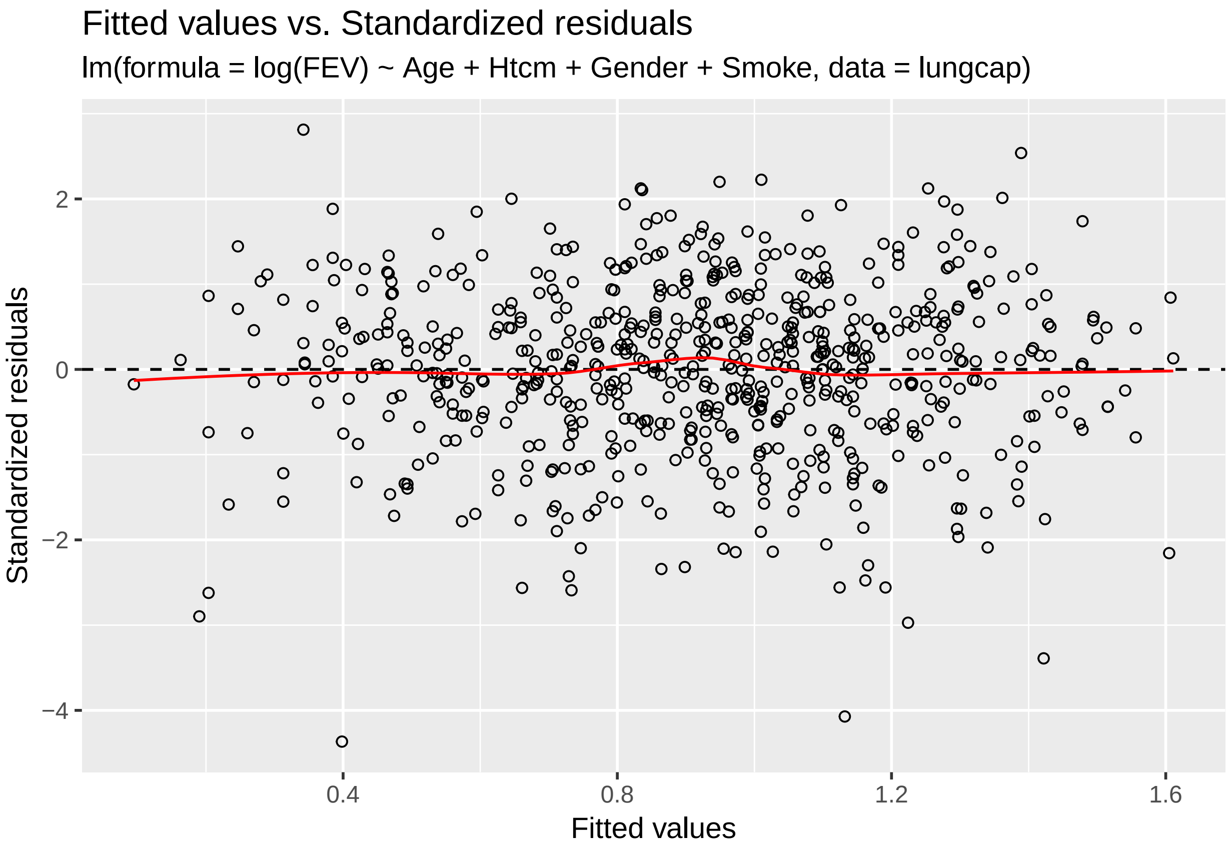
The proportion of variability is given by the coefficient of determination

$$R^2 = \frac{TSS - RSS}{TSS} = 0.8106$$

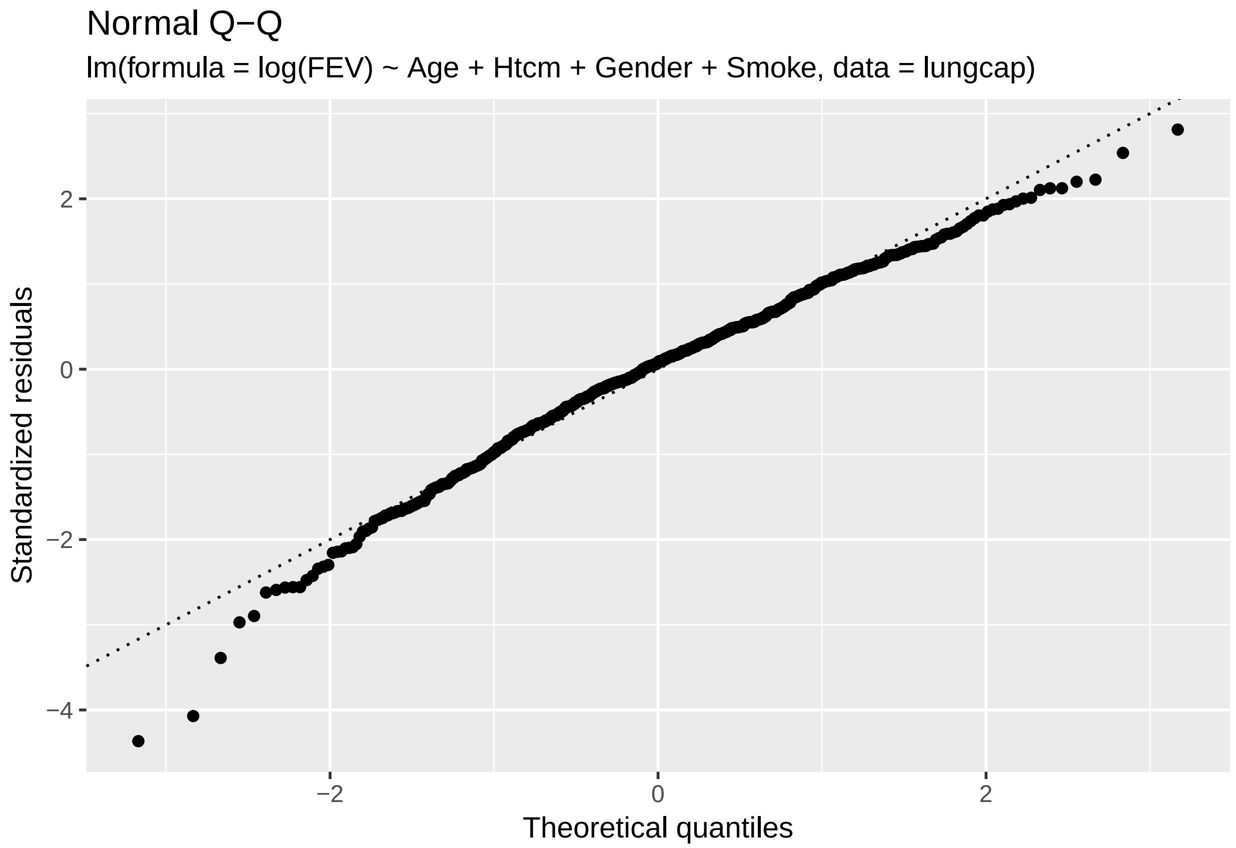
This means 81% of the variance in $\boldsymbol{Y} = \log(\text{FEV})$ is explained in our model, implying a good fit. The closer the coefficient is to 1, the better, as $R^2 = 1$ means all residuals are zero.

Q4:

```r
library(ggplot2)
# residuls vs fitted
ggplot(modelA, aes(.fitted, .stdresid)) + geom_point(pch = 21) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  geom_smooth(se = FALSE, col = "red", size = 0.5, method = "loess") +
  labs(x = "Fitted values", y = "Standardized residuals",
       title = "Fitted values vs. Standardized residuals",
       subtitle = deparse(modelA$call))
```



```r
# qq-plot of residuals
ggplot(modelA, aes(sample = .stdresid)) +
  stat_qq(pch = 19) +
  geom_abline(intercept = 0, slope = 1, linetype = "dotted") +
  labs(x = "Theoretical quantiles", y = "Standardized residuals",
       title = "Normal Q-Q", subtitle = deparse(modelA$call))
```



```
# normality test
library(nortest)
ad.test(rstudent(modelA))
```

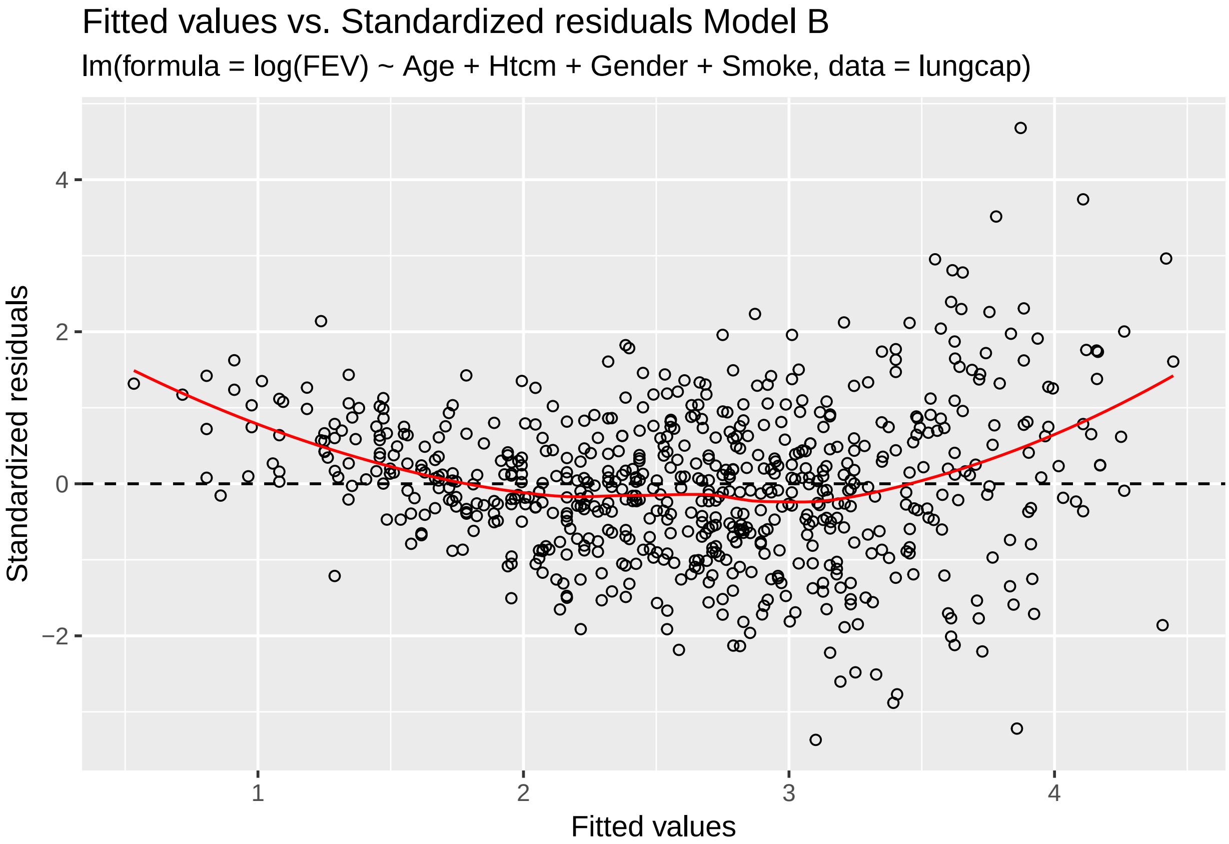
```
##
##  Anderson-Darling normality test
##
## data:  rstudent(modelA)
## A = 1.9256, p-value = 6.486e-05
```

Fitted values vs. Standardized residuals: This plot is used to test the assumption of a linear relationship between the covariates and the response by plotting residuals against fitted values of the model. For a linear relationship, we expect the average of the residuals to form a straight line across the fitted values. The assumption of a constant variance in the random errors can also be seen from the plot, where we expect a constant spread across all fitted values. From our plot, we can see that the residuals have a linear pattern and that the spread is seemily constant for all fitted values. This implies that our data fit some of the assumptions needed for a linear regression model.

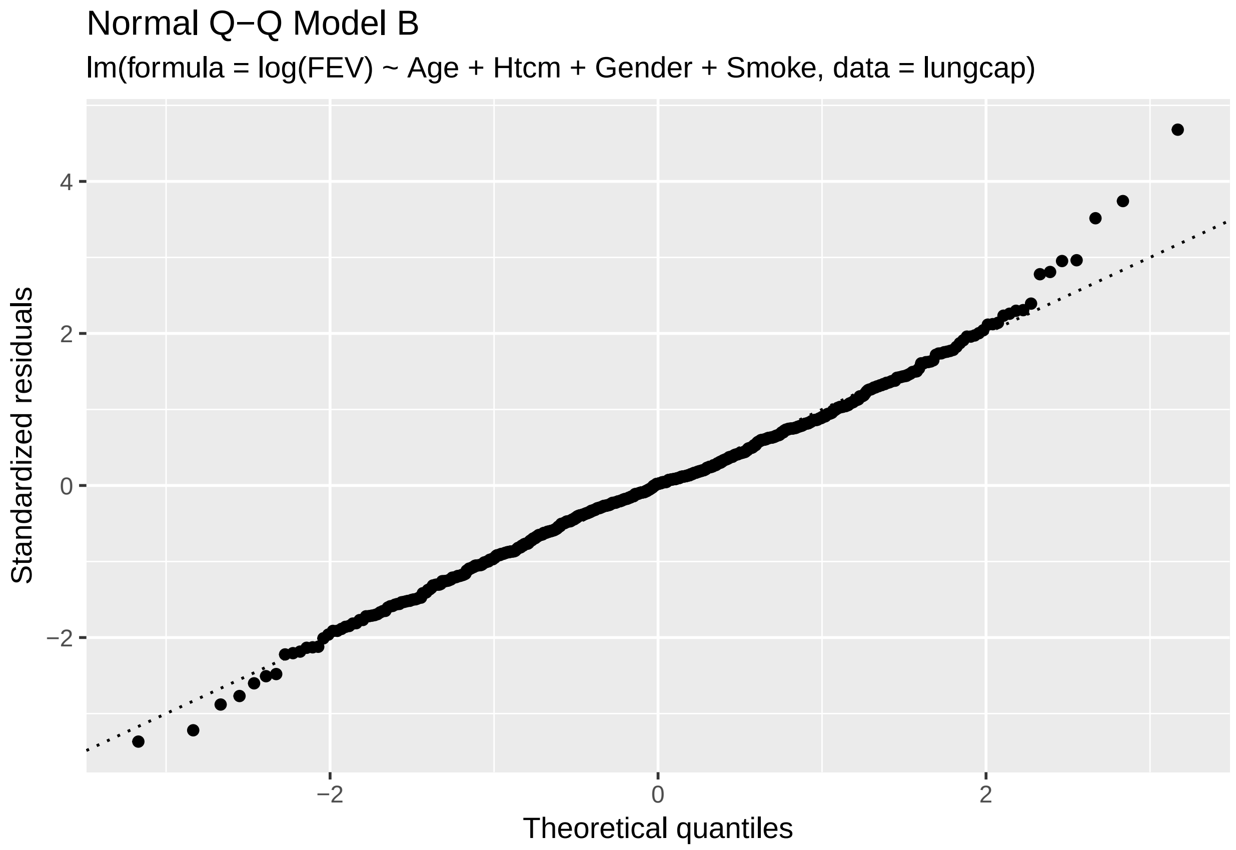
QQ-plot of standardized residuals: The QQ-plot shows whether the residuals are normally distributed or not, by comparing the distribution of the residuals against the normal distribution depicted by the straight line. The residuals form a small curve on top of the normal distribution with greater deviation at the tails, implying that they may not be normally distributed. This is further reinforced by the Anderson-Darling normality test, which rejects the hypothesis H_0 of normality when we apply a standard significance level of $\alpha = 0.05$. This can be seen from the p-value of the test, which is far less than α. The QQ-plot may indicate a violation of the assumption for normally distributed residuals, which again puts more uncertainty on our linear regression model.

Q5:

```r
modelB = lm(FEV ~ Age + Htcm + Gender + Smoke, data=lungcap)
summary(modelB)
library(ggplot2)
# residuals vs fitted Model B
ggplot(modelB, aes(.fitted, .stdresid)) + geom_point(pch = 21) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  geom_smooth(se = FALSE, col = "red", size = 0.5, method = "loess") +
  labs(x = "Fitted values", y = "Standardized residuals",
       title = "Fitted values vs. Standardized residuals Model B",
       subtitle = deparse(modelA$call))
```



```r
# qq-plot of residuals
ggplot(modelB, aes(sample = .stdresid)) +
  stat_qq(pch = 19) +
  geom_abline(intercept = 0, slope = 1, linetype = "dotted") +
  labs(x = "Theoretical quantiles", y = "Standardized residuals",
       title = "Normal Q-Q Model B", subtitle = deparse(modelA$call))
```



```r
# normality test
library(nortest)
ad.test(rstudent(modelB))
```

```
##
## Call:
## lm(formula = FEV ~ Age + Htcm + Gender + Smoke, data = lungcap)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.37656 -0.25033  0.00894  0.25588  1.92047
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.456974   0.222839 -20.001  < 2e-16 ***
## Age          0.065509   0.009489   6.904 1.21e-11 ***
## Htcm         0.041023   0.001873  21.901  < 2e-16 ***
## GenderM      0.157103   0.033207   4.731 2.74e-06 ***
## Smoke       -0.087246   0.059254  -1.472    0.141
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4122 on 649 degrees of freedom
## Multiple R-squared:  0.7754, Adjusted R-squared:  0.774
## F-statistic:   560 on 4 and 649 DF,  p-value: < 2.2e-16
##
##
```

```
##  Anderson-Darling normality test
##
## data:  rstudent(modelB)
## A = 1.2037, p-value = 0.003853
```

For `modelB`, the standarized residuals vs. fitted values shows a non-linear relationship and non-constant spread. However, the QQ-plot shows a better fit to the normal distribution, but with some outliers at the tails. The Anderson-Darling normality test reports a slightly larger p-value than in `modelA`, indicating that the residuals in `modelB` are more normally distributed, although not enough to justify normality. The R^2-statistic for `modelA` is better than for `modelB`, but the interpretation of the model may be more difficult as we take the logarithm of the response. In total, both models violate some of the assumptions for multiple linear regression and may therefore not be favorable when making inference. If we were to choose a model, `modelA` might be better suited for inference since the residuals show a linear relationship with the fitted values and have a fairly constant spread. Ex. a CI will have less randomness for different values of the fitted data for `modelA` than for `modelB`.

Q6:

Because σ^2 is unknown, we can perform a t -test. The T_0-statistic can be calculated by inserting the values for $\hat{\beta}_{age}$ and $\mathrm{SD}(\hat{\beta}_{age})$, shown in the `summary`, and testing the hypothesis $H_0 : \beta_{age} = 0$.

$$T_0 = \frac{\hat{\beta}_{age} - \mathrm{E}(\hat{\beta}_{age})}{\sqrt{\mathrm{Var}(\hat{\beta}_{age})}} = \frac{\hat{\beta}_{age} - \beta_{age}}{\mathrm{SD}(\hat{\beta}_{age})} = \frac{\hat{\beta}_{age} - 0}{\mathrm{SD}(\hat{\beta}_{age})} = 6.984$$

The p-value can then be calculated by inserting the numerical value of T_0 and calculating the probability of observing what we have observed or something more extreme in direction of the alternative hypothesis. The p-value for this test can be obtained from the `summary`.

$$P(|T| > t_0 | H_0 \text{true}) = P(|T| > 6.984) = 7.1 \cdot 10^{-12}$$

We reject the null hypothesis for a critical region given by $P(|T| > k | H_0\text{true}) = \alpha$. This probability is similar to the one used to calculate the p-value, where α is our significance level. It is then clear that we reject the null hypothesis for $\alpha \geqslant 7.1 \cdot 10^{-12}$.

Q7:

We start constructing the interval from the probability,

$$P(-t_{\frac{\alpha}{2},v} \leqslant T \leqslant t_{\frac{\alpha}{2},v}) = 1 - \alpha.$$

Writing out T in terms of $\hat{\beta}_{age}$, β_{age},$\sqrt{\mathrm{Var}(\hat{\beta}_{age})}$ and solving the inequalities for β_{age}, we can construct the intervall for a given significance α.

$$P(\hat{\beta}_{age} - t_{\frac{\alpha}{2},v}\sqrt{\mathrm{Var}(\hat{\beta}_{age})} \leqslant \beta_{age} \leqslant \hat{\beta}_{age} + t_{\frac{\alpha}{2},v}\sqrt{\mathrm{Var}(\hat{\beta}_{age})}) = 1 - \alpha$$

Our t-distribution has $v = n - p - 1 = 649$ degrees of freedom, approximating a normal distribution. Setting $\alpha = 0.01$ for a $(1-\alpha) \cdot 100\% = 99\%$ CI, we get $t_{\frac{0.01}{2},649} \approx z_{0.005} = 2.576$. Inserting our estimate for β_{age} and $\mathrm{SD}(\hat{\beta}_{age})$, we get the interval,

$$\beta_{age} \in [\hat{\beta}_{age} - 2.576 \cdot \mathrm{SD}(\hat{\beta}_{age}), \hat{\beta}_{age} + 2.576 \cdot \mathrm{SD}(\hat{\beta}_{age})] = [0.0147, 0.0320]$$

This interval can be verified in R with

```
confint(object=modelA, level=0.99)
```

```
##                     0.5 %       99.5 %
## (Intercept) -2.1471551289 -1.740841225
## Age          0.0147367391  0.032037689
## Htcm         0.0151410623  0.018556409
## GenderM     -0.0009546847  0.059593401
## Smoke       -0.1000874831  0.007952411
```

The confidence interval tells us that the true value of β_{age} lies within $[0.0147, 0.0320]$ in 95% of the intervals constructed if we were to make several intervals. The p-value of the test in **Q6** tells us the probability of $\beta_{age} = 0$. Since 0 is not a part of the 99% CI, we know that the p-value for the test is lower than $\alpha = 0.01$.

Q8:

Our best guess can be found by inserting the new observation into the fitted model, which is done in the R code below. The best guess for his log(FEV) is found to be 1.324. A prediction interval can be constructed by using the T-statistic, where we assume the new response is independent of previous observations.

$$T = \frac{Y_0 - \hat{Y}_0 - \mathrm{E}(Y_0 - \hat{Y}_0)}{\sqrt{\mathrm{Var}(Y_0 - \hat{Y}_0)}}$$

With

$$\mathrm{E}(Y_0 - \hat{Y}_0) = \mathrm{E}(Y_0) - \mathrm{E}(\hat{Y}_0) = \boldsymbol{x}_0^T\boldsymbol{\beta} - \boldsymbol{x}_0^T\mathrm{E}(\hat{\boldsymbol{\beta}}) = \boldsymbol{x}_0^T\boldsymbol{\beta} - \boldsymbol{x}_0^T\boldsymbol{\beta} = 0$$

$$\mathrm{Var}(Y_0 - \hat{Y}_0) = \mathrm{Var}(Y_0) + \mathrm{Var}(\hat{Y}_0) = \hat{\sigma}^2 + \boldsymbol{x}_0^T\mathrm{Var}(\hat{\boldsymbol{\beta}})\boldsymbol{x}_0 = \hat{\sigma}^2(1 + \boldsymbol{x}_0^T(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{x}_0)$$

The prediction interval is set up by solving the inequality in $P(-t_{\frac{\alpha}{2},v} \leqslant T \leqslant t_{\frac{\alpha}{2},v}) = 1 - \alpha$ for Y_0, resulting in

$$Y_0 \in [\hat{Y}_0 - t_{\frac{\alpha}{2},v} \cdot \sqrt{\hat{\sigma}^2(1 + \boldsymbol{x}_0^T(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{x}_0)}, \hat{Y}_0 + t_{\frac{\alpha}{2},v} \cdot \sqrt{\hat{\sigma}^2(1 + \boldsymbol{x}_0^T(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{x}_0)}]$$

Using the R code below, we find the 95% prediction interval for log(FEV) to be $Y_0 \in [1.036, 1.611]$. This can be transformed to normal units by taking e to the power of each limit. His FEV is then predicted to be in the interval $Y_0 \in [2.818, 5.010]$ in 95% of the prediction intervals we construct. We can compare the interval with properties of the data to justify whether it is useful or not. The prediction interval is very wide compared to the actual span in the data $(0.8, 5.8)$, and it contains values ranging from the mean to the upper tail of the data. Therefore it does not tell us much, and we conclude that our model is not very good at prediction.

```
new = data.frame(Age=16, Htcm=170, Gender="M", Smoke=0)
#best guess
best_guess = summary(modelA)$coeff[1,1]+summary(modelA)$coeff[2,1]*new$Age+summary(modelA)$coeff[3.1]*ne
best_guess
#prediction interval
pred.log=predict(modelA, new, interval = 'prediction') #using the default 95% prediction level
pred.log
pred=exp(pred.log)
pred
#properties of the data
mean(lungcap$FEV)
median(lungcap$FEV) #used to check how well the mean represents the data
min(lungcap$FEV)
max(lungcap$FEV)
```

```
## [1] 1.323802
##        fit     lwr      upr
## 1 1.323802 1.03616 1.611444
##       fit      lwr      upr
## 1 3.75768 2.818373 5.010038
## [1] 2.63678
## [1] 2.5475
## [1] 0.791
## [1] 5.793
```

Problem 2: Classification

```
library(class)# for function knn
library(caret)# for confusion matrices
```

```
## Loading required package: lattice
```

```
raw = read.csv("https://www.math.ntnu.no/emner/TMA4268/2019v/data/tennis.csv")
M = na.omit(data.frame(y=as.factor(raw$Result),
                       x1=raw$ACE.1-raw$UFE.1-raw$DBF.1,
                       x2=raw$ACE.2-raw$UFE.2-raw$DBF.2))
set.seed(4268) # for reproducibility
tr = sample.int(nrow(M),nrow(M)/2) # nrow(M)/2 distinct random values ranging from 1:nrow(M)
trte=rep(1,nrow(M)) # (1 x nrow(M)) vector of 1's
trte[tr]=0 # trte in random positions (set by tr) set to 0
Mdf=data.frame(M,"istest"=as.factor(trte))
```

Q9:

$$\hat{y}(x) = \hat{P}(Y = j|X = x) = \frac{1}{K}\sum_{i\in\mathcal{N}_\prime} I(y_i = j), \qquad j \in \{0, 1\}$$

Q10:

```
### Training set
ks = 1:30
yhat.train = sapply(ks, function(k) {
  class::knn(train = M[tr,-1], cl = M[tr,1],test = M[tr,-1], k = k)
})
train.equal = I(yhat.train == M[tr,1]) #Returns boolean matrix of correct/incorrect classifications
train.e = apply(train.equal == FALSE, 2, sum)  #Apply summation over all coloumns
train.e = train.e/nrow(train.equal)# Misclassification rate

### Test set
yhat.test = sapply(ks, function(k) {
  class::knn(train = M[tr,-1], cl = M[tr,1],test = M[-tr,-1], k = k)
})
test.equal = I(yhat.test == M[-tr,1])
test.e = apply(test.equal == FALSE, 2, sum)
test.e = test.e/nrow(test.equal)
```

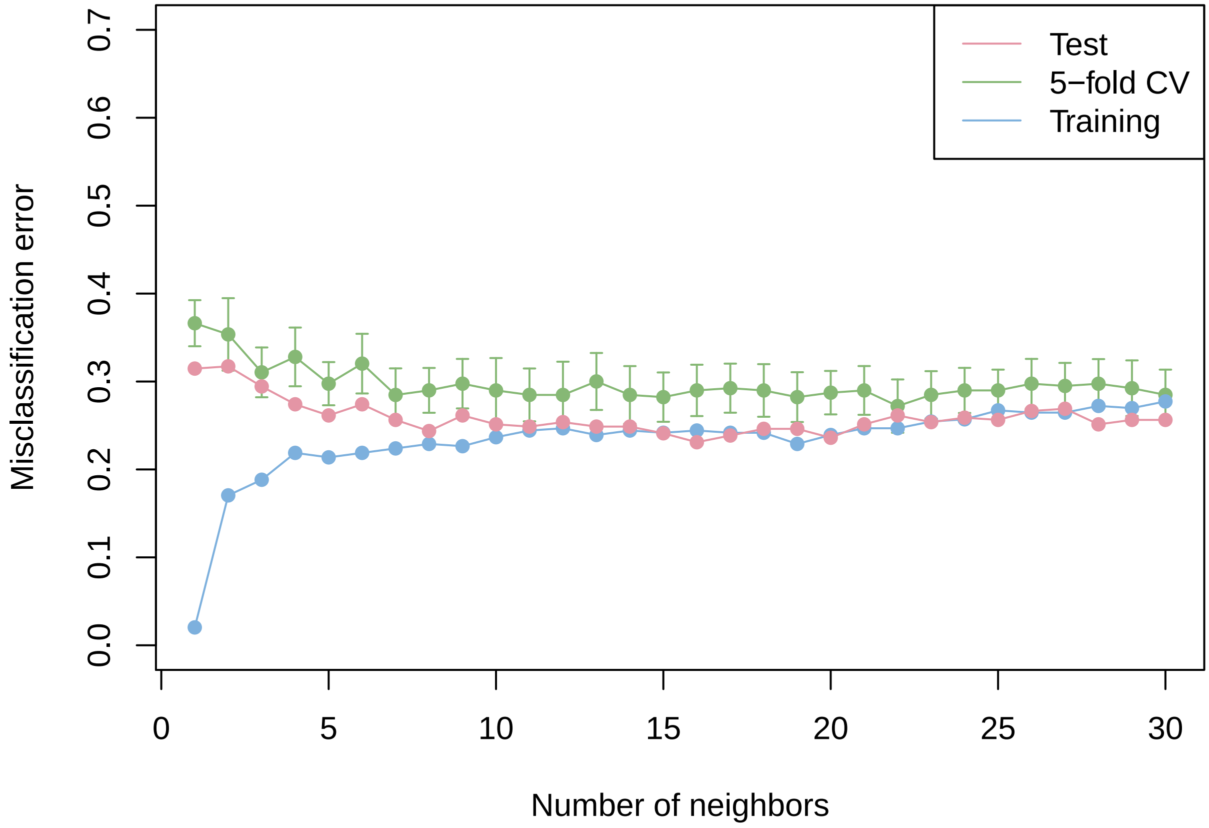
```
set.seed(0)
ks = 1:30 # Choose K from 1 to 30.
idx = createFolds(M[tr,1], k=5) # Divide the training data into 5 folds.
cv = sapply(ks, function(k){
  sapply(seq_along(idx), function(j) {
    yhat = class::knn(train=M[tr[ -idx[[j]] ], -1],
                cl=M[tr[ -idx[[j]] ], 1],
                test=M[tr[ idx[[j]] ], -1], k = k)
    mean(M[tr[ idx[[j]] ], 1] != yhat)
  })
})
```

Q11:

```
cv.e = numeric(30)
cv.se = numeric(30)
for (i in 1:30) {
  cv.e[i] <- mean(cv[,i])
  cv.se[i] <- sd(cv[,i])/sqrt(5)
}
k.min = col(cv)[cv==min(cv)]
```

Q12:

```
library(colorspace)
co = rainbow_hcl(3)
par(mar=c(4,4,1,1)+.1, mgp = c(3, 1, 0))
plot(ks, cv.e, type="o", pch = 16, ylim = c(0, 0.7), col = co[2],
     xlab = "Number of neighbors", ylab="Misclassification error")
arrows(ks, cv.e-cv.se, ks, cv.e+cv.se, angle=90, length=.03, code=3, col=co[2])
lines(ks, train.e, type="o", pch = 16, ylim = c(0.5, 0.7), col = co[3])
lines(ks, test.e, type="o", pch = 16, ylim = c(0.5, 0.7), col = co[1])
legend("topright", legend = c("Test", "5-fold CV", "Training"), lty = 1, col=co)
```

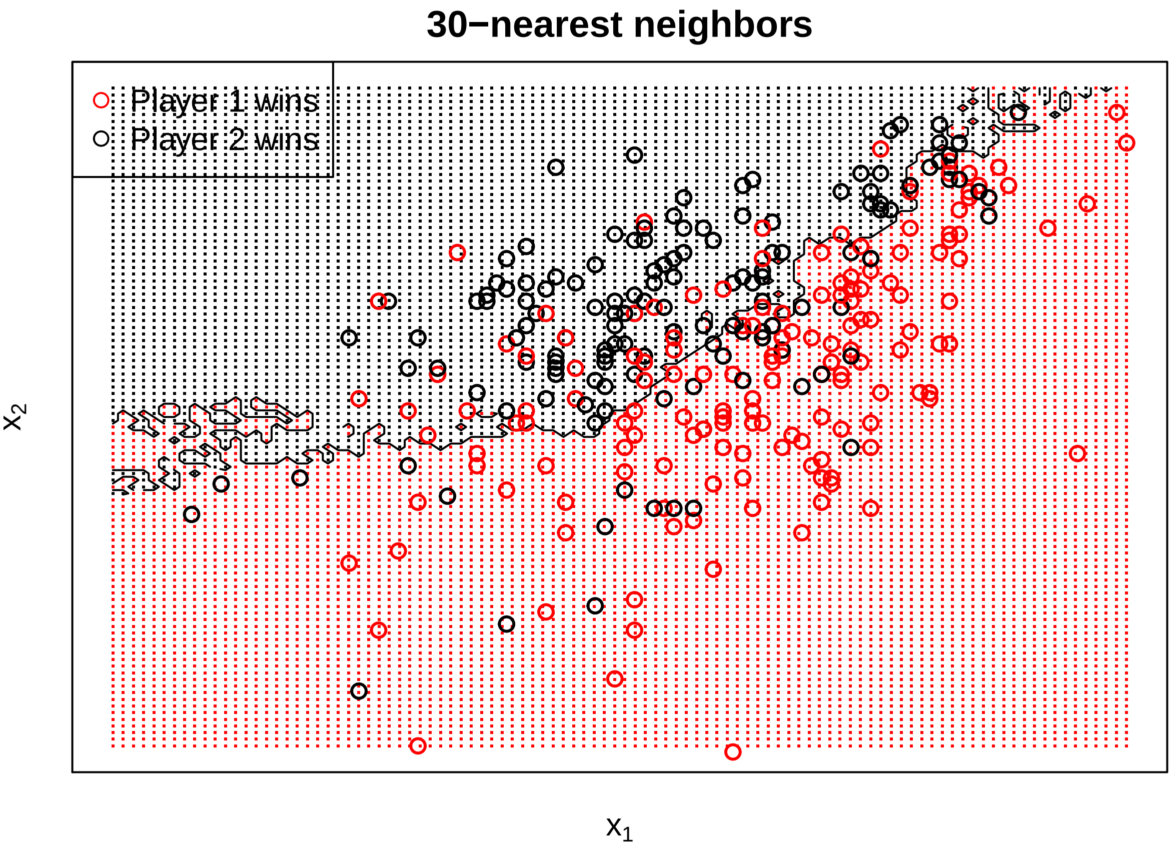


From the plot above we can see how K influences the misclassification rate, which is tied to the bias and variance of our classifier. As K increases, the bias increases and variance decreases. A low value for K will result in a flexible classifier, as the algorithm only looks to one neighbour. A flexible classifier will have a low bias, but in return have a high variance. This can be seen from comparing the graphs for `Training` and

`Test` above. We can see that a flexible classifier will fit the training set very well, but once it is introduced to new data it will misclassify a large proportion, a result of the high variance. The classifier looses its flexibility when K increases, as it takes a larger majority vote. Choosing the right K is thus a problem of weighing bias against variance for the best trade-off..

Q13:

```
k = tail(which(cv.e < cv.e[k.min] + cv.se[k.min]), 1)
size = 100
xnew = apply(M[tr,-1], 2, function(X) seq(min(X), max(X), length.out=size))
grid = expand.grid(xnew[,1], xnew[,2])
grid.yhat = class::knn(M[tr,-1], M[tr,1], k=k, test=grid)
np = 300
par(mar=rep(2,4), mgp = c(1, 1, 0))
contour(xnew[,1], xnew[,2], z = matrix(grid.yhat, size), levels=.5,
        xlab=expression("x"[1]), ylab=expression("x"[2]), axes=FALSE,
        main = paste0(k,"-nearest neighbors"), cex=1.2, labels="")
points(grid, pch=".", cex=1, col=grid.yhat)
points(M[1:np,-1], col=factor(M[1:np,1]), pch = 1, lwd = 1.5)
legend("topleft", c("Player 1 wins", "Player 2 wins"),
       col=c("red", "black"), pch=1)
box()
```



The proposed strategy is the *one standard error rule*, which chooses the simplest model given that the model error is within a standard error of the parameter giving the minimal error. In this case, we choose the largest K that satisfies the condition $\text{CV}(K) \leqslant \text{CV}(K_{min}) + \text{SE}(K_{min})$, where K_{min} gives the minimal error.

Q14:

```
K=30# your choice from Q13
KNNclass=class::knn(train=M[tr,-1], cl=M[tr,1], test=M[-tr,-1], k = K,prob=TRUE)
KNNprobwinning=attributes(KNNclass)$prob
KNNprob= ifelse(KNNclass == "0", 1-KNNprobwinning, KNNprobwinning)
# now KNNprob has probability that player 1 wins, for all matches in the test set
library(pROC)
```

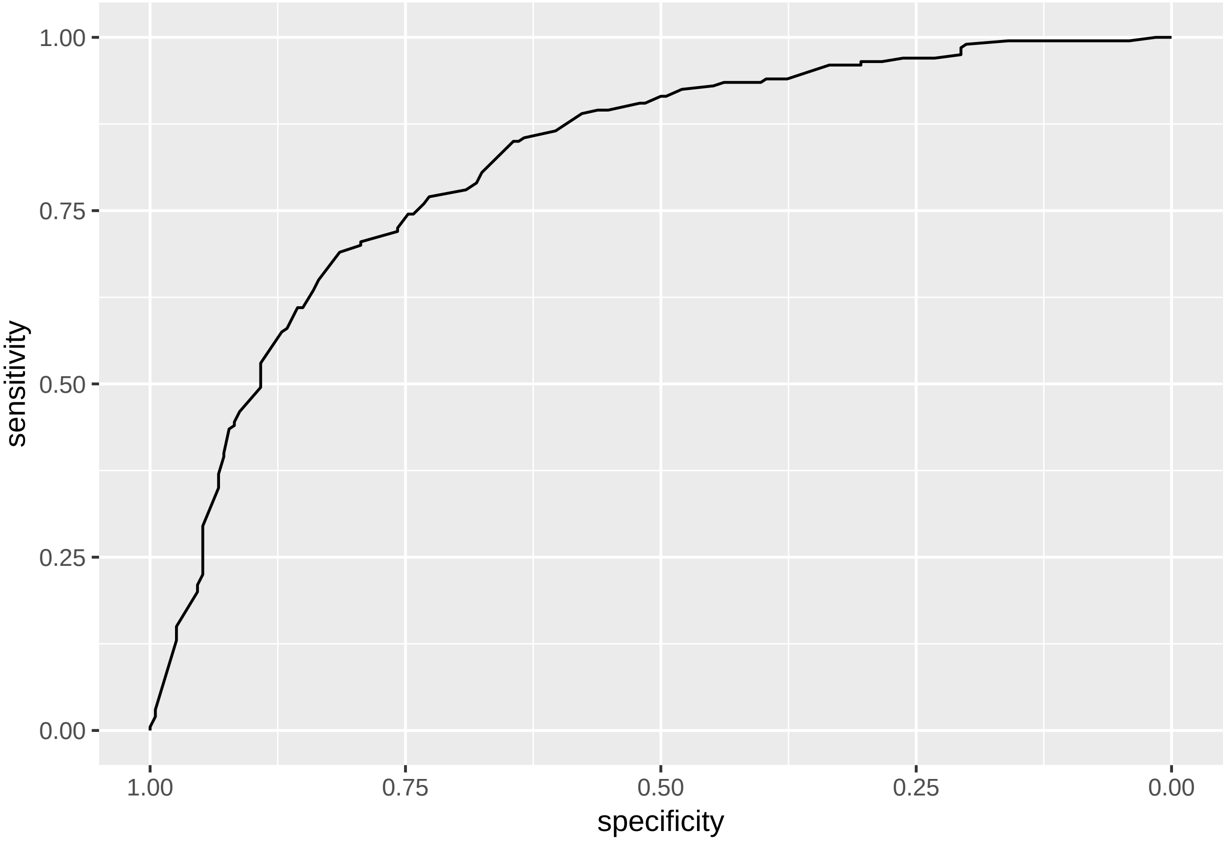
```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following object is masked from 'package:colorspace':
##
##     coords
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
tennis.roc = roc(response = M[-tr,1], predictor = KNNprob)
auc = tennis.roc$auc
ggroc(tennis.roc)
```



```
auc
```

```
## Area under the curve: 0.8178
```

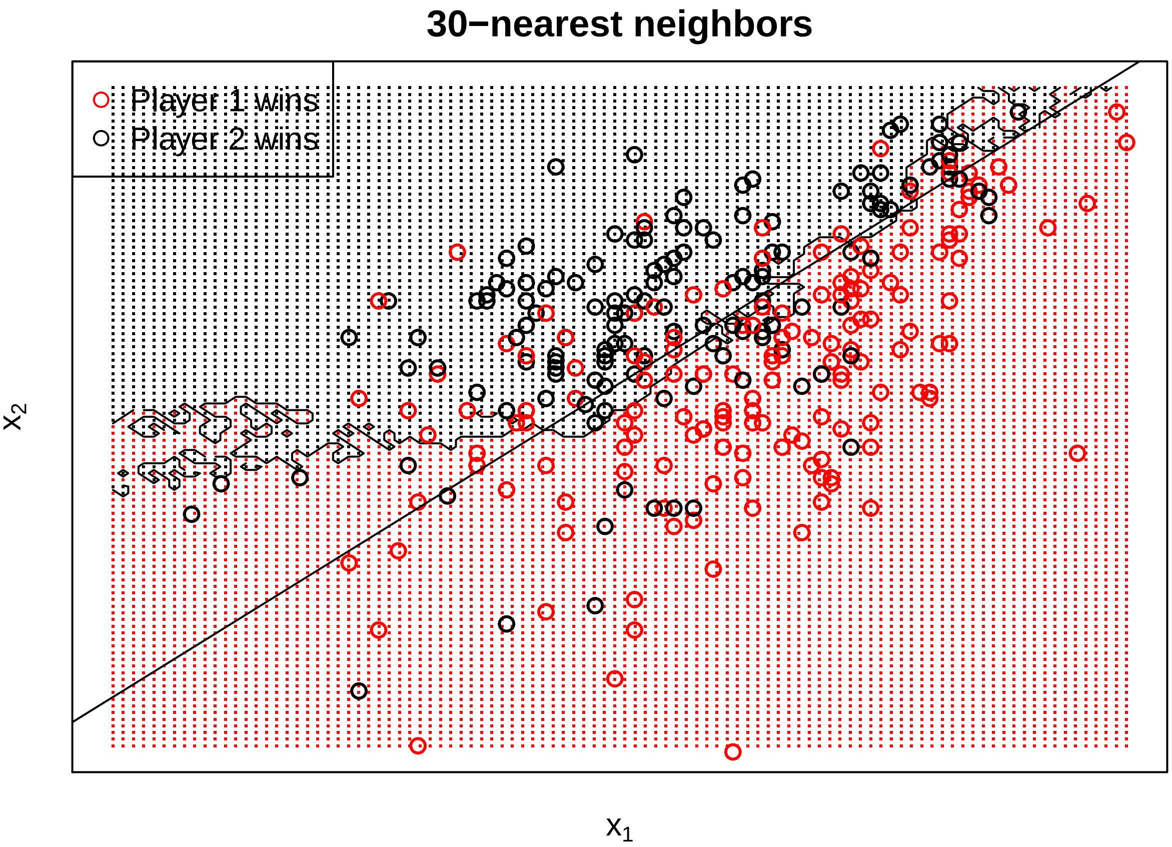
The ROC is produced by calculating the sensitivity and specificity for all cut-offs on the probability of classifying to a certain class, where $p(x) = \hat{P}(Y = j|X = x)$ is the probability of classifying to j. If

$p(x) \geqslant$ cut-off, we classify to j and plot the sensitivity and specificity. Repeating this procedure for all possible cut-offs in the range $(0, 1)$, we can produce the ROC. The AUC is the area under the ROC, which is used as a measure of how well a classifier classifies. Since sensitivity and specificity denotes correctly classifications, we want the AUC to be as close to 1 as possible. For our ROC the AUC is 0.8178, which is fairly good. If we were to use a random guessing classifier, the probability of classifying to a class is drawn from a uniform distribution. This means the fraction of true positives we guess out of all positives is the cut-off we set. The rest of our guesses will be negative, meaning the fraction of true negatives out of all negatives will be $1 -$ cut-off. For all cut-offs, the sensitivity will thus be $1 -$ specificity, resulting in a diagonal ROC. The area under the ROC will then be 0.5.

Q15:

```
#plot from Q13
k = tail(which(cv.e < cv.e[k.min] + cv.se[k.min]), 1)
size = 100
xnew = apply(M[tr,-1], 2, function(X) seq(min(X), max(X), length.out=size))
grid = expand.grid(xnew[,1], xnew[,2])
grid.yhat = class::knn(M[tr,-1], M[tr,1], k=k, test=grid)
np = 300
par(mar=rep(2,4), mgp = c(1, 1, 0))
contour(xnew[,1], xnew[,2], z = matrix(grid.yhat, size), levels=.5,
         xlab=expression("x"[1]), ylab=expression("x"[2]), axes=FALSE,
         main = paste0(k,"-nearest neighbors"), cex=1.2, labels="")
points(grid, pch=".", cex=1, col=grid.yhat)
points(M[1:np,-1], col=factor(M[1:np,1]), pch = 1, lwd = 1.5)
legend("topleft", c("Player 1 wins", "Player 2 wins"),
      col=c("red", "black"), pch=1)

#produces decision boundry for y.tilde as the diagonal x2 = x1
abline(0,1)
box()
```



```
library(caret)
ref = M[-tr,1]
y.hat = yhat.test[,30] #classifications for yhat with K=30
y.tilde <- numeric()
M.test=M[-tr,-1]
for (i in 1:dim(M.test)[1]) {
  if (M.test[i,1] > M.test[i,2]) {
    y.tilde[i] <- 1
  } else {
  y.tilde[i] <- 0
  }
}

confmat.argmax=table(y.tilde, ref)
confmat.yhat = table(y.hat,ref)
misclass.argmax=(sum(confmat.argmax) - sum(diag(confmat.argmax)))/sum(confmat.argmax)
misclass.yhat=(sum(confmat.yhat) - sum(diag(confmat.yhat)))/sum(confmat.yhat)
confmat.argmax
confmat.yhat
misclass.argmax
misclass.yhat
```

```
##         ref
## y.tilde   0   1
##       0 149  47
##       1  45 153
##       ref
## y.hat   0   1
##     0 137  44
##     1  57 156
## [1] 0.2335025
## [1] 0.2563452
```

We get the misclassification errors $\text{Ave}(I(y \neq \hat{y})) = 0.2563$ and $\text{Ave}(I(y \neq \widetilde{y})) = 0.2335$. As we prefer the classifier with the lowest misclassification error, we choose $\widetilde{y}(x)$.

Problem 3: Bias-variance trade-off

Least squares estimator:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

Q16:

$$\begin{aligned}
\text{E}(\hat{\boldsymbol{\beta}}) = \text{E}[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}] &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\text{E}(\mathbf{Y}) \\
&= (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{X})\boldsymbol{\beta} \\
&= I\boldsymbol{\beta} \\
&= \boldsymbol{\beta}
\end{aligned}$$

$$\begin{aligned}
\text{Cov}(\hat{\boldsymbol{\beta}}) = \text{Cov}[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}] &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\text{Cov}(\mathbf{Y})[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T]^T \\
&= \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}[(\mathbf{X}^{\mathbf{T}}\mathbf{X})^{\mathbf{-1}}]^{\mathbf{T}} \\
&= \sigma^2(\mathbf{X}^{\mathbf{T}}\mathbf{X})^{\mathbf{-1}}
\end{aligned}$$

Q17:

$$\begin{aligned}
\text{E}(\hat{f}(\mathbf{x}_0)) = \text{E}(\mathbf{x}_0^T\hat{\boldsymbol{\beta}}) &= \mathbf{x}_0^T\text{E}(\hat{\boldsymbol{\beta}}) \\
&= \mathbf{x}_0^T\boldsymbol{\beta} \\
&= f(\mathbf{x}_0)
\end{aligned}$$

$$\begin{aligned}
\text{Var}(\hat{f}(\mathbf{x}_0)) = \text{Var}(\mathbf{x}_0^T\hat{\boldsymbol{\beta}}) &= \mathbf{x}_0^T\text{Cov}(\hat{\boldsymbol{\beta}})\mathbf{x}_0 \\
&= \sigma^2\mathbf{x}_0^T(\mathbf{X}^{\mathbf{T}}\mathbf{X})^{\mathbf{-1}}\mathbf{x_0}
\end{aligned}$$

Q18:

$$\begin{aligned}
\text{E}[(Y_0 - \hat{f}(\mathbf{x}_0))^2] &= \text{E}[(Y_0^2 - 2Y_0\hat{f}(\mathbf{x}_0) + \hat{f}(\mathbf{x}_0)^2] \\
&= \text{E}(Y_0^2) - 2\text{E}[Y_0\hat{f}(\mathbf{x}_0)] + \text{E}[\hat{f}(\mathbf{x}_0)^2] \\
&= \text{Var}(Y_0) + \text{E}(Y_0)^2 - 2\text{E}(Y_0)\text{E}[\hat{f}(\mathbf{x}_0)] + \text{Var}[\hat{f}(\mathbf{x}_0)] + \text{E}[\hat{f}(\mathbf{x}_0)]^2 \\
&= \text{Var}(\varepsilon) + f(\mathbf{x}_0)^2 - 2f(\mathbf{x}_0)f(\mathbf{x}_0) + \text{Var}[\hat{f}(\mathbf{x}_0)] + f(\mathbf{x}_0)^2 \\
&= \text{Var}(\varepsilon) + \text{Var}[\hat{f}(\mathbf{x}_0)] \\
&= \sigma^2 + \sigma^2\mathbf{x}_0^T(\mathbf{X}^{\mathbf{T}}\mathbf{X})^{\mathbf{-1}}\mathbf{x_0}
\end{aligned}$$

Ridge estimator:

$$\widetilde{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}$$

Q19:

$$\begin{aligned}
\mathrm{E}(\widetilde{\boldsymbol{\beta}}) &= \mathrm{E}[(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}] \\
&= (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathrm{E}(\mathbf{Y}) \\
&= (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}(\mathbf{X}^T\mathbf{X})\boldsymbol{\beta}
\end{aligned}$$

$$\begin{aligned}
\mathrm{Cov}(\widetilde{\boldsymbol{\beta}}) &= \mathrm{Cov}[(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}] \\
&= (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathrm{Cov}(\mathbf{Y})[(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T]^T \\
&= \sigma^2(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{X}[(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}]^T \\
&= \sigma^2(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{X}[(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^T]^{-1} \\
&= \sigma^2(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}
\end{aligned}$$

Q20:

$$\begin{aligned}
\mathrm{E}(\widetilde{f}(\mathbf{x}_0)) &= \mathrm{E}(\mathbf{x}_0^T\widetilde{\beta}) \\
&= \mathbf{x}_0^T\mathrm{E}(\widetilde{\beta}) \\
&= \mathbf{x}_0^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}(\mathbf{X}^T\mathbf{X})\boldsymbol{\beta}
\end{aligned}$$

$$\begin{aligned}
\mathrm{Var}(\widetilde{f}(\mathbf{x}_0)) &= \mathrm{Var}(\mathbf{x}_0^T\widetilde{\beta}) \\
&= \mathbf{x}_0^T\mathrm{Cov}(\widetilde{\boldsymbol{\beta}})\mathbf{x}_0 \\
&= \sigma^2\mathbf{x}_0^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{x}_0
\end{aligned}$$

Q21:

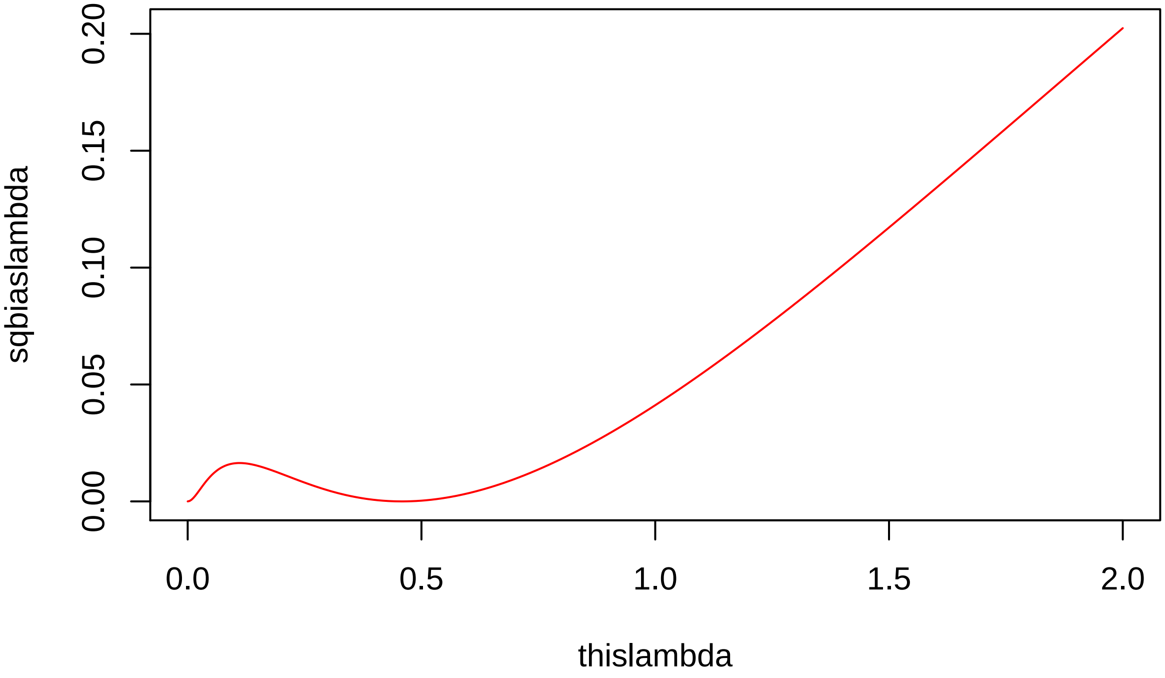
$$\begin{aligned}
\mathrm{E}[(Y_0 - \widetilde{f}(\mathbf{x}_0))^2] &= \mathrm{Var}(\varepsilon) + \mathrm{Var}(\widetilde{f}(\mathbf{x}_0)) + [\mathrm{E}(\widetilde{f}(\mathbf{x}_0) - f(\mathbf{x}_0)]^2 \\
&= \sigma^2 + \sigma^2\mathbf{x}_0^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{x}_0 + [\mathbf{x}_0^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}(\mathbf{X}^T\mathbf{X})\boldsymbol{\beta} - \mathbf{x}_0^T\beta]^2
\end{aligned}$$

```
values=dget("https://www.math.ntnu.no/emner/TMA4268/2019v/data/BVtradeoffvalues.dd")
X=values$X
dim(X)
x0=values$x0
dim(x0)
beta=values$beta
dim(beta)
sigma=values$sigma
sigma
```

```
## [1] 100  81
## [1] 81  1
## [1] 81  1
## [1] 0.5
```

Q22:

```
sqbias=function(lambda,X,x0,beta)
{
  p=dim(X)[2]
  value= (t(x0)%*%solve(t(X)%*%X+lambda*diag(p))%*%(t(X)%*%X)%*%beta-t(x0)%*%beta)^2
  return(value)
}
thislambda=seq(0,2,length=500)
sqbiaslambda=rep(NA,length(thislambda))
for (i in 1:length(thislambda)) sqbiaslambda[i]=sqbias(thislambda[i],X,x0,beta)
plot(thislambda,sqbiaslambda,col=2,type="l")
```

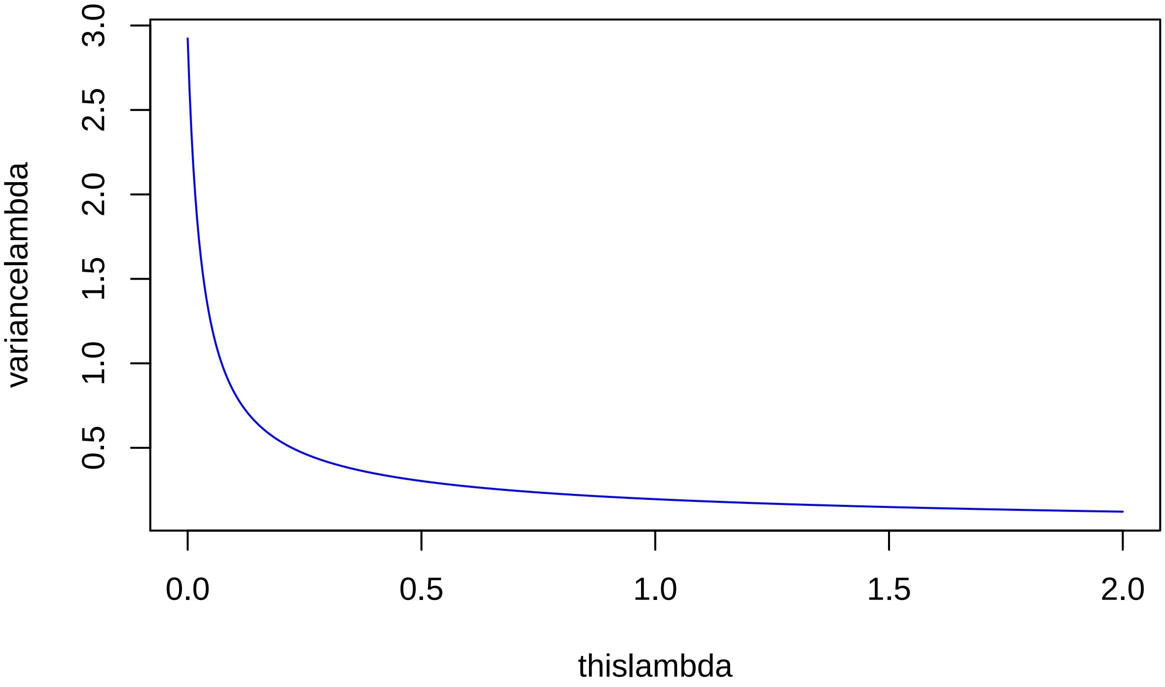


As we expect from a ridge regression, the bias increases with λ as the flexibility of the fit decreases. We can see from the curve that bias as a function of λ is not strictly increasing for all λ. This specific result comes from the data used in the ridge regression, and is not something we can predict before plotting the curve. The general trend in these curves is as expected though, namely that the bias increases with λ.

Q23:

```
variance=function(lambda,X,x0,sigma)
{
  p=dim(X)[2]
  inv=solve(t(X)%*%X+lambda*diag(p))
  var=sigma^2*inv%*%t(X)%*%X%*%inv
  value=t(x0)%*%var%*%x0
  return(value)
}
thislambda=seq(0,2,length=500)
variancelambda=rep(NA,length(thislambda))
```

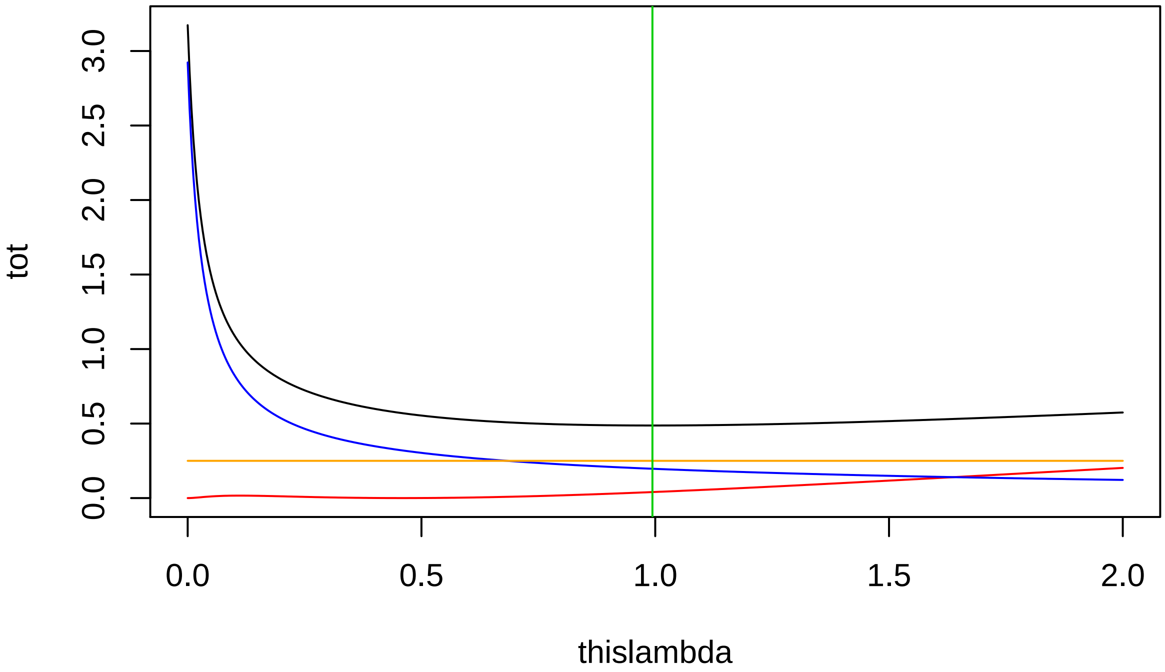
```
for (i in 1:length(thislambda)) variancelambda[i]=variance(thislambda[i],X,x0,sigma)
plot(thislambda,variancelambda,col=4,type="l")
```



Since the flexibility of the fit decreases with λ, the variance decreases, and we can see that the curve confirms this for our data.

Q24:

```
tot=sqbiaslambda+variancelambda+sigma^2
which.min(tot)
thislambda[which.min(tot)]
plot(thislambda,tot,col=1,type="l",ylim=c(0,max(tot)))
lines(thislambda, sqbiaslambda,col=2)
lines(thislambda, variancelambda,col=4)
lines(thislambda,rep(sigma^2,500),col="orange")
abline(v=thislambda[which.min(tot)],col=3)
```



```
## [1] 249
## [1] 0.993988
```

As we can see from the graph the optimal λ for this problem is $\lambda \approx 1$. This is neither the optimal choice for the bias or variance alone, meaning we make a trade-off between the two in order to get the minimum total mean squared error MSE.