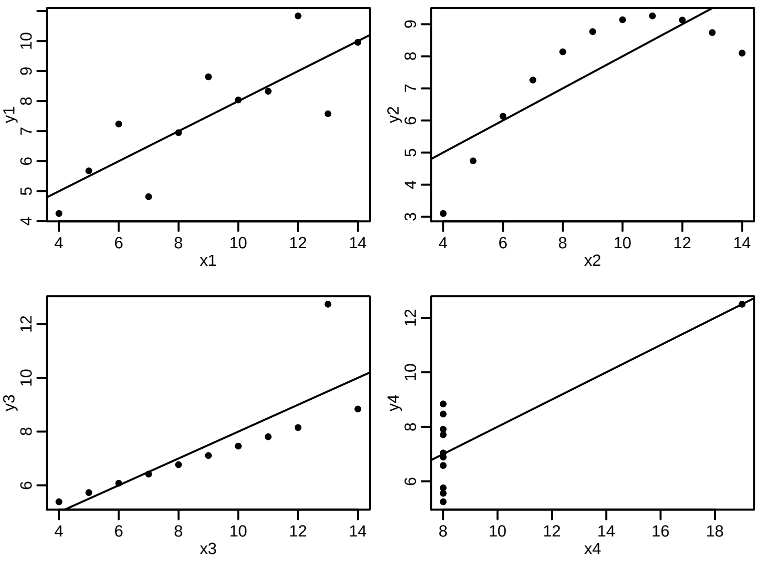
Solutions for Exercises for Applied Biostatistics II - FS 2019

1. (No solution.)

2. **a)** If you compare the four scatter plots, you can see that only in the first case a linear regression is reasonable. In the second case the relationship between X and Y is not linear but quadratic. In the third case an outlier has a huge influence on the estimated parameters. In the forth case the regression line is only dependent on one point.



b) This can be done in R the following way:

```
> ans <- lm(anscombe[, 5] ~ anscombe[, 1])
> summary(ans)
Call:
lm(formula = anscombe[, 5] ~ anscombe[, 1])

Residuals:
     Min       1Q   Median       3Q      Max
-1.92127 -0.45577 -0.04136  0.70941  1.83882

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)     3.0001     1.1247   2.667  0.02573 *
anscombe[, 1]   0.5001     0.1179   4.241  0.00217 **
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.237 on 9 degrees of freedom
Multiple R-squared:  0.6665,       Adjusted R-squared:  0.6295
F-statistic: 17.99 on 1 and 9 DF,  p-value: 0.00217
```

This is only an example for the first data set. The same can be done for the other 3 data sets in R. For all four models the estimated values for the intercept β_0, the slope β_1 and the standard deviation of error variables σ^2 as well as the quality criterion R^2 are identical. Pay attention that in the table we display the value σ^2, but R gives you the value σ:

	Modell 1	Modell 2	Modell 3	Modell 4
intercept ($\widehat{\beta_0}$)	3.000	3.001	3.002	3.002
slope ($\widehat{\beta_1}$)	0.500	0.500	0.500	0.500
$\widehat{\sigma}^2$	1.529	1.531	1.528	1.527
R^2	0.667	0.666	0.666	0.667

Conclusion: It is **not** sufficient to have a look at $\widehat{\beta_0}, \widehat{\beta_1}, \widehat{\sigma}^2$ and R^2. Here, in all models the estimated values are exactly the same, yet the data sets are completely different. So a graphical examination must always be done.

3. a)

```
> farm <- read.table("farm.dat", header = TRUE)
> fit <- lm(Dollar ~ cows, data = farm)
> summary(fit)
```

```
Call:
lm(formula = Dollar ~ cows, data = farm)

Residuals:
    Min      1Q  Median      3Q     Max 
-204.68  -80.02   15.48   54.57  284.43 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  694.019     50.039  13.869 4.75e-11 ***
cows          20.111      4.725   4.256 0.000475 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 122.9 on 18 degrees of freedom
Multiple R-squared:  0.5016,	Adjusted R-squared:  0.4739 
F-statistic: 18.11 on 1 and 18 DF,  p-value: 0.0004751
```

There is a significant dependence (e.g. on the 5% level) between income and number of cows, since the p-value of the regression coefficient is very small (0.000475).

b)

```
> predict(fit, newdata=data.frame(cows=c(0,20,8.85)), interval="confidence")
```

```
        fit       lwr       upr
1  694.0189  588.8902  799.1476
2 1096.2361  971.3953 1221.0768
3  872.0000  814.2627  929.7373
```

c) We first try to explain I with A:

```
> fit1 <- lm(Dollar~acres, data=farm)
> summary(fit1)
```

```
Call:
lm(formula = Dollar ~ acres, data = farm)

Residuals:
    Min      1Q  Median      3Q     Max 
-281.54 -113.94  -28.18   94.28  387.05 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 868.7363   105.9796   8.197 1.73e-07 ***
acres         0.0234     0.7066   0.033    0.974    
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 174.1 on 18 degrees of freedom
Multiple R-squared:  6.09e-05,       Adjusted R-squared:  -0.05549
F-statistic: 0.001096 on 1 and 18 DF,  p-value: 0.974
```

There seems to be no significant dependence. However, if we add C as a covariate, both variables are significant!

```
> fit2 <- lm(Dollar~acres+cows, data=farm)
> summary(fit2)
```

```
Call:
lm(formula = Dollar ~ acres + cows, data = farm)

Residuals:
     Min       1Q   Median       3Q      Max 
-145.064  -46.719   -9.992   55.149  133.664 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 285.4572    81.3793   3.508   0.0027 ** 
acres         2.1384     0.3936   5.434 4.47e-05 ***
cows         32.5690     3.7276   8.737 1.08e-07 ***
---
Signif. codes:  
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 76.45 on 17 degrees of freedom
Multiple R-squared:  0.8179,       Adjusted R-squared:  0.7965 
F-statistic: 38.17 on 2 and 17 DF,  p-value: 5.165e-07
```

It turns out that the covariates are collinear:

```
> corCA <- cor(farm$cows, farm$acres)
> corCA
```

```
[1] -0.615085
```

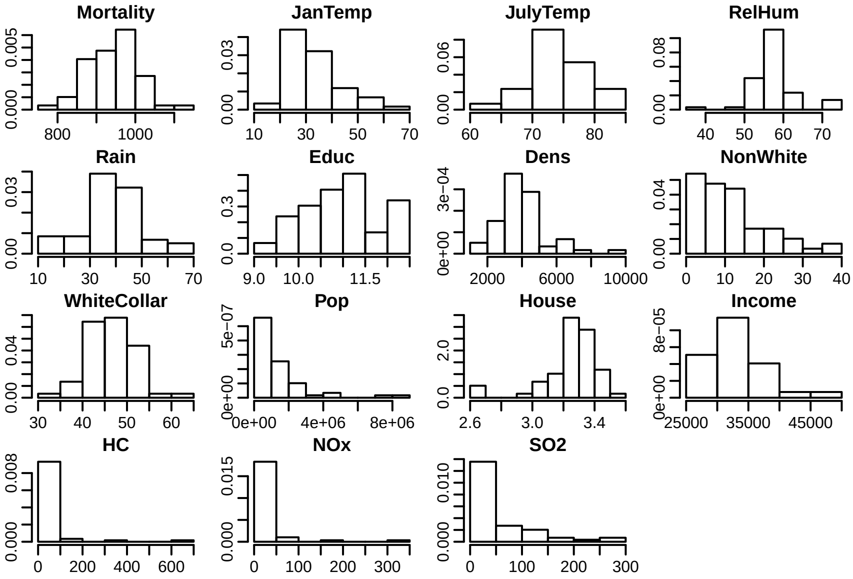
The income source *farm size* can only be identified if we control for the number of cows, i.e. comparing like with like. In colloquial terms, the positive correlation of I and C and the negative correlation of C and A cancel each other out. Thus the variable A is not considered significant in a univariate regression of I and A, though it is in the multiple linear regression of I, C and A.

4. **a)**

```
> ap <- read.table("airpollution.csv", header = TRUE, sep = ",")
> ap$City <- NULL
```

We have a look at the histograms we all ready saw in the lecture course.

```
> par(mfrow = c(4, 4), mar = c(1.5, 2, 1.5, 0.5), cex = 0.5)
> for (i in 1:ncol(ap)) hist(ap[, i], freq = FALSE, main = colnames(ap)[i],
   xlab = "", ylab = "")
```



We transform the following variables:

```
> ap$Pop          <- log(ap$Pop)
> ap$HC           <- log(ap$HC)
> ap$NOx          <- log(ap$NOx)
> ap$SO2          <- log(ap$SO2)
> names(ap)[names(ap)=="Pop"]<-"logPop"
> names(ap)[names(ap)=="HC"]<- "logHC"
> names(ap)[names(ap)=="NOx"]<-"logNOx"
> names(ap)[names(ap)=="SO2"]<-"logSO2"
```

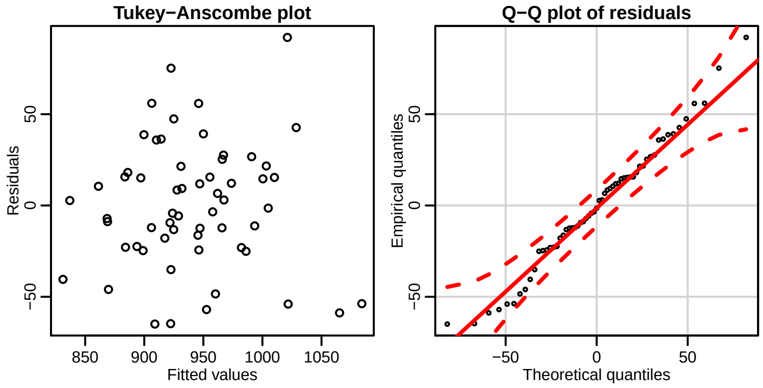
b) Full model:

```
> fit <- lm(Mortality ~ ., data=ap)
> summary(fit)
```

```
Call:
lm(formula = Mortality ~ ., data = ap)

Residuals:
    Min      1Q  Median      3Q     Max
-68.893 -20.704   0.586  24.129  74.604

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.297e+03  2.934e+02   4.422 6.32e-05 ***
JanTemp     -2.368e+00  8.851e-01  -2.676   0.0104 *
JulyTemp    -1.752e+00  2.031e+00  -0.863   0.3931
RelHum       3.420e-01  1.059e+00   0.323   0.7482
Rain         1.493e+00  5.898e-01   2.532   0.0150 *
Educ        -1.000e+01  9.087e+00  -1.101   0.2771
Dens         4.525e-03  4.223e-03   1.072   0.2897
NonWhite     5.152e+00  1.002e+00   5.143 6.01e-06 ***
WhiteCollar -1.883e+00  1.198e+00  -1.572   0.1232
logPop       4.391e+00  7.714e+00   0.569   0.5721
House       -4.574e+01  3.939e+01  -1.161   0.2518
Income      -6.892e-04  1.334e-03  -0.516   0.6081
logHC       -2.204e+01  1.523e+01  -1.447   0.1550
logNOx       3.397e+01  1.425e+01   2.384   0.0215 *
logSO2      -3.687e+00  7.359e+00  -0.501   0.6189
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 34.48 on 44 degrees of freedom
Multiple R-squared:  0.7685,       Adjusted R-squared:  0.6949
F-statistic: 10.43 on 14 and 44 DF,  p-value: 8.793e-10
```

```
> library(car)
> ap.fit <- lm(Mortality ~ ., data = ap)
> par(mfrow = c(1, 2), cex = 0.5)
> plot(fitted(ap.fit), resid(ap.fit),
   xlab = "Fitted values", ylab = "Residuals", main = "Tukey-Anscombe plot")
> qqPlot(resid(ap.fit), dist = "norm",
   mean = mean(resid(ap.fit)), sd = sd(resid(ap.fit)),
   xlab = "Theoretical quantiles", ylab = "Empirical quantiles",
   main = "Q-Q plot of residuals")
```

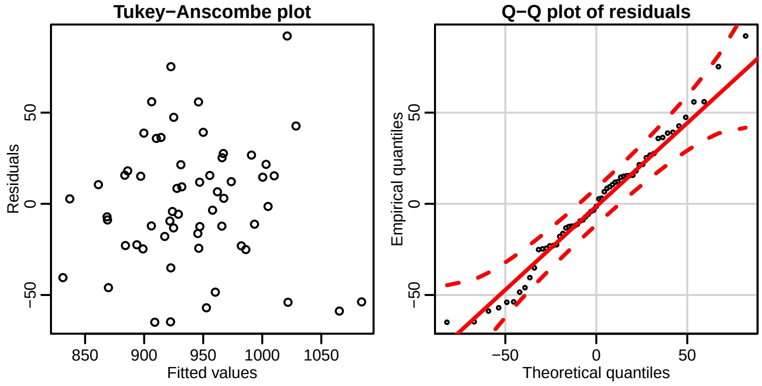


Even though most of the predictors seem to have no significant effect on the response, the model fits quite well. We do not see any violation of the model assumptions.

c) Now we just use the significant variables:

```
> fit2 <- lm(Mortality ~ JanTemp + Rain + NonWhite + logNOx, data=ap)
> summary(fit2)
```

```
Call:
lm(formula = Mortality ~ JanTemp + Rain + NonWhite + logNOx,
    data = ap)

Residuals:
    Min      1Q  Median      3Q     Max
-64.946 -22.658  -1.435  19.769  92.049

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 838.1347    27.9499  29.987  < 2e-16 ***
JanTemp      -2.4322     0.5220  -4.659 2.11e-05 ***
Rain          2.2528     0.4857   4.639 2.27e-05 ***
NonWhite      4.3129     0.6479   6.657 1.48e-08 ***
logNOx       20.2033     4.6113   4.381 5.47e-05 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 35.62 on 54 degrees of freedom
Multiple R-squared:  0.6968,       Adjusted R-squared:  0.6743
F-statistic: 31.02 on 4 and 54 DF,  p-value: 2.018e-13
```



Now all the variables are highly significant. As expected with fewer variables, the residuals are a little bigger now and R^2 decreased slightly. However, the difference in adjusted R^2 is very small, indicating that we have not lost much explanatory power.

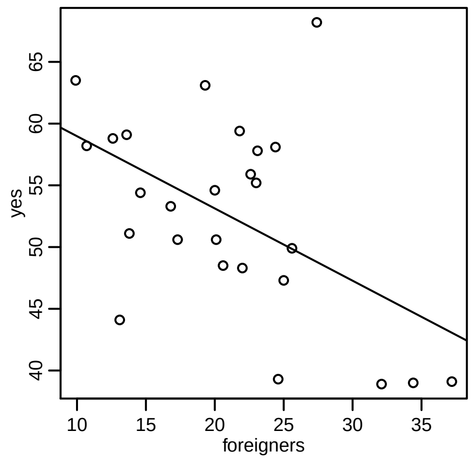
Even though leaving out all of the non-significant variable at once worked quite well here, this is not a good strategy in general. If the predictors are not mutually independent, leaving out one can have a huge effect on the significance of the others. A better way of pruning the model thus is to leave out predictors step by step, one at a time.

5. **a)** We need to fit a linear model to the data at first.

```
> mi <- read.table("massimmigration.csv", header = TRUE, sep = "\t")
> rownames(mi) <- mi$canton
> mi$canton <- NULL
> plot(yes ~ foreigners, data = mi)
> mi.fit <- lm(yes ~ foreigners, data = mi)
> summary(mi.fit)
```

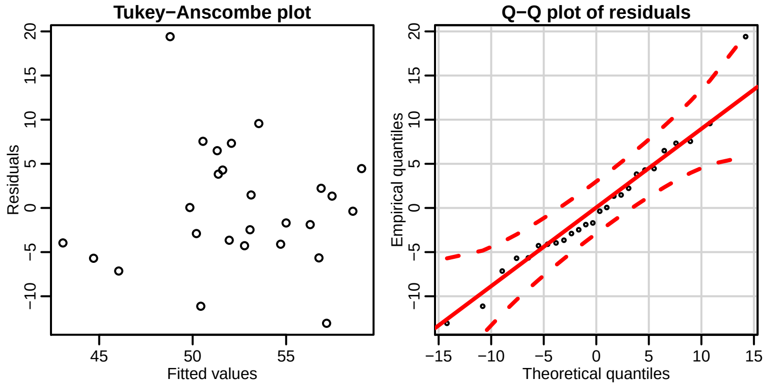
```
Call:
lm(formula = yes ~ foreigners, data = mi)

Residuals:
    Min      1Q  Median      3Q     Max
-13.065  -4.070  -1.034   4.179  19.405

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  64.8317     4.4174  14.676 1.74e-13 ***
foreigners   -0.5853     0.2001  -2.925   0.0074 **
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.006 on 24 degrees of freedom
Multiple R-squared:  0.2629,	Adjusted R-squared:  0.2321
F-statistic: 8.558 on 1 and 24 DF,  p-value: 0.007404
```

```
> abline(mi.fit)
```



To check the model assumptions we do a qq plot and a Turkey-Anscombe plot.

```
> par(mfrow = c(1, 2), cex = 0.5)
> plot(fitted(mi.fit), resid(mi.fit),
   xlab = "Fitted values", ylab = "Residuals", main = "Tukey-Anscombe plot")
> library(car)
> qqPlot(resid(mi.fit), dist = "norm",
   mean = mean(resid(mi.fit)), sd = sd(resid(mi.fit)),
   xlab = "Theoretical quantiles", ylab = "Empirical quantiles",
   main = "Q-Q plot of residuals")
```



Both plots look okay. So the model assumption are not violated.

b) Like we saw in the lecture course (on Slide 11) the R-function predict is a good idea to start with. We get the following lines in R.

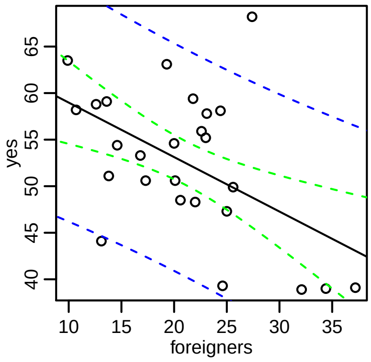
```
> x.val <- seq(0, 40, by = 1)
> pred.band <- predict(mi.fit, level = 0.9,
   newdata = data.frame(foreigners = x.val), interval = "prediction")
> conf.band <- predict(mi.fit, level = 0.9,
   newdata = data.frame(foreigners = x.val), interval = "confidence")
> lines(x.val, pred.band[, "lwr"], col = "blue", lty = 2)
> lines(x.val, pred.band[, "upr"], col = "blue", lty = 2)
> lines(x.val, conf.band[, "lwr"], col = "green", lty = 2)
> lines(x.val, conf.band[, "upr"], col = "green", lty = 2)
```

```
> mi <- read.table("massimmigration.csv", header = TRUE, sep = "\t")
> rownames(mi) <- mi$canton
> mi$canton <- NULL
> plot(yes ~ foreigners, data = mi)
> mi.fit <- lm(yes ~ foreigners, data = mi)
> summary(mi.fit)
```

```
Call:
lm(formula = yes ~ foreigners, data = mi)
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-13.065  -4.070  -1.034   4.179  19.405 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  64.8317     4.4174  14.676 1.74e-13 ***
foreigners   -0.5853     0.2001  -2.925   0.0074 ** 
---
Signif. codes:  
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.006 on 24 degrees of freedom
Multiple R-squared:  0.2629,	Adjusted R-squared:  0.2321 
F-statistic: 8.558 on 1 and 24 DF,  p-value: 0.007404
```

```
> abline(mi.fit)
> x.val <- seq(0, 40, by = 1)
> pred.band <- predict(mi.fit, level = 0.9,
   newdata = data.frame(foreigners = x.val), interval = "prediction")
> conf.band <- predict(mi.fit, level = 0.9,
   newdata = data.frame(foreigners = x.val), interval = "confidence")
> lines(x.val, pred.band[, "lwr"], col = "blue", lty = 2)
> lines(x.val, pred.band[, "upr"], col = "blue", lty = 2)
> lines(x.val, conf.band[, "lwr"], col = "green", lty = 2)
> lines(x.val, conf.band[, "upr"], col = "green", lty = 2)
```



The Confidence band indicates the accuracy of estimation of the true regression line.
The Prediction interval indicates, between which boundaries the y-value of a new measurement will be with certainty 1-α, at position x.

c) We first compute R^2 with the formula from the lecture on slide 44.

$$R^2 = \frac{\left\|\hat{Y} - \overline{Y}\right\|^2}{\left\|Y - \overline{Y}\right\|^2}$$

In R, this can be done as follows:

```
> yhat <- fitted(mi.fit)
> ybar <- mean(mi$yes)
> (R.sq <- sum((yhat - ybar)^2)/sum((mi$yes - ybar)^2))
[1] 0.2628564
```

The F statistic is given by

$$F = \frac{\|\hat{Y} - \bar{Y}\|^2/(q-1)}{\|Y - \hat{Y}\|^2/(n-q)} ,$$

where q is the number of regression parameters to fit (including the intercept!). In R, this can be done as follows:

```
> n <- nrow(mi)
> q <- 2
> (F <- sum((yhat - ybar)^2)*(n - q)/sum((mi$yes - yhat)^2)/(q - 1))
[1] 8.558108
```

To check if our R^2 and F are correct, we have a look at the values that R computes.

```
> summary(mi.fit)
Call:
lm(formula = yes ~ foreigners, data = mi)

Residuals:
    Min      1Q  Median      3Q     Max 
-13.065  -4.070  -1.034   4.179  19.405 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  64.8317     4.4174  14.676 1.74e-13 ***
foreigners   -0.5853     0.2001  -2.925   0.0074 ** 
---
Signif. codes:  
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.006 on 24 degrees of freedom
Multiple R-squared:  0.2629,	Adjusted R-squared:  0.2321 
F-statistic: 8.558 on 1 and 24 DF,  p-value: 0.007404
```

d) We need to add the new variable density and then do a multiple linear model. Then we remove at each step the least significant variable, and we check with the F-test if our model is now significantly better or not. If the old model was better, we keep the old model and are done. If the new model is better, we again remove the least significant variable and check the model with an F-test. This gives us in R the following result.

```
> mi$density <- mi$inhabitants/mi$area
> mi.full <- lm(yes ~ ., data = mi)
> summary(mi.full)
Call:
lm(formula = yes ~ ., data = mi)

Residuals:
     Min       1Q   Median       3Q      Max 
-13.6334  -2.7179   0.5134   4.2234  17.7448 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  6.281e+01  5.026e+00  12.499 3.42e-11 ***
area        -5.972e-04  8.599e-04  -0.695    0.495    
inhabitants -3.474e-06  5.018e-06  -0.692    0.496    
foreigners  -3.353e-01  2.573e-01  -1.303    0.207    
density     -2.513e-03  1.693e-03  -1.484    0.153    
---
Signif. codes:  
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.983 on 21 degrees of freedom
Multiple R-squared:  0.3593,	Adjusted R-squared:  0.2373 
F-statistic: 2.945 on 4 and 21 DF,  p-value: 0.0445
> mi.red1 <- update(mi.full, . ~ . - inhabitants)
> anova(mi.red1, mi.full)
Analysis of Variance Table
```

```
Model 1: yes ~ area + foreigners + density
Model 2: yes ~ area + inhabitants + foreigners + density
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     22 1047.3
2     21 1024.0  1    23.366 0.4792 0.4964
> summary(mi.red1)
Call:
lm(formula = yes ~ area + foreigners + density, data = mi)

Residuals:
     Min       1Q   Median       3Q      Max
-13.1777  -2.6501   0.1105   3.6335  18.3475

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 63.3799985  4.8999629  12.935 9.29e-12 ***
area        -0.0008504  0.0007690  -1.106    0.281
foreigners  -0.3955019  0.2393073  -1.653    0.113
density     -0.0024644  0.0016717  -1.474    0.155
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.9 on 22 degrees of freedom
Multiple R-squared:  0.3447,        Adjusted R-squared:  0.2554
F-statistic: 3.858 on 3 and 22 DF,  p-value: 0.02334
> mi.red2 <- update(mi.red1, . ~ . - area)
> summary(mi.red2)
Call:
lm(formula = yes ~ foreigners + density, data = mi)

Residuals:
     Min       1Q   Median       3Q      Max
-12.5467  -4.3562  -0.1261   3.5218  17.6284

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 62.313648   4.827398  12.908 5.08e-12 ***
foreigners  -0.419684   0.239459  -1.753    0.093 .
density     -0.001998   0.001625  -1.229    0.231
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.933 on 23 degrees of freedom
Multiple R-squared:  0.3083,        Adjusted R-squared:  0.2481
F-statistic: 5.126 on 2 and 23 DF,  p-value: 0.01442
```

In the next step, we would get back the model fitted in task a).

6. a) The model assumed by the biologists would be of the form:

$$\texttt{jump}_i = \beta_0 + \beta_1 \texttt{length}_i + E_i, \text{ with } E_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2) \text{ for } i = 1, \ldots, n$$

And given the output, this would result in the fitted model:

$$\widehat{\texttt{jump}_i} = 51.742 + 0.349 \cdot \texttt{length}_i, \text{ for } i = 1, \ldots, n$$

b) The residual standard error has $(n-q)$ degrees of freedom. Given that $q=2$, and $(n-q)=9$, it follows that $n=9+2=11$ frogs

c) No, because the coefficient of determination R^2 is very small. Moreover, the variable `length` is not significant.

d) From the answer to b), values for the residual standard error, R^2, and F statistic, plus the formulas for these (as given in the lecture), can be recovered:

$$\begin{aligned}
\texttt{MS}_{\text{resid}} &= \hat{\sigma}^2 = \texttt{Residual standard error}^2 = (18.15)^2 = 329.42\\
\texttt{SS}_{\text{resid}} &= (n-q)\cdot \texttt{MS}_{\text{resid}} = 9\cdot(18.15)^2 = 2964.80\\
\texttt{MS}_{\text{regression}} &= F\cdot \texttt{MS}_{\text{resid}} = 0.7755\cdot(18.15)^2 = 255.47\\
\texttt{SS}_{\text{regression}} &= (q-1)\cdot \texttt{MS}_{\text{regression}} = 1\cdot 255.47 = 255.47\\
\texttt{SS}_{\text{total}} &= \texttt{SS}_{\text{resid}} + \texttt{SS}_{\text{regression}} = 2964.80 + 255.47 = 3220.27
\end{aligned}$$

Double check using R^2:

$$R^2 = \texttt{SS}_{\text{regression}}/\texttt{SS}_{\text{total}} = 255.47/3220.27 = 0.0793$$

Correct! So the table now looks like:

Source of variation	**df (degrees of freedom)**	**SS (sum of squares)**	**MS (mean square)**
regression	1	255.47	255.47
errors / resid.	9	2964.80	329.42
total around global mean	10	3220.27	

7. **a)** The null hypothesis claims that there is no significant difference in average daffodil stem lenght between different locations (i.e. groups). So that, given the model

$$Y_{ij} = \mu + \alpha_i + E_{ij},\ \text{ with } \alpha_i \text{ the groupeffect for } \ i = 2,\ldots,5,$$

$$\text{where } (1 = \text{East}),\ 2 = \text{North},\ 3 = \text{Open},\ 4 = \text{South},\ 5 = \text{West},$$

$$\text{and } \ E_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,\sigma^2) \text{ for } \ i = 2,\ldots,5,\ j = 1,\ldots,n_i$$

the null hypothesis is:

$$H_0 : \alpha_2 = \alpha_3 = \alpha_4 = \alpha_5 = 0$$

b) No, especially daffodils growing in the group "Open" display different growing behaviour, i.e. we expect $\alpha_3 \neq 0$

c)

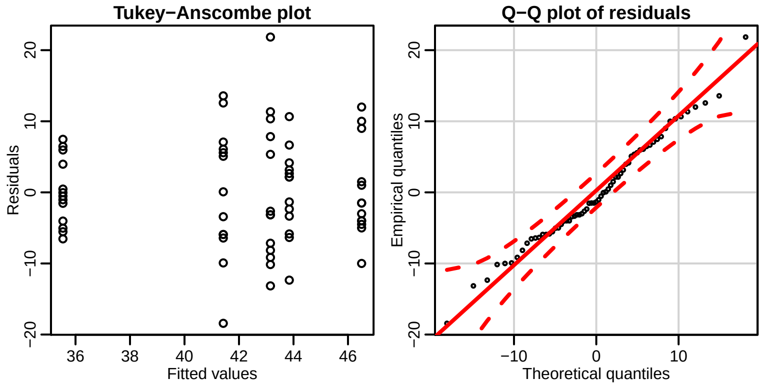
```
> fit <- lm(Length ~ Side, data=daffodils)
> summary(fit)
Call:
lm(formula = Length ~ Side, data = daffodils)

Residuals:
    Min      1Q  Median      3Q     Max
-18.423  -5.038  -1.346   5.577  21.846

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  43.8462     2.1449  20.442  < 2e-16 ***
SideNorth    -2.4231     3.0334  -0.799  0.42755
SideOpen     -8.3077     3.0334  -2.739  0.00811 **
SideSouth     2.6538     3.0334   0.875  0.38513
SideWest     -0.6923     3.0334  -0.228  0.82024
```

```
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.734 on 60 degrees of freedom
Multiple R-squared:  0.1954,       Adjusted R-squared:  0.1417
F-statistic: 3.642 on 4 and 60 DF,  p-value: 0.01009
```

At a significance level of 10% the daffodils from ”Open” differ significantly, so that the null hypothesis gets rejected, since $\alpha_3 \neq 0$

d)

```
>   library(car)
> daf.fit <- lm(Length ~ Side, data = daffodils)
> par(mfrow = c(1, 2), cex = 0.5)
> plot(fitted(daf.fit), resid(daf.fit),
      xlab = "Fitted values", ylab = "Residuals", main = "Tukey-Anscombe plot")
> qqPlot(resid(daf.fit), dist = "norm",
        mean = mean(resid(daf.fit)), sd = sd(resid(daf.fit)),
        xlab = "Theoretical quantiles", ylab = "Empirical quantiles",
        main = "Q-Q plot of residuals")
```



The Tukey-Ascombe plot shows a fairly similar spread of fitted values, given location, though ”Open” does seem somewhat more condensed than the other areas. The Q-Q plot suggests that the residuals are more or less normal. So overall the model (where indeed only ”Open” has a significant effect) fits quite well.

e)

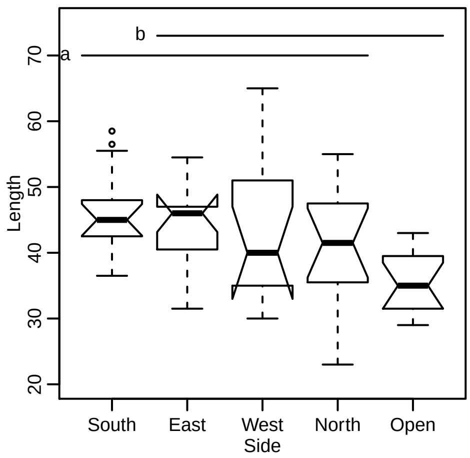
```
> pairwise.t.test (x=daffodils$Length, g=daffodils$Side, p.adj="bonf")

        Pairwise comparisons using t tests with pooled SD

data:  daffodils$Length and daffodils$Side

      East   North  Open   South
North 1.0000 -      -      -
Open  0.0811 0.5709 -      -
South 1.0000 0.9940 0.0062 -
West  1.0000 1.0000 0.1477 1.0000

P value adjustment method: bonferroni
```

The table reveals that upon comparing all locations with one another, the open area daffodils might grow differently from the ones around the building, but only ”Open” vs. ”South” showed a significant average stem length difference at 5%.
From the summary in c), we see that the cell means are ordered as

$$\mu_{\text{South}} > \mu_{\text{East}} > \mu_{\text{West}} > \mu_{\text{North}} > \mu_{\text{Open}} \ .$$

On a 5% level, the groups “South” and “Open” significantly differ, but “South” and “North” do not: hence we can draw a bar (labelled “a”) over the groups “South” to “North”. Since “Open” does not show a significant difference to any other location than “South”, we draw another bar (labelled “b”) over the groups “East” to “Open”.

```
> side.ord <- factor(daffodils$Side, c("South", "East", "West", "North", "Open"))
> boxplot(daffodils$Length ~ side.ord, notch = TRUE, ylim = c(20, 75),
   xlab = "Side", ylab = "Length")
> lines(c(0.6, 4.4), c(70, 70))
> text(0.6, 70, "a", pos = 2)
> lines(c(1.6, 5.4), c(73, 73))
> text(1.6, 73, "b", pos = 2)
```



8. **a)** You can see in the table that the variable "activity" has 2 degrees of freedom. So there must be 3 different activity groups. As there are all in all 28 degrees of freedom, their must be 29 women.

   b) As the p-value is smaller than 0.05, we can reject the null hypothesis. Therefore we know that not all sport courses lead to the same flexibility. We just don't know which one differs from the other or if they all differ.

   c) spool= $\sqrt{0.5799}$

9. We use the Bonferroni-adjusted significance level $\alpha = \overline{\alpha}/3 = 0.05/3$ and build confidence intervals for cell means μ_i using the empirical means of our data

$$\left[\overline{y}_{\bullet i} - \Phi^{-1}(1-\alpha/2)\frac{s_{\text{pool}}}{\sqrt{n_i}}, \overline{y}_{\bullet i} + \Phi^{-1}(1-\alpha/2)\frac{s_{\text{pool}}}{\sqrt{n_i}}\right]$$

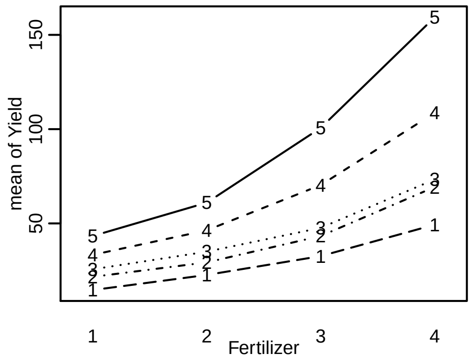
where $s_{\text{pool}} = \hat{\sigma} = \sqrt{\texttt{MS(within)}}$. We get:

```
> alpha <- 0.05/3
> z <- qnorm(1-alpha/2)
> spool <- sqrt(2.071)
> sqn <- sqrt(36)
> ciB <- c(5.4 - z*spool/sqn, 5.4 + z*spool/sqn)
> ciC <- c(6.2 - z*spool/sqn, 6.2 + z*spool/sqn)
> ciD <- c(5.0 - z*spool/sqn, 5.0 + z*spool/sqn)
> list(ciB, ciC, ciD)

[[1]]
[1] 4.825805 5.974195

[[2]]
[1] 5.625805 6.774195

[[3]]
[1] 4.425805 5.574195
```

Since not one confidence interval included the control group's mean (0.8), we can conclude with 95% confidence that all treatments significantly heigthened the mean selenium concentration in the cows' blood.

10. **a)** First we need to convert the variables CropSpecies and Fertilizer to factors. Then we can do an interaction plot.

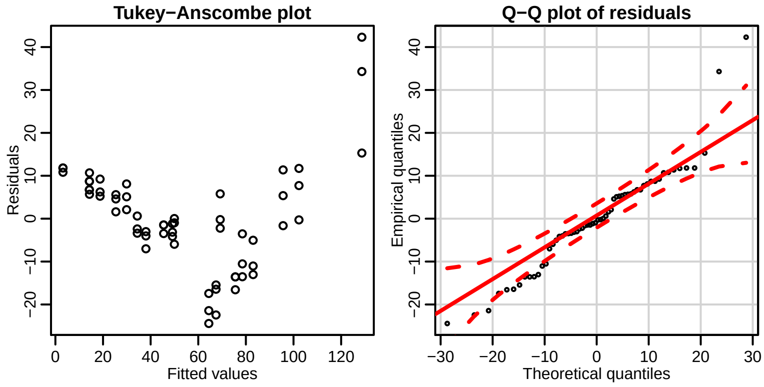
```
> fertilizer <- read.table("fertilizer.dat", header = TRUE)
> fertilizer$Fertilizer <- factor(fertilizer$Fertilizer)
> fertilizer$CropSpecies <- factor(fertilizer$CropSpecies)
> interaction.plot(fertilizer$Fertilizer, fertilizer$CropSpecies, fertilizer$Yield,
   type = "b", legend = FALSE,, xlab="Fertilizer", ylab="mean of Yield")
```



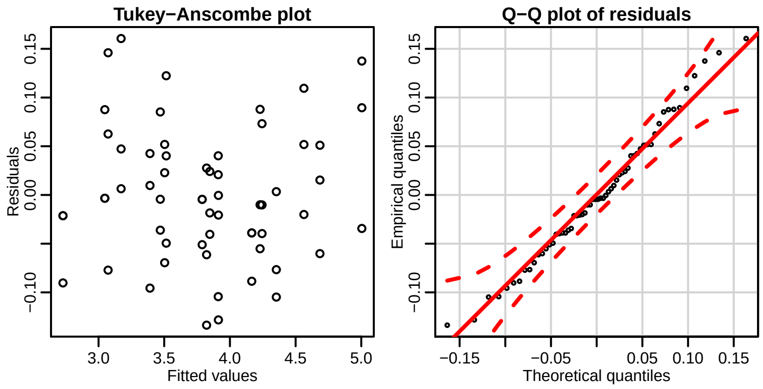
We can see that all five lines in the plot act similarly. Their could be interaction between our data.

b)

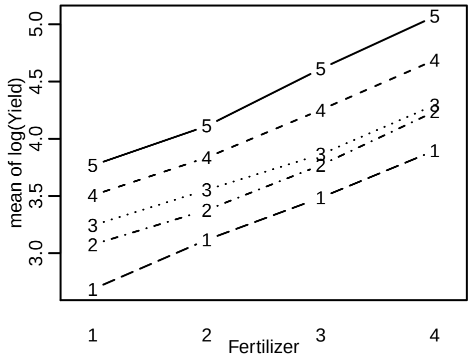
```
> fertilizer.fit <- lm(Yield ~ Fertilizer + CropSpecies, data = fertilizer)
> par(mfrow = c(1, 2), cex = 0.5)
> plot(fitted(fertilizer.fit), resid(fertilizer.fit),
   xlab = "Fitted values", ylab = "Residuals", main = "Tukey-Anscombe plot")
> library(car)
> qqPlot(resid(fertilizer.fit), dist = "norm",
   mean = mean(resid(fertilizer.fit)), sd = sd(resid(fertilizer.fit)),
   xlab = "Theoretical quantiles", ylab = "Empirical quantiles",
   main = "Q-Q plot of residuals")
```



We clearly see in both plots that this is a bad fit. We try to get rid of the right-skewness with a log-transformation of yield.



Now both plots clearly fit better. The interaction plot looks as follows:



c) We try the model log(Yield) ∼ Fertilizer * CropSpecies + Fertilizer + CropSpecies.

```
> fertilizer.int <- lm(log(Yield) ~ Fertilizer*CropSpecies, data = fertilizer)
> anova(fertilizer.fit, fertilizer.int)
```

```
Analysis of Variance Table

Model 1: log(Yield) ~ Fertilizer + CropSpecies
Model 2: log(Yield) ~ Fertilizer * CropSpecies
  Res.Df     RSS Df Sum of Sq      F Pr(>F)
1     52 0.27539
2     40 0.18253 12   0.09286 1.6958 0.1045
```

We can see that the model is not yet significant.

d) If we follow slide 41 from the lecture, point 3 says, remove the interaction variable if the model with this variable is not significant. Our model in part c) is not significant, so we have to remove the interaction term between Fertilizer and CropSpecies. We then get the model fertilizer.fit = (log(Yield)∼Fertilizer+CropSpecies). Now we need to check if one of the variables Fertilizer or CropSpecies is not significant.

```
> summary(fertilizer.fit)
```

```
Call:
lm(formula = log(Yield) ~ Fertilizer + CropSpecies, data = fertilizer)

Residuals:
      Min        1Q    Median        3Q       Max
-0.133656 -0.042664 -0.004453  0.043701  0.160569

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.72933    0.02657  102.71  < 2e-16 ***
Fertilizer2   0.34356    0.02657   12.93  < 2e-16 ***
Fertilizer3   0.74080    0.02657   27.88  < 2e-16 ***
Fertilizer4   1.18167    0.02657   44.47  < 2e-16 ***
CropSpecies2  0.31858    0.02971   10.72 8.83e-15 ***
CropSpecies3  0.44230    0.02971   14.89  < 2e-16 ***
```

```
CropSpecies4  0.77417    0.02971   26.06  < 2e-16 ***
CropSpecies5  1.09320    0.02971   36.80  < 2e-16 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.07277 on 52 degrees of freedom
Multiple R-squared:  0.9866,        Adjusted R-squared:  0.9847
F-statistic: 545.2 on 7 and 52 DF,  p-value: < 2.2e-16
```

```
> cropspecies.fit <- lm(formula = log(Yield) ~ CropSpecies, data = fertilizer)
> summary(cropspecies.fit)
```

```
Call:
lm(formula = log(Yield) ~ CropSpecies, data = fertilizer)

Residuals:
     Min       1Q   Median       3Q      Max
-0.70016 -0.34042 -0.01546  0.34043  0.75262

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    3.2958     0.1347  24.476  < 2e-16 ***
CropSpecies2   0.3186     0.1904   1.673 0.100020
CropSpecies3   0.4423     0.1904   2.323 0.023925 *
CropSpecies4   0.7742     0.1904   4.065 0.000154 ***
CropSpecies5   1.0932     0.1904   5.741  4.2e-07 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4665 on 55 degrees of freedom
Multiple R-squared:  0.4159,        Adjusted R-squared:  0.3734
F-statistic:  9.79 on 4 and 55 DF,  p-value: 4.716e-06
```

```
> fert.fit <- lm(formula = log(Yield) ~ Fertilizer, data = fertilizer)
```

We can see that both variable are significant. So we found our final model in log(Yield)∼Fertilizer+CropSpecies.

11. a)

```
> strawb <- read.table("strawb.dat", header = TRUE)
> strawb$land <- factor(strawb$land)
> strawb.fit.block <- lm(yield ~ gtype + land, data = strawb)
> anova(strawb.fit.block)
```

```
Analysis of Variance Table

Response: yield
          Df Sum Sq Mean Sq F value Pr(>F)
gtype      2 289.65 144.824  5.4056 0.0145 *
land       9 115.97  12.886  0.4810 0.8687
Residuals 18 482.25  26.792
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The factor “genotype” is significant on a 5% level, but not on a 1% level. The block factor “land” does not have much influence on the yield.

b) We analyse the data without the block factor.

```
> strawb.fit.nb <- lm(yield ~ gtype, data = strawb)
> anova(strawb.fit.nb)
```

```
Analysis of Variance Table

Response: yield
          Df Sum Sq Mean Sq F value   Pr(>F)
gtype      2 289.65 144.824  6.5364 0.004841 **
Residuals 27 598.22  22.156
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The factor ”genotype” is now significant on a 1% level.

c) It is obvious that gtype is more significant if we test without the land variable. The degree of freedom changes because we removed the land variable and so the degree of freedom of this variable ($df = 9$) is added to the degree of freedom of the residuals. I would use the second model, because we estimated too many parameters in the first model. Moreover, the adjusted R^2 is higher in the second model.

12. a)

```
> rikz <- read.table("RIKZ.txt", header = TRUE, sep = "\t")
> str(rikz)
```

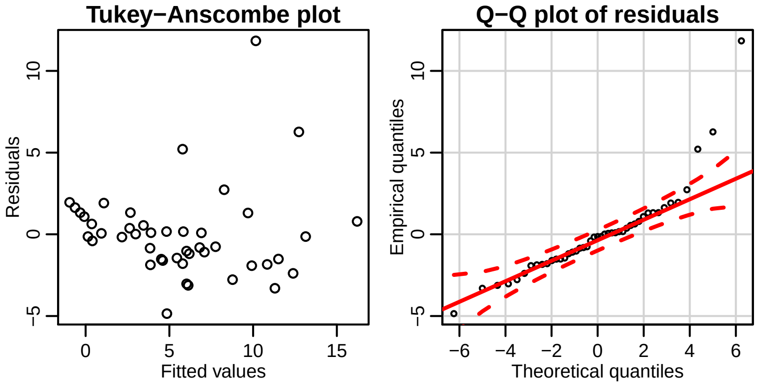
```
'data.frame':       45 obs. of  5 variables:
 $ Sample  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Richness: int  11 10 13 11 10 8 9 8 19 17 ...
 $ Exposure: int  10 10 10 10 10 8 8 8 8 8 ...
 $ NAP     : num  0.045 -1.036 -1.336 0.616 -0.684 ...
 $ Beach   : int  1 1 1 1 1 2 2 2 2 2 ...
```

```
> rikz$Beach <- factor(rikz$Beach)
> rikz$Sample <- NULL
```

We apply corrections so as to ignore the indexing numbers from ”Sample” (which do not inform the model) and turn ”Beach” into a factor instead of an integer variable.

b)

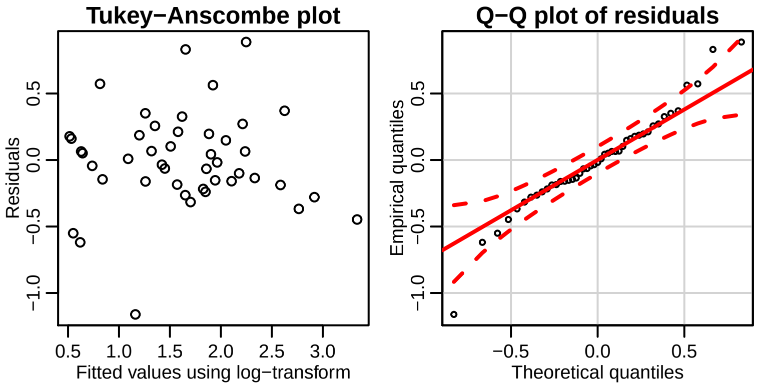
```
Call:
lm(formula = Richness ~ ., data = rikz)

Residuals:
    Min      1Q  Median      3Q     Max
-4.8518 -1.5188 -0.1376  0.7905 11.8384

Coefficients: (1 not defined because of singularities)
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  64.1726    21.5206   2.982  0.00519 **
Exposure     -5.4367     2.0506  -2.651  0.01196 *
NAP          -2.4928     0.5023  -4.963 1.79e-05 ***
Beach2       -7.7952     5.4060  -1.442  0.15821
Beach3       -0.9683     1.9841  -0.488  0.62858
Beach4       -0.5962     1.9420  -0.307  0.76066
Beach5       -0.8983     2.0105  -0.447  0.65778
Beach6        0.2136     1.9616   0.109  0.91393
Beach7            NA         NA      NA       NA
Beach8       -4.5530     1.9972  -2.280  0.02883 *
Beach9       -3.7820     2.0060  -1.885  0.06770 .
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.06 on 35 degrees of freedom
Multiple R-squared:  0.7025,       Adjusted R-squared:  0.626
F-statistic: 9.183 on 9 and 35 DF,  p-value: 5.645e-07
```



Problems are the lack of a coefficient for ”Beach7” due to singularities (though once the data are transformed this problem disappears), the slight cone-shape visible in the Tukey-Anscombe plot and deviation of the residuals from a normal distribution in the upper quantiles.

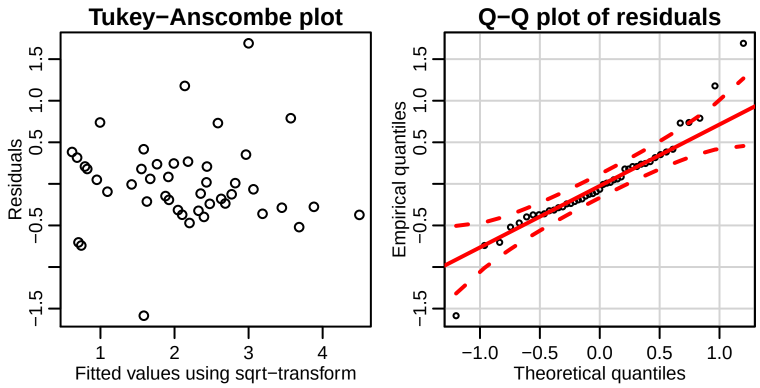
c) To improve the model fit we can try applying either a log or a square root transformation, as these lessen extreme values in end tails. However, log(0) = `-inf` which would seriously distort the model. So to avoid this from happening, add 1 to all ”Richness” values.

```
> rikz.fit1 <- lm(log(1 + Richness) ~ ., data = rikz)
> par(mfrow = c(1, 2), cex = 0.6)
> plot(fitted(rikz.fit1), resid(rikz.fit1),
      xlab = "Fitted values using log-transform", ylab = "Residuals", main = "Tukey-Anscombe p
> qqPlot(resid(rikz.fit1), dist = "norm",
   mean = mean(resid(rikz.fit1)), sd = sd(resid(rikz.fit1)),
   xlab = "Theoretical quantiles", ylab = "Empirical quantiles",
        main = "Q-Q plot of residuals")
```



The log transform makes for a better fit

```
> rikz.fit2 <- lm(sqrt(Richness) ~ ., data = rikz)
> par(mfrow = c(1, 2), cex = 0.6)
> plot(fitted(rikz.fit2), resid(rikz.fit2),
       xlab = "Fitted values using sqrt-transform", ylab = "Residuals", main = "Tukey-Anscombe
> qqPlot(resid(rikz.fit2), dist = "norm",
   mean = mean(resid(rikz.fit2)), sd = sd(resid(rikz.fit2)),
   xlab = "Theoretical quantiles", ylab = "Empirical quantiles",
        main = "Q-Q plot of residuals")
```



Also the square root transform produces somewhat better model fit.

d) We do a backward selection on the log-transformed data, so that ”Exposure” got removed and ”NAP” and ”Beach” kept, though beaches 2, 5 and 9 have non-significant coefficients.

```
> library(MASS)
> rikz.bw <- stepAIC(rikz.fit1, direction = "backward", trace = 0)
> summary(rikz.bw)
```

```
Call:
lm(formula = log(1 + Richness) ~ NAP + Beach, data = rikz)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.16096 -0.18464 -0.01796  0.18628  0.88732 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  2.23644    0.18456  12.118 4.44e-14 ***
NAP         -0.51043    0.06671  -7.651 5.63e-09 ***
Beach2       0.42025    0.26193   1.604  0.11760    
Beach3      -0.83201    0.25905  -3.212  0.00283 ** 
Beach4      -0.80163    0.26609  -3.013  0.00479 ** 
Beach5      -0.24501    0.26704  -0.917  0.36516    
Beach6      -0.58716    0.26142  -2.246  0.03112 *  
Beach7      -0.69613    0.27237  -2.556  0.01509 *  
Beach8      -0.57004    0.26527  -2.149  0.03864 *  
Beach9      -0.53503    0.26644  -2.008  0.05240 .  
---
Signif. codes:  
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4065 on 35 degrees of freedom
Multiple R-squared:  0.7807,	Adjusted R-squared:  0.7243 
F-statistic: 13.84 on 9 and 35 DF,  p-value: 3.82e-09
```

Again backward selection on the square root transformed data, with the same results except Beach 9 is now significant.

```
> rikz.bw2 <- stepAIC(rikz.fit2, direction = "backward", trace = 0)
> summary(rikz.bw2)
```

```
Call:
lm(formula = sqrt(Richness) ~ NAP + Beach, data = rikz)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.58544 -0.28653 -0.06544  0.23657  1.69043 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)  2.99457    0.26711  11.211 3.92e-13 ***
NAP         -0.66410    0.09655  -6.878 5.49e-08 ***
Beach2       0.61544    0.37909   1.623  0.11346
Beach3      -1.21158    0.37491  -3.232  0.00268 **
Beach4      -1.13596    0.38510  -2.950  0.00564 **
Beach5      -0.32863    0.38648  -0.850  0.40093
Beach6      -0.90219    0.37835  -2.385  0.02265 *
Beach7      -0.98741    0.39419  -2.505  0.01705 *
Beach8      -0.89080    0.38392  -2.320  0.02628 *
Beach9      -0.79350    0.38561  -2.058  0.04712 *
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5882 on 35 degrees of freedom
Multiple R-squared:  0.7596,        Adjusted R-squared:  0.6978
F-statistic: 12.29 on 9 and 35 DF,  p-value: 1.744e-08
```

13. 1. The interest lies in the influence of the physiological quantities, not the patients; patients should therefore be modelled as a random effect.

2. The cells were selected out of many and could have been chosen differently: the biological replicates should be modelled as a random effect.

3. The yeast strain in a gene expression study should be modelled as fixed effect, because the yeast strain is the variable we are interested in.

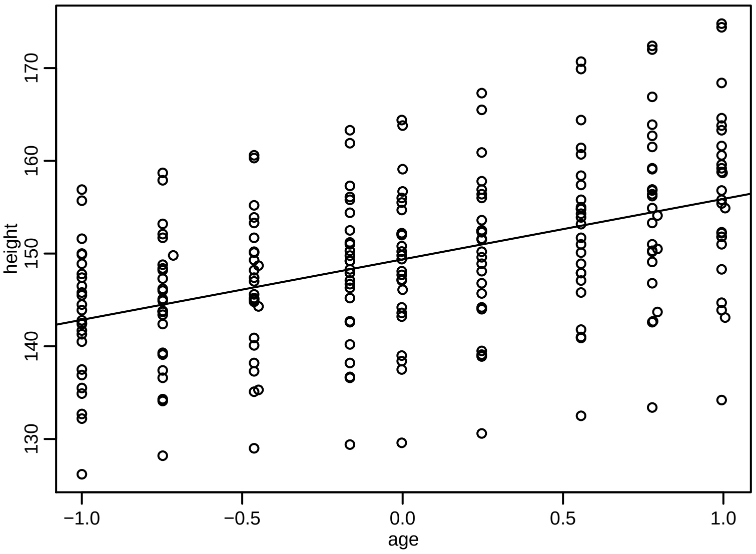
4. Since we are interested in the effect of different machines, we must model them as a fixed effect. (Note that there are few DNA sequencing technologies on the market, and each one has specific effects.)

5. The litter a rat in a behavioural study comes from should be a random effect, because we could have chosen other rats from other litters.

14. **a)** We read in the data set as follows:

```
> oxboys <- read.table("oxboys.csv", header = TRUE, sep = ";")
> oxboys$subject <- factor(oxboys$subject)
> n.boys <- length(levels(oxboys$subject))
```

```
> plot(height ~ age, data = oxboys)
> oxboys.lm <- lm(height ~ age, data = oxboys)
> abline(oxboys.lm)
```

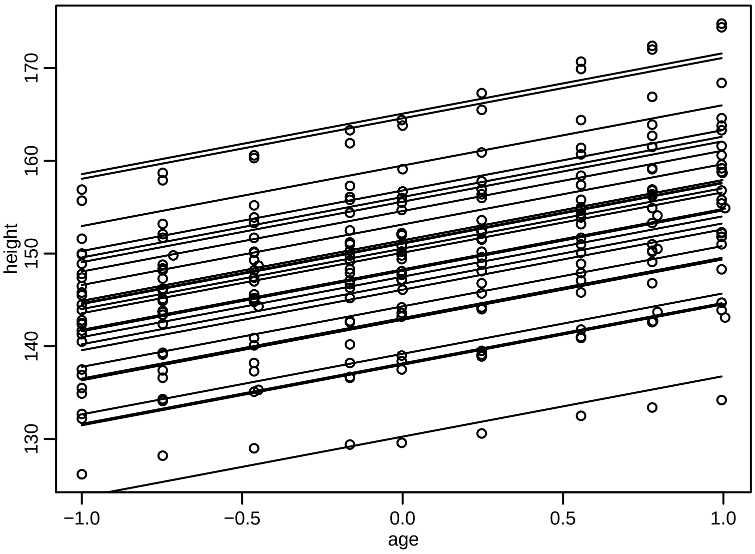


b) The R^2 value can be determined in R with the summary function.

```
> summary(oxboys.lm)

Call:
lm(formula = height ~ age, data = oxboys)

Residuals:
     Min       1Q   Median       3Q      Max
-21.6570  -5.1403   0.4872   4.7514  18.9430

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 149.3718     0.5286 282.599  < 2e-16 ***
age           6.5210     0.8170   7.982 6.64e-14 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.081 on 232 degrees of freedom
Multiple R-squared:  0.2154,        Adjusted R-squared:  0.2121
F-statistic: 63.71 on 1 and 232 DF,  p-value: 6.635e-14
```

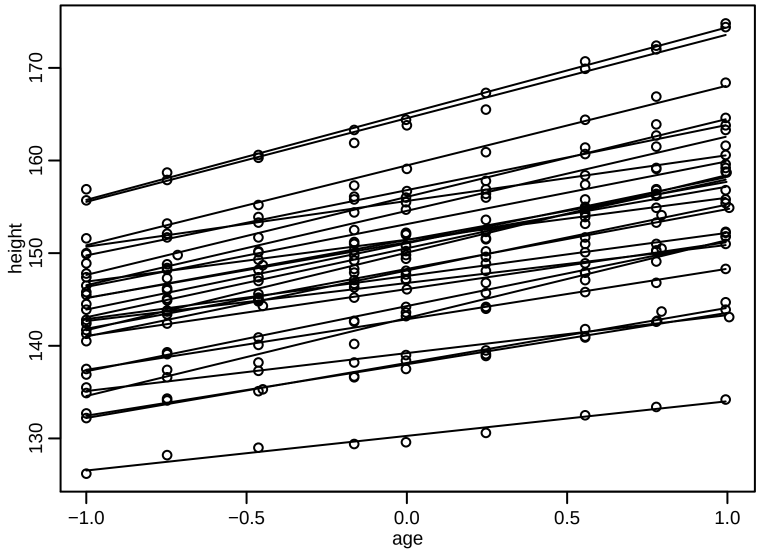
c)

```
> library(lme4)
> oxboys.rand.int <- lmer(height ~ age + (1|subject), data = oxboys)
> plot(height ~ age, data = oxboys)
> for (i in 1:n.boys)
    lines(oxboys$age[oxboys$subject == i],
      fitted(oxboys.rand.int)[oxboys$subject == i])
```



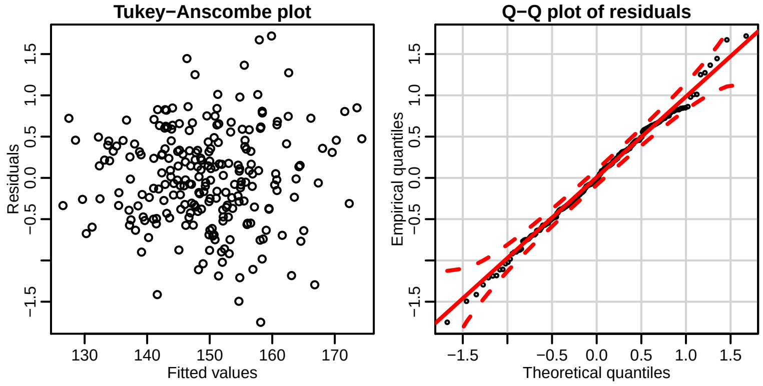
d)

```
> oxboys.rand.int.slp <- lmer(height ~ age + (1 + age|subject), data = oxboys)
> plot(height ~ age, data = oxboys)
> for (i in 1:n.boys)
    lines(oxboys$age[oxboys$subject == i],
      fitted(oxboys.rand.int.slp)[oxboys$subject == i])
```



We can clearly see that this gives a better fit to the data, as we know don't overestimate the smaller datas anymore. It is also better for the bigger datas.

e) As we can see below, the data fits really good.



f) As we have seen in serie 2, be can compute R^2 the following way.

```
> yhat <- fitted(oxboys.rand.int.slp)
> ybar <- mean(oxboys$height)
> (R.sq <- sum((yhat - ybar)^2)/sum((oxboys$height - ybar)^2))
[1] 0.9936777
```

The difference to the value in b) is, that the value in f) is bigger, because we added a random intercept model to the data. This helps to model the mixed effects.

15. **a)** ANOVA models look like

$Y_{ij} = \mu + \alpha_i + \beta_i + E_{ij}$, with α_i, β_i the groupeffect(s) (here: for batches, casks and/or samples)

$$\text{and } E_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2) \text{ for } i = 1, \dots, g, \; j = 1, \dots, n_i$$

Possible models would be:

```
> fit1 <- lm(strength ~ sample, data=Pastes)
> fit2 <- lm(strength ~ batch, data=Pastes)
> fit3 <- lm(strength ~ batch+sample, data=Pastes)
> fit4 <- lm(strength ~ batch+cask, data=Pastes)
```

But there are various problems and limitations: too many variables with very small groupsizes (29 coefficients based on 2 observations each [fit1, fit3]); many missing variables (fit3 especially); coefficients for single factors only (i.e. no knowledge on distribution of strength for two batches simultaneously, or various samples together [fit1, fit2]); finally, these models clearly display the loss of randomness of the variables. For example, ANOVA assumes that casks 'a', 'b' and 'c' are the same casks in each batch (fit4), which they are not. Alltogether these problems show the inappropriateness of fitting an ANOVA model with fixed effects to data where random effects are present and ought to be taken into account.

b) The random group effects of `batch`, `cask` and `sample` on `strength` can be modeled in different combinations: 'batch & cask', 'batch & sample', 'cask & sample'. However, even as a random effect, `cask` remains an impracticle variable to use (casks of different batches may share the same cask label without being the same), so we try 'batch & sample'. It is also worth noting that this experiment has a **nested design**, as first the batch is randomly selected and then from each selected batch the casks are randomly taken. This limits the choice of indices available in the formula for this two-way ANOVA with random effects:

$$Y_{ijk} = \mu + a_i + b_{ij} + E_{ijk}$$

with, for batches: $i = 1, \dots, g$; casks: $j = 1, \dots, n_i$; assay: $k = 1, 2$

a_i: random effect of i-th batch, $a_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_{\text{a}}^2)$

b_{ij}: random effect of j-th cask in i-th batch, $b_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_{\text{b}}^2)$

E_{ijk}: random error, $E_{ijk} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$

and in R

```
> pastes.fit <- lmer(strength ~ 1 + (1|batch) + (1|sample), data = Pastes)
```

c) Looking at confidence intervals for the 'batch & sample' model coefficients (i.e. their standard deviation), there does not appear to be a significant difference in strength between batches since 0 is included in the interval (note: `.sig01` denotes the first variable according to the summary, i.e. `sample` and `.sig02` is `batch`):

```
> set.seed(42)
> confint(pastes.fit)
```

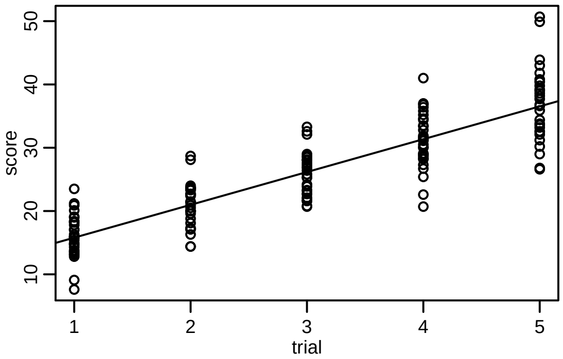
```
                 2.5 %    97.5 %
.sig01        2.1579337  4.053589
.sig02        0.0000000  2.946591
.sigma        0.6520234  1.085448
(Intercept) 58.6636504 61.443016
```

And neither between casks in the 'cask & sample' model (as indeed the coefficient for cask is 0)

```
> pastes.fit2 <- lmer(strength ~ 1 + (1|cask) + (1|sample), data = Pastes)
> set.seed(42)
> confint(pastes.fit2)
```

```
                 2.5 %    97.5 %
.sig01        2.4306465  4.122011
.sig02        0.0000000  1.939822
.sigma        0.6520234  1.085448
(Intercept) 58.8051444 61.301527
```

16. a) Score vs. Trial with added regression line:



b) The linear regression model equation assumes independent and identically distributed errors:

$$\texttt{score}_i = \beta_0 + \beta_1 \cdot \texttt{trial}_i + E_i,\ i = 1, \dots, 150,\ E_1, \dots, E_{150} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$$

However, the 150 trials are really grouped in five trials per subject, making the independency of all errors rather unlikely to hold. So, to improve the model, we could consider `trial` as a random effect by grouping factor `id`. Additionally, how well a subject does in video games, may well be dependent on their age, so inclusion of (an interaction term of) `age` might offer further improvement.

c) This is an example of a **mixed effects model**, including fixed effects for age, trial and the interaction between them, a random intercept for each proband and a random slope w.r.t. trial for each proband. The mathematical formula would be, including the normality assumptions of both random effects and residuals:

$$\texttt{score}_{ij} = \beta_0 + \beta_1 \cdot \texttt{age}_i + \beta_2 \cdot \texttt{trial}_j + \beta_3 \cdot \texttt{(age * trial)}_{ij} + b_{i1} + b_{i2} \cdot \texttt{trial}_{ij} + E_{ij}$$

$$\text{for probands } i = 1, \ldots, 30 \text{ and trials } j = 1, \ldots, 5, \text{ with } \begin{pmatrix} b_{i1} \\ b_{i2} \end{pmatrix} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_b), E_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$$

In R:

```
> library(lme4)
> video.full <- lmer(score ~ age*trial + (1 + trial|id), data = video)
```

d) Step 1: start with full model `video.full`
Step 2: compare full to model without random slope and assess difference in AIC score:

```
> video.rand.int <- lmer(score ~ age*trial + (1|id), data = video)
> AIC(video.rand.int, video.full)
               df      AIC
video.rand.int  6 729.0754
video.full      8 731.8198
```

A very small difference in AIC suggests removing the random slope (this is backed up by the summary-statistics of `video.full` which reveal a correlation coefficient of 1(!) between the random slope and intercept, suggesting one of the two is redundant).
(Alternatively, one can use the Bootstrapping method:

```
> library(pbkrtest)
> PBmodcomp(video.full, video.rand.int)
Parametric bootstrap test; time: 43.44 sec; samples: 1000 extremes: 340;
large : score ~ age * trial + (1 + trial | id)
small : score ~ age * trial + (1 | id)
        stat df p.value
LRT    1.267  2  0.5307
PBtest 1.267     0.3407
```

which also results in a non-significant difference between the two models, suggesting the choice of the simpler model, i.e. `video.rand.int`)
Step 3: compare reduced model (incl. random intercept) to a model without random effects:

```
> video.fixed <- lm(score ~ age*trial, data = video)
> AIC(video.fixed, video.rand.int)
               df      AIC
video.fixed     5 790.9366
video.rand.int  6 729.0754
```

Now the sizable AIC difference suggests retaining `video.rand.int` which includes the random intercept.

e) Step 4: assess the significance of the fixed effects, starting with the interaction term:

```
> video.rand.int.red  <- lmer(score ~ age + trial + (1|id), data = video)
> KRmodcomp (video.rand.int, video.rand.int.red)
F-test with Kenward-Roger approximation; computing time: 0.14 sec.
large : score ~ age * trial + (1 | id)
small : score ~ age + trial + (1 | id)
        stat    ndf    ddf F.scaling   p.value
Ftest 155.14   1.00 119.73         1 < 2.2e-16 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Since the interaction between age and trial is highly significant, it is kept in the model (and consequentially, so are the main effects due to the hierarchical nature of interaction terms). So, our final model is `video.rand.int`, i.e.

$$\texttt{score}_{ij} = \beta_0 + \beta_1 \cdot \texttt{age}_i + \beta_2 \cdot \texttt{trial}_j + \beta_3 \cdot \texttt{(age * trial)}_{ij} + b_{i2} \cdot \texttt{trial}_{ij} + E_{ij}$$

17. **a)**

```
> library(MASS)
> cement.null <- lm(y ~ 1, data = cement)
> cement.fw <- stepAIC(cement.null, scope = ~ x1 + x2 + x3 + x4,
   direction = "forward", trace = 0)
> summary(cement.fw)
Call:
lm(formula = y ~ x4 + x1 + x2, data = cement)

Residuals:
     Min       1Q   Median       3Q      Max
-0.20553 -0.11976  0.01703  0.08520  0.25913

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.582e-16  4.256e-02   0.000   1.0000
x4          -2.632e-01  1.928e-01  -1.365   0.2054
x1           5.677e-01  4.575e-02  12.410 5.78e-07 ***
x2           4.304e-01  1.920e-01   2.242   0.0517 .
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1535 on 9 degrees of freedom
Multiple R-squared:  0.9823,       Adjusted R-squared:  0.9764
F-statistic: 166.8 on 3 and 9 DF,  p-value: 3.323e-08
```

b) 1. We first need to delete the third row in our data set.

```
> cement3 <- cement[-c(3), ]
```

Now we can can repeat our analysis.

```
> cement3.null <- lm(y ~ 1, data = cement3)
> cement3.fw <- stepAIC(cement3.null, scope = ~ x1 + x2 + x3 + x4,
   direction = "forward", trace = 0)
> summary(cement3.fw)
Call:
lm(formula = y ~ x4 + x1 + x3, data = cement3)

Residuals:
     Min       1Q   Median       3Q      Max
-0.18605 -0.12171  0.01736  0.06743  0.22509

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.01664    0.04390   0.379  0.71453
x4          -0.72584    0.04826 -15.040 3.77e-07 ***
x1           0.41051    0.08399   4.888  0.00121 **
x3          -0.18552    0.08187  -2.266  0.05321 .
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1517 on 8 degrees of freedom
Multiple R-squared:  0.9842,       Adjusted R-squared:  0.9782
F-statistic: 165.7 on 3 and 8 DF,  p-value: 1.539e-07
```

2. We now repeat the same for row 10

```
> cement10 <- cement[-c(10), ]
> cement10.null <- lm(y ~ 1, data = cement10)
> cement10.fw <- stepAIC(cement10.null, scope = ~ x1 + x2 + x3 + x4,
   direction = "forward", trace = 0)
> summary(cement10.fw)
Call:
lm(formula = y ~ x2 + x1, data = cement10)

Residuals:
     Min       1Q   Median       3Q      Max
-0.17269 -0.09164 -0.06213  0.08767  0.30454

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.01548    0.04877  -0.317    0.758
x2           0.69564    0.04985  13.955 2.11e-07 ***
x1           0.53311    0.06899   7.727 2.92e-05 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1625 on 9 degrees of freedom
Multiple R-squared:  0.9762,       Adjusted R-squared:  0.9709
F-statistic: 184.7 on 2 and 9 DF,  p-value: 4.931e-08
```

We got now 3 different models by removing different rows from our data set. So we need to think of a better way to find a model for this data set than with the help of the stepAIC function.

c) We get the following results for the difference off $\hat{\beta}_2 - \hat{\beta}_4$. Don't forget that a coefficient is 0 if the corresponding variable doensn't belong to the model.
i) full model: 0.6936
ii) model without row 3: 0.72584
iii) model without row 10: 0.69564

We get 3 times more or less the same value. This is, because x_2 and x_4 are correlated with approximately value -1.

d)

```
> cement.ridge <- lm.ridge(y ~ ., data = cement, lambda = seq(0, 1, by = 0.01))
> which.min(cement.ridge$GCV)
0.32
  33
```

e)

```
> cement.ridge2  <- lm.ridge(y ~ ., data = cement, lambda = which.min(cement.ridge$GCV))
> coef(cement.ridge2)
                          x1            x2            x3
 1.927723e-16  1.588718e-01  1.684314e-01 -1.060207e-01
           x4
-1.738900e-01
> cement3.ridge <- lm.ridge(y ~ ., data = cement3, lambda = which.min(cement.ridge$GCV))
> coef(cement3.ridge)
                     x1          x2          x3          x4
-0.02156153  0.15193007  0.16098364 -0.10053798 -0.16672916
> cement10.ridge <- lm.ridge(y ~ ., data = cement10, lambda = which.min(cement.ridge$GCV))
> coef(cement10.ridge)
                     x1          x2          x3          x4
-0.07069794  0.16858567  0.16432162 -0.08131035 -0.16043173
```

We can see that the coefficients $\beta_1, \beta_2, \beta_3$ and β_4 are more or less the same in all 3 models.

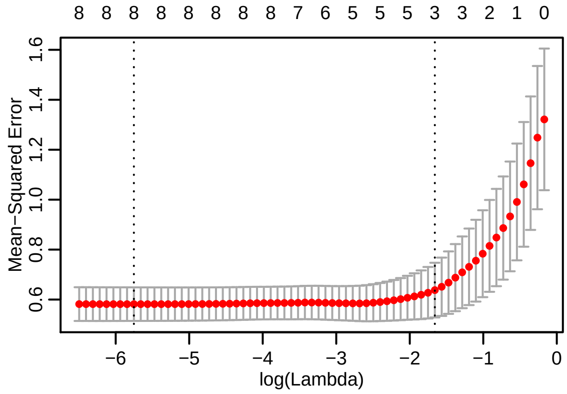
18. a) We start by defining the model matrix and setting a seed for `cv.glmnet`:

```
> library(glmnet)
> X <- as.matrix(Prostate[,-ncol(Prostate)])
> set.seed(7)
```

The command `set.seed(7)` is only needed for reproducibility. This means that we need it to be able to do the same test a second time. Otherwise the random number generator used in `glmnet` would have a different starting point all the time.

We use cross validation implemented in `cv.glmnet` to assess the "optimal" regularization parameter λ:

```
> pro.lasso.cv <- cv.glmnet(X,Prostate$lpsa)
> plot(pro.lasso.cv)
```



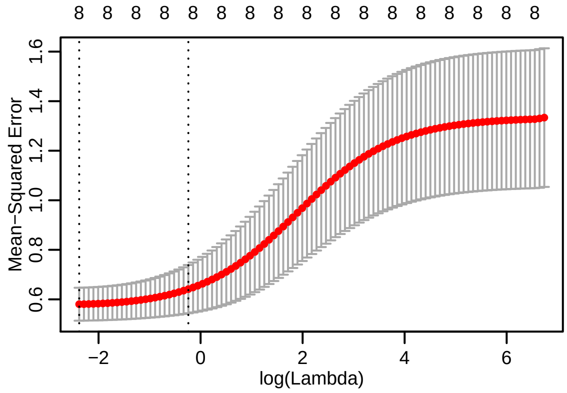
Apparently, the MSE is very flat at its minimum given by `pro.lasso.cv$lambda.min`. Therefore, it is better to choose λ according to the "one standard error-rule": taking the largest lambda for which the MSE is not more than one standard deviation above the minimal one. The corresponding λ can also be found in the object produced by `cv.glmnet`:

```
> pro.lasso.cv$lambda.1se
[1] 0.1903632
> coef(pro.lasso.cv, s = pro.lasso.cv$lambda.1se)
9 x 1 sparse Matrix of class "dgCMatrix"
                    1
(Intercept) 1.0785980
lcavol      0.4720240
lweight     0.1869505
age         .
lbph        .
svi         0.3680476
lcp         .
gleason     .
pgg45       .
```

Apart from the intercept, this model contains three non-zero coefficients: the ones corresponding to `lcavol`, `lweight`, `svi`.

b) We again look at the MSE values as a function of λ:

```
> set.seed(7)
> pro.ridge.cv <- cv.glmnet(X, Prostate$lpsa, alpha = 0)
> plot(pro.ridge.cv)
```



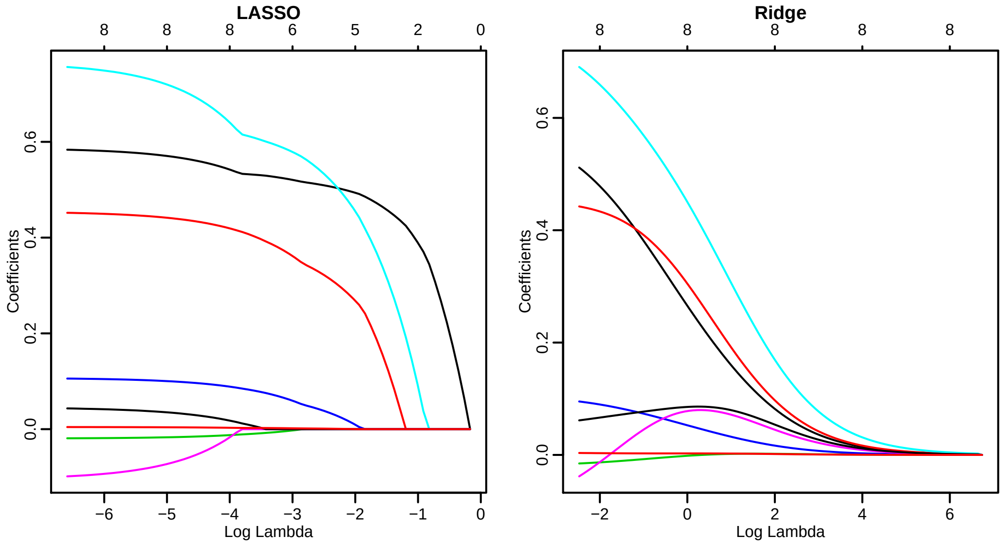
It is again more reasonable to choose λ according to the “one standard error-rule”:

```
> pro.ridge.cv$lambda.1se
[1] 0.7865832
> coef(pro.ridge.cv, s = pro.ridge.cv$lambda.1se)
9 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept)  0.329373785
lcavol       0.293325788
lweight      0.329625420
age         -0.002824404
lbph         0.057882314
svi          0.481514820
lcp          0.073907673
gleason      0.084083801
pgg45        0.002657979
```

Here, no coefficient has been set to zero, as expected for ridge regression.

c)

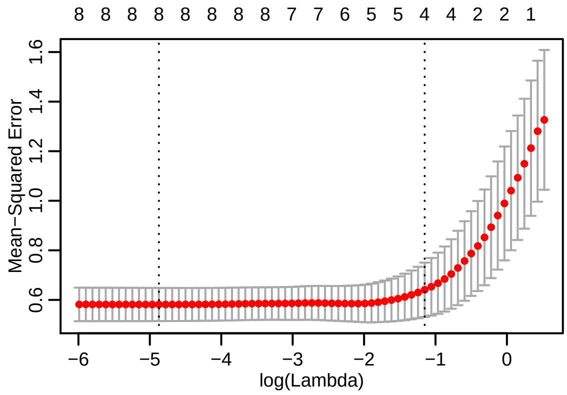
```
> set.seed(7)
> pro.lasso <- glmnet(X, Prostate$lpsa, alpha = 1)
> set.seed(7)
> pro.ridge <- glmnet(X, Prostate$lpsa, alpha = 0)
> par(mfrow = c(1, 2), cex = 0.5, mar = c(3, 3, 4, 1))
> plot(pro.lasso, xvar = "lambda", main = "LASSO")
> plot(pro.ridge, xvar = "lambda", main = "Ridge")
```



One difference is, that the Ridge graph has smooth curves. The curves from the Lasso graph have ”edges”. This has the reason that the once a coefficient gets zero, the other coefficients get adapted. The other difference is the way the coefficients converge to zero. The curves from the ridge curves converges slowly to zero whereas the curves from the lasso graph go straight to zero once they decrease.

d)

```
> set.seed(7)
> pro.net.cv <- cv.glmnet(X, Prostate$lpsa, alpha = 0.5)
> plot(pro.net.cv)
```



For the optimal λ, the elastic net apparently also performs variable selection: a lot of coefficients are put to zero:

```
> pro.net.cv$lambda.1se
[1] 0.3160858
> coef(pro.net.cv, s = pro.net.cv$lambda.1se)
9 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept) 0.9367303269
lcavol      0.4160596154
lweight     0.2366797804
age         .
lbph        .
svi         0.4394358352
lcp         .
gleason     .
pgg45       0.0008333994
```

From the CV plots in a), b) and d), we can already read off that the MSE values are very similar for the λ's chosen by the one standard error-rule (right dotted lines in the plots). We can verify that as follows:

```
> pro.lasso.cv$cvm[pro.lasso.cv$lambda == pro.lasso.cv$lambda.1se]
[1] 0.6378093
> pro.ridge.cv$cvm[pro.ridge.cv$lambda == pro.ridge.cv$lambda.1se]
[1] 0.6413064
> pro.net.cv$cvm[pro.net.cv$lambda == pro.net.cv$lambda.1se]
[1] 0.6406738
```

19. a) The best λ has the smallest leave-one-out cross validation value, which seems to be around $e^{-4} \approx 0.018$ (see right figure) so the optimal range is $[0.005, 0.05]$.

b) The model contains 8 variables:

$$Y = 0.600 - 0.040X_4 + 0.103X_8 + 0.001X_9 + 0.039X_{15} + 0.008X_{17} + 0.112X_{20} - 0.316X_{22} + 0.150X_{25} + \epsilon$$

c) First, those variables with coefficient 0 in the model with a smaller λ (such as $\lambda = 0.018$), will never return in a model using a larger λ (such as $\lambda = 0.05$). So from part (b) we know immediately that X_{14} will not be included in the new model.

Furthermore, the left figure shows that the coefficients of all non-zero variables from (b)'s model go to zero for $\lambda = 0.05$, except for the variable with the maximum and minimum coefficient value. The table from (b) tells us, that these are X_{25} and X_{22} respectively. So the answer is X_{22}, X_{25}.

20. a)

```
> library(glmnet)
> set.seed(40)
> X <- as.matrix(Prostate[,-ncol(Prostate)])
> Y <- Prostate$lpsa
```

From series 8, exercise 2.a followed that the “optimal” regularization parameter $\lambda = 0.19$. Now lets fit the Lasso model

```
> fit.lasso <- glmnet(X, Y, lambda=0.19)
> # lasso-model based on all data points and optimal lambda
> coef(fit.lasso)
```

```
9 x 1 sparse Matrix of class "dgCMatrix"
                   s0
(Intercept) 1.0760234
lcavol      0.4721785
lweight     0.1875648
age         .
lbph        .
svi         0.3686123
lcp         .
gleason     .
pgg45       .
```

and calculate the leave-one-out cross validation value for this λ by hand:

```
> crossval <- function(lambda)
 {
   n <- nrow(Prostate) # find out number of observations
   error <- numeric(n) # create empty vector of length n
  # start the 'for'-loop to do the leave-one-out cross validation
 for (i in 1:n) {
   subX <- X[-i,]
   subY <- Y[-i]
   fit.lasso <- glmnet(subX, subY, lambda=lambda) # lasso-model based on n-1 data points
   Yhat <- predict(fit.lasso, newx=X[i, , drop=FALSE], s=lambda, type="link")
   error[i] <- Yhat - Y[i] # calculate residual
 }
 return (mean(error^2))
 }
> cvr <- cv.glmnet(X,Y, lambda=c(1,0.19), nfolds=nrow(Prostate)) # in-build R version
> # Note: requires at least two lambda values to be tested
>
> list(cvr$cvm, crossval(0.19))
```

```
[[1]]
[1] 1.346356

[[2]]
[1] 0.6159392
```

The mean squared error at the optimal lambda value from the in-build R function and the by-hand function are the same!

b)

```
> library(MASS)
> prostate.null <- lm(Prostate$lpsa ~ 1, data = Prostate)
> prostate.all <- lm(Prostate$lpsa ~ ., data = Prostate) # model including all variables
> prostate.fw <- stepAIC(prostate.null, scope=as.formula(prostate.all),
                         direction = "forward", trace = 0)
> summary(prostate.fw)
```

```
Call:
lm(formula = Prostate$lpsa ~ lcavol + lweight + svi + lbph +
    age, data = Prostate)

Residuals:
     Min       1Q   Median       3Q      Max
-1.83506 -0.39395  0.00412  0.46336  1.57887

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.95102    0.83174   1.143 0.255870
lcavol       0.56561    0.07459   7.583 2.77e-11 ***
lweight      0.42369    0.16687   2.539 0.012815 *
svi          0.72096    0.20902   3.449 0.000854 ***
lbph         0.11184    0.05805   1.927 0.057160 .
age         -0.01489    0.01075  -1.385 0.169527
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7073 on 91 degrees of freedom
Multiple R-squared:  0.6441,        Adjusted R-squared:  0.6245
F-statistic: 32.94 on 5 and 91 DF,  p-value: < 2.2e-16
```

This returns a linear model including the `lcavol, lweight, svi, lbph & age` variables. Now lets rerun the MSE calculations:

```
>     n <- nrow(Prostate)
>     error <- numeric(n)
>     for (i in 1:n) {
        fit.forw <- lm(formula(prostate.fw), data=Prostate, subset = -i)
        Yhat <- predict(fit.forw, newdata=data.frame(Prostate[i,]))
        error[i] <- Yhat - Y[i]
      }
>     mean(error^2)
[1] 0.5430164
>
```

The resulting MSE is only somewhat smaller than that of the Lasso-model, suggesting that collinearity between explanatory variables is not an issue here.

c) Now we expand the Lasso model by including interaction terms. For this new model we find the optimal λ using `cv.glmnet` :

```
> XX <- model.matrix(lpsa ~ .^2, data = Prostate)
> lasso.int.cv <- cv.glmnet(XX,Y)
> lam.int <- lasso.int.cv$lambda.1se
> lam.int
[1] 0.2794947
```

Currently, the optimal $\lambda = 0.27$, with which we fit the model

```
> fit.lasso.int <- glmnet(XX, Y, lambda=lam.int)
> coef(fit.lasso.int)

38 x 1 sparse Matrix of class "dgCMatrix"
                         s0
(Intercept)     1.717755278
(Intercept)     .
lcavol          .
lweight         .
age             .
lbph            .
svi             0.043370204
lcp             .
gleason         .
pgg45           .
lcavol:lweight  0.115273863
lcavol:age      .
lcavol:lbph     .
lcavol:svi      0.065072919
lcavol:lcp      .
lcavol:gleason  .
lcavol:pgg45    .
lweight:age     .
lweight:lbph    .
lweight:svi     .
lweight:lcp     .
lweight:gleason 0.005428533
lweight:pgg45   .
age:lbph        .
age:svi         .
age:lcp         .
```

```
age:gleason     .                        svi:lcp          .
age:pgg45       .                        svi:gleason      .
lbph:svi        .                        svi:pgg45        .
lbph:lcp        .                        lcp:gleason      .
lbph:gleason    .                        lcp:pgg45        .
lbph:pgg45      .                        gleason:pgg45    .
```

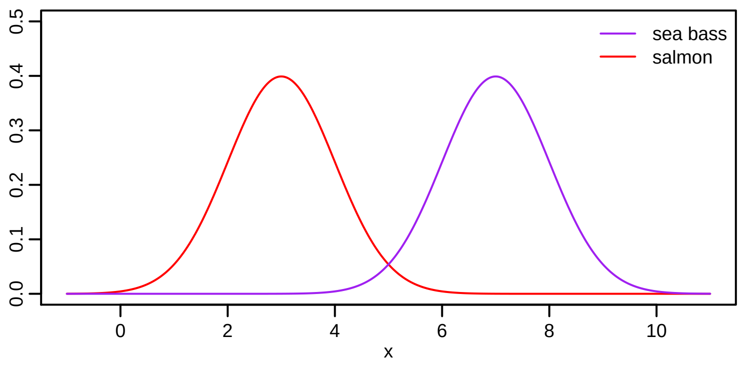
Next we calculate by hand the MSE of this model by building a function dependent only on the value of λ:

```
> crossval.int <- function(lambda)
 {
   n <- nrow(Prostate)
   error <- numeric(n)

 for (i in 1:n) {
   subX <- XX[-i,]
   subY <- Y[-i]
   fit.lasso.int <- glmnet(subXX, subY, lambda=lambda)
   Yhat <- predict(fit.lasso, newx=XX[i, , drop=FALSE], s=lambda, type="link")
   error[i] <- Yhat - Y[i]
 }
 return (mean(error^2))
 }
> crossval(lam.int)
```

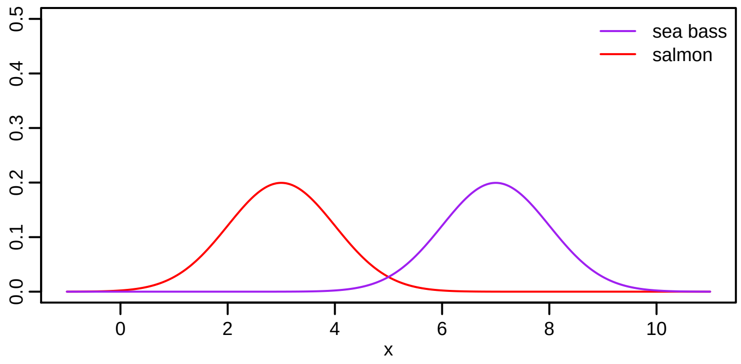
```
[1] 0.6993408
```

Including the interactions in the Lasso-model results in a MSE of 0.6993408, compared to 0.6159392 for the model without interactions. So the interaction model actually performs a bit worse than the one from (a), as the residuals are typically larger.

21. **a)** The class-conditional probability density functions look as follows:

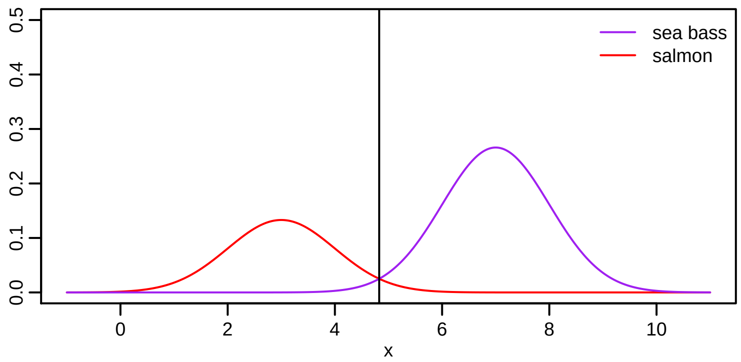


The class-conditional probabilities only differ by their mean. As the prior class probabilities of both fish species are the same, this is also true for the product of prior probability and class-conditional probability density:



The Bayes classifier predicts the species of a fish of lightness x as the one for which the line in the plot above is higher; we hence only need to find the point of intersection of the two lines. Because of symmetry, this is obviously the mid-point between both means, $x = 5$.
A fish with lightness $x > 5$ will hence be classified as “sea bass”, a fish with lightness $x < 5$ as “salmon”.

b) We start with the same plot as in a). The product of prior class probabilities and class-conditional densities now looks as follows:



Unfortunately we now have the problem that we can’t determine the point of intersection by just looking at the picture. So we need to caculate it.

$$\begin{aligned}
\frac{1}{3}\cdot\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{1}{2}\left(\frac{x-3}{\sigma}\right)^2} &= \frac{2}{3}\cdot\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{1}{2}\left(\frac{x-7}{\sigma}\right)^2}\\
e^{-\frac{(x-3)^2}{2}} &= 2e^{-\frac{(x-7)^2}{2}}\\
(x-3)^2 &= 2\cdot\log(2) + (x-7)^2\\
8x &= 40 - 2\cdot\log(2)\\
x &= 4.8267
\end{aligned}$$

Therefore there is only one point of intersection and it is at $x = 4.8267$. We can see that this x-value is slightly smaller than the one from exercise a).

c) Both species of fishes have a given class-conditional probability density function. The only other factor in the Bayes classifier is the prior probability of each species. So if we change the priors, the threshold is going to change aswell. In part a) we had a higher proportion of sea brass. Therefore the density of the sea brass was multiplied with a higher factor and therefore the point of intersection was pushed further to the left. In other words: if sea brass is more frequent overall, the Bayes classifier tends to classify more fishes as sea brass.

22. **a)**

```
> babies <- read.table("baby.dat", header = TRUE)
> babies.full <- glm(Survival ~ ., data = babies, family = "binomial")
> survival.pred <- (predict(babies.full, type = "response") >= 0.5)
> mean(survival.pred != babies$Survival)
```

```
[1] 0.2105263
```

As we have only the values 0 and 1 for the variable Survival, we can say that ”error term” is one if we predict another value than the variable actually has.

b)

```
> library(MASS)
> babies.red <- stepAIC(babies.full, direction = "backward", trace = 0)
> summary(babies.red)
```

```
Call:
glm(formula = Survival ~ Weight + Age + X1.Apgar, family = "binomial",
    data = babies)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4320  -0.7431   0.4180   0.7694   1.9416
```

```
Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -8.4841905  1.8177415  -4.667 3.05e-06 ***
Weight       0.0037911  0.0008449   4.487 7.22e-06 ***
Age          0.1652973  0.0745653   2.217   0.0266 *
X1.Apgar     0.1429887  0.0795671   1.797   0.0723 .
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 319.28  on 246  degrees of freedom
Residual deviance: 236.56  on 243  degrees of freedom
AIC: 244.56

Number of Fisher Scoring iterations: 5
> mean((predict(babies.red, type = "response") >= 0.5) != babies$Survival)
[1] 0.2145749
```

c)

```
> loocv <- function(formula)
 {
   n <- nrow(babies)
   err <- logical(n)
   for (i in 1:n) {
     babies.fit <- glm(formula, data = babies[-i, ], family = "binomial")
     pred <- predict(babies.fit, type = "response", newdata = babies[i, ]) >= 0.5
     err[i] <- pred != babies$Survival[i]
   }
   return(mean(err))
 }
> loocv(formula(babies.red))
[1] 0.2186235
```

d) We can see that the misclassification rate is rising from part a) on to part c), although all numbers are very close. Difference from a) to b): larger models tend to overfit the data. Difference from b) to c): the *actual* misclassification rate tends to be over-optimistic compared to the *expected* misclassification rate estimated by cross validation.

23. We will work with the classifier given in the exercise sheet:

```
> logRegClassifier <- function(formula, data, training, test)
 {
   # Fit the classifier on the training data and get the index of the predicted class label
   fit <- glm(formula, data[training, ], family = "binomial")
   pred.index <- (predict(fit, newdata = data[test, ], type = "response") >= 0.5) + 1

   # Convert the response index to the correct format: a factor as in the data set
   response <- all.vars(formula)[1] # name of the response variable
   factor(pred.index, levels = 1:2, labels = levels(data[[response]]))
 }
```

A k-fold cross validation function could look as follows:

```
> kfoldcv <- function(formula, data, k, classifier)
 {
   response <- all.vars(formula)[1] # name of the response variable
   n <- nrow(data) # find out number of observations
   mis <- numeric(k) # create empty vector of length k
```

```
  # ss: subset index; for data point i, ss[i] denotes the subset
  # the data point belongs to. ss[i] can take values 1, ..., k.
  ss <- rep(1:k, times = ceiling(n/k)) # a bit too long in general...
  ss <- ss[1:n]                    # ... so we cut at the end...
  ss <- sample(ss)                 # ... and randomly shuffle

  # start the 'for'-loop to do the k-fold cross validation
  for (j in 1:k) {
    # Indices of training and test data
    training <- which(ss != j)
    test <- which(ss == j)

    # Misclassification rate on j-th subset
    pred <- classifier(formula, data, training, test)
    mis[j] <- mean(pred != data[ss == j, response])
  }
  return(mean(mis)) # overall misclassification rate
}
```

Test it on the `babies` dataset:

```
> babies <- read.table("baby.dat", header=TRUE)
> babies$Survival <- factor(babies$Survival)
> set.seed(42)
> kfoldcv(Survival ~ ., babies, k=8, classifier=logRegClassifier)

[1] 0.218414

> kfoldcv(Survival ~ ., babies, k=10, classifier=logRegClassifier)

[1] 0.2355

> kfoldcv(Survival ~ ., babies, k=nrow(babies), classifier=logRegClassifier)

[1] 0.2307692
```

The last line performs leave-one-out cross validation: LOOCV is equivalent to n-fold CV, where n is the sample size of the data set (why?).

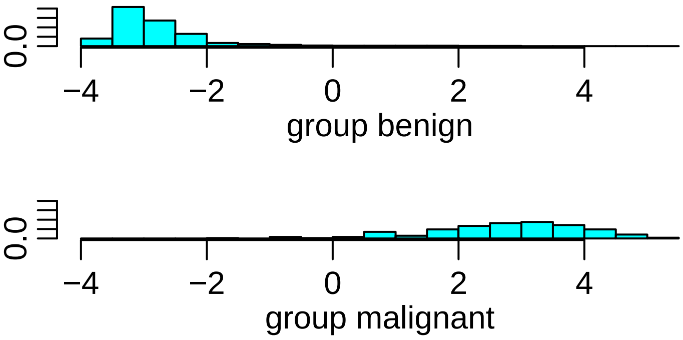
NOTE: different outcomes when rerun due to randomization of the partitioning when you don't call `set.seed`, or when you call `set.seed` with different values!

24. **a)** Let's first manually adjust the dataset to make it more manageable

```
> library(MASS)
> BreastCancer$Id <- NULL
> BreastCancer$Cl.thickness <- as.integer(BreastCancer$Cl.thickness)
> BreastCancer$Cell.size <- as.integer(BreastCancer$Cell.size)
> BreastCancer$Cell.shape <- as.integer(BreastCancer$Cell.shape)
> BreastCancer$Marg.adhesion <- as.integer(BreastCancer$Marg.adhesion)
> BreastCancer$Epith.c.size <- as.integer(BreastCancer$Epith.c.size)
> BreastCancer <- na.omit(BreastCancer) # losing 16 women
```

and now fit an LDA model

```
> bcLDA <- lda(Class~., BreastCancer)
> plot(bcLDA)
```



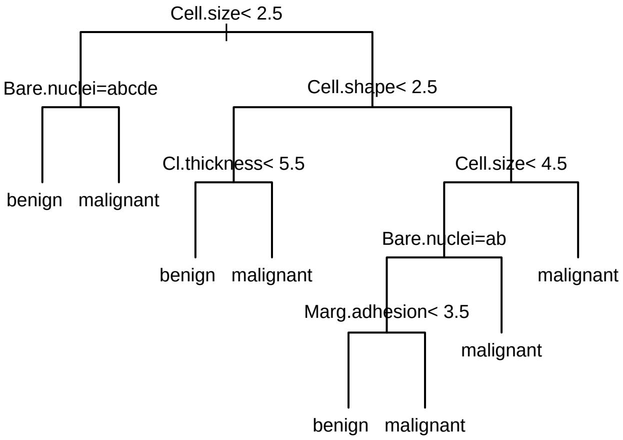
However, with four remaining factorial variables (and 49 levels in total!) it is rather doubtful that normality applies to each class-conditional density. Also, the assumption that the benign and malignant group have the same covariance structure will probably not hold.

b)

```
> set.seed(42)
> kfoldcv(Class ~ ., BreastCancer, k=10, classifier=LDAClassifier)
[1] 0.03655158
```

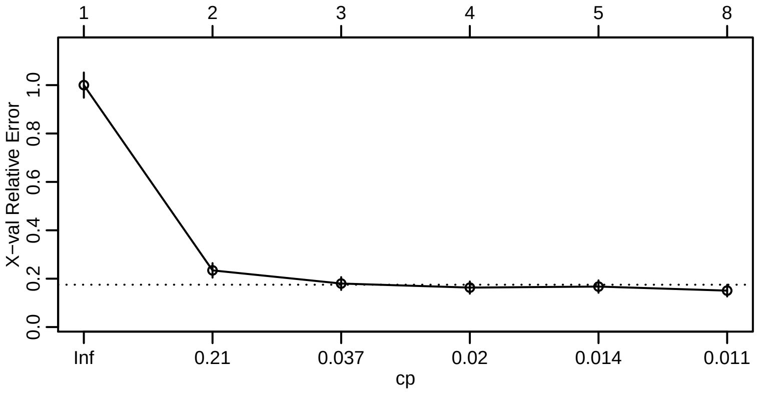
c)

```
> library(rpart)
> set.seed(42)
> fit.xl <- rpart(Class ~ ., BreastCancer, method="class", minsplit=10)
> plot(fit.xl, uniform = TRUE, margin = 0.1)
> text(fit.xl)
```



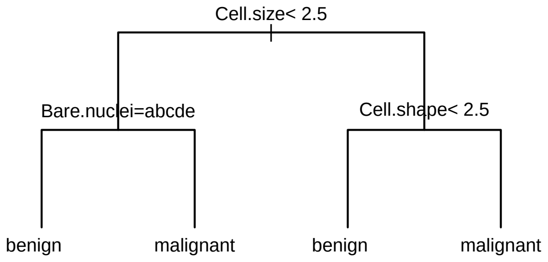
Initially, the overfitted model includes 8 nodes.

```
> plotcp(fit.xl)
```



However, when pruning to the simplest tree with a missclassification rate that is less than 1 standard error away (see ”X-val”-bar in `plotcp`), a tree with 4 nodes is returned.

```
> fit.prune <- prune(fit.xl, cp=0.02)
> plot(fit.prune, uniform = TRUE, margin = 0.2)
> text(fit.prune)
```



This reduced model bases its classification solely on the values of `Cell.size` and `Cell.shape`.

d) We calculate the expected misclassification rate of the CART classifier *without tree pruning*:

```
> set.seed(42)
> kfoldcv(Class ~., BreastCancer, k=10, classifier=CartClassifier)
[1] 0.0556266
```

It is also possible to perform k-fold cross validation using a *manually pruned tree*, although this is a bit more intricate. . . For the sake of completeness, we present a possible solution here:

```
> prunedCartClassifier <- function(formula, data, training, test)
 {
   # Fit the classifier on the training data and get the index of the
   # predicted class label
   fit <- rpart(formula, data[training, ], method="class")
   fit.prune <- prune(fit, cp = 0.02)
   pred.index <- (predict(fit.prune, newdata = data[test, ])[, 2] >= 0.5) + 1

   # Convert the response index to the correct format: a factor as in the data set
   response <- all.vars(formula)[1] # name of the response variable
   factor(pred.index, levels = 1:2, labels = levels(data[[response]]))
 }
> set.seed(42)
> kfoldcv(Class ~ ., BreastCancer, k = 10, classifier = prunedCartClassifier)
[1] 0.05856777
```

Both CART-models result in somewhat higher misclassification rates than the LDA-model — and the manually pruned one is slightly worse than the unpruned tree (although probably not *significantly* worse)!

e)

```
> library(e1071)
> fit <- svm(Class ~ ., data=BreastCancer, scale=FALSE)
> summary(fit)
Call:
svm(formula = Class ~ ., data = BreastCancer, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.02439024

Number of Support Vectors:  97

 ( 39 58 )
```

```
Number of Classes:  2

Levels:
 benign malignant
```

There were 97 support vectors fitted.

```
> fit <- svm(Class~., data=BreastCancer, scale=TRUE)
> summary(fit)
Call:
svm(formula = Class ~ ., data = BreastCancer, scale = TRUE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.02439024

Number of Support Vectors:  94

 ( 47 47 )


Number of Classes:  2

Levels:
 benign malignant
```

Now all the variables are scaled to zero mean and unit variance (i.e. standardized) and this time 94 support vectors were fitted, so the difference between standardizing and not standardizing is small in this example.

f)

```
> svmClassifier <- function(formula, data, training, test)
 {
   fit <- svm(formula, data[training, ], scale=TRUE)
   predict(fit, data[test, ])
 }
> set.seed(42)
> kfoldcv(Class ~ ., BreastCancer, k=10, svmClassifier)
[1] 0.02926257
```

The SVM-classifier seems equivalent to the LDA, typically somewhat smaller, and both are a little better than the CART.

25. a) Assay 1: for the full tree, we start with mgo=2.5. This is smaller as 2.695 so we head to the left to float. There we have a al2o3 value of 1.5. This is bigger than 1.42 and we head to the right to nonfloat. As the ri value of 1.6 is bigger than 1.517 we end up at nonfloat.

For the pruned tree we start again with the mgo value of 2.5. This is smaller as 2.695 and we head left to float. The al2o3 value is 1.5 and therefore is bigger than 1.42. Therefore we head to the right to nonfloat.

Assay 2: from the trees in the plot, we see that the assay cannot be classified by the full tree since the refractive index (variable `ri`) is missing. The pruned tree assigns the sample to class `nonfloat`.

b) Assay 1: The logistic regression models for $Z_{\cdot j}$ are already done. So we need to find out for which j the fitted posterior class probability $\hat{\pi}_j(x_1, \ldots, x_p)$ will be maximal. Remember the logistic regression model (see the slides):

$$\log\left(\frac{\hat{\pi}_j(x_1,\ldots,x_p)}{1-\hat{\pi}_j(x_1,\ldots,x_p)}\right) = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \ldots + \hat{\beta}_p x_p$$

Since the logistic transformation

$$\pi_j \mapsto \frac{\pi_j}{1-\pi_j}$$

is monotonely increasing, the glass type j with the maximal posterior probability $\hat{\pi}_j(\ldots)$ is the one for which the *right-hand side* of the logistic regression model is maximized, i.e. the scalar product

$$\hat{\beta}_0 + \hat{\beta}_1 x_1 + \ldots + \hat{\beta}_p x_p \ .$$

We calculate that product for the three glass types:

- `float` ($j = 1$):

$$\begin{aligned}\hat{\beta}_0 + \hat{\beta}_1 x_1 + \ldots + \hat{\beta}_p x_p &= 444.2 - 524.3 \cdot 1.6 + 3.2 \cdot 13.0 + 6.1 \cdot 2.5 - 0.5 \cdot 1.5 \\ &\quad + 3.4 \cdot 70.0 + 3.0 \cdot 0.5 + 4.8 \cdot 8.0 + 4.5 \cdot 0.2 - 2.3 \cdot 0.1 \\ &= -60.7\end{aligned}$$

- `nonfloat` ($j = 2$): $\hat{\beta}_0 + \hat{\beta}_1 x_1 + \ldots + \hat{\beta}_p x_p = 28.4$
- `nonwindow` ($j = 3$): $\hat{\beta}_0 + \hat{\beta}_1 x_1 + \ldots + \hat{\beta}_p x_p = 43.1$

The biggest number is 28.4 for $j = 2$; therefore we would predict `nonfloat`.

Assay 2: As there are missing values for assay 2, we can not do the algorithm. A prediction is not possible.