Compulsory exercise 1: Group 2

TMA4268 Statistical Learning V2019

Anders Thallaug Fagerli, Rebecca Sandstø

20 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: Write down the equation for the fitted modelA.

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 $\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{\epsilon}$. 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 sqaures RSS or maximizing the likelihood function with respect to $\boldsymbol{\beta}$, resulting in the estimated vector of coefficients $\hat{\boldsymbol{\beta}} = (\mathbf{X^T X})^{-1}\mathbf{X^T Y}$. 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 $\text{Cov}(\hat{\beta}) = \sigma^2(\mathbf{X^T X})^{-1}$. 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:

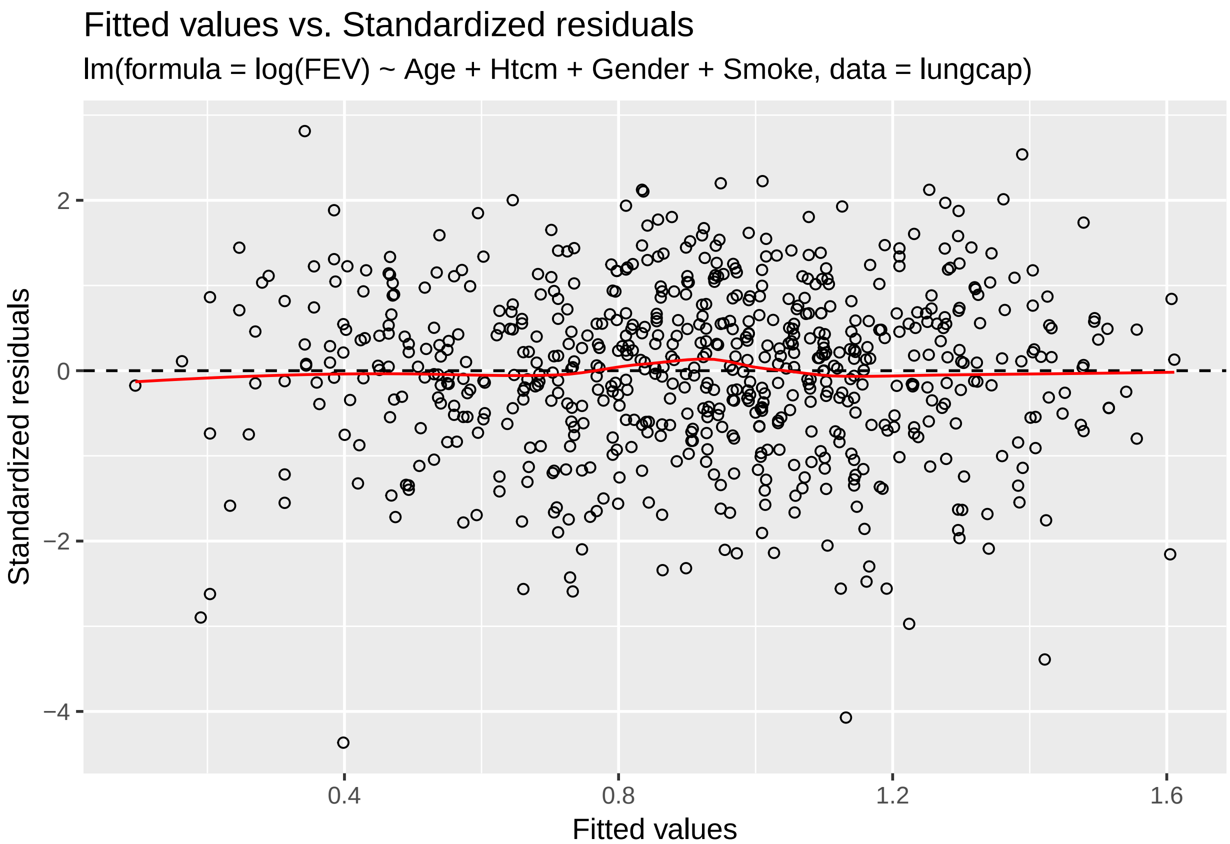
The proportion of variability is given by the coefficient of determination

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

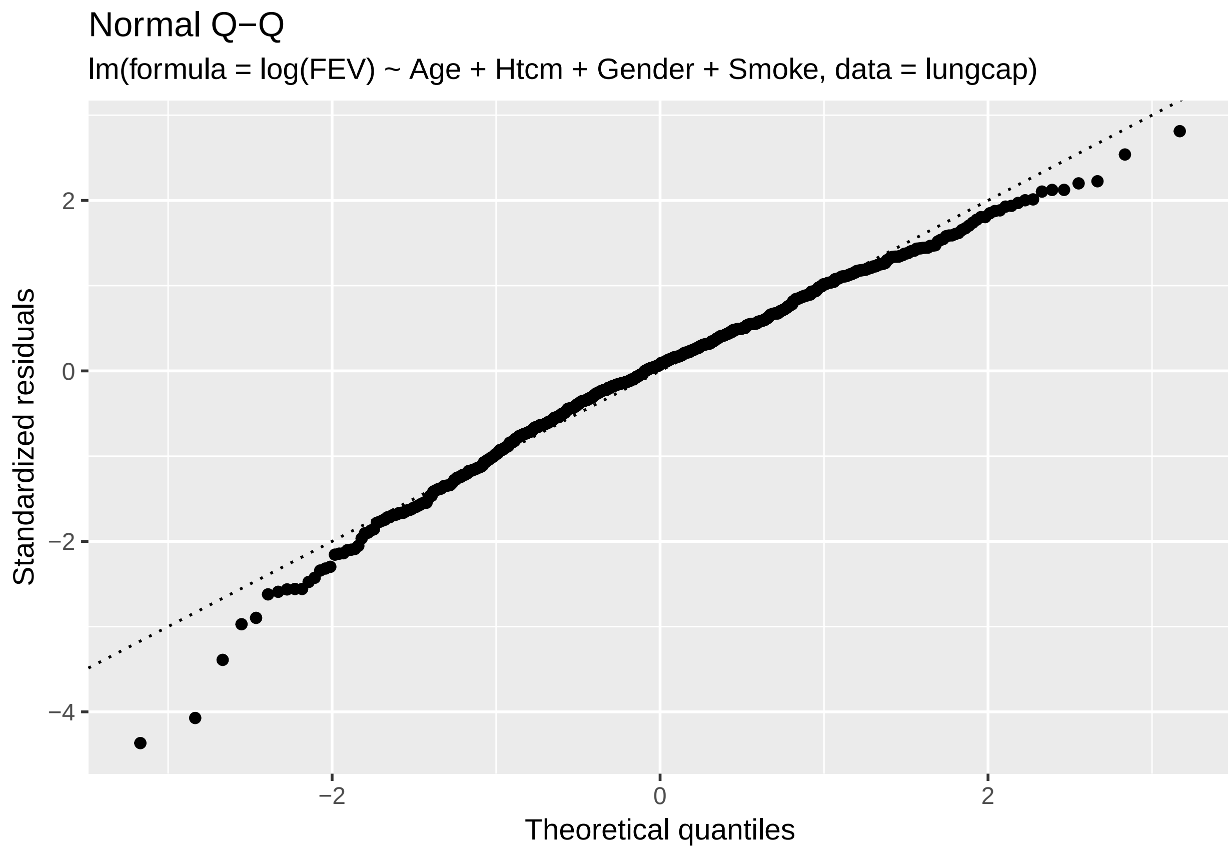
This means 81% of the variance in $\boldsymbol{Y} = log(\text{FEV})$ is explained in our model, implying a good fit. The closer the coefficient is to 1, the better, as $R^2 = 1$ means all residuals are zero.

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))
```




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

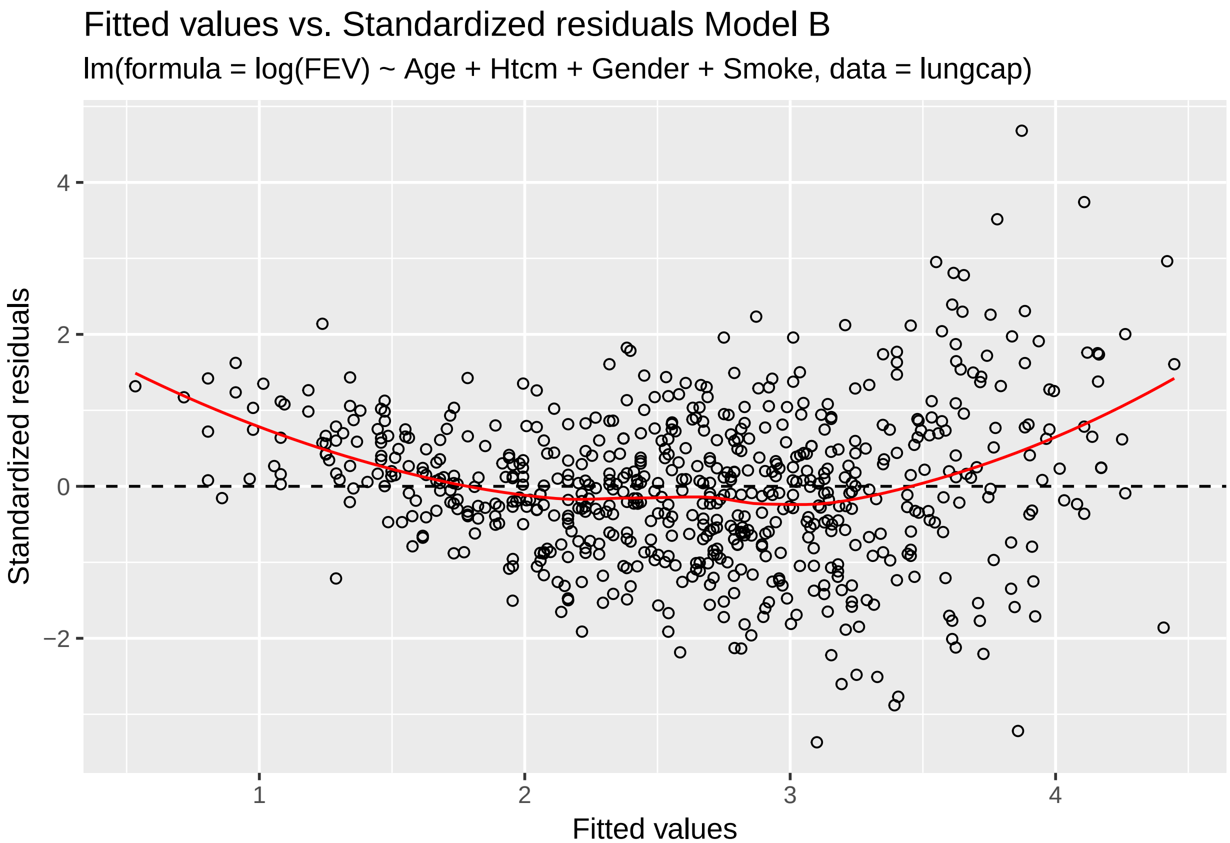
```
##
##  Anderson-Darling normality test
##
## data:  rstudent(modelA)
## A = 1.9256, p-value = 6.486e-05
##
##
##  Anderson-Darling normality test
##
## data:  rstudent(modelA)
## A = 1.9256, p-value = 6.486e-05
```

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 the assumptions needed for a linear regression model.

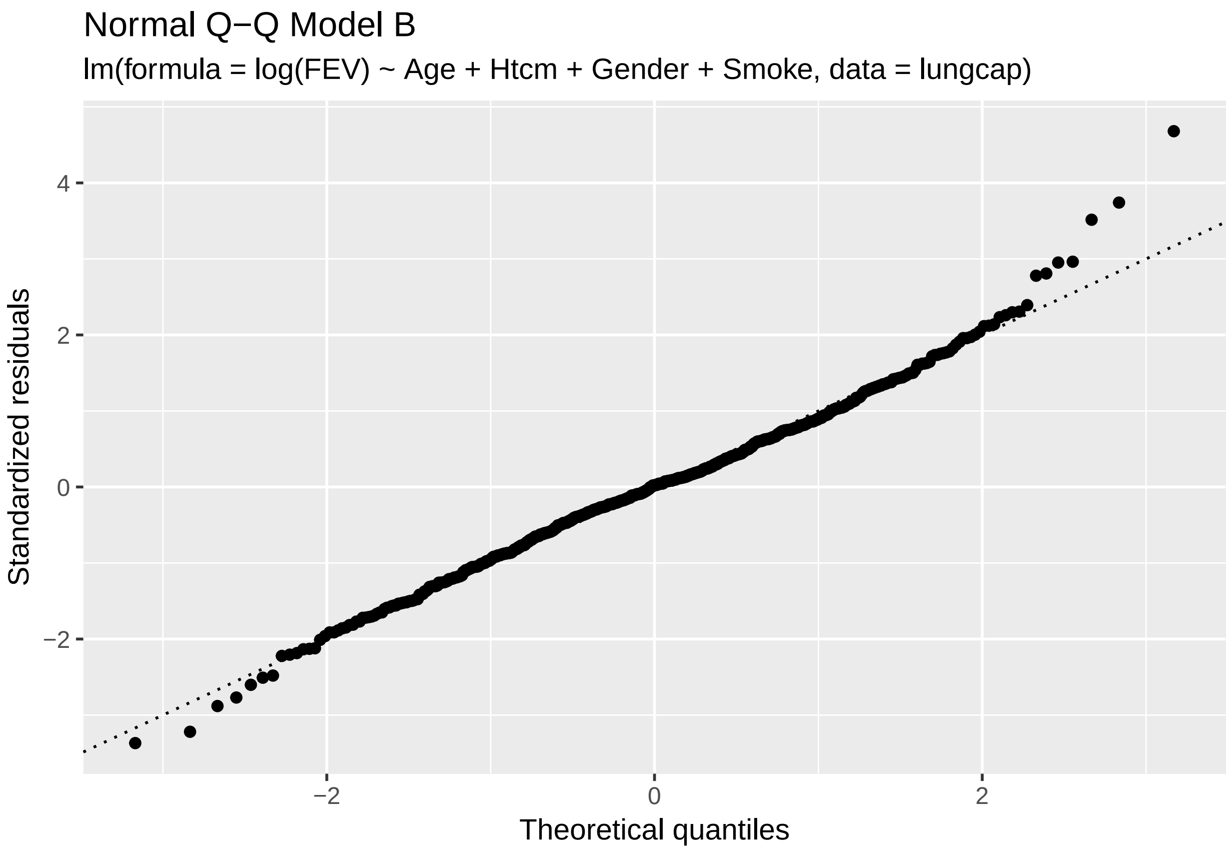
QQ-plot of standardized residuals: The QQ-plot shows whether the residuals are normally distributed or not, by comparing the distribution of the residuals against the normal distribution depicted by the straight line. The residuals form a small curve on top of the normal distribution with greater deviation 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.

Q5:

```
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))
```




```
# 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))
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
##
##
##  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. WHICH MODEL TO CHOOSE??

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 $\text{SD}(\hat{\beta}_{age})$, shown in the `summary`, and testing the hypothesis $H_0 : \beta_{age} = 0$.

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

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 intervall 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},$\sqrt{\text{Var}(\hat{\beta}_{age})}$ and solving the inequalities for β_{age}, we can construct the intervall for a given significance α.

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

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 $\text{SD}(\hat{\beta}_{age})$, we get the interval,

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

This intervall can be verified in R with

```r
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 intervall tells us that the true value of β_{age} lies within $[0.0147, 0.0320]$ in 95% of the intervalls constructed if we were to make several intervalls. 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:

```r
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]*16+summary(modelA)$coeff[3.1]*170+su
best_guess
#predicition interval
pred.log=predict(modelA, new, interval = 'prediction') #using the default 95% prediction level
pred.log
pred=exp(pred.log)
pred
```

```
## [1] 1.277734
##        fit     lwr      upr
## 1 1.323802 1.03616 1.611444
##       fit      lwr      upr
## 1 3.75768 2.818373 5.010038
```

The prediction interval is very wide compared to the actual span in the data (0.7 to 5.7), and it contains all the values for other males his age. Therefore it does not tell us much, and we conclude that our model is not very good at prediction.

Problem 2: Classification

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

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

```r
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:

```r
# here you write your code
### 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) #Oppgi test = M[tr,-1], altså test-
})
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, resulting in vector
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) #Riktig måte for test-set??
})
test.equal = I(yhat.test == M[-tr,1]) #Returns boolean matrix of correct/incorrect classifications
test.e = apply(test.equal == FALSE, 2, sum) # Apply summation over all coloumns, resulting in vector of
test.e = test.e/nrow(test.equal) # Misclassification rate for each k
```

```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 = 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:

```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)
```

Bias does in general increase with K, as the KNN-algorithm tends to overfit the data with low values of K, resulting in a low bias. The variance however decreases with K. Explain why, and justify from plot

Q13:

The proposed strategy uses the "one standard error rule". It chooses the simplest model (e.g.highest value of K) whose error is smaller than the the minimal error + its standard error. In our case: K=30.

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

Q14:

Random guessing will produce the ROS as a straight line along the diagonal.. Along this line 50% of the guesses will have been right, and the rest wrong. With a 50/50 chance of guessing right this line will represent random guessing… As the AUC is the area under the curve, it will in that case be 0.5.

```
K=30# your choice from Q13

# 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
KNNclass=class::knn(train=M[tr,-1], cl=M[tr,1], test=M[-tr,-1], k = K,prob=TRUE)
KNNprobwinning=attributes(KNNclass)$prob
KNNprob= ifelse(KNNclass == "0", 1-KNNprobwinning, KNNprobwinning)
# now KNNprob has probability that player 1 wins, for all matches in the test set
library(pROC)
# now you use predictor=KNNprob and response=M[-tr,1]
# in your call to the function roc in the pROC library
tennis_roc = roc(response = M[-tr,1], predictor = KNNprob)
auc = tennis_roc$auc
ggroc(tennis_roc)+ggtitle("ROC")
tennis_roc
auc
```

Q15:

```
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
  }
}
```

```r
#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 = 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)
#
# segments(-88, -96, 15, 12) #desicion boundary
#
# legend("topleft", c("Player 1 wins", "Player 2 wins"),
#        col=c("red", "black"), pch=1)
# box()
library(caret)
library(e1071)
ref = M[tr,1]
#confusion matrix for KNN
#confmat_KNN=(table(predicted classes KNN, ref))
#confmat_KNN
#misclass_KNN=(sum(confmat_KNN) - sum(diag(confmat_KNN)))/sum(confmat_KNN)
#misclass_KNN
#confusion matrix for argmax
#confmat2=confusionMatrix(table(y_tilde, ref))
#confmat2
confmat_argmax=(table(y_tilde, ref))
confmat_argmax
misclass_argmax=(sum(confmat_argmax) - sum(diag(confmat_argmax)))/sum(confmat_argmax)
misclass_argmax
```

```
##        ref
## y_tilde   0   1
##       0 151  61
##       1  47 134
## [1] 0.2748092
```

We prefer classifier:.. with misclassification rate XX %

Problem 3: Bias-variance trade-off

Here you see how to write formulas with latex (needed below)

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

Q16:

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

$$\mathrm{Cov}(\hat{\boldsymbol\beta}) = \mathrm{Cov}((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}) = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathrm{Cov}(\mathbf{Y})[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T]^T = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}[(\mathbf{X}^\mathbf{T}\mathbf{X})^{-\mathbf{1}}]^\mathbf{T} = \sigma^\mathbf{2}(\mathbf{X}^\mathbf{T}\mathbf{X})^{-\mathbf{1}} = \sigma$$

Q17:

$$\mathrm{E}(\hat{f}(\mathbf{x}_0)) = \mathrm{E}(\mathbf{x}_0^T\hat{\boldsymbol\beta}) = \mathbf{x}_0^T\mathrm{E}(\hat{\boldsymbol\beta}) = \mathbf{x}_0^T\boldsymbol\beta = f(\mathbf{x}_0)$$

$$\mathrm{Var}(\hat{f}(\mathbf{x}_0)) = \mathrm{Var}(\mathbf{x}_0^T\hat{\boldsymbol\beta}) = \mathrm{Var}(\hat\beta_0) + (x_{01}^2\mathrm{Var}(\hat\beta_1)) + ... + (x_{0p}^2\mathrm{Var}(\hat\beta_p)) = \mathbf{x}_0^T \circ \mathbf{x}_0^T\mathrm{diag}(\mathrm{Cov}(\hat{\boldsymbol\beta})) = \sigma^2\sum_{i=1}^{p+1}\mathbf{x}_{0i}^{T2}\mathbf{A}_{jj}$$

Q18:

$$\mathrm{E}[(Y_0-\hat{f}(\mathbf{x}_0))^2] = [\mathrm{E}(\hat{f}(\mathbf{x}_0)-f(\mathbf{x}_0)]^2+\mathrm{Var}(\hat{f}(\mathbf{x}_0))+\mathrm{Var}(\varepsilon) = \mathrm{E}[(Y_0-\hat{f}(\mathbf{x}_0))^2] = \mathrm{E}[(Y_0^2-2Y_o\hat{f}(\mathbf{x}_0)+\hat{f}(\mathbf{x}_0)^2] = \mathrm{Var}(Y_0)+\mathrm{E}(Y_0)^2$$

Ridge estimator:

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

Q19:

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

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

Q20:

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

$$\mathrm{Var}(\widetilde{f}(\mathbf{x}_0)) = \mathrm{Var}(\mathbf{x}_0^T\widetilde{\beta}) = \mathrm{Var}(\widetilde{\beta}_0 + x_{01}\widetilde{\beta}_1 + ... + x_{0p}\widetilde{\beta}_p) = \mathbf{x}_0^T \circ \mathbf{x}_0^T\mathrm{diag}(\mathrm{Cov}(\widetilde{\boldsymbol\beta})) = \sigma^2\sum_{i=1}^{p+1}\mathbf{x}_i^2C_{ii}$$

Q21:

$$\mathrm{E}[(Y_0-\widetilde{f}(\mathbf{x}_0))^2] = [\mathrm{E}(\widetilde{f}(\mathbf{x}_0)-f(\mathbf{x}_0)]^2+\mathrm{Var}(\widetilde{f}(\mathbf{x}_0))+\mathrm{Var}(\varepsilon)$$

$$\mathrm{E}[(Y_0-\hat{f}(\mathbf{x}_0))^2] = \mathrm{E}[(Y_0^2-2Y_o\widetilde{f}(\mathbf{x}_0)+\widetilde{f}(\mathbf{x}_0)^2] = \mathrm{Var}(Y_0)+\mathrm{E}(Y_0)^2-2\mathrm{E}(Y_0)\mathrm{E}(\widetilde{f}(\mathbf{x}_0))+\mathrm{Var}(\widetilde{f}(\mathbf{x}_0))+\mathrm{E}(\widetilde{f}(\mathbf{x}_0))^2 = \mathrm{Var}(\varepsilon)+\mathrm{Var}(\widetilde{f}(\mathbf{x}$$

```
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
```

Hint: we perform matrix multiplication using `%*%`, transpose of a matrix `A` with `t(A)` and inverse with `solve(A)`.

Q22:

In general, increased lambda gives increased bias.

```
#library(Metrics)
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")
```

Q23:

As expected, the variance decreases when lambda increases.

```
variance=function(lambda,X,x0,sigma)
{
  p=dim(X)[2]
  inv=solve(t(X)%*%X+lambda*diag(p))
  var=diag(sigma*inv%*%t(X)%*%X%*%inv)
  x_prod = t(x0)*t(x0)
  value=x_prod[1,]%*%var
  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:

Optimal lambda is:

```
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)
```