Compulsory exercise 1: Group 13

TMA4268 Statistical Learning V2020

Vemund Tjessem, Erik Andre Klepp Vik

16 mars, 2020

```
# install.packages('knitr') #probably already installed
# install.packages('rmarkdown') #probably already installed
# install.packages('ggplot2') #plotting with ggplot install.packages('ggfortify')
# install.packages('MASS') install.packages('dplyr')
library(knitr)
library(rmarkdown)
library(ggplot2)
library(ggfortify)
library(MASS)
library(dplyr)
```

Problem 1

a)

The expected MSE for the function $\hat{f}(x_i)$ is

$$MSE_{train} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{f}(x_i))^2 \tag{1}$$

{i}

Further, the expected test mean squared error (MSE) at x_0

$$E[y_0 - \hat{f}(x_i)]^2 \tag{2}$$

b)

This shows that the error term can be decomposed into three terms, that is the irreducible error, the variance of prediction and the squared bias, respectively.

$$\begin{aligned}
E[y_0 - \hat{f}(x_0)]^2 &= E[(y_0 - E(\hat{f}(x_0)) - \hat{f}(x_0))^2] && (3)\\
&= [(y_0 - E(y_0))^2 + 2((y_0 - E(y_0))(E(y_0) - \hat{f}(x_0))(E(y_0) - \hat{f}(x_0))^2] && (4)\\
&= E[(y_0 - E(y_0))^2] + E[(E(y_0) - \hat{f}(x_0))^2] + \epsilon && (5)\\
&= Var(\epsilon) + Var(\hat{f}(x_0)) + (f(x_0) - E[\hat{f}(x_0)])^2 && (6)
\end{aligned}$$

c)

- Irreducible error: This term cannot be reduced regardless how well our statistical model fits the data.
- Variance of the prediction at $\hat{f}(x_0)$. Relates to the amount by which $\hat{f}(x_0)$ is expected to change for different training data. If the variance is high, there is large uncertainty associated with the prediction.
- Squared bias. The bias gives an estimate of how much the prediction differs from the true mean. If the bias is low the model gives a prediction which is close to the true value.

d)

(i) TRUE
(ii) TRUE
(iii) FALSE
(iv) TRUE

e)

(i) TRUE
(ii) TRUE
(iii) TRUE
(iv) FALSE

f)

(ii) 0.17

g)

Contour plot with $\sigma_x = 1$, $\sigma_y = 2$ and $\rho = 0.1$. This implies best correlation with figure C.

Problem 2

```
id <- "1nLen1ckdnX4P9n8ShZeU7zbXpLc7qiwt"  # google file ID
d.worm <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download", id))
head(d.worm)
```

```
##   Gattung Nummer GEWICHT FANGDATUM MAGENUMF
## 1      Oc     32    0.19  23.09.97     1.56
## 2      Oc     34    0.59  23.09.97     1.63
## 3      Oc     48    0.09  23.09.97     1.69
## 4      Oc     55    0.23  23.09.97     1.69
## 5      Oc     41    0.24  23.09.97     1.75
## 6      Oc     24    0.19  23.09.97     1.81
```

```
str(d.worm)
```

```
## 'data.frame':    143 obs. of  5 variables:
##  $ Gattung  : Factor w/ 3 levels "L","N","Oc": 3 3 3 3 3 3 3 3 3 3 ...
##  $ Nummer   : int  32 34 48 55 41 24 39 35 45 27 ...
##  $ GEWICHT  : num  0.19 0.59 0.09 0.23 0.24 0.19 0.26 0.19 0.15 0.34 ...
##  $ FANGDATUM: Factor w/ 3 levels "12.10.97","15.09.97",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ MAGENUMF : num  1.56 1.63 1.69 1.69 1.75 1.81 1.81 1.88 2 2.13 ...
```

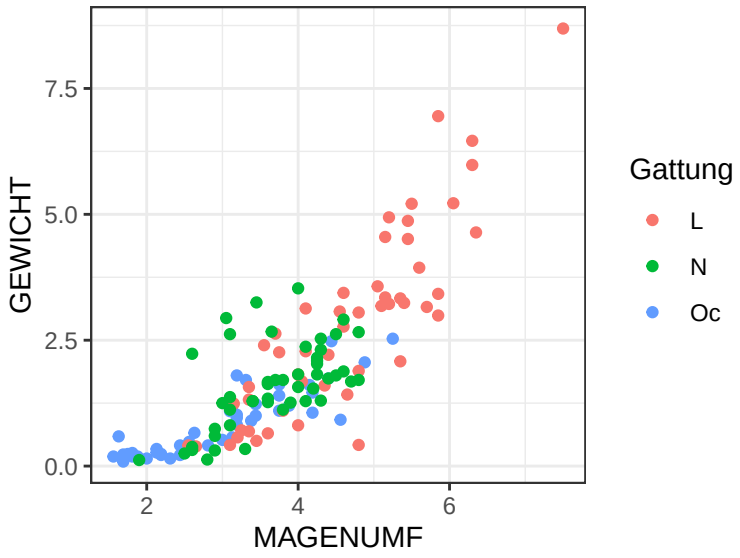
```
attach(d.worm)
```

a)

The worm dataset has 143 rows and 5 columns. That is, 5 variables are recorded per worm observation. Out of these, Gattung, Fangdatum are qualitative variables, and Nummer, GEWICHT and MAGENUMF are quantitative. That is, we have 2 qualitative and 3 quantitative variables.

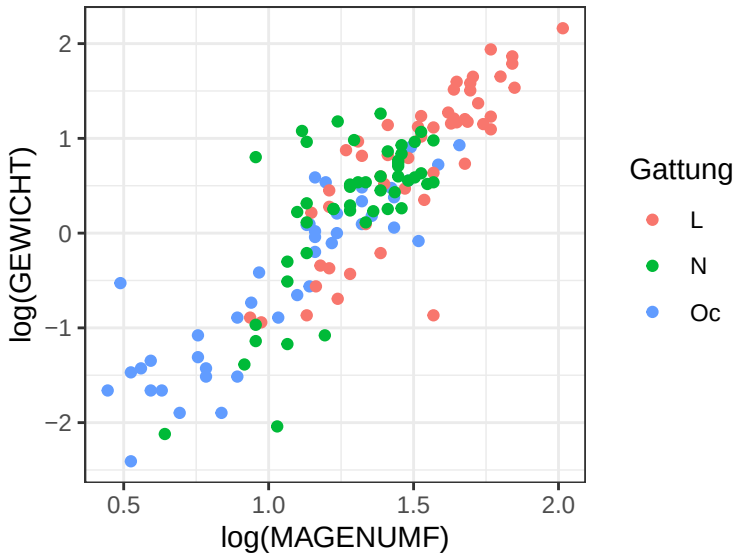
b)

```
d.worm$Gattung <- as.factor(d.worm$Gattung)
ggplot(d.worm, mapping = aes(x = MAGENUMF, y = GEWICHT, colour = Gattung)) + geom_point() +
    theme_bw()
```

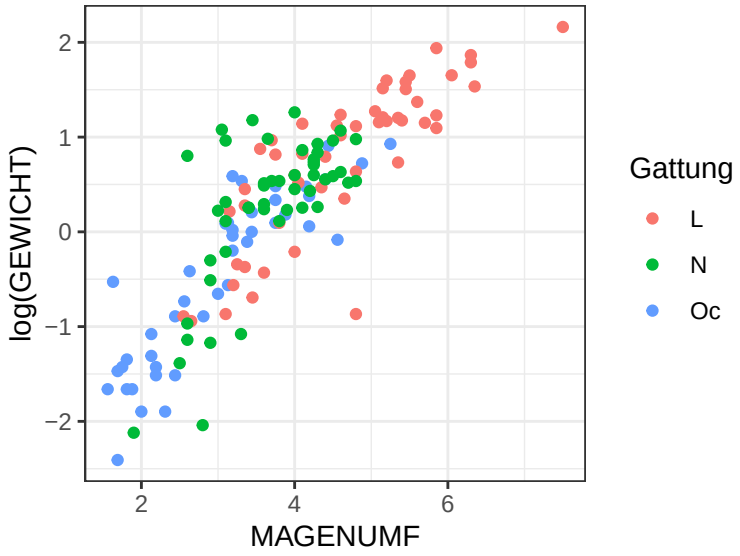


The relationship does not look linear, therefore different log responses were tried out.

```
ggplot(d.worm, mapping = aes(x = log(MAGENUMF), y = log(GEWICHT), colour = Gattung)) +
    geom_point() + theme_bw()
```




```
ggplot(d.worm, mapping = aes(x = MAGENUMF, y = log(GEWICHT), colour = Gattung)) +
    geom_point() + theme_bw()
```



It shows that log transformations were needed to make the relationship look linear. To simplify the rest of the task, only the log of the response is used.

c)

Plot with linear interaction term:

```
lm.fit <- lm(log(GEWICHT) ~ MAGENUMF + Gattung, data = d.worm)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = log(GEWICHT) ~ MAGENUMF + Gattung, data = d.worm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.74894 -0.27575  0.02197  0.28513  1.30867
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.53555    0.22147 -11.449   <2e-16 ***
## MAGENUMF     0.71187    0.04529  15.719   <2e-16 ***
## GattungN     0.17801    0.11009   1.617    0.108
## GattungOc   -0.09073    0.12791  -0.709    0.479
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5059 on 139 degrees of freedom
## Multiple R-squared:  0.7367, Adjusted R-squared:  0.731
## F-statistic: 129.6 on 3 and 139 DF,  p-value: < 2.2e-16
```

```
anova(lm.fit)
```

```
## Analysis of Variance Table
##
## Response: log(GEWICHT)
##            Df Sum Sq Mean Sq  F value  Pr(>F)
## MAGENUMF    1 97.802  97.802 382.0947 < 2e-16 ***
## Gattung     2  1.725   0.862   3.3691 0.03725 *
## Residuals 139 35.579   0.256
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The possible linear models are for GattungL, GattungN and GattungOc respectively

$$Y_1 = -2.53555 + 0.71187 * MAGENUMF \tag{7}$$
$$Y_2 = -2.35754 + 0.71187 * MAGENUMF \tag{8}$$
$$Y_3 = -2.62628 + 0.71187 * MAGENUMF \tag{9}$$

Gattung seems to have a small impact as a predictor for the models.

d)

```
lm.fitInteraction = lm(data = d.worm, formula = log(GEWICHT) ~ MAGENUMF * Gattung)
summary(lm.fitInteraction)
```

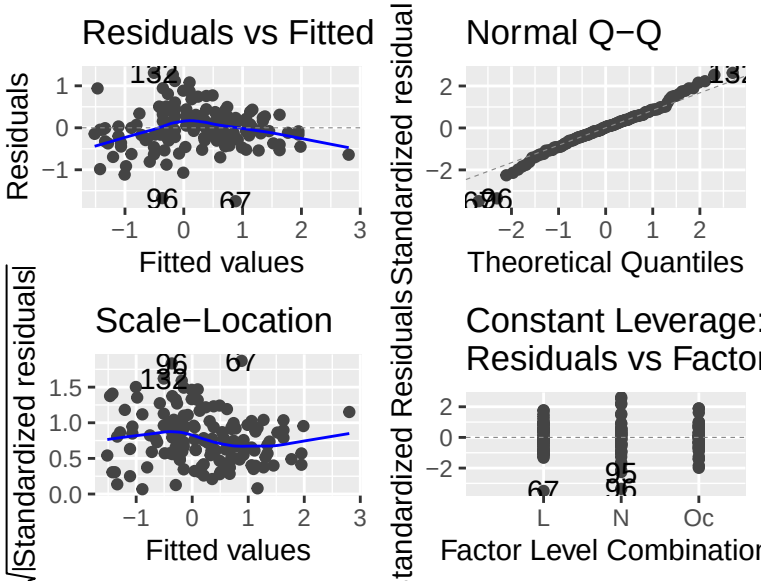
```
##
## Call:
## lm(formula = log(GEWICHT) ~ MAGENUMF * Gattung, data = d.worm)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.73426 -0.29551  0.04012  0.28248  1.37537
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -2.12744    0.30860  -6.894 1.82e-10 ***
## MAGENUMF            0.62379    0.06485   9.620  < 2e-16 ***
## GattungN           -0.45791    0.49399  -0.927   0.3556
## GattungOc          -0.78106    0.39669  -1.969   0.0510 .
## MAGENUMF:GattungN   0.15005    0.12178   1.232   0.2200
## MAGENUMF:GattungOc  0.18177    0.10200   1.782   0.0769 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.503 on 137 degrees of freedom
## Multiple R-squared:  0.7434, Adjusted R-squared:  0.7341
## F-statistic:  79.4 on 5 and 137 DF,  p-value: < 2.2e-16
```

We can observe that an interaction term is not relevant and despite beeing a relevant predictor, can be neglected.

e)

```
autoplot(lm.fit)
```



For the residual plot, we see a pattern, and it looks like linearity assumption is not met, because the variance follows a pattern. If the linearity condition was met, there would not be a pattern. For the Q-Q plot it looks like the plot follows the center line, which means the normal distribution assumption is true. For the standarized and residual plots we also see that there is no covariance, supporting the theory that the fit is good.

f)

For the analysis in e) we have used a linear regression model, which was made under 4 assumptions:

- Linearity and additivity of the relationship between dependent and independent variables:
- Statistical independence of the errors
- Homoscedasticity (constant variance) of the errors
- Normality of the error distribution.

The analysis of residuals plays an important role in validating the regression model. Since the statistical tests for significance are also based on assumptions, the conclusions resulting from these significance tests are called into question if the assumptions regarding ϵ are not satisfied.

The i-th residual is the difference between the observed value of the dependent variable, y_i, and the value predicted by the estimated regression equation, $\hat{y}_i$. These residuals, computed from the available data, are treated as estimates of the model error, ϵ. As such, they are used by statisticians to validate the assumptions concerning ϵ. Good judgment and experience play key roles in residual analysis.

How to fix: Consider applying a nonlinear transformation to the dependent and/or independent variables if you can think of a transformation that seems appropriate. For example, if the data are strictly positive, the log transformation is an option. If a log transformation is applied to the dependent variable only, this is equivalent to assuming that it grows (or decays) exponentially as a function of the independent variables. If a log transformation is applied to both the dependent variable and the independent variables, this is equivalent to assuming that the effects of the independent variables are multiplicative rather than additive in their original units. This means that, on the margin, a small percentage change in one of the independent variables induces a proportional percentage change in the expected value of the dependent variable, other things being equal. Models of this kind are commonly used in modeling price-demand relationships, as illustrated on the beer sales example on this web site.

g)

(i) FALSE
(ii) FALSE
(iii) FALSE
(iv) TRUE

Problem 3

Loading the files:

```
id <- "1GNbIhjdhuwPOBr0Qz82JMkdjUVBuSoZd"
tennis <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download", id),
    header = T)
head(tennis)
```

```
##         Player1      Player2 Result ACE.1 UFE.1 ACE.2 UFE.2
## 1     M.Koehler   V.Azarenka      0     2    18     3    14
## 2    E.Baltacha   F.Pennetta      0     0    10     4    14
## 3     S-W.Hsieh      T.Maria      1     1    13     2    29
## 4      A.Cornet       V.King      1     4    30     0    45
## 5  Y.Putintseva   K.Flipkens      0     2    28     6    19
## 6 A.Tomljanovic B.Jovanovski      0     6    42    11    40
```

a)

We have

$$p_i = \frac{e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\beta_3 x_{i3}+\beta_4 x_{i4}}}{1+e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\beta_3 x_{i3}+\beta_4 x_{i4}}} \tag{10}$$

which gives

$$\text{logit}(p_i) = \log\left(\frac{p_i}{1-p_i}\right) = \log\left(\frac{\frac{e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\beta_3 x_{i3}+\beta_4 x_{i4}}}{1+e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\beta_3 x_{i3}+\beta_4 x_{i4}}}}{1-\frac{e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\beta_3 x_{i3}+\beta_4 x_{i4}}}{1+e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\beta_3 x_{i3}+\beta_4 x_{i4}}}}\right) \tag{11}$$

Multiplying both the numerator and denominator in Equation 11 by $1+e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\beta_3 x_{i3}+\beta_4 x_{i4}}$ gives

$$\text{logit}(p_i) = \log\left(\frac{e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\beta_3 x_{i3}+\beta_4 x_{i4}}}{1+e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\beta_3 x_{i3}+\beta_4 x_{i4}}-(e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\beta_3 x_{i3}+\beta_4 x_{i4}})}\right) \tag{12}$$

which further results in

$$\text{logit}(p_i) = \log(e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\beta_3 x_{i3}+\beta_4 x_{i4}}) = \beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\beta_3 x_{i3}+\beta_4 x_{i4} \tag{13}$$

It can be seen from Equation 13 that $\text{logit}(p_i)$ is a linear function of the covariates x_{i1}, x_{i2}, x_{i3} and x_{i4}.

b)

```
r.tennis = glm(Result ~ ACE.1 + ACE.2 + UFE.1 + UFE.2, data = tennis, family = "binomial")
summary(r.tennis)
```

```
##
## Call:
## glm(formula = Result ~ ACE.1 + ACE.2 + UFE.1 + UFE.2, family = "binomial",
##     data = tennis)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0517  -0.8454   0.3725   0.8773   2.0959
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.02438    0.59302  -0.041 0.967211
## ACE.1        0.36338    0.10136   3.585 0.000337 ***
## ACE.2       -0.22388    0.07369  -3.038 0.002381 **
## UFE.1       -0.09847    0.02840  -3.467 0.000527 ***
## UFE.2        0.09010    0.02479   3.635 0.000278 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 163.04  on 117  degrees of freedom
## Residual deviance: 124.96  on 113  degrees of freedom
```

```
## AIC: 134.96
##
## Number of Fisher Scoring iterations: 4
```

Since the β_1 is positive one more ace for player one would increase the probability of player 1 winning. By incerasing x_i by 1 the odds ratio for $Y_i = 1$ increases by $\exp(\beta_1)$.

c)

```
# make variables for difference
tennis$ACEdiff = tennis$ACE.1 - tennis$ACE.2
tennis$UFEdiff = tennis$UFE.1 - tennis$UFE.2

# divide into test and train set
n = dim(tennis)[1]
n2 = n/2
set.seed(1234)  # to reproduce the same test and train sets each time you run the code
train = sample(c(1:n), replace = F)[1:n2]
tennisTest = tennis[-train, ]
tennisTrain = tennis[train, ]
```

The code for fitting a logistic regression model is given below.

```
# Fitting a logistic regression model on the form Result ~ ACEdiff + UFEdiff on
# the training set
fit.3c = glm(Result ~ ACEdiff + UFEdiff, data = tennisTrain, family = "binomial")
summary(fit.3c)
```

```
##
## Call:
## glm(formula = Result ~ ACEdiff + UFEdiff, family = "binomial",
##     data = tennisTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8546  -0.8968   0.4204   0.8247   1.9382
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.28272    0.31175   0.907  0.36447
## ACEdiff      0.22355    0.07959   2.809  0.00497 **
## UFEdiff     -0.08607    0.02832  -3.039  0.00237 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 80.959  on 58  degrees of freedom
## Residual deviance: 63.476  on 56  degrees of freedom
## AIC: 69.476
##
## Number of Fisher Scoring iterations: 4
```

The class boundary will be where $\hat{P}(Y = 1|\boldsymbol{x}) = 0.5$. When the probability is 0.5 we have $\text{logit}(p_i) = \log(1) = 0$

$$0 = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \tag{14}$$

This gives the class boundary

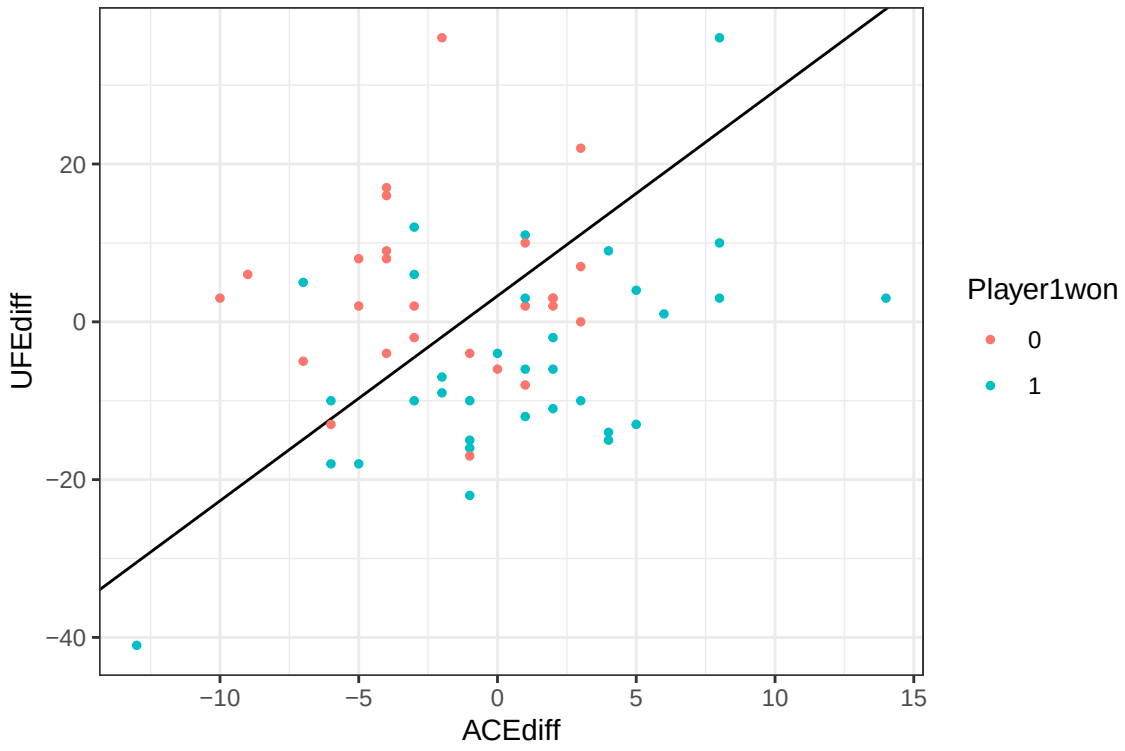
$$x_2 = -\frac{\beta_0}{\beta 2} - \frac{\beta_1}{\beta_2} x_1 \tag{15}$$

```
cof = coef(fit.3c)
a = -cof[1]/cof[3]
b = -cof[2]/cof[3]
```

The class boundary will be $x_2 = 3.285 + 2.597 x_1$

Making a plot of the training observations and the class boundary.

```
df = data.frame(tennisTrain, Player1won = as.factor(tennisTrain$Result))
ggplot(df, aes(x = ACEdiff, y = UFEdiff, color = Player1won)) + geom_abline(intercept = a,
    slope = b) + geom_point(size = 1) + theme_bw()
```



Making a confusion matrix

```
prd = predict(fit.3c, tennisTest, type = "response")
confMat = table(tennisTest$Result, prd > 0.5)
confMat
```

```
##
##      FALSE TRUE
##   0     22    7
##   1      6   24
```

The sensitivity is 0.8 and the specificity is 0.759

d)

- π_k is the prior class probabilities $\pi_k = \Pr(Y = k)$. In this case it will be the probability for player 1 winning and for player 1 losing.
- $\boldsymbol{\mu}_k$ is the mean of class k. In this case it will be a vector with the mean of difference in aces and difference in unforced errors.
- $\boldsymbol{\Sigma}$ is the covariance matrix, and in this case it is assumed equal for both classes since LDA is used. The diagonal elements will be the variance of the difference in aces the variance of the difference in unforced errors and the off-diagonal elements will be the covariance.
- $f_k(\boldsymbol{x})$ is the probability density function for $\boldsymbol{X}$ in class k, and is here assumed to be multivariate normally distributed with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}$.

e)

Part 1

$$P(Y = 0|\mathbf{X} = \boldsymbol{x}) = P(Y = 1|\mathbf{X} = \boldsymbol{x}) \tag{16}$$

The first step is to insert for the probability and the probability distribution.

$$\frac{\pi_0}{\sum_{l=1}^{K} \pi_l f_l(\boldsymbol{x}) 2\pi|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_0})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_0})} = \frac{\pi_1}{\sum_{l=1}^{K} \pi_l f_l(\boldsymbol{x}) 2\pi|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_1})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_1})} \tag{17}$$

Next is taking the logarithm on both sides, which gives

$$\log(\pi_0) - \frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0 + \boldsymbol{\mu}_0^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0 - \log\left(\sum_{l=1}^{K} \pi_l f_l(\boldsymbol{x}) 2\pi|\boldsymbol{\Sigma}|^{1/2}\right) = \tag{18}$$

$$\log(\pi_1) - \frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \log\left(\sum_{l=1}^{K} \pi_l f_l(\boldsymbol{x}) 2\pi|\boldsymbol{\Sigma}|^{1/2}\right) \tag{19}$$

After removing the terms not depending on k, which are equal on both sides

$$\log(\pi_0) + \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0 + \frac{1}{2}\boldsymbol{\mu}_0^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0 = \log(\pi_1) + \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 \tag{20}$$

which equals

$$\delta_0(\boldsymbol{x}) = \delta_1(\boldsymbol{x}) \tag{21}$$

Part 2

The class boundary will be the values for $\boldsymbol{x}$ where $\delta_0(\boldsymbol{x}) = \delta_1(\boldsymbol{x})$, hence Equation 20 can be used to find the class boundary.

$$\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1) = \log\left(\frac{\pi_1}{\pi_0}\right) + \frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \frac{1}{2}\boldsymbol{\mu}_0^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0 \tag{22}$$

Doing the matrix multiplications here will result in a equation on the form $ax_1 + bx_2 = c$, which can be used to find the class boundary on the form $x_2 = \frac{c}{b} - \frac{a}{b}x_1$.

Since π_k, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}$ are unknown, they have to be estimated. The estimators used are

- $\hat{\pi}_k = \frac{n_k}{n}$
- $\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k}\sum_{i:y_i=k}\mathbf{X}_i$
- $\hat{\boldsymbol{\Sigma}}_k = \frac{1}{n_k-1}\sum_{i:y_i=k}(\mathbf{X}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{X}_i - \hat{\boldsymbol{\mu}}_k)^T$
- $\hat{\boldsymbol{\Sigma}} = \sum_{k=1}^{K}\frac{n_k-1}{n-K}\cdot\hat{\boldsymbol{\Sigma}}_k$

```
k0s = subset(tennisTrain, tennisTrain$Result < 0.5)
k1s = subset(tennisTrain, tennisTrain$Result > 0.5)
n0 = length(k0s$UFEdiff)
n1 = length(k1s$UFEdiff)
pi0 = n0/(n0 + n1)
pi1 = n1/(n0 + n1)
mu0 = matrix(c(mean(k0s$ACEdiff), mean(k0s$UFEdiff)), nrow = 2)
mu1 = matrix(c(mean(k1s$ACEdiff), mean(k1s$UFEdiff)), nrow = 2)
mu0
```

```
##           [,1]
## [1,] -2.076923
## [2,]  3.730769
```

```
mu1
```

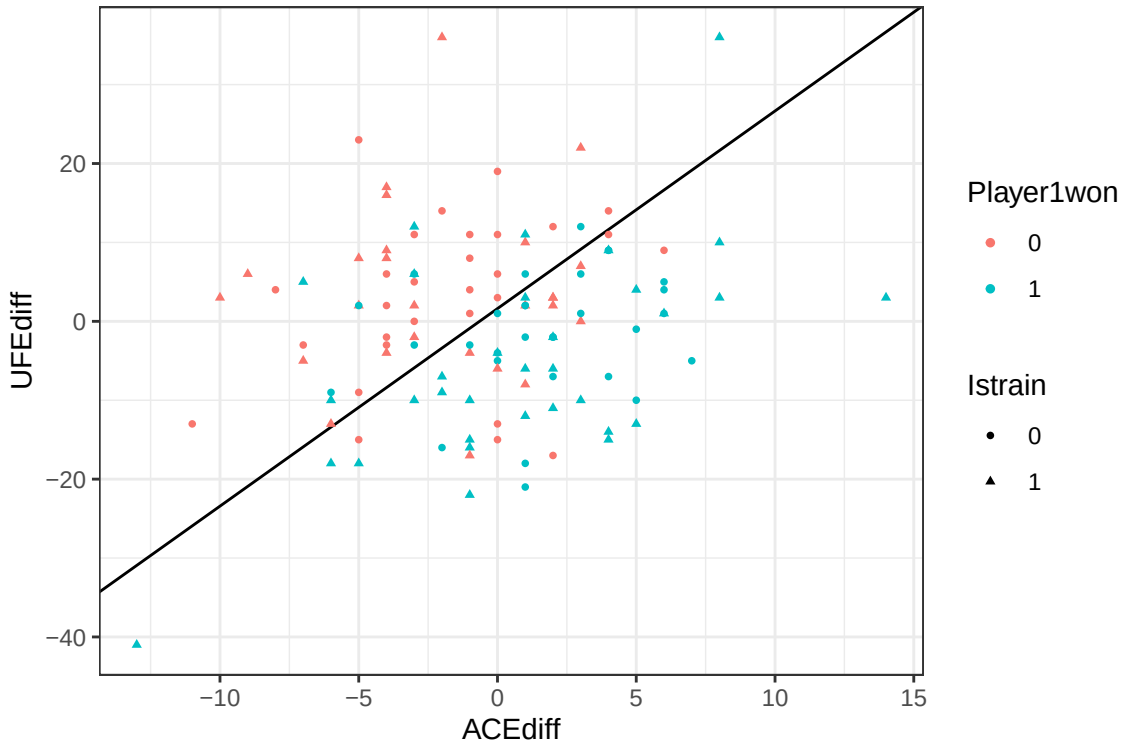
```
##            [,1]
## [1,]  0.7575758
## [2,] -5.0303030
```

```
covmat0 = cov(cbind(k0s$ACEdiff, k0s$UFEdiff))
covmat1 = cov(cbind(k1s$ACEdiff, k1s$UFEdiff))
covK = 1/(n0 + n1) * ((n0 - 1) * covmat0 + (n1 - 1) * covmat1)
c = log(pi1/pi0) + 0.5 * t(mu1) %*% solve(covK) %*% mu1 - 0.5 * t(mu0) %*% solve(covK) %*%
    mu0
lhs = solve(covK) %*% (mu0 - mu1)
a = lhs[1]
b = lhs[2]
```

Find that $a = -0.228$, $b = 0.091$ and $c = 0.147$, which gives the class boundary $x_2 = 1.616 + 2.504x_1$

Part 3

```
tennis$istrain = 1
tennis[-train, ]$istrain = 0
df2 = data.frame(tennis, Player1won = as.factor(tennis$Result), Istrain = as.factor(tennis$istrain))
ggplot(df2, aes(x = ACEdiff, y = UFEdiff, color = Player1won, shape = Istrain)) +
    geom_abline(intercept = c/b, slope = -a/b) + geom_point(size = 1) + theme_bw()
```



f)

```
lda.fit = lda(Result ~ ACEdiff + UFEdiff, data = tennisTrain)
lda.fit.p = predict(lda.fit, tennisTest)$class
confMat = table(lda.fit.p, tennisTest$Result)
confMat
```

```
##
## lda.fit.p  0  1
##         0 20  5
##         1  9 25
```

The sensitivity is 0.735 and the specificity is 0.8

g)

```
qda.fit = qda(Result ~ ACEdiff + UFEdiff, data = tennisTrain)
qda.fit.p = predict(qda.fit, tennisTest)$class
confMat = table(qda.fit.p, tennisTest$Result)
confMat
```

```
##
## qda.fit.p  0  1
##         0 20  6
##         1  9 24
```

The sensitivity is 0.727 and the specificity is 0.769

h)

Looking at the confusion matrices, QDA has the lowest sensitivity and the specificity is lower than for LDA, so QDA should not be used. Looking at the plot of the QDA decision boundary we see that to the left in the plot around `UFEdiff`=0 and below QDA classifies as win while glm and LDA would classify as loss, this may be why QDA is worse than the other two. glm has the highest sensitivity and the lowest specificity while LDA has better sensitivity than QDA and the best spesificity. Summing up the two values glm has the highest value, so that is what should be used.

Problem 4

a)

Given a set of values for K, 10-fold cross validation is performed by first randomly dividing the data into a training set and a testing set, the testing set is not used until the very end. The training dataset is then randomly divided into 10 more or less equal parts, C_1, C_2, ..., C_{10}. C_k denotes the indices of the observations in part k. 9 parts are used for training the model and 1 is used for testing the model. This is done 10 times with a new set used as test set each time. The error is then calculated using the loss function in Equation 23.

$$\text{CV}_{10} = \sum_{k=1}^{10} \frac{n_k}{10} \text{Err}_k \tag{23}$$

where n_k is the number of observations in part k. The error for part k is

$$\text{Err}_k = \sum_{i \in C_k} \frac{\text{I}(y_i \neq \hat{y}_i)}{n_k} \tag{24}$$

where I is the indicator function defined as

$$\text{I}(a \neq \hat{a}) = \begin{cases} 1 \text{ if } a \neq \hat{a} \\ 0 \text{ else} \end{cases} \tag{25}$$

This is done for each value of K we want to consider. This will result in a plot of CV_{10} against K. Based on this plot the best model can be selected. The best model will typically be the one with the lowest CV_{10}. The model is then fit using the whole training dataset, and tested using the test set which has not been used yet.

b)

(i) TRUE
(ii) TRUE
(iii) FALSE
(iv) FALSE

c)

```
id <- "1I6dk1fA4ujBjZPo3Xj8pIfnzIa94WKcy"  # google file ID
d.chd <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download", id))
fit.4c = glm(chd ~ sbp + sex, data = d.chd, family = "binomial")
summary(fit.4c)
```

```
##
## Call:
## glm(formula = chd ~ sbp + sex, family = "binomial", data = d.chd)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0647  -0.8697  -0.7749   1.4191   1.7794
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.386252   0.790657  -3.018  0.00254 **
## sbp          0.011337   0.006273   1.807  0.07075 .
## sex          0.322764   0.235786   1.369  0.17103
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 427.61  on 349  degrees of freedom
## Residual deviance: 422.63  on 347  degrees of freedom
## AIC: 428.63
##
## Number of Fisher Scoring iterations: 4
```

```
eta <- summary(fit.4c)$coef[, 1] %*% c(1, 140, 1)
pchd = exp(eta)/(1 + exp(eta))
```

The probability of a male with $sbp = 140$ having coronary heart disease is 0.383.

d)

Will now perform bootstrapping with $B = 1000$. For each iteration the model is fit with the resampled data and is used to create a bootstrap estimate of the probability for `chd`, given `sbp`=140 and `sex`=male, p. The bootstrap estimates, $\hat{p}_i$, are stored in the `estimator` variable.

```r
B = 1000
n = dim(d.chd)[1]
estimator = rep(NA, B)
for (b in 1:B) {
    newind = sample(x = c(1:n), size = n, replace = TRUE)
    newsample = d.chd[newind, 1:3]
    fit.4d = glm(chd ~ sbp + sex, data = newsample, family = "binomial")
    eterm <- summary(fit.4d)$coef[, 1] %*% c(1, 140, 1)
    estimator[b] = exp(eterm)/(1 + exp(eterm))
}
Mxb = mean(estimator)
SE = sqrt(1/(B - 1) * sum((estimator - Mxb)^2))
confinterval = quantile(estimator, probs = c(2.5, 97.5)/100)
```

Now that there are 1000 different values for the probability the standard error can be calculated. The mean is simply

$$\bar{\hat{p}} = \frac{1}{B}\sum_{i=1}^{B}\hat{p}_i \tag{26}$$

The standard error is then

$$\mathrm{SE}_B(\hat{p}) = \sqrt{\frac{1}{B-1}\sum_{i=1}^{B}(\hat{p}_i - \bar{\hat{p}})^2} = 0.0478892 \tag{27}$$

The standard error is 0.0479. The confidence interval for $\hat{p}$ is (0.29, 0.476)