Compulsory exercise 1: Group 37

TMA4268 Statistical Learning V2019

Anders Bendiksen and Helge Bergo

20 February, 2020

Problem 1

a)

The mean squared error(MSE) for the function $\hat{f}(x_i)$ is

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

This gives the MSE at x_0

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

b)

The expected MSE from above is decomposed to the terms irreducible error, the variance of prediction and the squared bias.

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

c)

Irreducible error: The error that cannot be reduced, regardless how well our statistical model fits the given data.

Variance of the prediction: The amount $\hat{f}(x_0)$ is expected to change for different sets of training data. Higher variance means higher uncertainty for the prediction.

Squared bias: An estimate of how much the prediction differs from the true mean. A lower bias gives a prediction closer to the true value.

d)

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

e)

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

f)

(ii) 0.17

g)

Plot D, since it is a countour plot with $\sigma_x = 1$, $\sigma_y = 2$ and $\rho = 0.1$.

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) not shown here
str(d.worm)
```

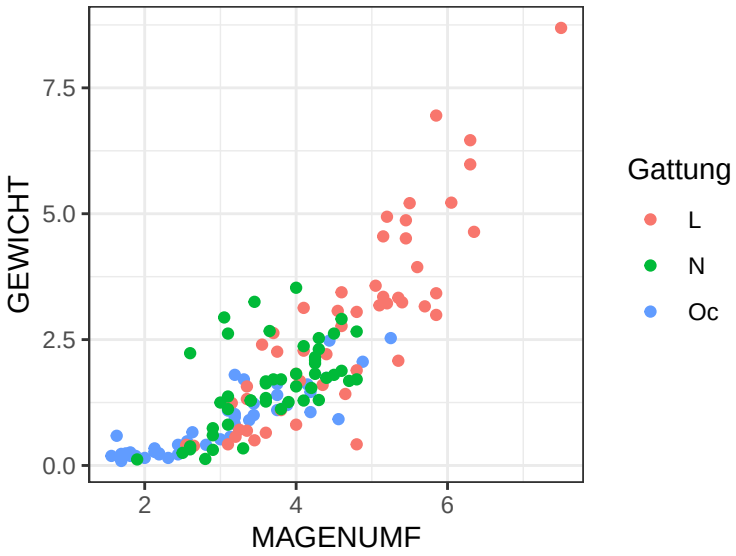
```
## 'data.frame':    143 obs. of  5 variables:
##  $ Gattung  : Factor w/ 3 levels "L","N","Oc": 3 3 3 3 3 3 3 3 3 3 ...
##  $ Nummer   : int  32 34 48 55 41 24 39 35 45 27 ...
##  $ GEWICHT  : num  0.19 0.59 0.09 0.23 0.24 0.19 0.26 0.19 0.15 0.34 ...
##  $ FANGDATUM: Factor w/ 3 levels "12.10.97","15.09.97",..: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

The dimensions are 5x143, the qualitative variables are `Gattung` and `Fangdatum`, and the quantitative variables are `Nummer`, `Gewicht` and `Magenumf`.

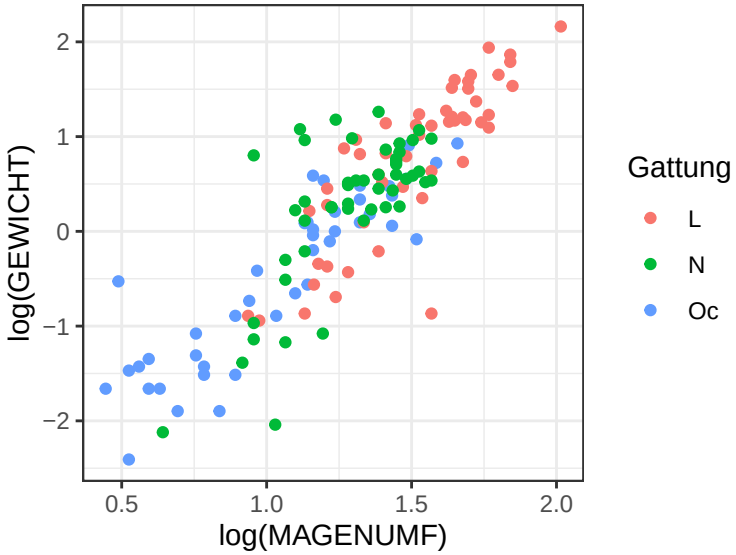
b)

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



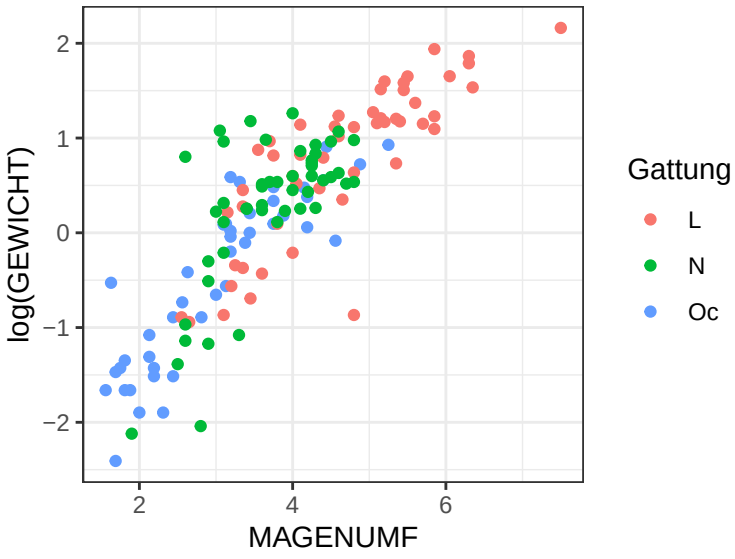
Here we see the simple linear model where `Magenumf` is plotted against `Gewicht`. It is clear that the relationship is not linear, so a logarithmic fit was tried instead.

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



We also tried to plot the relationship with only the log of `Gewicht`, and not `Magenumf`, as shown below. This seems like the best fit so far.

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



c)

Using the transformed version of the variables with log(`GEWICHT`) as output, the regression model was fitted with `Magenumf` and `Gattung` as factors.

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

```
##
## 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
```

When using `Gattung` as a factor, Gattung L becomes the main factor, and the total model become

$$y = 0.712M + 0.178 * G_1 - 0.091 * G_2 - 2.536,$$

where G_1 is species N, and G_2 is species Oc.

Separating these into three different equations, we get the following.

$$y_L = 0.712M - 2.536,$$
$$y_N = 0.712M + 0.178 * G_1 - 2.536,$$
$$y_{Oc} = 0.712M - 0.091 * G_2 - 2.536,$$

`Gattung` is not a relevant predictor, as the P values for GattungN and GattungOc are very high. This means that it is not significant for the model.

d)

To test `Gattung` as an interaction term, the model was fitted again.

```
d.worm$Gattung <- as.factor(d.worm$Gattung)
lm.fitInteraction=lm(data=d.worm, 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
```

```
anova(lm.fitInteraction)
```

```
## Analysis of Variance Table
##
## Response: log(GEWICHT)
##                   Df Sum Sq Mean Sq  F value  Pr(>F)
## MAGENUMF           1 97.802  97.802 386.5547 < 2e-16 ***
## Gattung            2  1.725   0.862   3.4084 0.03592 *
## MAGENUMF:Gattung   2  0.917   0.458   1.8112 0.16735
## Residuals        137 34.662   0.253
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From this, it can be seen that the interaction term is not that significant, and can be neglected in the model, according to the low P values.

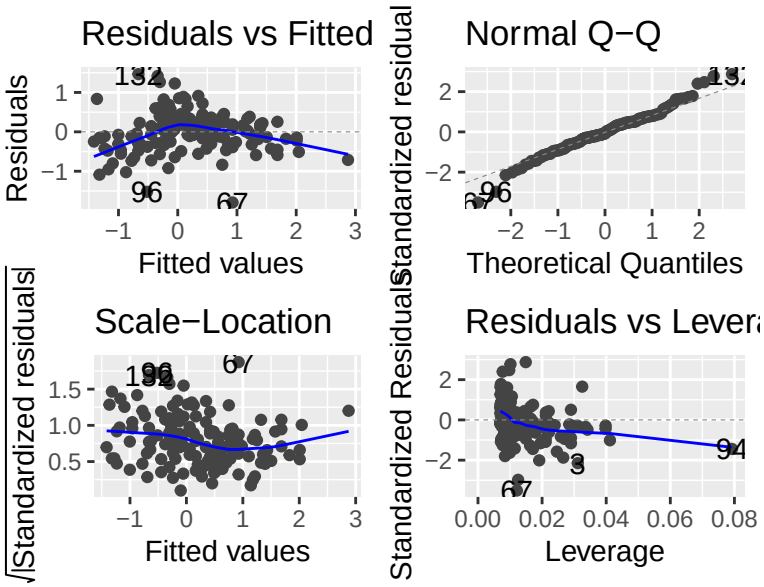
e)

The model was plotted using `autoplot()`, without the interaction term.

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

```
##
## Call:
## lm(formula = log(GEWICHT) ~ MAGENUMF, data = d.worm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.79265 -0.30299 -0.04173  0.28027  1.46536
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.54069    0.14970  -16.97   <2e-16 ***
## MAGENUMF     0.72205    0.03755   19.23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5144 on 141 degrees of freedom
## Multiple R-squared:  0.7239, Adjusted R-squared:  0.7219
## F-statistic: 369.7 on 1 and 141 DF,  p-value: < 2.2e-16
```

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




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

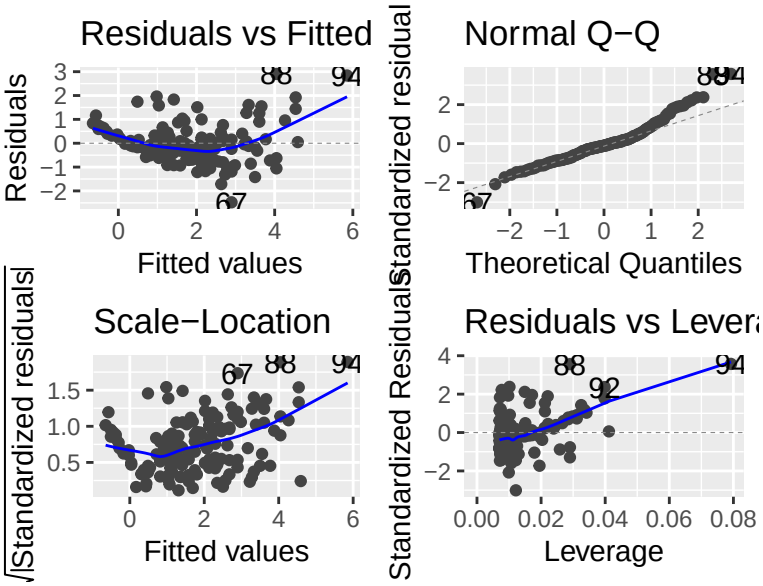
```
## Analysis of Variance Table
##
## Response: log(GEWICHT)
##            Df Sum Sq Mean Sq F value    Pr(>F)
## MAGENUMF    1 97.802  97.802  369.67 < 2.2e-16 ***
## Residuals 141 37.304   0.265
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It seems like the linearity assumptions are not met, since there is a pattern in the residual plot. The normal distribution assumption also seems to be true, as the Q-Q plot follows the center line reasonably well.

Comparing with the residual plot of the untransformed model, our model is better, but still needs some work.

```
lm.fitOriginal = lm(data = d.worm, GEWICHT ~ MAGENUMF)
autoplot(lm.fitOriginal)
```



f)

Analysing residual plots plays an important role in validating regression models. 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. If the assumptions are not fulfilled, the model could have a limited use area, or be wrong.

If assumptions are violated, applying nonlinear transformations to some of the variables may help, for example a logarithmic transformation to the dependent or independent variables.

g)

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

Problem 3

a)

A linear combination of linear combinations will result in a linear combination.And this is exactly what we are dealing with here. The expression for Pi is a linear combination of the covariates $X_{i,1}, X_{i,2}$ etc.

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

```
##         Player1          Player2       Result           ACE.1
##  S.Lisicki  : 6   K.Flipkens : 5   Min.   :0.0000   Min.   : 0.000
##  M.Bartoli  : 5   N.Li       : 5   1st Qu.:0.0000   1st Qu.: 1.000
##  A.Radwanska: 4   D.Cibulkova: 3   Median :1.0000   Median : 2.000
##  P.Kvitova  : 4   E.Makarova : 3   Mean   :0.5339   Mean   : 2.975
##  R.Vinci    : 4   F.Pennetta : 3   3rd Qu.:1.0000   3rd Qu.: 4.000
##  S.Stephens : 4   K.Kanepi   : 3   Max.   :1.0000   Max.   :14.000
##  (Other)    :91   (Other)    :96
##      UFE.1           ACE.2           UFE.2
##  Min.   : 4.00   Min.   : 0.000   Min.   : 2.00
##  1st Qu.:13.00   1st Qu.: 1.000   1st Qu.:12.00
##  Median :18.00   Median : 2.000   Median :18.00
##  Mean   :20.18   Mean   : 3.271   Mean   :20.47
##  3rd Qu.:25.75   3rd Qu.: 5.000   3rd Qu.:27.00
##  Max.   :54.00   Max.   :15.000   Max.   :55.00
##
```

b)

The estimate of the slope is positive, therefore an Ace for player one will increase his or her chance for a win.

c)

$$\hat{P(Y = 1|x)} > 0.5$$

$$\hat{P(Y = 1|x)} = \frac{e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}}}{1 + \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}} > 0.5$$

Solve the equation above with regards to x1 and x2 and insert the values of $\beta_0 = 0.216$, $\beta_1 = 0.273$ and $\beta_2 =$ -0.091. And we get $a = 2.9747$ and $b = 0.2354$, in the expression $y = ax + b$.

```
# 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, ]

logistic <- glm(Result ~ ACEdiff + UFEdiff, data=tennisTrain, family = "binomial")

summary(logistic)
```

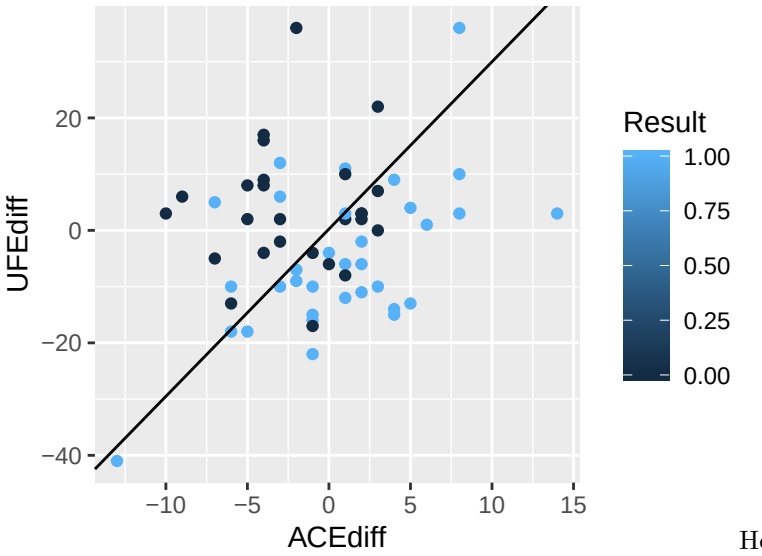
```
##
## Call:
## glm(formula = Result ~ ACEdiff + UFEdiff, family = "binomial",
##     data = tennisTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8546  -0.8968   0.4204   0.8247   1.9382
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (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
```

```
p <- predict(logistic, type="response")
round(p)
```

```
##  28  80  22 101 108 111   9   5  38  16   4  98 103  86  90  70  79  78
##   1   1   0   0   1   0   0   0   1   1   1   1   1   0   1   0   1   0
##  14  56  62 114  97  21  40  84  99  67 113  66  47  94  93  48   3  41
##   1   1   1   1   1   1   0   0   1   0   1   1   0   1   1   1   1   1
##  72  32  42  43   2  54  49  92  51   6  76  82  57   8  26  17  63  69
##   1   1   0   1   0   1   0   1   0   0   1   1   1   1   1   1   0   0
## 117  81  88 116  35
##   0   0   1   1   1
```

In order to plot the results, I now use ggplot to plot the class boundary line against the set.

```
ggplot(tennisTrain, aes(y=UFEdiff, x=ACEdiff, color=Result))+geom_point() + geom_abline(slope=2.9747, in
```



Here one can see the class boundary plottet against the training set.

```
p <- predict(logistic, newdata = tennisTest, type="response")
r <- round(as.numeric(p))
r
```

```
##  [1] 0 0 1 1 0 1 0 0 1 1 0 0 1 1 1 1 0 0 0 1 0 1 0 1 1 1 0 1 0 1 1 1 1 0 0
## [36] 0 0 1 0 1 0 1 1 1 0 0 0 0 0 1 1 1 1 0 1 0 0 1 1
```

```
tennisTest$Result
```

```
##  [1] 0 0 1 0 0 1 1 0 1 0 1 0 1 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 1 1 1 1 0 1
## [36] 1 1 1 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 1 0 0 1 0
```

```
table(predict = r, true = tennisTest$Result)
```

```
##        true
## predict  0  1
##       0 22  6
##       1  7 24
```

```
confMat3 = table(tennisTest$Result, p>0.5)
confMat3
```

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

Here the result from the logistic model on the testing data is shown. It is presented in a confusion matrix.

d)

$$\pi_k = Classprobabilities \mu_k = mean \Sigma = Covariancematrix f_k(x) = Distribution$$

e)

The class boundary exists where the probability of 1 equals to the probability of 0. As such;

$$\delta_0(x) = \delta_1(x)$$

We also know from the lectures that:

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + log\pi_k$$

Thus we also know that:

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

From this we get:

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

To find the prior probabilities the confusion matrix is found, and the number of correct guesses is divided by the length of the vector. The average

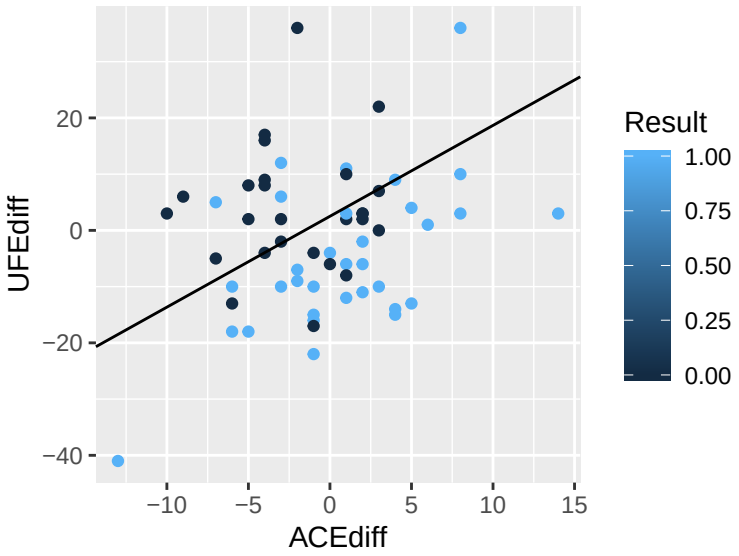
```
lda.fit = lda(Result ~ ACEdiff + UFEdiff, data = tennisTrain)
lda.fit
```

```
## Call:
## lda(Result ~ ACEdiff + UFEdiff, data = tennisTrain)
##
## Prior probabilities of groups:
##        0        1
## 0.440678 0.559322
##
## Group means:
##      ACEdiff   UFEdiff
## 0 -2.0769231  3.730769
## 1  0.7575758 -5.030303
##
## Coefficients of linear discriminants:
##                 LD1
## ACEdiff  0.18650205
## UFEdiff -0.07448027
```

```
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
```

```
ggplot(tennisTrain, aes(y=UFEdiff, x=ACEdiff, color=Result))+geom_point() + geom_abline(slope=1.616, in
```




```
qda.fit = qda(Result ~ ACEdiff + UFEdiff, data = tennisTrain)
qda.fit
```

```
## Call:
## qda(Result ~ ACEdiff + UFEdiff, data = tennisTrain)
##
## Prior probabilities of groups:
##        0        1
## 0.440678 0.559322
##
## Group means:
##      ACEdiff   UFEdiff
## 0 -2.0769231  3.730769
## 1  0.7575758 -5.030303
```

```
qda.fit.p = predict(qda.fit, tennisTest)$class
confMat2 = table(qda.fit.p, tennisTest$Result)
confMat2
```

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

Here i perform a LDA and a QDA on the training set, and then validate with the training set. The results are than printed in two confusion matrices.

From the confusion matrices, It seems that LDA is the best method. It has the same specificity, and one better case of sensitvity. The GLM model on the other hand is way off, this may be however user error and not because glm is a bad model for this case.

The best one is LDA.

Problem 4 - Classification

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, $C_1, C_2, \ldots, C_{10}$. 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 7.

$$\text{CV}_{10} = \sum_{k=1}^{10} \frac{n_k}{10} \text{Err}_k \tag{7}$$

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

$$\text{Err}_k = \sum_{i \in C_k} \frac{\text{I}(y_i \neq \hat{y}_i)}{n_k} \tag{8}$$

where I is the indicator function defined as

$$\text{I}(a \neq \hat{a}) = \begin{cases} 1 \text{ if } a \neq \hat{a} \\ 0 \text{ else} \end{cases} \tag{9}$$

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)

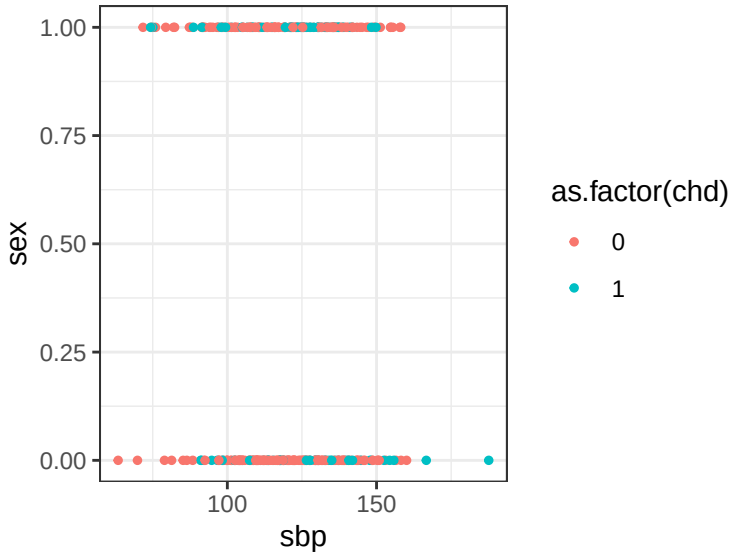
```
id <- "1I6dk1fA4ujBjZPo3Xj8pIfnzIa94WKcy" # google file ID
d.chd <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download", id))
#summary(d.chd)
str(d.chd)
```

```
## 'data.frame':    350 obs. of  3 variables:
##  $ sbp: num  133 124 188 141 129 ...
##  $ sex: int  1 0 0 0 0 0 0 0 1 0 ...
##  $ chd: int  0 0 1 1 1 0 0 0 0 0 ...
```

```
lm.fit = glm(data = d.chd, chd ~ sbp + sex, family = "binomial")
#summary(lm.fit)
eta <- summary(lm.fit)$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`.

```
ggplot(d.chd, aes(x = sbp, y = sex, color=as.factor(chd))) + geom_point(size=1) + theme_bw()
```



d)

```
B = 1000
n = dim(d.chd)[1]
estimator = rep(NA, B)

for (b in 1:B) {
  i = sample(x = c(1:n), size = n, replace = TRUE)
  newSample = d.chd[i,1:3]
  fit.4d = glm(chd ~ sbp + sex, data = newSample, family = "binomial")
  e <- summary(fit.4d)$coef[, 1] %*% c(1, 140, 1)
  estimator[b] = exp(e)/(1 + exp(e))
}

meanEstimator = mean(estimator)
SE = sqrt(1 / (B - 1) * sum((estimator - meanEstimator)^2))
confinterval = quantile(estimator, probs = c(2.5, 97.5)/100)
```

The standard error is then `0.048`, and the confidence interval for is [`0.29, 0.476`].