Compulsory exercise 1: Group 24

TMA4268 Statistical Learning V2019

Silje Anfindsen and Clara Panchaud

17 February, 2020

Problem 1

a)

The expected test mean squared error (MSE) at x_0 is

$$E[(y_0 - \hat{f}(x_0))^2]$$

for $y_0 = f(x_0) + \epsilon$ where f is the true regression function and ϵ the error term.

b)

Now we can derive this expression into three terms.

$$\begin{aligned} E[(y_0 - \hat{f}(x_0))^2] = & \quad \text{definition of y0} \\ E[(f(x_0) + \epsilon + \hat{f}(x_0))^2] = & \quad \text{by linearity of the expectation} \\ E[f(x_0)^2] + E[\epsilon^2] + E[\hat{f}(x_0)^2] - 2E[f(x_0)\hat{f}(x_0)] + 0 = & \quad \text{using the definition of variance} \\ f(x_0)^2 + Var(\epsilon) + Var(\hat{f}(x_0)) + E[\hat{f}(x_0)]^2 - 2f(x_0)E[\hat{f}(x_0)] = & \\ \underbrace{\text{Var}(\varepsilon)}_{\text{Irreducible error}} + \underbrace{\text{Var}(\hat{f}(x_0))}_{\text{Variance of prediction}} + \underbrace{\Big(f(x_0) - \text{E}[\hat{f}(x_0)]\Big)^2}_{\text{Squared bias}} & \end{aligned}$$

c)

We can interpret the three terms as follows:

- Irreducible error, $Var(\epsilon)$ is a measure of the amount of noise in our data which cannot be reduced by creating a better model.
- Variance of prediction, $Var(\hat{f}(x_0))$ is the variability or spread of the model prediction for a given data point x_0. The estimate at this data point will varry when the training data, and therefore $\hat{f}(x_0)$ changes.
- Squared bias, $\Big(f(x_0) - \text{E}[\hat{f}(x_0)]\Big)^2$ is the difference between the expected value for the prediction of our model $E[\hat{f}(x_0)]$ and the true value $f(x_0)$ in a given data point x_0.

d)

True, False, False, True

e)

True, False, True, False

f)

(ii)

g)

C

Problem 2

In this problem we will work with the earthworm dataset. Let's start by loading it and having a first look.

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

a)

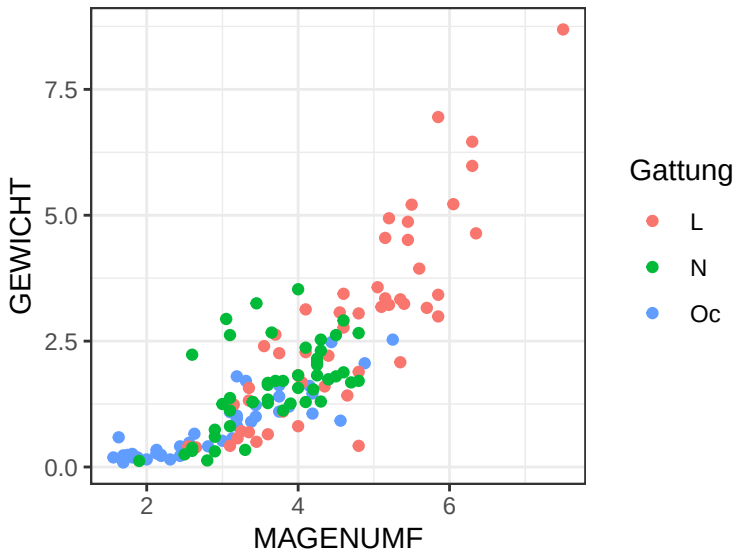
There are 143 observations of 5 variables, so 143 rows and 5 columns.

The qualitative variables are $Gattung$,$FANGDATUM$ and $Nummer$. We see that the first two are indeed encoded as factors, and $Nummer$ is just the worm-specific ID so it does not have a relevant numerical value. The two remaining ones, $GEWICHT$ and $MAGENUMF$ are quantitative.

b)

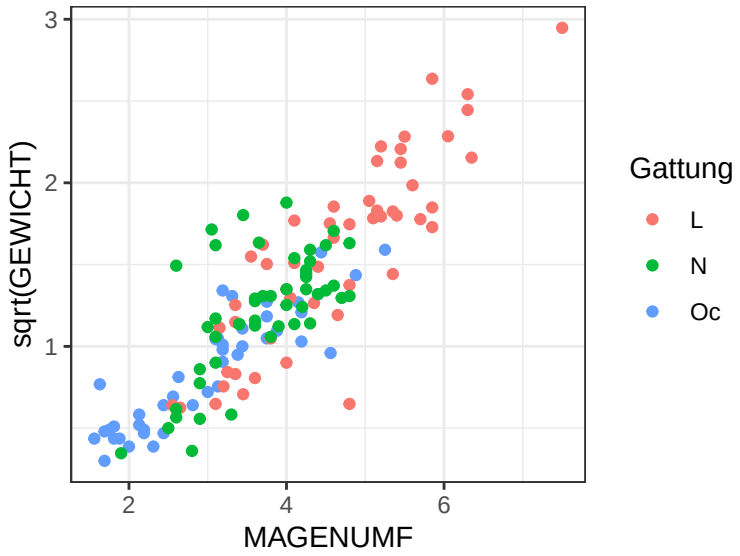
We want to make a scatterplot of the weight against the circumfence of stomach for each of the three species. Since we want to fit a linear regression, we would like to find a linear relationships between the weight and the circumference.

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



The relationship does not look linear, it seems like it has a quadratic shape suggesting $GEWICHT = MAGENUMF^2$. We therefore decided to try again by plotting $MAGENUMF$ versus $\sqrt{GEWICHT}$.

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



It now seems like the relationship is linear.

c)

We will now fit a regression model that predicts the weight given the circumference of the stomach and the species. We will use the transformed version of the variable $GEWICHT$ found in b).

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

```
##
## Call:
## lm(formula = sqrt(GEWICHT) ~ MAGENUMF + Gattung, data = d.worm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.99731 -0.17453  0.00161  0.13317  0.74534
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.234647   0.115510  -2.031   0.0441 *
## MAGENUMF     0.391673   0.023620  16.582   <2e-16 ***
## GattungN     0.009343   0.057419   0.163   0.8710
## GattungOc   -0.082327   0.066712  -1.234   0.2193
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2639 on 139 degrees of freedom
## Multiple R-squared:  0.7632, Adjusted R-squared:  0.7581
## F-statistic: 149.3 on 3 and 139 DF,  p-value: < 2.2e-16
```

We find three different regression models, one for each species. They are specified as follows:

$$Lumbricus: \sqrt{GEWICHT} = -0.235 + 0.392 * MAGENUMF$$
$$Nicodrilus: \sqrt{GEWICHT} = -0.235 + 0.009 + 0.392 * MAGENUMF = -0.226 + 0.392 * MAGENUMF$$
$$Octolasion: \sqrt{GEWICHT} = -0.235 - 0.082 + 0.392 * MAGENUMF = -0.317 + 0.392 * MAGENUMF$$

Since *Gattung* is a predictor with three levels, we need to test for H0: $\beta_2 = \beta_3 = 0$ in order to determine if it is a relevant predictor. This requires an F-test and is done by the anova function below.

```
anova(model)
```

```
## Analysis of Variance Table
##
## Response: sqrt(GEWICHT)
##            Df  Sum Sq Mean Sq  F value Pr(>F)
## MAGENUMF    1 31.0031 31.0031 445.2735 <2e-16 ***
## Gattung     2  0.1879  0.0939   1.3491 0.2629
## Residuals 139  9.6782  0.0696
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This would suggest that *Gattung* is not a relevant predictor since we have a pretty high p-value.

d)

We will now test if including an interaction between the species and MAGENUMF would be relevant to predict the weight of a worm. We therefore fit a model including the interatction term.

```
model_2<-lm(sqrt(GEWICHT)~MAGENUMF*Gattung,data=d.worm)
summary(model_2)
```

```
##
## Call:
## lm(formula = sqrt(GEWICHT) ~ MAGENUMF * Gattung, data = d.worm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.00772 -0.15569 -0.00232  0.11959  0.71671
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -0.52408    0.15915  -3.293  0.00126 **
## MAGENUMF             0.45414    0.03344  13.580  < 2e-16 ***
## GattungN             0.46682    0.25476   1.832  0.06906 .
## GattungOc            0.40417    0.20458   1.976  0.05021 .
## MAGENUMF:GattungN   -0.10818    0.06281  -1.722  0.08725 .
## MAGENUMF:GattungOc -0.12789    0.05260  -2.431  0.01634 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2594 on 137 degrees of freedom
## Multiple R-squared:  0.7744, Adjusted R-squared:  0.7662
## F-statistic: 94.07 on 5 and 137 DF,  p-value: < 2.2e-16
```

We test similarly as in c).

```
anova(model_2)
```

```
## Analysis of Variance Table
##
## Response: sqrt(GEWICHT)
##                    Df  Sum Sq Mean Sq  F value  Pr(>F)
## MAGENUMF            1 31.0031 31.0031 460.7282 < 2e-16 ***
## Gattung             2  0.1879  0.0939   1.3959 0.25111
## MAGENUMF:Gattung    2  0.4592  0.2296   3.4122 0.03579 *
## Residuals         137  9.2189  0.0673
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

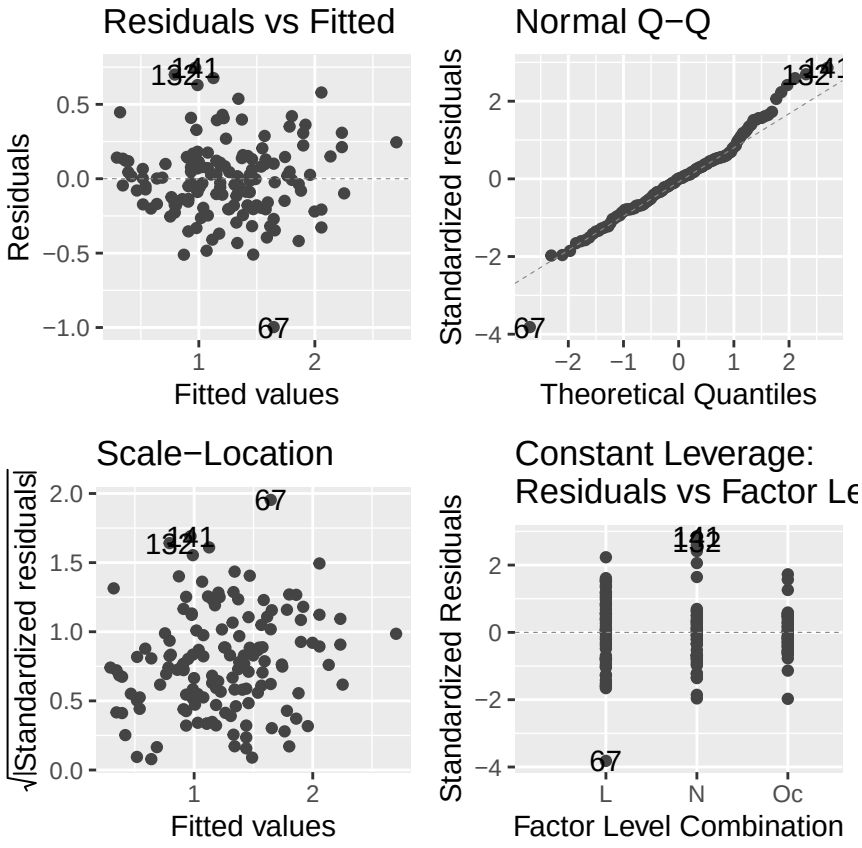
The significance level can usually be decided, but we will take the usual 0.05 level. Sine this p value is less than the usual significance level, we can conclude that there is no evidence that the interaction term is irrelevant and we would keep the interaction in our model.

e)

We will now carry out a residual analysis for the model without interaction term. Recall that $\epsilon_i = y_i - \hat{y}_i \sim N(0, \sigma^2)$. We want to verify the assumptions of the linear model, which are:

1. The expected value of ϵ_i is 0: $E(\epsilon_i) = 0$.
2. All ϵ_i have the same variance: $Var(\epsilon_i) = \sigma^2$.
3. The ϵ_i are normally distributed.
4. The ϵ_i are independent of each other.

```
library(ggfortify)
autoplot(model,smooth.colour = NA)
```

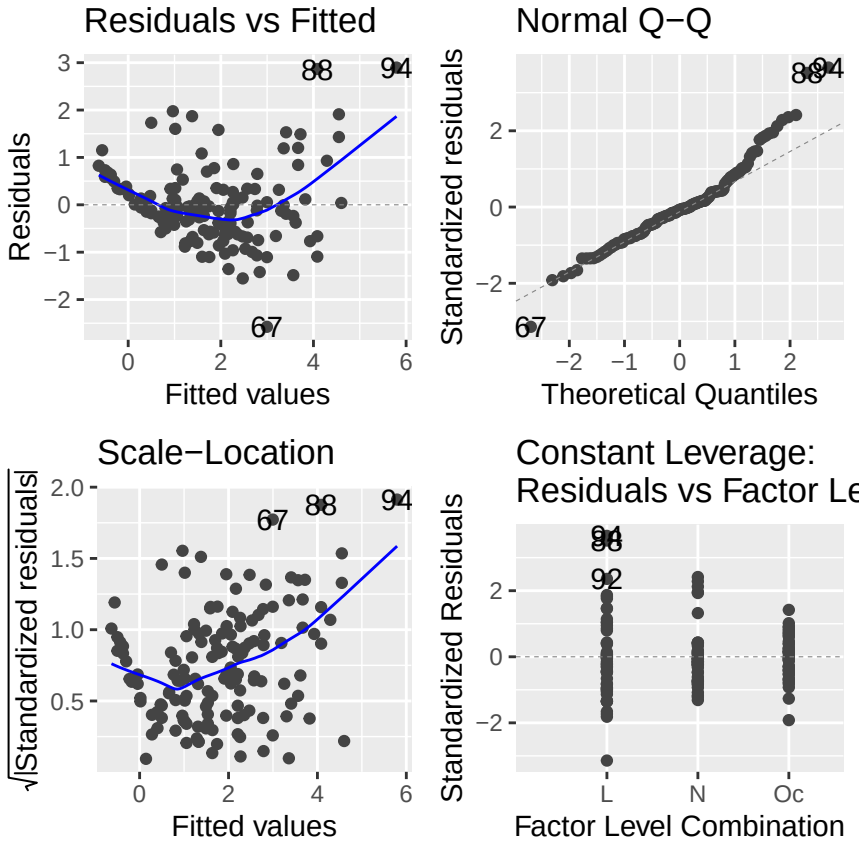


Looking at the Residual vs. Fitted plot there are no presence of a pattern between the residuals $e_i = y_i - \hat{y}_i$ and the predicted values $\hat{y}_i$. This helps us check assumptions 1 and 2. We don´t see a particular pattern so it seems fine. We don´t see any trend in the Scale-Location plot either, reinforcing that the homoscedasticity assumption should be correct. The QQ-plot does not show any evidence against assumption 3. The leverage plot is not so relevant here since we are looking at a variable encoded as a factor. The observation 07 seems to be an outlier, it would be good to pay attention to it in the future.

In conclusion, according to those plots we have no evidence against the assumptions of the linear model so it seems like a good choice.

We will make a residual plot for the model without any variable transformations.

```
library(ggfortify)
simple_model<-lm(GEWICHT~MAGENUMF+Gattung, data=d.worm)
autoplot(simple_model)
```



The difference is mostly in the Residuals vs. Fitted and in the Scale-Location plot. We can see some patterns in both so we decided to add a smoothing line that makes it even clearer. This would suggest that at least the homoscedasticity assumption is violated and a linear model on this data is not appropriate.

f)

Residual plots are a usefull tool to identify if the assumptions of a linear regression are fulfilled. If they are not, a linear model will not be appropriate and any predictions or inference made from it will be misleading and untrustworthy. The residual plots are useful when we have many covariates because then it is harder to plot them all against each other and identify patterns like we did in this problem.

If the residual plots indicates that some assumptions are violated, we can look closer at the data and try to identify non-linear transformations of the variables that can be used instead.

g)

False,False,False, True

Problem 3

a)

We will create a logistic regression model where the probability for player 1 to win has the form:

$$P(Y_i = 1|\mathbf{X} = \boldsymbol{x}_i) = p_i = \frac{e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\beta_3 x_{i3}+\beta_4 x_{i4}}}{1 + e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\beta_3 x_{i3}+\beta_4 x_{i4}}}$$

where x_{i1} is the number of aces for player 1 in match i, x_{i2} is the number of aces for player 2 in match i, and x_{i3} and x_{i4} are the number of unforced errors commited by player 1 and 2 in match i. $Y_i = 0$ represents player 1 loosing match i, $Y_i = 1$ represents player 1 winning match i.

Let $\alpha = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4}$

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

b)

Taking the exponential of the formula in a) we get the following formula for the odds:

$$\frac{p_i}{1-p_i} = \frac{P(Y_i = 1|X = x_i)}{P(Y_i = 0|X = x_i)} = e^{\beta_0} e^{\beta_1 x_{i1}} ... e^{\beta_4 x_{i4}}$$

To interpret β_1 we think about what would happen if we increased x_{i1} by 1 while the other covariates remain the same. We do this by looking at the following ratio of the odds

$$\frac{odds(Y_i|X_1 = x_{i1} + 1)}{odds(Y_i|X_1 = x_{i1})} = e^{\beta_1}$$

We see that the odds ratio is changed by a factor e^{β_1}.Therefore, if the number of aces of player 1 increases by 1, this will change his odds ratio of winning by a factor e^{β_1}. Since more aces is a good thing to have in a tennis match, we expect β_1 to take a positive value. Let's now fit the model to see.

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

We get $\beta_1 = 0.363$ which is positive as expected.

c)

Now we will reduce the number of covariates. The vector of covariates is now $\mathbf{x} = (x_1, x_2)$ where x_1 denotes the difference between aces perfored by player 1 and player 2, while x_2 denotes the difference in unforced errors made by the players.

We divide the data into a training and a test set.

```
# The new variables
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)
train = sample(c(1:n), replace = F)[1:n2]
tennisTest = tennis[-train, ]
tennisTrain = tennis[train, ]
```

Now we can fit a logistic regression model on the training set.

```
r2.tennis = glm(Result ~ ACEdiff + UFEdiff, data = tennisTrain, family = "binomial")
summary(r2.tennis)
```

```
##
## 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
```

With the new variables that we defined, the model now has the following form:

$$logit(p) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

We can solve for x_2 which gives

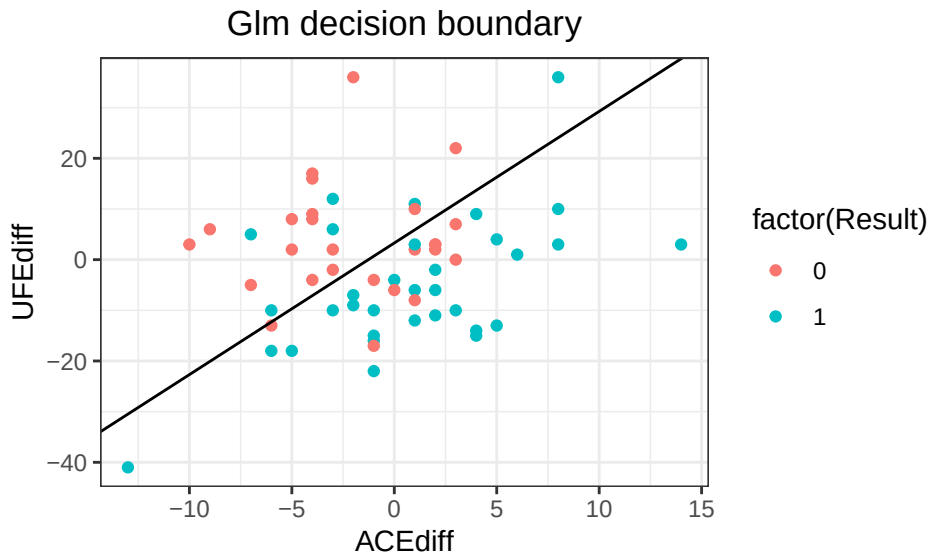
$$x_2 = \frac{logit(p) - \beta_0 - \beta_1 x_1}{\beta_2} = a + b x_1$$

where $a = \frac{logit(p) - \beta_0}{\beta_2}$ and $b = \frac{-\beta_1}{\beta_2}$. Since we want to use a 0.5 cutoff as decision rule, we replace p by 0.5 in those formulas to find the a and b that define our class boundary.

```
#define logit
logit <-function(x) {return(log(x/(1-x)))}
#find a and b
a = (logit(0.5)-r2.tennis$coefficients[1])/(r2.tennis$coefficients[3])
b = -r2.tennis$coefficients[2]/r2.tennis$coefficients[3]
```

We plot the training observations with the boundary line.

```
ggplot(r2.tennis,aes(x=ACEdiff,y=UFEdiff ,color=factor(Result))) + geom_point() +
  theme_bw() + geom_abline(slope=b, intercept=a)+ ggtitle("Glm decision boundary")+theme(plot.title = el
```



It seems like the boundary line separates the two classes quite well on the training set. We can use the boundary line to predict the result of a new test set. We will now look at the confusion table to see how good the classification rule is.

```
#make a vector of the result of the matches consisting of Y=0 or Y=1
probs = predict(r2.tennis, tennisTest, type = "response")
for (i in 1:length(probs)){
  if (probs[i]<0.5){
    probs[i]=0
  }
  else{probs[i]=1}
}

#confusion table of
t<-table(tennisTest$Result, probs)
colnames(t)=c("Predicted Loss","Predicted Win")
rownames(t)=c("Actual Loss","Actual Win")
t
```

```
##              probs
##               Predicted Loss Predicted Win
##   Actual Loss             22             7
##   Actual Win               6            24
```

The confusion table shows the performance of the classification rule, logisitc regression. The sum of the diagonal is the total number of correct classifications. The sum of all elements off the diagonal is the total number of misclassifications. We notice that there are some misclassifications for both classes. We will now calculate the sensitivity and specificity in order to understand this table better.

$$Sensitivity = \frac{True\ Positive}{Actual\ Positive}$$

$$Specificity = \frac{True\ Negative}{Actual\ Negative}$$

```r
Sensitivity<-(t[2,2]/(t[2,2]+t[2,1]))
Specificity<-(t[1,1]/(t[1,1]+t[1,2]))
print(c("Sensitivity: ",Sensitivity,"Specificity: ", Specificity))
```

```
## [1] "Sensitivity: "     "0.8"               "Specificity: "
## [4] "0.758620689655172"
```

Sensitivity is the proportion of correctly classified positive observations, while specificity is the proportion of correctly classified negative observations. Since a high sensitivity and specificity yields a good classification rule, we are happy with the obtained values above.

d)

First of all, $\pi_k = P(Y = k)$, k=0,1, are the prior probabilities. Recall that $Y = 0$ denotes player 1 loosing the match. For $k = 0$, the prior probability for $Y = 0$ is estimated by $\pi_0 = \frac{\#loss}{\#observations}$ while for k=1 we have $\pi_1 = \frac{\#victories}{\#observations}$.

The function $f_k(\mathbf{x}) = P(\mathbf{X} = \mathbf{x}|Y = k)$ is the probability to obtain a certain $\mathbf{x}$, given the winning player. It follows a multivariate normal distribution with a class specific mean $\mu_\mathbf{k}$ and a common covariance matrix $\mathbf{\Sigma}$ that we will discuss now.

The mean $\mu_\mathbf{k} = (\mu_{k1}, \mu_{k2})$ for k=0,1, is estimated as the average of all training observations (x_1, x_2) from the kth class. For example μ_{01} is calculated by averaging the differences in aces performed by the players, over the games where player 2 won.

On the other hand the covariance matrix is common to both classes. In order to do this the covariance matrix is estimated for each class by a weighted average of the sample variances. Then a pooled version is calculated by combining the two class specific covariance matrices.

e)

We will now look at the decision boundary for LDA between the two classes. Start with inserting the expression for each class distribution into Bayes formula to obtain the posterior probability $p_k(x) = P(\mathbf{Y} = k|\mathbf{X} = x)$.

$$
\begin{aligned}
P(\mathbf{Y} = 0|\mathbf{X} = x) &= P(\mathbf{Y} = 1|\mathbf{X} = x) \\
\frac{\pi_0 f_0(x}{\sum_{i=1}^{2} \pi_i f_i(x)} &= \frac{\pi_1 f_1(x}{\sum_{i=1}^{2} \pi_i f_i(x)} \\
\pi_0 f_0(x) &= \pi_1 f_1(x) \\
\pi_0 e^{-\frac{1}{2}(x-\mu_0)^T \Sigma (x-\mu_0)} &= \pi_1 e^{-\frac{1}{2}(x-\mu_1)^T \Sigma (x-\mu_1)} \\
log(\pi_0) - \frac{1}{2}(x-\mu_0)^T \Sigma (x-\mu_0)) &= log(\pi_1) - \frac{1}{2}(x-\mu_1)^T \Sigma (x-\mu_1)) \\
log(\pi_0) - \frac{1}{2}\Big(x^T\Sigma^{-1}x - \mu_0^T\Sigma^{-1}x + \mu_0^T\Sigma^{-1}\mu_0 - x^T\Sigma^{-1}\mu_0\Big) &= log(\pi_1) - \frac{1}{2}\Big(x^T\Sigma^{-1}x - \mu_1^T\Sigma^{-1}x + \mu_1^T\Sigma^{-1}\mu_1 - x^T\Sigma^{-1}\mu_1\Big) \\
log(\pi_0) + x^T\Sigma^{-1}\mu_0 - \frac{1}{2}\mu_0\Sigma^{-1}\mu_0 &= log(\pi_1) + x^T\Sigma^{-1}\mu_1 - \frac{1}{2}\mu_1\Sigma^{-1}\mu_1 \\
\delta_0(x) &= \delta_1(x)
\end{aligned}
$$

The calculations between the second and third last line removes some terms. Here we use that Σ is a positive semidefinitt matrix: $\Sigma^{-1} = (\Sigma^{-1})^T = (\Sigma^T)^{-1} = \Sigma^{-1}$. It follows that $\Big(\mu_0^T\Sigma^{-1}x\Big)^T = x^T\Sigma^{-1}\mu_0$.

We have now found the discriminant score $\delta_k(x)$. We will assign an observation to class k when $\delta_k(x)$ is largest. In other words, we use the rule to classify to class 1 for an observation with covariates x if $\hat{P}(\mathbf{Y} = 1 \mid \mathbf{x}) > 0.5$. We will now find the formula for the boundary between the classes $ax_1 + bx_2 + c = 0$.

$$\delta_0(x) = \delta_1(x)$$
$$log(\pi_0) + x^T\Sigma^{-1}\mu_0 - \frac{1}{2}\mu_0\Sigma^{-1}\mu_0 = log(\pi_1) + x^T\Sigma^{-1}\mu_1 - \frac{1}{2}\mu_1\Sigma^{-1}\mu_1$$

We do the matrix multiplication of the terms above in order to find an expression for a, b and c. Let $x = [x_1, x_2]^T$, $\mu_k = [\mu_{k1}, \mu_{k2}]$, and

$$\Sigma^{-1} = \begin{bmatrix} \sigma_{11} & \sigma_{21} \\ \sigma_{21} & \sigma_{22} \end{bmatrix}$$

We are now ready to define the constants a,b and c.

$$a = \sigma_{11}(\mu_{01} - \mu_{11}) + \sigma_{12}(\mu_{02} - \mu_{12})$$
$$b = \sigma_{21}(\mu_{01} - \mu_{11}) + \sigma_{22}(\mu_{02} - \mu_{12})$$
$$c = -\frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0 + \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + log\Big(\frac{\pi_0}{\pi_1}\Big)$$

We rearrange the forumla for the boundary in order to find a linear form.

$$x_2 = -\frac{ax_1 + c}{b}$$

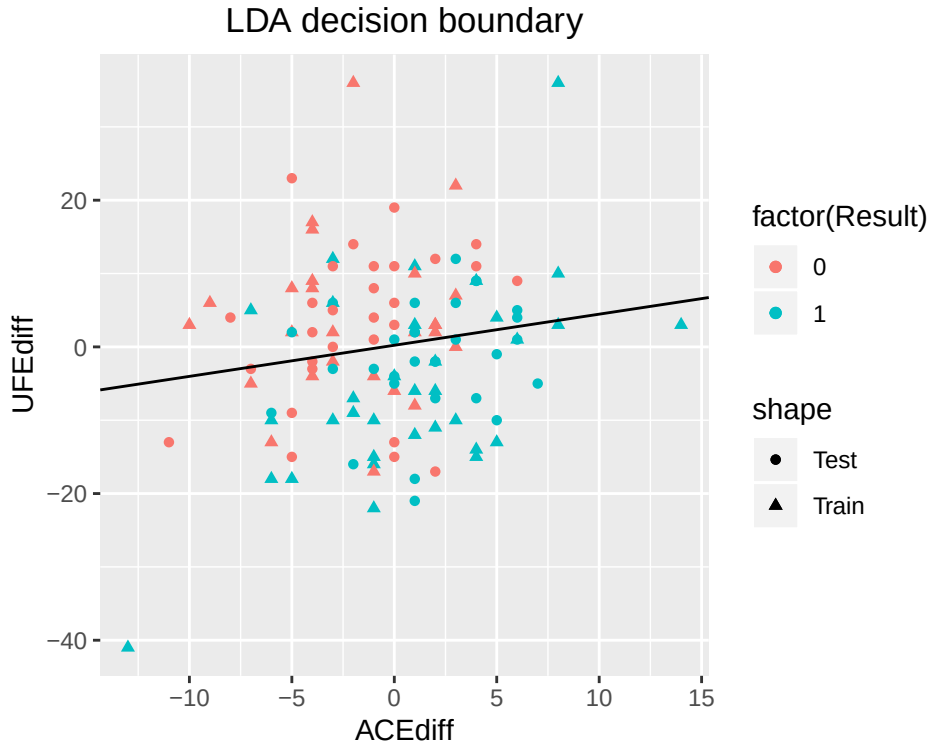
We will now use R for the calculations of a,b and c in order to plot the boundary line.

```
##Start by estimating mu,pi and sigma
##Recall Y=0 means player 1 loosing the match
tennisTrain_loss <- tennisTrain[tennisTrain$Result==0,] #player 1 loose
tennisTrain_win <- tennisTrain[tennisTrain$Result==1,] #player 1 win

## mu_k is the mean vector consisting of ACEdiff and UFEdiff for Y=0 and Y=1
mu_0 <- c(mean(tennisTrain_loss$ACEdiff), mean(tennisTrain_win$UFEdiff)) #player 1 loose
mu_1 <- c(mean(tennisTrain_win$ACEdiff), mean(tennisTrain_loss$UFEdiff)) #player 1 win

##pi_k is the lost/won matches over total number of matches
pi_0 <- nrow(tennisTrain_loss)/nrow(tennisTrain)
pi_1 <- nrow(tennisTrain_win)/nrow(tennisTrain)

## First find covariance matrices for the each class
sigma_0 <- cov(tennisTrain_loss[,c("ACEdiff", "UFEdiff")])
sigma_1 <- cov(tennisTrain_win[,c("ACEdiff", "UFEdiff")])

## pooled version of covariance matrices
sigma <- (((nrow(tennisTrain_loss) - 1)) *
            sigma_0 + (nrow(tennisTrain_win) - 1) * sigma_1) / (nrow(tennis) - 2)

## inverse of the covariance matrix
sigma.inv <- solve(sigma)

##We can now solve for a,b and c
```

```
a = sigma.inv[1,1]*(mu_0[1]-mu_1[1])+sigma.inv[1,2]*(mu_0[2]-mu_1[2])
b = sigma.inv[2,1]*(mu_0[1]-mu_1[1])+sigma.inv[2,2]*(mu_0[2]-mu_1[2])
c = -0.5*t(mu_0)%*%sigma.inv%*%mu_0+0.5*t(mu_1)%*%sigma.inv%*%mu_1 + log(pi_0/pi_1)
```

We will now make a plot with both train and test set, as well as the boundary line for LDA.

```
ggplot(r2.tennis,aes(x=ACEdiff,y=UFEdiff ,color=factor(Result))) +
geom_point(data = tennisTrain, aes(shape="Train")) + geom_point(data=tennisTest,aes(shape="Test")) +
  geom_abline(slope=a/c, intercept=b/c)+ggtitle("LDA decision boundary")+
  theme(plot.title = element_text(hjust = 0.5))
```



The boundary line manages to seperate the biggest amount of observations from the two classes, but there are some observations from both the train and test set that lay on the wrong side of the boundary line.

f)

For this task we perform LDA on the training data.

```
LDA=lda(Result~ACEdiff+UFEdiff,data=tennisTrain)
```

We are now ready to classify the results of the test set. We make a confusion table to look at our results.

```
pred <- predict(LDA, tennisTest)$class ##list of results for the matches
table2<-table(tennisTest$Result,pred) ##confusion table
colnames(table2)=c("Predicted Loss","Predicted Win")
rownames(table2)=c("Actual Loss","Actual Win")
table2
```

```
##              pred
##               Predicted Loss Predicted Win
##   Actual Loss             20             9
##   Actual Win               5            25
```

The confusion tables shows that most of the predictions are correct, with a misclassification of 9 and 5, there are a remarkeble chance for the prediction to be wrong.

```
##calculate specificity and sensitivity
Sensitivity<-(table2[2,2]/(table2[2,2]+table2[2,1]))
Specificity<-(table2[1,1]/(table2[1,1]+table2[1,2]))
print(c("Sensitivity: ", Sensitivity,"Specificity: ",Specificity))
```

```
## [1] "Sensitivity: "     "0.833333333333333" "Specificity: "
## [4] "0.689655172413793"
```

The values are pretty high which is good.

g)

We will now use QDA on the training set and make the confusion table for the test set. The difference between LDA and QDA is that in LDA we had a common covariance matrix while in QDA the covariance matrix is specific to each class. Therefore in our model we have two different covariance matrices.

```
QDA=qda(Result~ACEdiff+UFEdiff,data=tennisTrain)
pred <- predict(QDA, tennisTest)$class ##result list for y-values
table3<-table(tennisTest$Result,pred)##confusion table
colnames(table3)=c("Predicted Loss","Predicted Win")
rownames(table3)=c("Actual Loss","Actual Win")
table3
```

```
##              pred
##               Predicted Loss Predicted Win
##   Actual Loss             20             9
##   Actual Win               6            24
```

Again the table for QDA is quite similar as the table for LDA. This can indicate that a quadratic fit may not make it better.

```
Sensitivity<-(table3[2,2]/(table3[2,2]+table3[2,1]))
Specificity<-(table3[1,1]/(table3[1,1]+table3[1,2]))
print(c("Sensitivity: ", Sensitivity,"Specificity: ",Specificity))
```

```
## [1] "Sensitivity: "     "0.8"               "Specificity: "
## [4] "0.689655172413793"
```

The sensitivity is exactly the same as in LDA but the specificity is now a bit lower.

h)

We will now compare the plots for the different classification approaches we have discussed: logistic regression, LDA and QDA. We know that QDA assumes a quadratic decision boundary which appears clearly on the plot. The two other decision boundaries are linear and they also seem to separate the two classes quite well.

Recall that usually QDA is recommended when we have a huge amount of data for the training set so that the variance of the classifier is not a major concern, or if the assumption of common covariance matrix is wrong. In this case it looks like QDA is overfitting so we would not choose this method. In LDA we assumed that the observations are drawn from a Gaussian distribution with common covariance matrix in each class. Maybe it would be useful to investigate this assumption more in order to pick between LDA and glm. The confusion matrices confirm this since they are very similar for both LDA and glm. If we compare the tables of QDA and LDA we see that QDA makes one more mistake than LDA which is not much but it agrees with our conclusion that QDA overfits the data.

Problem 4

a)

The set of possible values for K is $1, 2, 3, ..., N_0$.

To perform 10-fold cross validation, we split the training data into 10 (more or less) equal parts. We fit the model on 9 of the parts and validate it on the remaining one. This is done 10 times, where each part take the role of the validation part once.

Each time we fit the model, we calculate the MSE on the validation set as we had explained in Problem 1 b). This yields $(MSE_1, MSE_2, ..., MSE_{10})$. Then we can find the validation error using the following formula:

$$\mathrm{CV}_{10} = \frac{1}{10} \sum_{i=1}^{10} \mathrm{MSE}_i \ .$$

b)

True, True, False, False

c)

We will now look at a bootstrap example. We want to fit a model that predicts the probability for coronary heart disease (chd) from systolic blood pressure (sbp) and sex (0=female, 1=male). We load the data and perform logistic regression with chd as outcome and sbp and sex as covariates.

```
id <- "1I6dk1fA4ujBjZPo3Xj8pIfnzIa94WKcy"  # google file ID
d.chd <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download",
    id))
head(d.chd)
```

```
##        sbp sex chd
## 1 132.6730   1   0
## 2 124.3902   0   0
## 3 187.5981   0   1
## 4 140.7011   0   1
```

```
## 5 129.4785   0   1
## 6 124.8995   0   0
```

```
#logistic regresseion as classification rule
model<-glm(chd ~ sbp+sex, data = d.chd, family = "binomial")
summary(model)
```

```
##
## 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
```

```
#predict the probability of chd for a male with a sbp=140
new_data = data.frame(sbp = 140,sex=1)
p = predict(model,new_data,type="response")
p
```

```
##         1
## 0.3831131
```

The estimated probability of chd for a male with a sbp=140 in the given dataset is 0.38.

d)

We now use the bootstrap to estimate the uncertainty of the probability derived in b).

```
##start by defining some variables to estimate the probability in each iteration
n=nrow(d.chd) ##number of observations in each sample
new_data=data.frame(sbp = 140,
                    sex=1)

estimated_prob = c() ##list of probabilities from each bootstrap sample
```

```
beta_0=c()
B = 1000 ##iterate through 1000 bootstrap samples
for (i in 1:B) {
   index<- sample(n, n, replace = T) ##make a bootstrap sample of the data consisting of n observations
   model<-glm(chd ~ sbp+sex, data = d.chd, subset = index,family = "binomial") ##fit a model
   estimated_prob[i]=predict(model,new_data,type="response") ##save probability for newdata
   beta_0[i]=model$coefficients[1] ##save intercept
}
```

We derive the standard error for the set of estimated probabilities.

```
standard_error<-sd(estimated_prob)
print(c("standard error:",standard_error))
```

```
## [1] "standard error:"    "0.0478892368258885"
```

Our estimation $\hat{p}$ of the probability is the mean of the estimated probabilities from the bootstrap sample. Its 95% confidence interval is found by the following formula:

$$\hat{p} \pm 1.96 \cdot SD(\hat{p})$$

```
#calculate confidence interval
c(mean( estimated_prob) - 1.96 * sd(estimated_prob),mean(estimated_prob) + 1.96 * sd(estimated_prob))
```

```
## [1] 0.2894396 0.4771654
```

A confidence interval is calculated given a set of observations in order to consider the true value for chd and if it is laying in the interval.This means that the true probability of chd for a male with sbp=140 would be found in this interval with probability 0.95. We notice that the interval is a bit wide, this means that the there is a great uncertantity when predicting the chd for a male with sbp=140.

References

James, G., D. Witten, T. Hastie, and R. Tibshirani. 2013. An Introduction to Statistical Learning with Applications in R. New York: Springer.