Compulsory exercise 1

TMA4268 Statistical Learning V2019

Christian Lehre, Axel R??nold & Erik B??e

20 February, 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:

Model A:

$$\log(\text{FEV}) = \beta_0 + \beta_1 AGE + \beta_2 HTCM + \beta_3 GENDERM + \beta_4 SMOKE + \varepsilon$$

Fitted Model:

$$\log(\hat{\text{FEV}}) = -1.944 + 0.023AGE + 0.017HTCM + 0.029GENDERM - 0.046SMOKE$$

Q2:

- `Estimate` - in particular interpretation of `Intercept`
  Estimated regression coefficients given by

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

.
When increasing covariate x_j with one unit, and keeping all other covariates constant, the response variable changes with a factor of $\hat{\beta}_j$. Positive estimates reduce the value of the response, while negative estimates increase the value. The intercept is the value of the response when all covariates are zero.

- `Std.Error`
  Estimated standard deviation of the estimated regression coefficients, i.e the average amount that the estimated regression coefficients vary from the actual value. The Std.Error is given by

$$\hat{\text{SD}}(\hat{\beta}_j) = \sqrt(\hat{\sigma}^2(\mathbf{X}^T\mathbf{X})^{-1}_{jj})$$

  , where $\mathbf{X}$ is the design matrix of the regression, and $\hat{\sigma}$ is the estimated standard deviation of the response.

- `Residual standard error`
  Estimate of the standard deviation of the error term ϵ in the regression model. The residual standard error is given by

$$\hat{\sigma} = \frac{RSS}{n-p-1} = \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n-p-1}$$

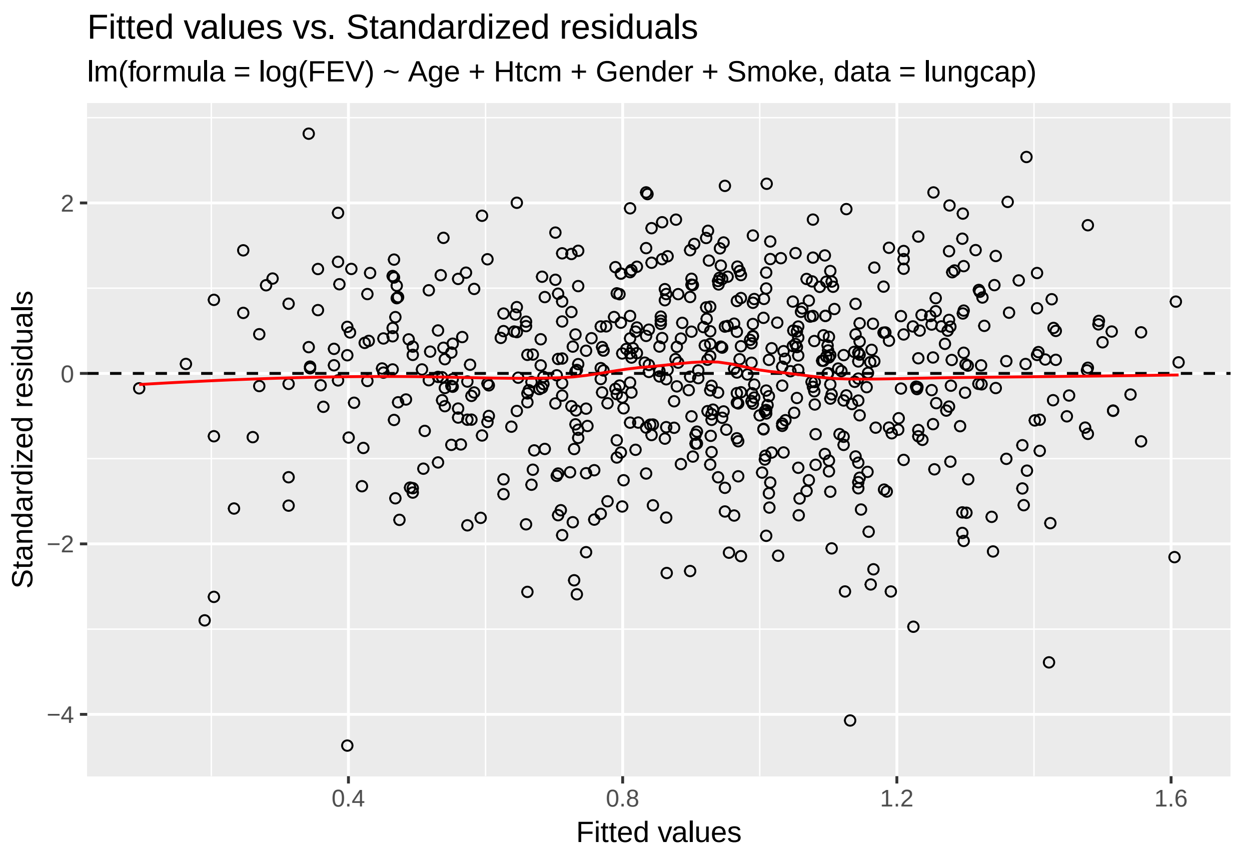
  , where n is the number of observations and p is the number of covariates (or predictors) in the fitted model.

- `F-statistic`
  The F-statistic is used to test the hypothesis that all estimated regression coefficients are zero, i.e to check whether there is a relationship between response and predictors.

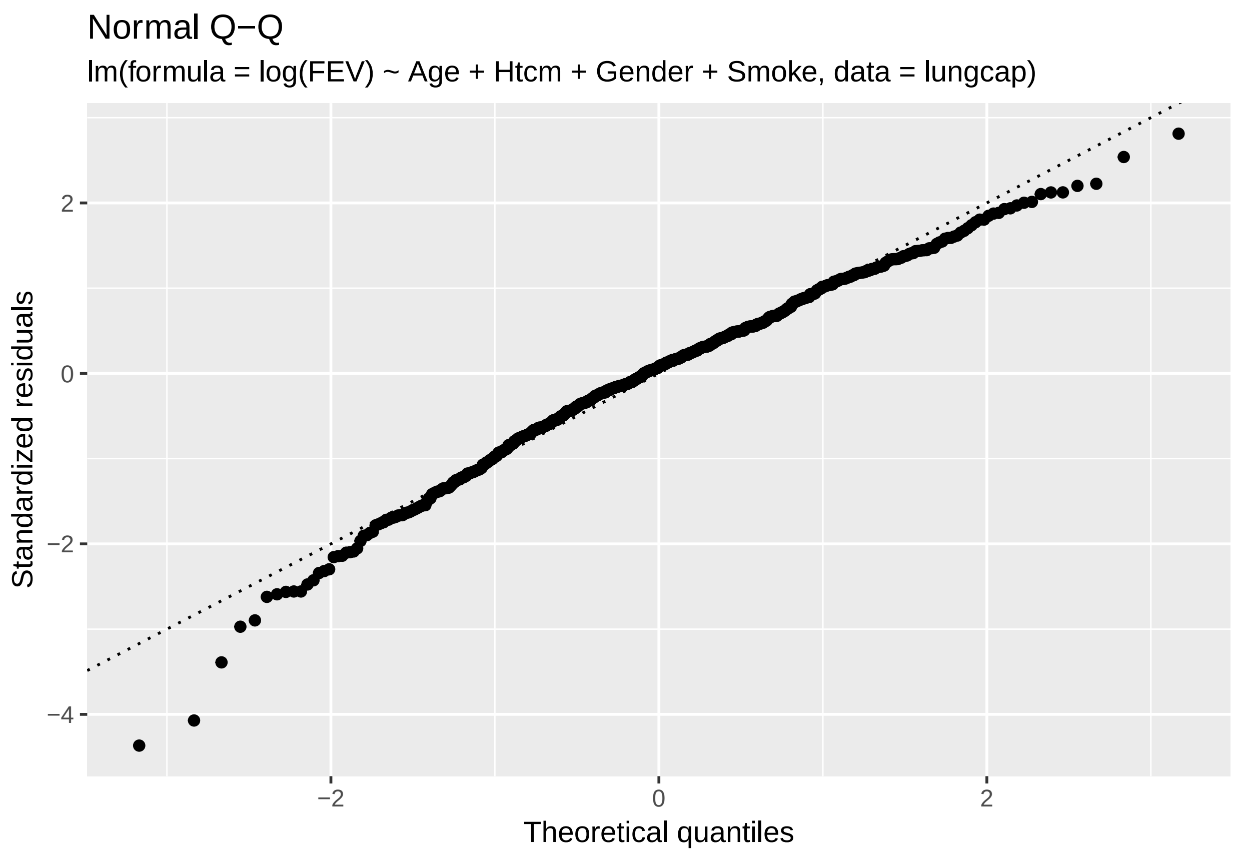
Q3:
The proportion of variability of the model fit is given by the R^2-statistic. The multiple R^2 will always increase with an increasing number of covariates, and might result in too optimistic model assessment.The adjusted R^2 takes this into account, and adjust according to the number of covaraites. Thus, the adjusted R^2-statistic is usually the preferred measure of explained variability in the fitted model when doing multiple linear regression. For model A, the multiple R^2 is 0.8106, while the adjusted R^2 is 0.8095. Thus, 81% of the variability is explained in modelA.

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))
```



```
# 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))
```



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

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

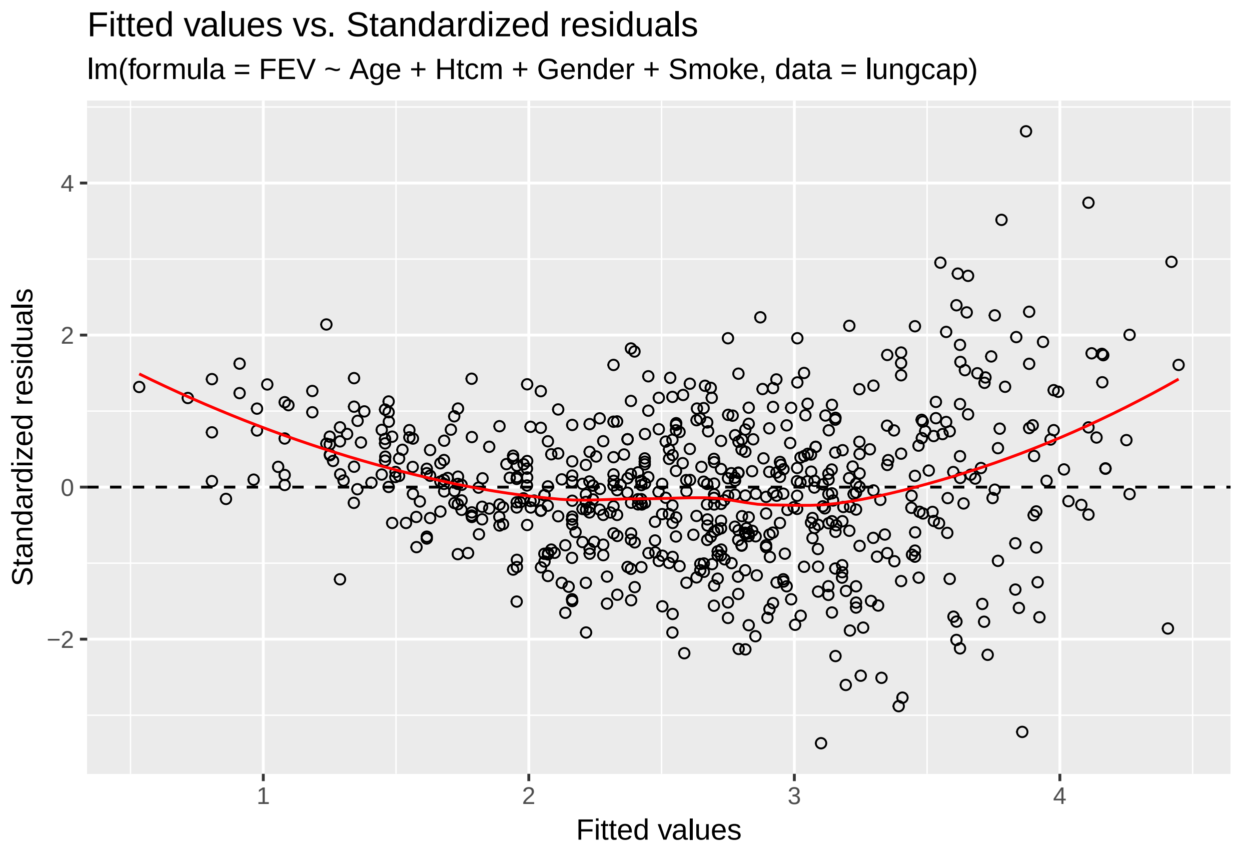
In this problem we consider two diagnostic plots, namely residuals vs. fitted values and the Q-Q plot. The former plot showing residuals vs. fitted values looks good. The residuals are spread seemingly random, and the model seems to have homoscedatic error variances. There is no evidence of correlated errors, so the model seems to include all explanatory factors (the residuals contains no predictive features).

As for the Q-Q plot, the data quantiles seems to bend away from the normal quantiles at the tails. That is, the residuals are heavy-tailed rather than being normally distributed.

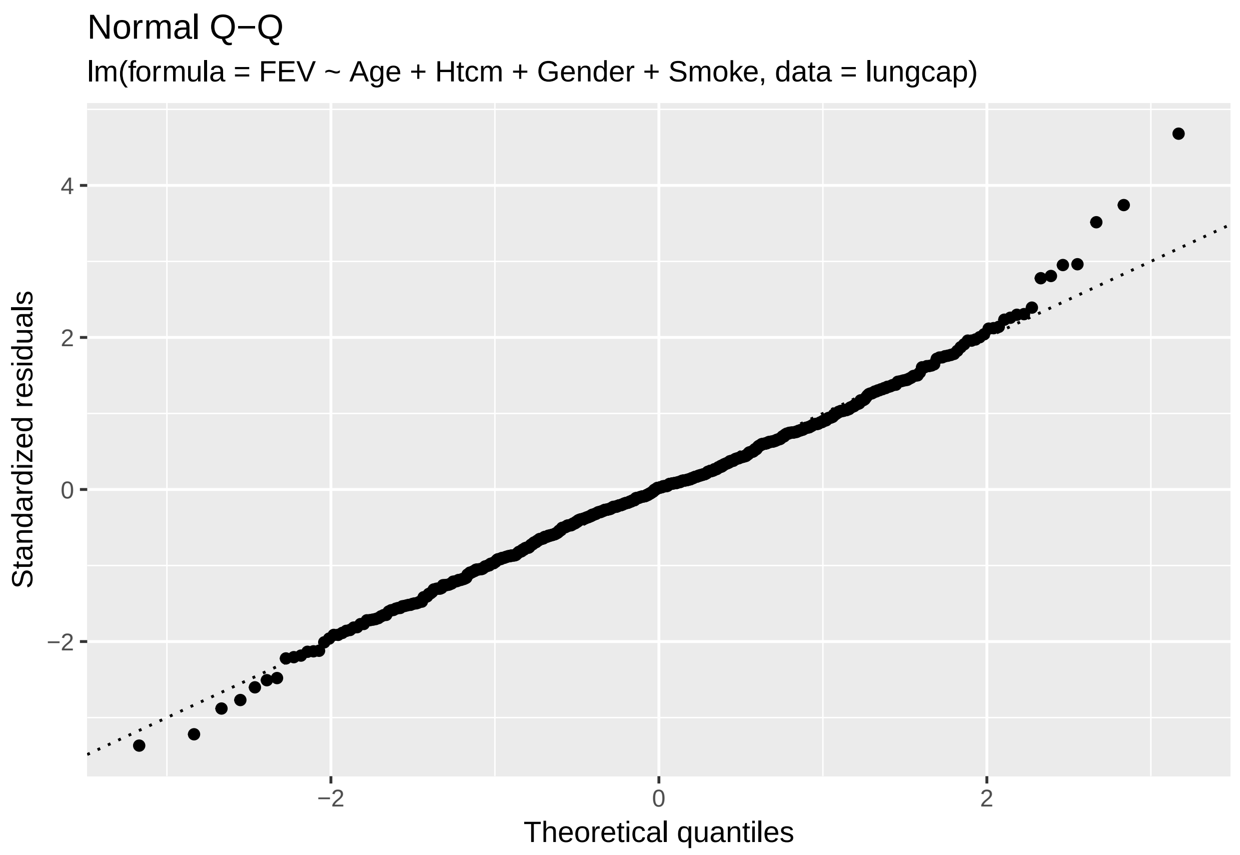
The normality test yields a low p-value, and the normality hypothesis is rejected.

Q5:

```r
modelB = lm(FEV ~ Age + Htcm + Gender + Smoke, data=lungcap)
summary(modelB)

#make diagnostic plots for modelB
# residuls vs fitted
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",
       subtitle = deparse(modelB$call))
```



```
# 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", subtitle = deparse(modelB$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
```

Looking at the residuals vs. fitted values plot, one can see that the residuals are heteroscedatic. There is also evidence of autocorrelated errors, as the residuals shows a non-linear explanatory trend.

The Q-Q plot is similar for modelB as for modelA, i.e the residuals are not normally distributed as the model assumes.

As for model fit B, the Smoke covariate is no longer significant, as its corresponding p-value is relatively large. The coefficient of determination, R^2, which is a measure of the explained variability of the model, is also lower for model B than model A. In other words, model A explains a greater proportion of variability than model B.

The standard error, or estimated standard deviation of the estimated regression coefficients, are lower in model A for all covarariates. These quantities are used in e.g constructing confidence intervals for the regression coefficients. As these are lower in model A, one will obtain more accurate confidence intervals for β_j when using model A.

When doing inference about FEV, the interpretation might be easier when using model B, but the accuracy is greater when using model A.

Q6: Test if the covariate AGE has an effect on log(FEV), i.e conduct the following hypothesis test

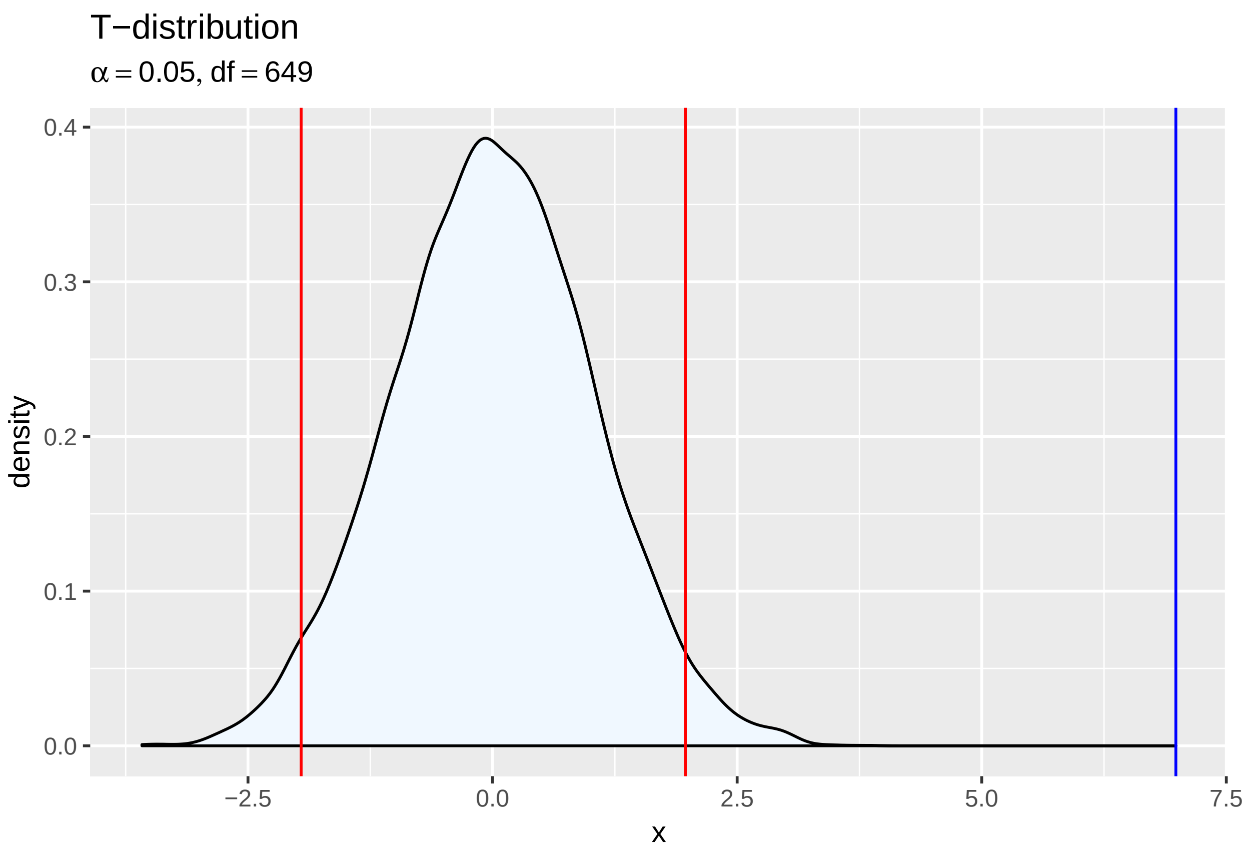
$$H_0 : \beta_{\text{AGE}} = 0 \quad \text{vs.} \quad H_1 : \beta_{\text{AGE}} \neq 0$$

using a two-sided t-test based on

$$T_{\text{AGE}} = \frac{\hat{\beta_{\text{AGE}}} - \beta_{\text{AGE}}}{\hat{SD}(\hat{\beta_{\text{AGE}}})} \sim t_{n-p-1} = t_{654-4-1}$$

where $\hat{\text{SD}}(\hat{\beta}_j)$ is the standard error as explained in Q2 above.

```r
#p-value AGE
p_AGE = summary(modelA)$coefficients['Age',4]

#or equivalently
n = dim(lungcap)[1]
p = 4
df = n-p-1
alpha = 0.05
critical_values = c(qt(alpha/2,df),qt(1-alpha/2,df))
t_dist = rt(10000,df)

t_stat = abs(modelA$coefficients['Age']/summary(modelA)$coefficients['Age','Std. Error'])

#equivalently,
t_stat = summary(modelA)$coefficients['Age','t value']

#calculate p-value from t-statistic
p1_age = 2*pt(-abs(t_stat),df)

ggplot(data.frame(x = t_dist)) + geom_density(aes(x = x),fill = 'aliceblue') + geom_vline(xintercept =
```



For a given significance level α, the null-hypothesis is rejected if the p-value is less than α. The p-value is

```
## [1] "P-value from summary of the fitted model:"
## [1] 7.09641e-12
## [1] "P-value computed from the t-statistic:"
## [1] 7.09641e-12
```

Thus, the null-hypothesis is rejected for $\alpha \geq 7 \cdot 10^{-12}$.
As an example, observe in the above figure a student-t distribution with 649 degrees of freedom corresponding to the problem description. The critical values for a significance level $\alpha = 0.05$ is marked with red, vertical lines, whereas the t-statistic assuming the null-hypothesis is maarked with a blue line. The t-statistic is clearly within the rejection region, and the null-hypothesis is rejected.

Q7: The $(1-\alpha)\%$ confidence interval (CI) for β_{Age} for a given significance level α is given by

$$[\hat{\beta} - t_{\frac{\alpha}{2},n-p-1}\hat{SD}(\hat{\beta}), \hat{\beta} + t_{\frac{\alpha}{2},n-p-1}\hat{SD}(\hat{\beta})]$$

```r
#99% confidence interval for beta_Age
betahat = summary(modelA)$coefficients['Age','Estimate']
sd = summary(modelA)$coefficients['Age','Std. Error']
alpha = 0.01
ta2 = qt(alpha/2, df)
UCI = betahat + ta2*sd
LCI = betahat - ta2*sd
CI = c(UCI,LCI)
#or, equivalently
print("built-in CI:")
confint(modelA,'Age',level = 0.99)
print(" ")
print("Numerically from formula:")
CI
```

```
## [1] "built-in CI:"
##          0.5 %     99.5 %
## Age 0.01473674 0.03203769
## [1] " "
## [1] "Numerically from formula:"
## [1] 0.01473674 0.03203769
```

Since the 99% CI for β_{Age} does not include 0, the p-value is less than $\alpha = 0.01$.

Q8: Prediction of a 16 year old male that is 170cm and non-smoking

```r
new = data.frame(Age=16, Htcm=170, Gender="M", Smoke=0)
pred = predict(modelA,newdata = new, type = 'response')
# Construct 95% prediction interval for FEV, note that modelA's response is log(FEV)
pred

f.exp = function(x){
  return (exp(x))
}
fev.pred = f.exp(pred)

fev.predint = predict(modelA,newdata = new, level = 0.95, interval = 'prediction')
f.exp(fev.predint)
```

```
##        1
## 1.323802
##       fit      lwr      upr
## 1 3.75768 2.818373 5.010038
```

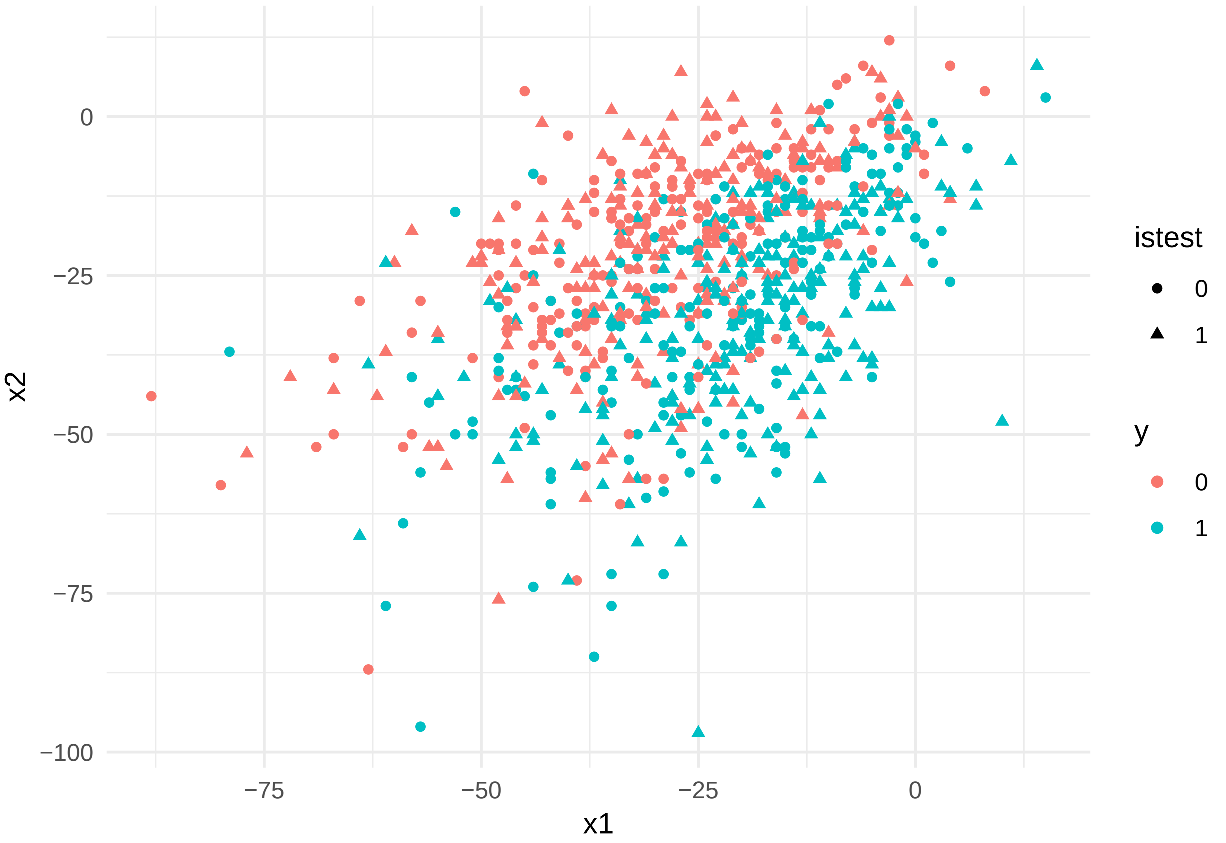
Best prediction for log(FEV) is 1.324. The 95% prediction interval for the forced expiratory volume, FEV, of the new observation is $[2.812, 5.010]$. The prediction interval is too wide to be useful. A person with the given characteristic will on average have a 95% chance of having a FEV between 2.8 and 5.0

Problem 2: Classification

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

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

```r
library(ggplot2)# for ggplot
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)
trte=rep(1,nrow(M))
trte[tr]=0
Mdf=data.frame(M,"istest"=as.factor(trte))
ggplot(data=Mdf,aes(x=x1,y=x2,colour=y,group=istest,shape=istest))+
  geom_point()+theme_minimal()
```



Q9: The mathematical formula for the K-nearest neighbour esitimator $\hat{y} \in \{0, 1\}$ is given by

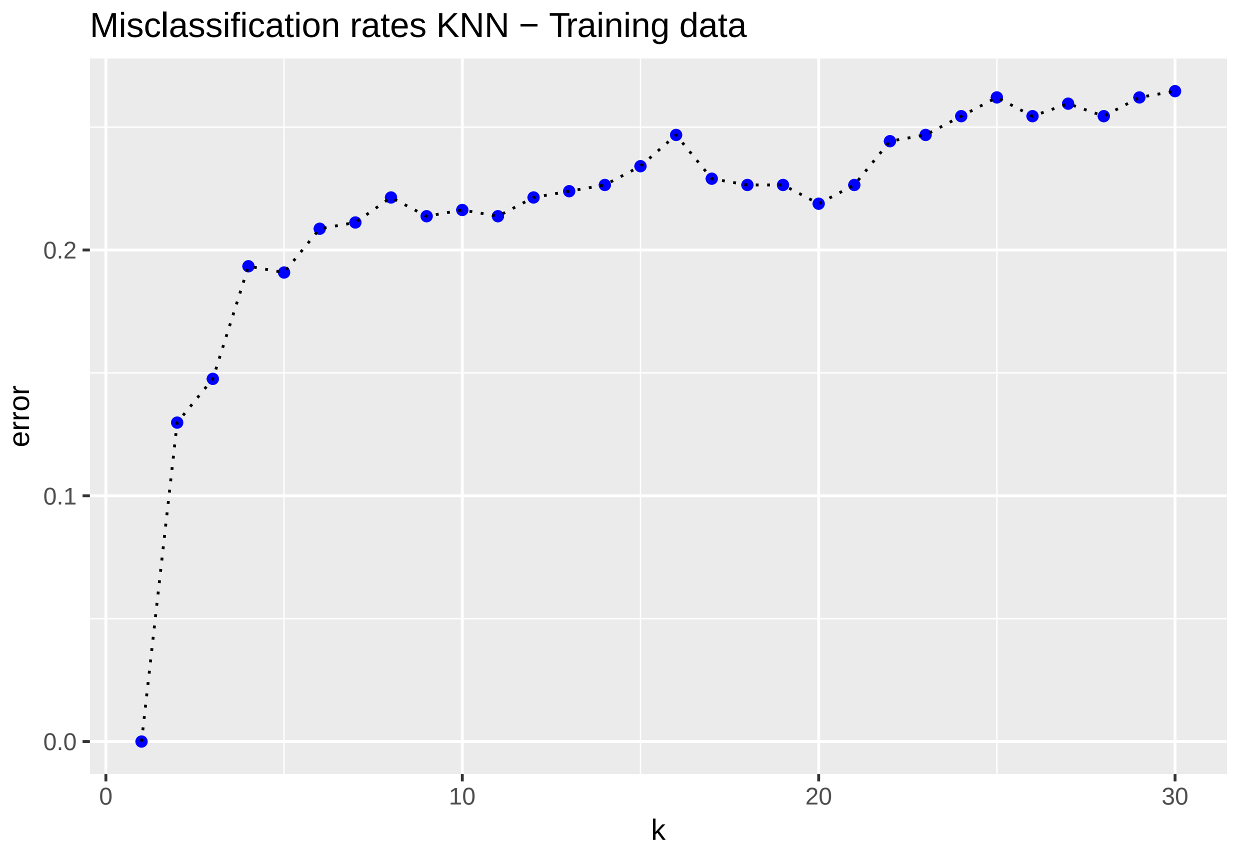
$$\hat{P}(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j)$$

where K is the flexibility parameter determining how many points are in the neighbouring set N_0. The classifier identifies the K nearest points to a test observation x_0 in the training data, and calculates the conditional, or posterior, probability of being in class j as the fraction of points in N_0 where the response is j. $I(y_i = j)$ is a binomial function yielding 1 if $y_i = j$, and 0 elsewhere.

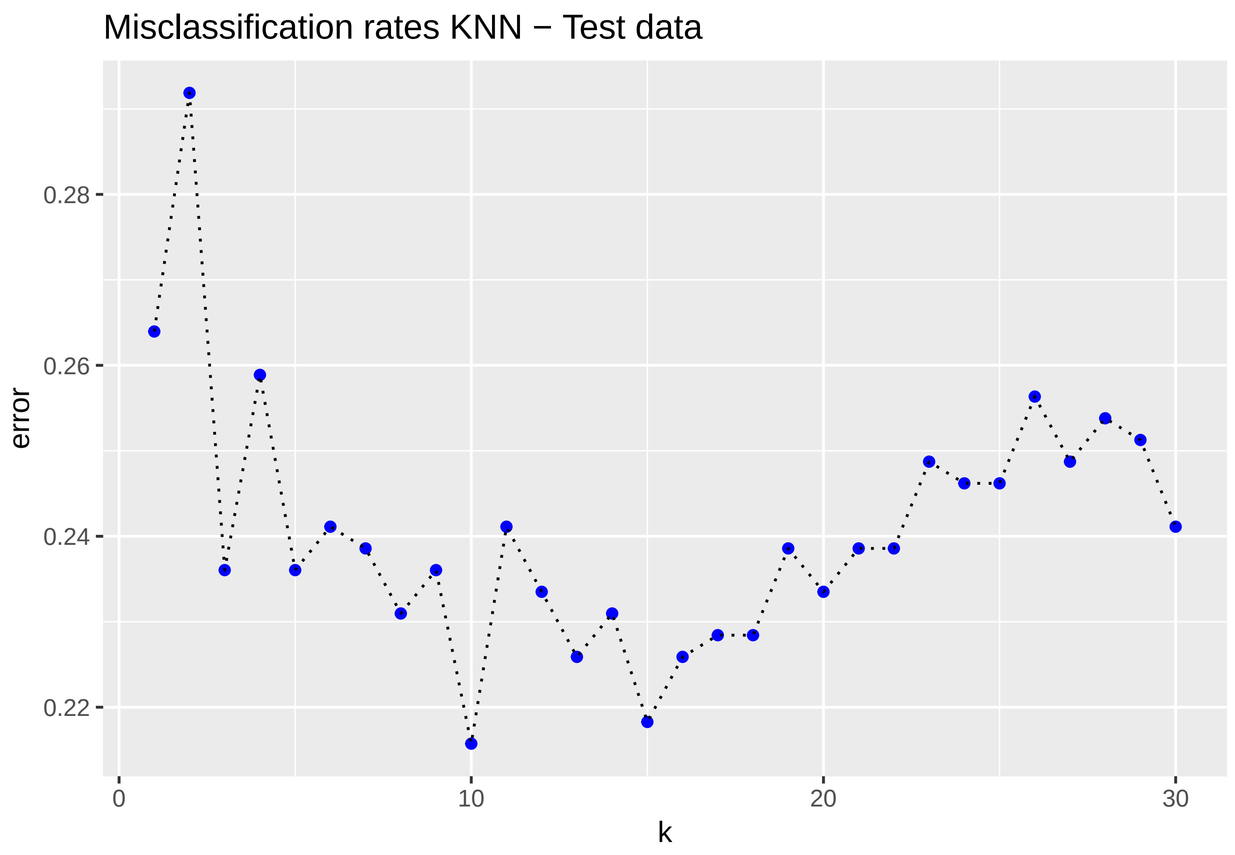
Q10:

```
# Misclassification error for training and test data using Knn with k = 1:30
knn.train = Mdf[tr,]
knn.test = Mdf[-tr,]

K = 30
train.error = rep(NA,K)
test.error = rep(NA,K)
for (k in 1:K){
  test.pred = class::knn(train = knn.train,test = knn.test,cl = knn.train$y,k = k)
  train.pred = class::knn(train = knn.train, test = knn.train, cl = knn.train$y, k =k)
  test.error[k] = mean(test.pred != knn.test$y)
  train.error[k] = mean(train.pred != knn.train$y)
}
train.e = train.error
test.e = test.error
```

```r
train.error.df = data.frame(k=1:K, error = train.error)
ggplot(train.error.df, aes(x=k, y=error))+geom_point(col="blue")+geom_line(linetype="dotted") + ggtitle
```



```r
test.error.df = data.frame(k=1:K, error = test.error)
ggplot(test.error.df, aes(x=k, y=error))+geom_point(col="blue")+geom_line(linetype="dotted") + ggtitle(
```



```r
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.
# "Sapply" is a more efficient for-loop.
# We loop over each fold and each value in "ks"
# and compute error rates for each combination.
# All the error rates are stored in the matrix "cv",
# where folds are rows and values of $K$ are columns.
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)
  })
})
```

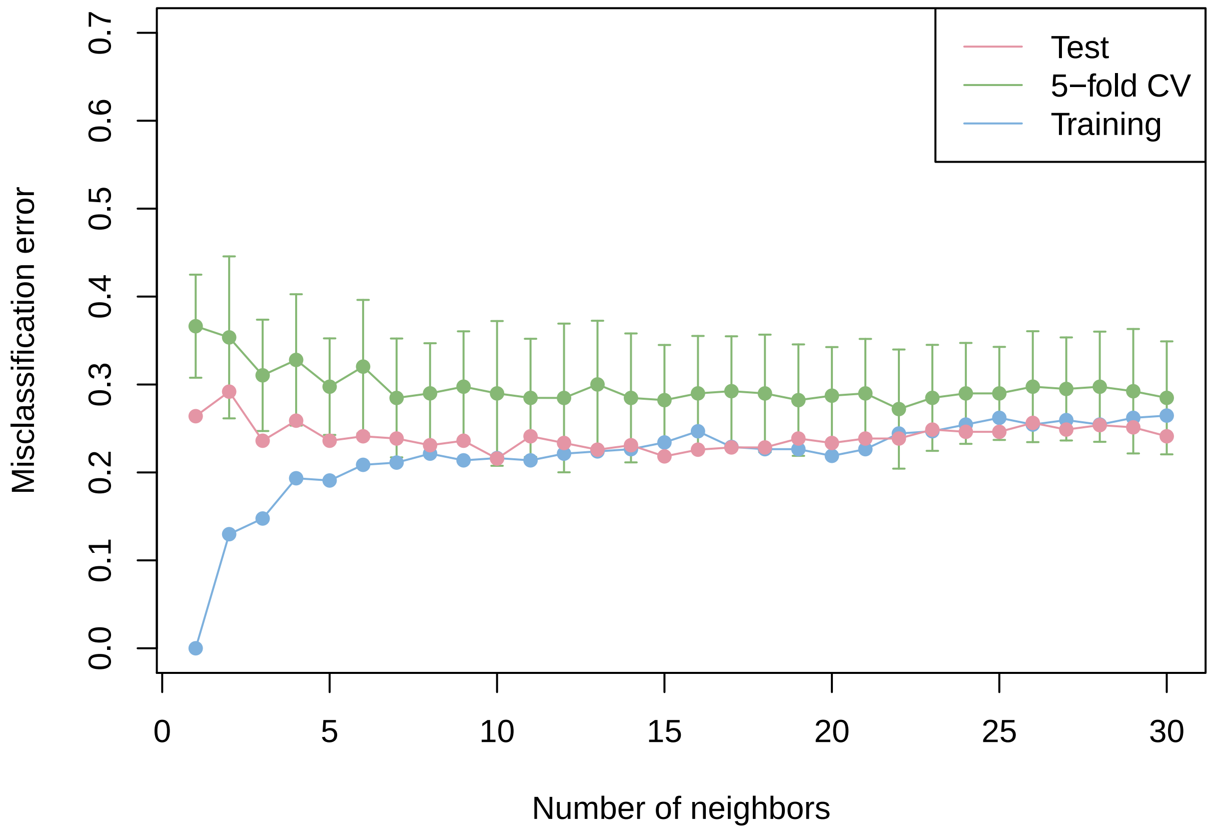
Q11:

```r
cv.e = colMeans(cv)
cv.se = apply(cv,2,sd)

k.min = which.min(cv.e)
print('K corresponding to the smallest CV error:')
k.min
```

```
## [1] "K corresponding to the smallest CV error:"
## [1] 22
```

Q12:

```r
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)
```

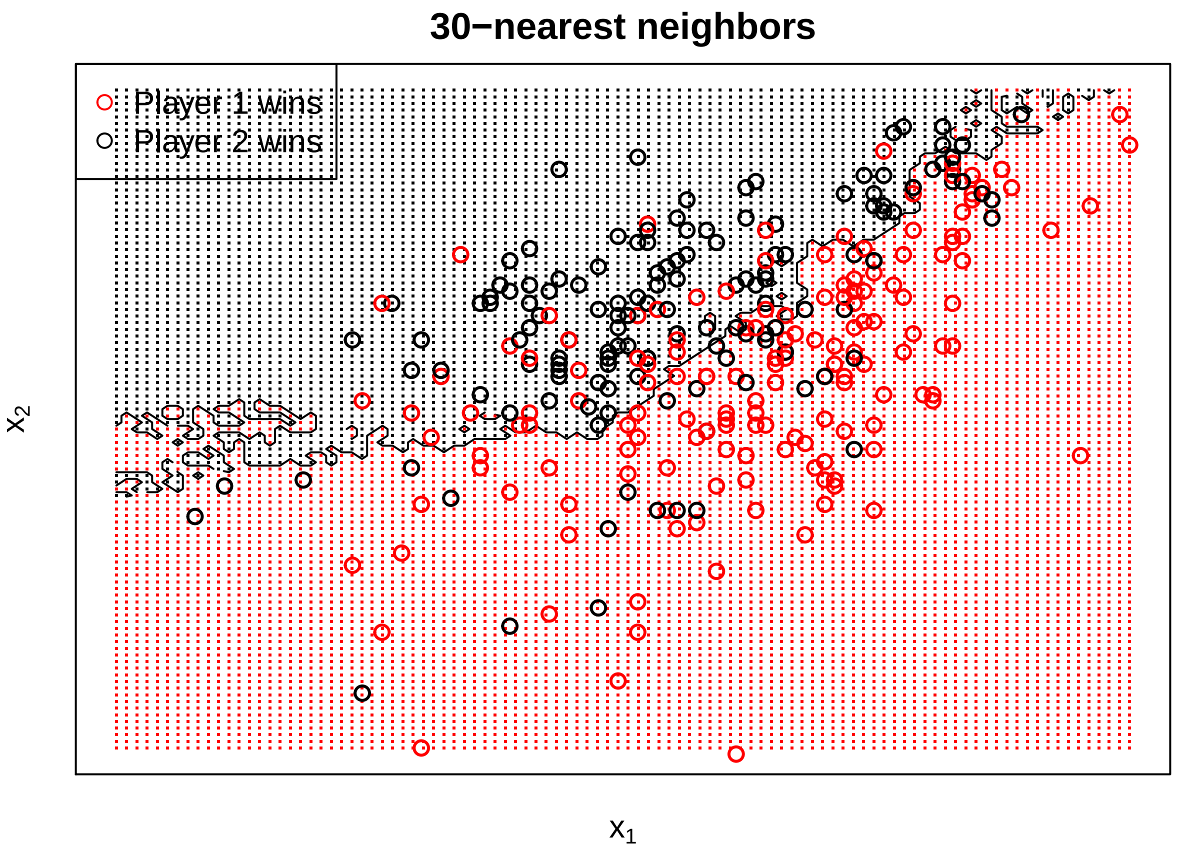


In K-nearest neighbour, the degree of flexibility decrease with K. That is, the lower the K, the lower the bias. Therefore, the bias increase with K. According to the bias-variance tradeoff, the variance will decrease with K.

Q13:

```r
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 = 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)
```

```r
legend("topleft", c("Player 1 wins", "Player 2 wins"),
       col=c("red", "black"), pch=1)
box()
```



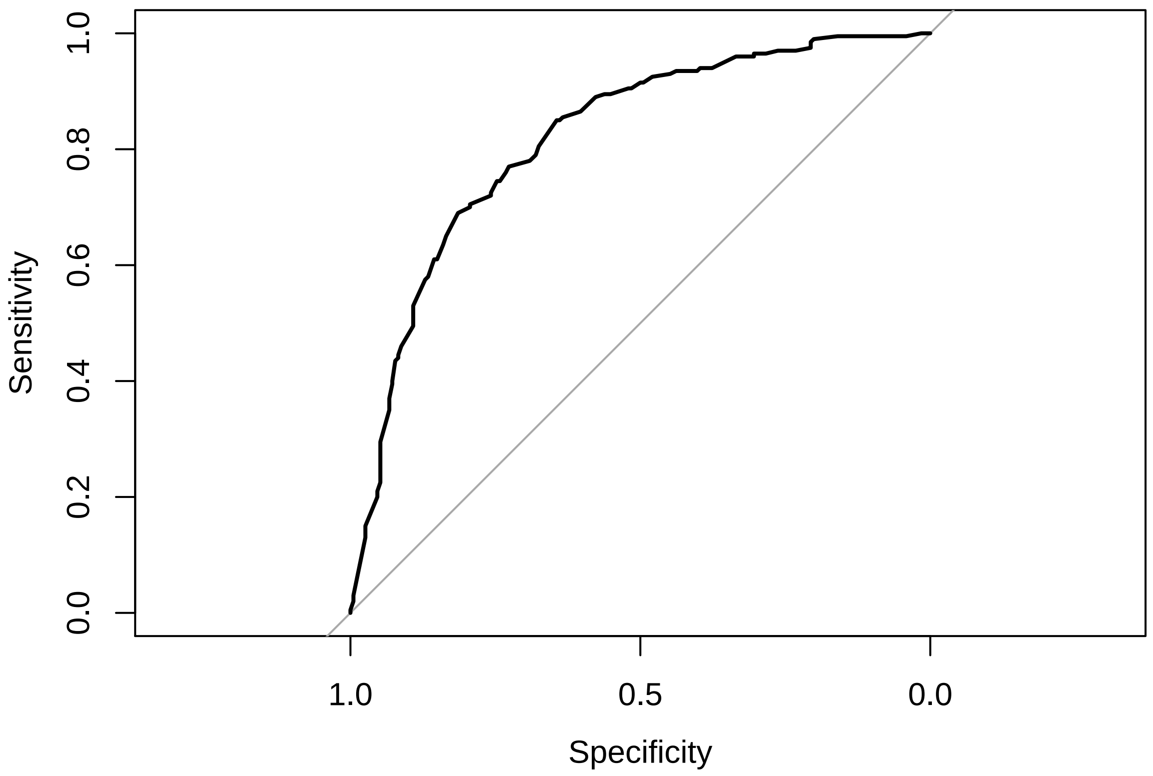
The strategy for finding K in the above code chunk is finding the greatest K in which the average K over all folds is less than the minimum K + its standard deviation.

Q14:

```r
K= 30
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
```

```r
# knn with prob=TRUE outputs the probability of the winning class
# therefore we have to do an extra step to get the probability of player 1 winning
auc = rep(NA,K)
for (k in 1:K){
  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

  KNN.roc = roc(response =M[-tr,1],predictor = KNNprob)
  auc[k] = auc(KNN.roc)
}
print("Maximum Area under the curve")
max(auc)
print('Occuring at K = ')
which(auc == max(auc))

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
KNN.roc = roc(response =M[-tr,1],predictor = KNNprob)
plot(KNN.roc)
```



```
print("K = 30")
auc(KNN.roc)
```

```
## [1] "Maximum Area under the curve"
## [1] 0.8200387
## [1] "Occuring at K = "
## [1] 17
## [1] "K = 30"
## Area under the curve: 0.8178
```

The ROC-curve is found by calculating the sensitivity and specificity of the model, where all possible values of the decicion thresholds are being considered (each point corresponds to a single threshold value). The AUC is the area under the ROC-curve, and a good classifier has a high value of AUC $\in [0, 1]$. The AUC for K = 30, our choice from the previous question, is 0.8178.

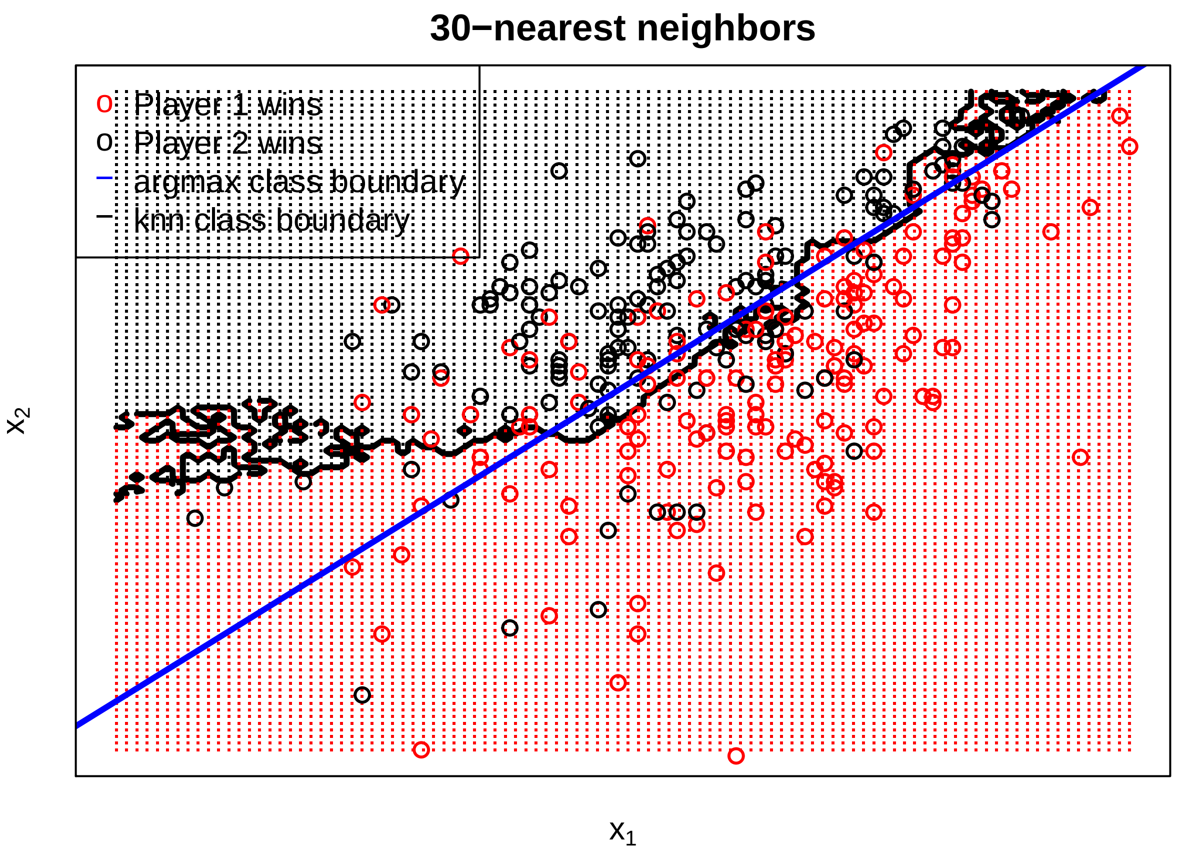
We also found that K = 17 yields the optimal, or largest value for the AUC (0.82003)

The interpretation of the AUC is the proportion of time the model ranks a random positive observation (true positive) higher than a random negative (false positive) observation.

Random guessing would rank a random positive observation higher than a negative observation 50% of the time, and thus the AUC of random guessing is 0.50.

Q15:

```
# new classifier y_hat(x) = argmax_k(x_k)

new_classifier = ifelse(M[-tr,2] > M[-tr,3],1,0)
argmax = as.factor(new_classifier)

conf_knn = confusionMatrix(test.pred,Mdf[-tr,1])
conf_knn$table

conf_argmax = confusionMatrix(argmax,Mdf[-tr,1])
conf_argmax$table


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 = 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="",col = 'black',lwd = 3)
points(grid, pch=".", cex=1, col=grid.yhat)
points(M[1:np,-1], col=factor(M[1:np,1]), pch = 1, lwd = 1.5)
abline(0,1,col='blue',pch='-',lwd = 3)

legend("topleft", c("Player 1 wins", "Player 2 wins",'argmax class boundary','knn class boundary'),
       col=c("red", "black",'blue','black'), pch=c('o','o','-','-'))
box()
```



```
table1 = conf_knn$table
table2 = conf_argmax$table
knn.misclasserror = (table1[1,2] + table1[2,1])/sum(table1)
knn.misclasserror
argmax.misclasserror = (table2[1,2] + table2[2,1])/sum(table2)
argmax.misclasserror
```

```
##           Reference
## Prediction   0   1
##          0 142  43
##          1  52 157
##           Reference
## Prediction   0   1
##          0 149  47
##          1  45 153
## [1] 0.2411168
## [1] 0.2335025
```

We see that the argmax classifier yields a lower misclassification error on the test set, i.e the argmax classifies to the correct class a greater proportion of time. Therefore, based on this metric, the argmax classifier is preferrable.

Problem 3: Bias-variance trade-off

Q16:

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

Expected value

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

$$= (X^TX)^{-1}X^TE[\mathbf{Y}]$$

$$= (X^TX)^{-1}X^TE[X\boldsymbol{\beta} + \boldsymbol{\epsilon}]$$

$$= (X^TX)^{-1}(X^TX)(E[\boldsymbol{\beta}] + E[\boldsymbol{\epsilon}])$$

$$= \boldsymbol{\beta}$$

Where we are assuming $\boldsymbol{\epsilon} \sim N(0, I\sigma^2)$.

$$\text{Cov}[\hat{\boldsymbol{\beta}}] = \text{Cov}[(X^TX)^{-1}X^T\mathbf{Y}]$$

$$= (X^TX)^{-1}X^T\text{Cov}[\mathbf{Y}]\bigg((X^TX)^{-1}X^T\bigg)^T$$

$$= (X^TX)^{-1}X^T\sigma^2I\bigg((X^TX)^{-1}X^T\bigg)^T$$

$$= (X^TX)^{-1}(X^TX)((X^TX)^{-1})^T\sigma^2$$

$$= \sigma^2((X^TX)^{-1})^T$$

$$= \sigma^2(X^TX)^{-1}$$

where we have used $\text{Cov}[\mathbf{Y}] = \sigma^2I$.

Q17: Expected value and variance of $\hat{f}(\mathbf{x_0})$

Expected value

$$E[\hat{f}(\boldsymbol{x_0})] = E[\boldsymbol{x_0^T}\hat{\boldsymbol{\beta}}]$$

$$= \boldsymbol{x_0}^TE[\hat{\boldsymbol{\beta}}]$$

$$= \boldsymbol{x_0^T}\boldsymbol{\beta}$$

Variance

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

Q18:

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

The Ridge estimator is given as

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

Q19: Expected value of the ridge estimator

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

Covariance of the ridge estimator

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

Q20: Expected value and variance of $\widetilde{f}(\mathbf{x_o}) = \mathbf{x_0}^T\widetilde{\boldsymbol{\beta}}$

Expected value

$$E[\widetilde{f}(\mathbf{x_o})] = E[\mathbf{x_0^T}\widetilde{\beta}]$$

$$= \mathbf{x_0^T} E[\widetilde{\beta}]$$

$$= \mathbf{x_0}^T (X^T X + \lambda I)^{-1} X^T X \boldsymbol{\beta}$$

Variance

$$\text{Var}[\widetilde{f}(\mathbf{x_0})] = \text{Var}[\mathbf{x_0}^T \widetilde{\beta}]$$

$$= \mathbf{x_0}^T \text{Var}[\widetilde{\beta}]$$

$$= \sigma^2 \mathbf{x_0}^T (X^T X + \lambda I)^{-1} X^T X (X^T X + \lambda I)^{-1} \mathbf{x_0}$$

Q21:

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

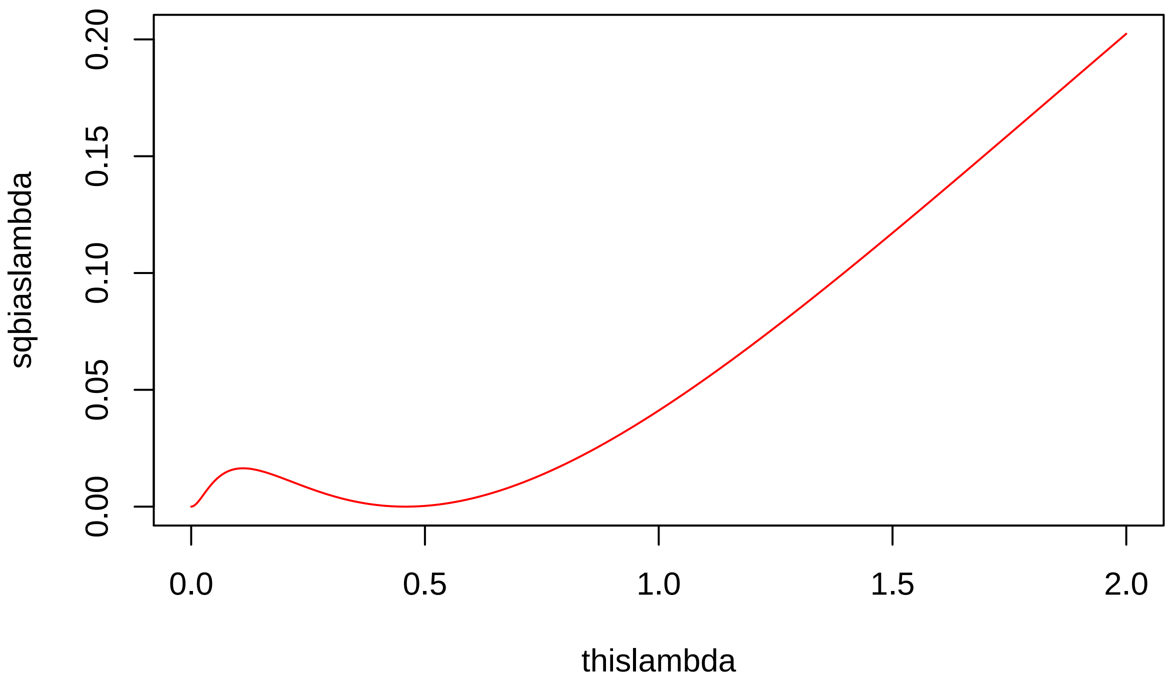
```r
values=dget("https://www.math.ntnu.no/emner/TMA4268/2019v/data/BVtradeoffvalues.dd")
X=values$X
dim(X)
x0=values$x0
dim(x0)
beta=values$beta
dim(beta)
sigma=values$sigma
sigma
```

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

Q22:

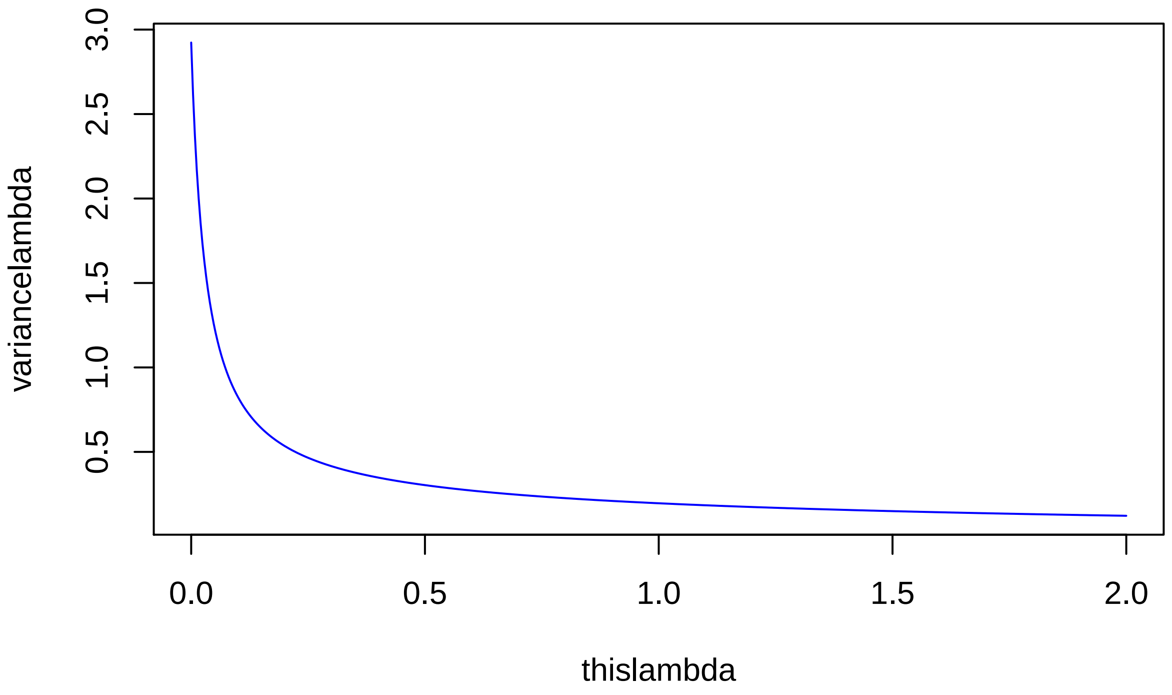
```r
sqbias=function(lambda,X,x0,beta)
{
  p=dim(X)[2]
  inv=solve(t(X)%*%X+lambda*diag(p))
  value=(t(x0)%*%inv%*%t(X)%*%X%*%beta-t(x0)%*%beta)^2
  #value=(t(x0)%*%(diag(p)+lambda*solve(t(X)%*%X))%*%beta-t(x0)%*%beta)^2
  return(value)
}
thislambda=seq(0,2,length=500)
sqbiaslambda=rep(NA,length(thislambda))
```

```r
for (i in 1:length(thislambda)) sqbiaslambda[i]=sqbias(thislambda[i],X,x0,beta)
plot(thislambda,sqbiaslambda,col=2,type="l")
```



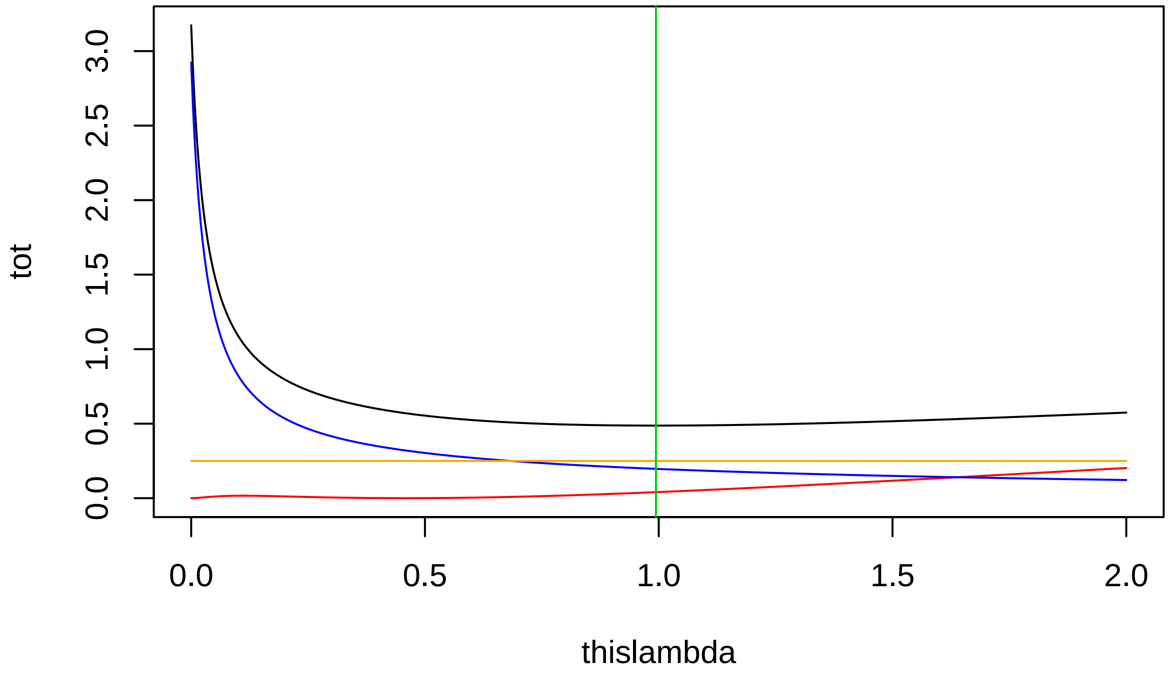
Q23:

```r
variance=function(lambda,X,x0,sigma)
{
  p=dim(X)[2]
  inv=solve(t(X)%*%X+lambda*diag(p))
  value=sigma^2*t(x0)%*%inv%*%t(X)%*%X%*%t(inv)%*%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")
```



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
```

We observe that the 295. lambda value in the thislambda-vector yields the lowest value for the total value of $\mathrm{E}[(Y_0 - \widetilde{f}(\mathbf{x}_0))^2]$. This corresponds to $\lambda = 0.998 \approx 1$.