TMA4300 Computer Intensive Statistical Methods

Exercise3, Spring 2014

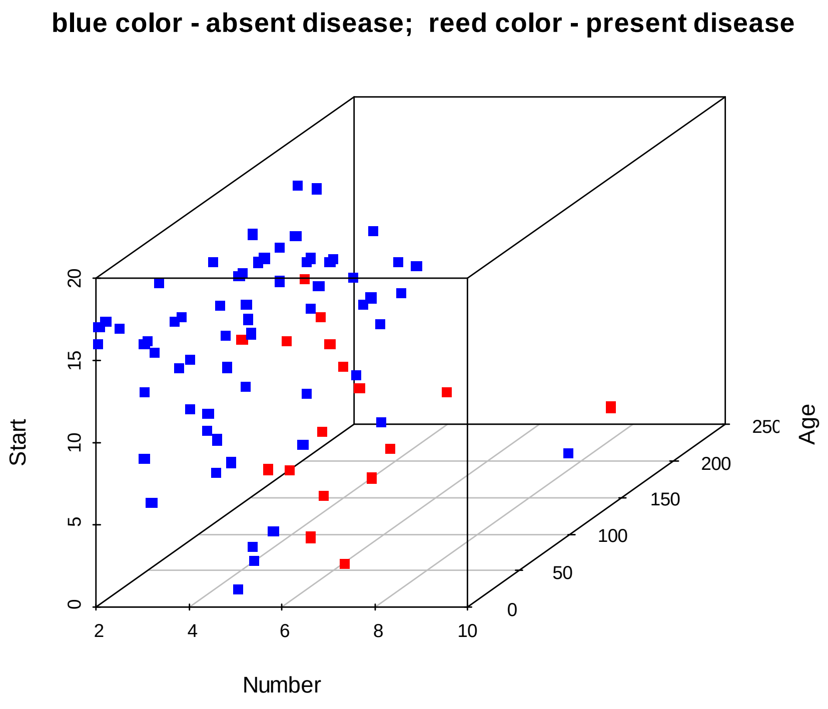
Mateusz Samiec

Problem A: Classification and cross validation

In this problem under our consideration is the data `kyphosis` that contains two groups of individuals: diseased and un-diseased. Each individual has three predictors: age, number and start.

1. Linear and quadratic discriminant models to the `kyphosis` data set are derived thanks to `lda` and `qda` functions, that are available in the library `MASS`. To have some view into the data, 3d plot and prior distribution are enclosed bellow.

```
data(kyphosis)
table(kyphosis[,1])/length(kyphosis[,1])
# Piror distributions
   absent   present
0.7901235 0.2098765
library(scatterplot3d)
scatterplot3d(kyphosis[, 3], kyphosis[, 2], kyphosis[, 4],
color = c("blue", "red")[kyphosis$Kyphosis], pch = 15,
main="blue color - absent disease;  reed color - present
disease", xlab="Number", ylab="Age", zlab="Start")
```



In order to check how precise are the models, the misclassification rates have to be estimated. For this reason function performing K-fold cross validation test was implemented. Its' code is as follows.

```
Kfold_CV <- function(K){
   tot = c(0, 0)
   names(tot) <- c("lda", "qda")
   n = length(kyphosis$Kyphosis)
   train <- seq(1:n)
   T = n %/% K
   rest = n %% K
   for(i in 1:T){
      lda(Kyphosis ~ ., kyphosis,
      subset = train[-(((i-1)*10+1):i*10)]) -> zlda
      qda(Kyphosis ~ ., kyphosis,
      subset = train[-(((i-1)*10+1):i*10)]) -> zqda
      for(j in 1:10){
         if(predict(zlda, kyphosis[(i-1)*10+j,])$class
         != kyphosis$Kyphosis[(i-1)*10+j]){
            tot[1] = tot[1] + 1
         }
         if(predict(zqda, kyphosis[(i-1)*10+j,])$class
         != kyphosis$Kyphosis[(i-1)*10+j]){
            tot[2] = tot[2] + 1
         }
      }
   }
   if(rest != 0){
      lda(Kyphosis ~ ., kyphosis,
      subset = train[-((T*10+1):(T*10+rest))]) -> zlda
      qda(Kyphosis ~ ., kyphosis,
      subset = train[-((T*10+1):(T*10+rest))]) -> zqda
      for(i in 1:rest){
         if(predict(zlda, kyphosis[T*10+i,])$class
         != kyphosis$Kyphosis[T*10+i]){
            tot[1] = tot[1] + 1
         }
         if(predict(zqda, kyphosis[T*10+i,])$class
         != kyphosis$Kyphosis[T*10+i]){
            tot[2] = tot[2] + 1
         }
      }
   }
   tot = tot / n
   tot
}
```

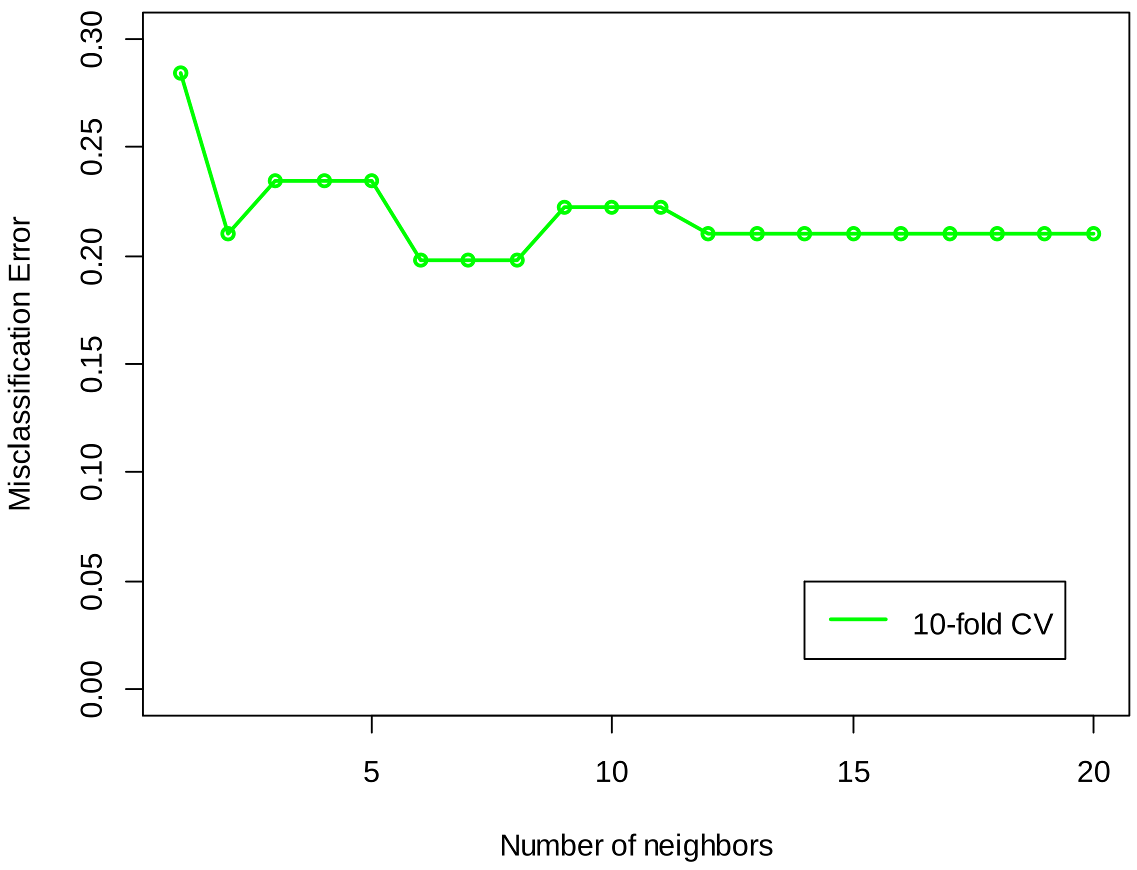
Because for 10-fold cross validation, linear and quadratic models seem to have misclassification rates on the same levels, I decided to check as well the estimates using 11-fold cross validation. As we see for this particular case misclassification rate is smaller for quadratic model.

```
Kfold_CV(10) # Misclassification rates for K=10
      lda       qda
0.1604938 0.1604938
Kfold_CV(11) # Misclassification rates for K=11
      lda       qda
0.1728395 0.1604938
```

2. A k-nearest neighbor classifier to the `kyphosis` data set is construct bellow. I is used to find the best value for k by 10-fold cross validation.

```
KNN <- function(K, neighbours){
   n = length(kyphosis$Kyphosis)
   T = n %/% K
   rest = n %% K
   tot = 0
   for(i in 1:T){
      test <- cbind(kyphosis[((i-1)*K+1):(i*K),2],
                    kyphosis[((i-1)*K+1):(i*K),3],
                    kyphosis[((i-1)*K+1):(i*K),4])
      train <- cbind(kyphosis[-(((i-1)*K+1):(i*K)),2],
                     kyphosis[-(((i-1)*K+1):(i*K)),3],
                     kyphosis[-(((i-1)*K+1):(i*K)),4])
      control_data <- knn(train, test,
            kyphosis[-(((i-1)*K+1):(i*K)),1], k = neighbours)
      for(j in 1:K){
         if(control_data[j] != kyphosis$Kyphosis[(i-1)*K+j]){
            tot = tot + 1
         }
      }
   }
   if(rest != 0){
      test <- cbind(kyphosis[(T*K+1):(T*K+rest),2],
                    kyphosis[(T*K+1):(T*K+rest),3],
                    kyphosis[(T*K+1):(T*K+rest),4])
      train <- cbind(kyphosis[-((T*K+1):(T*K+rest)),2],
                     kyphosis[-((T*K+1):(T*K+rest)),3],
                     kyphosis[-((T*K+1):(T*K+rest)),4])
      control_data <- knn(train, test,
           kyphosis[-((T*K+1):(T*K+rest)),1], k = neighbours)
      for(i in 1:rest){
         if(control_data[i] != kyphosis$Kyphosis[T*K+i]){
            tot = tot + 1
         }
      }
   }
   tot = tot / n
   tot
}
# Misclassification rates as a function of knn
ten_fold_cv <- rep(0,20)
for(i in 1:20){
   ten_fold_cv[i] <- KNN(10, i)
}
# Plotting routine
plot(ten_fold_cv, type = "o", col = "green", lwd = 2,
     ylim = c(0,0.3), xlab = "Number of neighbors", ylab =
"Misclassification Error")
legend(14, 0.05, col ="green", lty = 1,lwd=2, "10-fold CV")
```

For such a data, misclassification as a function of k-nearest neighbors has the following shape. On the basis of this experiment the optimal value of k maight be chosen. In this example it would be 6, 7 or 8.



Problem B: Comparing AR(2) parameter estimators using resmapling of residuals

1. In order to evaluate the relative performance of the two parameter estimators the resampling bootstrap method was used. This method contains other sub methods that were uploaded on the homework web side.

```
source("probBhelp.R")
source("probBdata.R")
residualSampling <- function(data, B){
   betas <- matrix(nrow = B, ncol = 4)
   for(i in 1:B){
      ARp.beta.est(data, 2)$LS -> betaLS
      ARp.beta.est(data, 2)$LA -> betaLA
      ARp.resid(data, betaLS) -> e.observedLS
      ARp.resid(data, betaLA) -> e.observedLA
      sample(e.observedLS, size=100, replace=TRUE) -> eLS
      sample(e.observedLA, size=100, replace=TRUE) -> eLA
      idx = sample(1:99, 1)
      x0 <- c(data[idx], data[idx+1])
      ARp.filter(x0, betaLS, eLS) -> xLS
      ARp.filter(x0, betaLA, eLA) -> xLA
      ARp.beta.est(xLS, 2)$LS -> betaLS
      ARp.beta.est(xLA, 2)$LA -> betaLA
      betas[i, 1] = betaLS[1]
      betas[i, 2] = betaLS[2]
      betas[i, 3] = betaLA[1]
      betas[i, 4] = betaLA[2]
   }
   betas
}
```

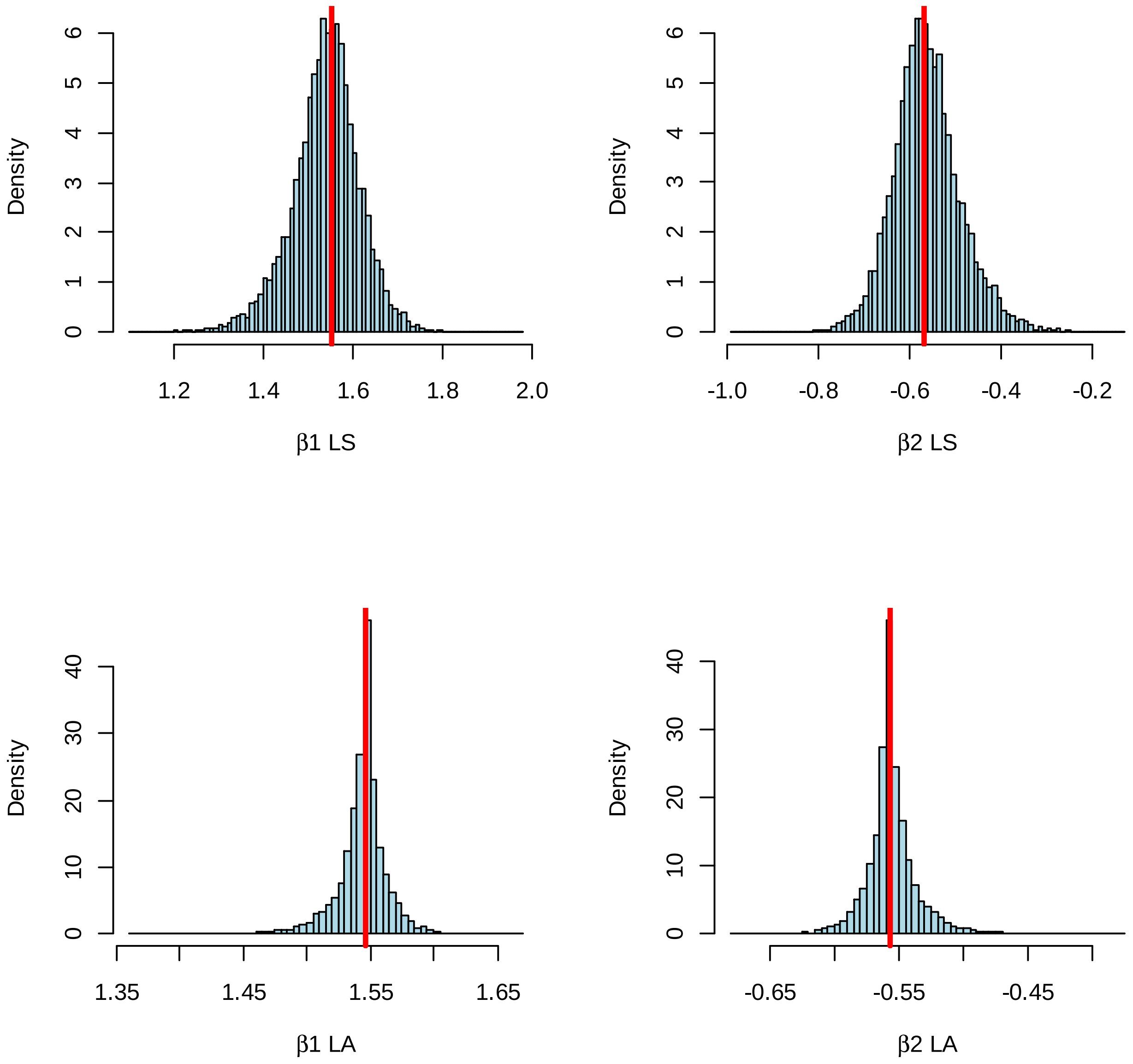
To obtain good estimator 10000 bootstrap samples were performed. The code that return the statistics together witch the plots are enclosed bellow.

```
ARp.beta.est(data3A$x, 2)$LS -> betaLS_Original
ARp.beta.est(data3A$x, 2)$LA -> betaLA_Original
set.seed(9925)
residualSampling(data3A$x, 10000, betaLS_Original,
betaLA_Original) -> betas
# Plotting routine
par(mfrow=c(2,2))
truehist(betas[,1], prob=TRUE, ylab="Density",
xlab=expression(beta*"1 LS"), col="lightblue")
abline(v=betaLS_Original[1], col=2, lwd=3)

truehist(betas[,2], prob=TRUE, ylab="Density",
xlab=expression(beta*"2 LS"), col="lightblue")
abline(v=betaLS_Original[2], col=2, lwd=3)

truehist(betas[,3], prob=TRUE, ylab="Density",
xlab=expression(beta*"1 LA"), col="lightblue")
abline(v=betaLA_Original[1], col=2, lwd=3)

truehist(betas[,4], prob=TRUE, ylab="Density",
xlab=expression(beta*"2 LA"), col="lightblue")
abline(v=betaLA_Original[2], col=2, lwd=3)
```



Above plots give initial intuition about the estimators. The (LA) estimator seem to be better than (LS) estimator. To validate this assumption, variance and bias of the estimators are estimated. Results are enclosed on the next page.

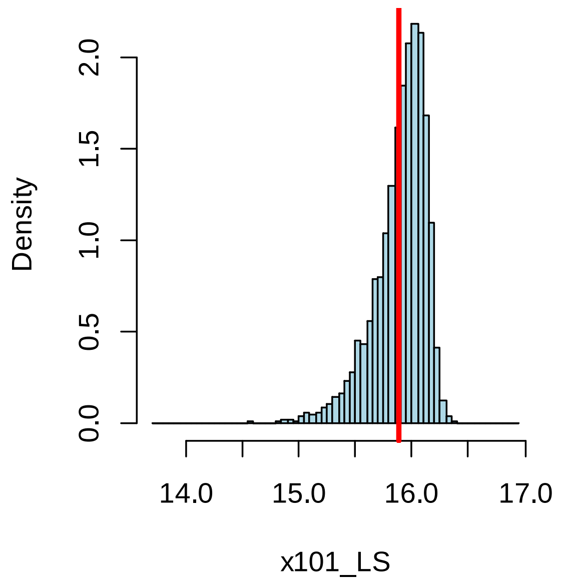
```
> mean(betas[,1] - betaLS_Original[1])
[1] -0.01273881      # LS bias[beta1]
> mean(betas[,2] - betaLS_Original[2])
[1] 0.006697618      # LS bias[beta2]
> mean(betas[,3] - betaLA_Original[1])
[1] -0.002424196     # LA bias[beta1]
> mean(betas[,4] - betaLA_Original[2])
[1] 0.001867013      # LA bias[beta2]
> var(betas[,1])
[1] 0.00573468       # LS Var[beta1]
> var(betas[,2])
[1] 0.005589221      # LS Var[beta2]
> var(betas[,3])
[1] 0.0003909377     # LA Var[beta1]
> var(betas[,4])
[1] 0.0003869638     # LA Var[beta2]
```

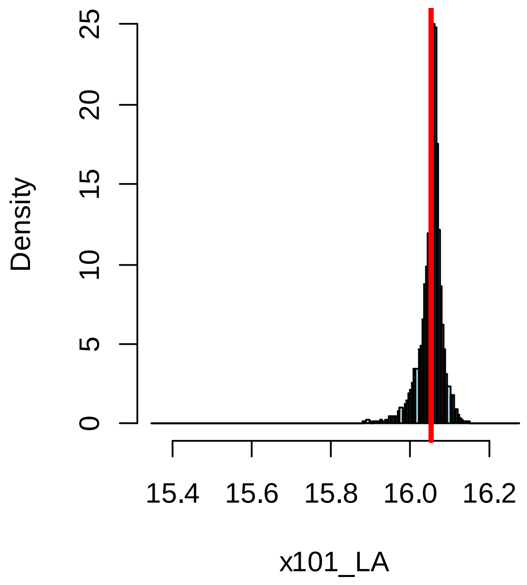
The results confirm the assumption that (LA) is a better estimator than (LS) in this particular case. Therefore we can not say that the (LS) optimal for this problem.

2. In order to compute the prediction intervals for x_{101} some preliminary steps have to be performed. First, using the corresponding parameter estimates obtained in the previous part, predictions of a value x_{101} for each bootstrap iteration have to be computed. Than confidence interval might be derived based on the 95% guantile. The code and distribution of the predictions are enclosed bellow.

```
x101LS <- rep(0, 10000)
x101LA <- rep(0, 10000)
for(i in 1:10000){
   x101LS[i] =
betas[i,1]*data3A$x[99]+betas[i,2]*data3A$x[100]
   x101LA[i] =
betas[i,3]*data3A$x[99]+betas[i,4]*data3A$x[100]
}
# Plotting routine
par(mfrow=c(1,2))
truehist(x101LS, prob=TRUE, ylab="Density",
xlab="x101_LS", col="lightblue")
abline(v=mean(x101LS), col=2, lwd=3)

truehist(x101LA, prob=TRUE, ylab="Density",
xlab="x101_LA", col="lightblue")
abline(v=mean(x101LA), col=2, lwd=3)
```





Once more the analysis reveal the advantage of using (LA) estimator in spite of (LS) one. 95% prediction intervals for x_{101} based on both estimators are enclosed bellow.

```
# Prediction intervals based on (LS) estimator
> quantile(x101LS,  probs = c(0.025, 0.975))
    2.5%    97.5%
15.29008 16.20942
# Prediction intervals based on (LA) estimator
> quantile(x101LA,  probs = c(0.025, 0.975))
    2.5%    97.5%
15.96243 16.10723
```

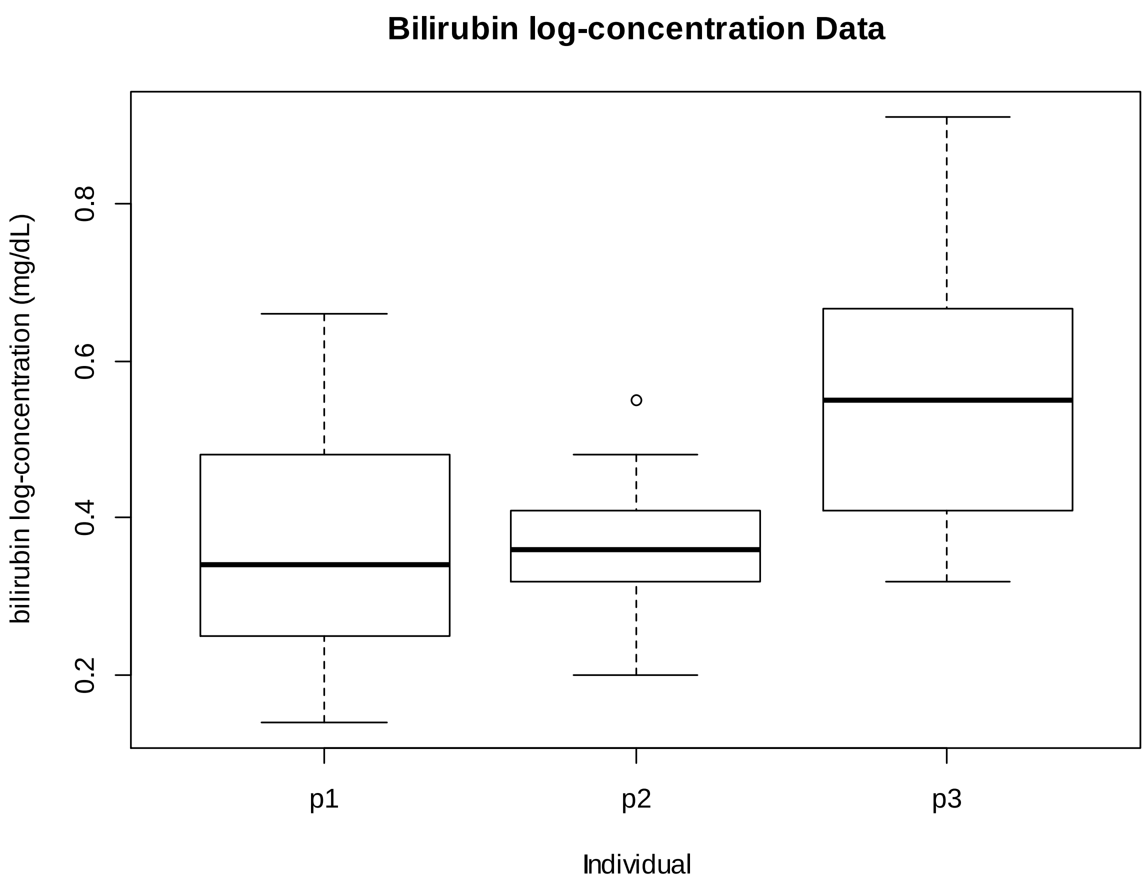
Problem C: Permutation test

In the following problem the F-statistic is employed to perform a permutation test. Under the consideration is data of bilirubin concentration in blood samples taken from three individuals.

1. In order to inspect the logarithms of the concentrations for each individual the following code is used.

```
bilirubin <- read.table("bilirubin.txt",header=T)
boxplot(meas~pers,data=bilirubin,   main="Bilirubin   log-
concentration  Data",  xlab="Individual",  ylab="bilirubin
log-concentration (mg/dL)")
```

   The Box plot reveal the differences between the log distributions of bilirubin concentration for each individualist. The first and second individualist seem to have quite similar median of the log-concentration. When it comes to the third box-plot, it is shifted up respectively to the first two distributions.



2. In order to check if the measured outcomes does not depend on individualist some steps have to be performed. The first one is to find the model describing our data. Mentioned model has the following representation:

$$\log Y_{ij} = \beta_i + \epsilon_{ij}, \quad \text{with } i = 1,2,3 \text{ and } j = 1,\dots,n_i$$

where $n_1 = 11, n_2 = 10, n_3 = 8,$ and $\epsilon_{ij} \sim N(0,\sigma^2)$.

To fit the regression model, following code was used.

```
x1 <- c(rep(1, 11), rep(0, 18))
x2 <- c(rep(0, 11), rep(1, 10), rep(0, 8))
x3 <- c(rep(0, 21), rep(1, 8))
summary(lm(log(bilirubin[,1])~x1+x2+x3-1)) -> fit
fit$fstatistic[1] -> Fval
fit
```

```
Call:
lm(formula = log(bilirubin[, 1]) ~ x1 + x2 + x3 - 1)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.87215 -0.26246  0.03131  0.20236  0.67844 

Coefficients:
   Estimate Std. Error t value Pr(>|t|)
x1  -1.0940     0.1175  -9.312 9.15e-10 ***
x2  -1.0298     0.1232  -8.359 7.70e-09 ***
x3  -0.6291     0.1377  -4.567 0.000105 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3896 on 26 degrees of freedom
Multiple R-squared:  0.8722,    Adjusted R-squared:  0.8575 
F-statistic: 59.15 on 3 and 26 DF,  p-value: 9.526e-12
```

$\text{Fval} = 59.15$ is F-statistic for original data.

3. The next step that is needed to valid $H_0: \beta_1 = \beta_2 = \beta_3$ is permutation test that generates permutation of the data between the three individuals, consequently fits the model given in previous point and finally returns the value of the F-statistic.

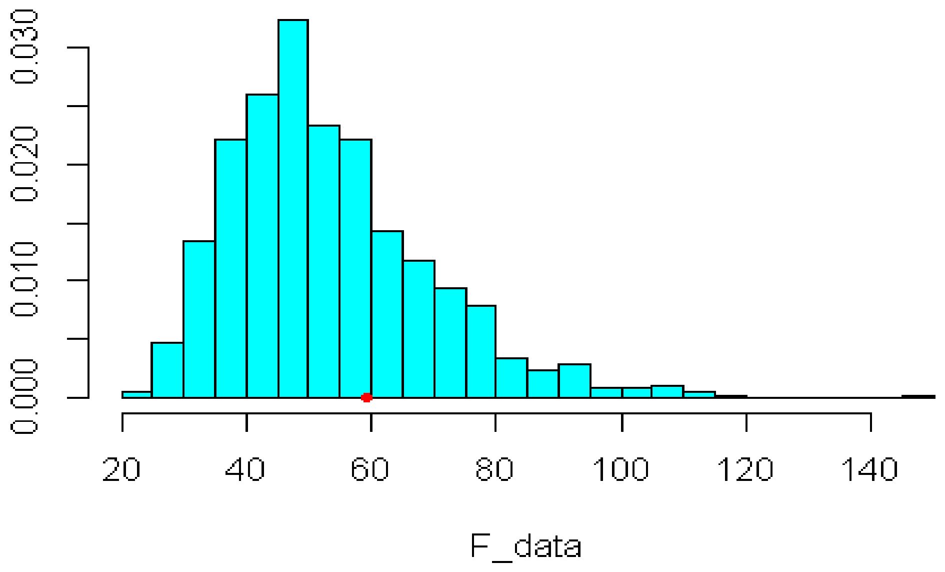
R implementation of that function is enclosed bellow.

```
      permTest <- function(){
         N = length(bilirubin[,1])
         sample(1:N, N, replace = F) -> idx
         y <- bilirubin[idx,1]
         summary(lm(log(y)~x1+x2+x3-1))$fstatistic[1]
}
```

4. The last step is the performance of a permutation test using the function `permTest` to generate a sample of size 999 for the F-statistic. At this stage the `Fval`, which was computed in the 2nd point, is needed to derive p-value for the sample.
   Representation of the R code enclosed is enclosed bellow.

```
B = 999
tot = 0
F_data = rep(0, B)
for(i in 1:B){
   F <- permTest()
   if ( F > Fval)
   {
      tot = tot + 1
   }
   F_data[i] = F
}
tot = tot/B
tot
# p-value based on a test quantity F(x)
[1] 0.2892893
# plotting routine
library(MASS)
truehist(F_data)
points(Fval,0,col=2, pch=20)
```

To have a better insight in to the results of the permutation test additional plot was used. It puts together all values of the F-statistic (for permuted samples) together witch Fval (red point).



We observed that p-value is greater than 0.05. Thus $H_0: \beta_1 = \beta_2 = \beta_3$ is not rejected.

Problem D: EM-algorithm

Under the consideration is two-way table of y_{ij} for $i = 1{,}2$ and $j = 1{,}2{,}3$ with one missing value y_{22}:

5	8	7
10	●	12

As in the lecture a linear model:

$$y_{ij} = \mu + \alpha_i + \beta_j + \epsilon_{ij}$$

where $\sum_i \alpha_i = \sum_j \beta_j = 0$ and $\epsilon_{ij} \sim N(0, \sigma^2)$.

1. An R function that impute the missing value using the EM-algorithm consists of several steps:

$$1)\ \hat{\mu} = \bar{y}$$
$$2)\ \widehat{\alpha_1} = \overline{y_{1,}} - \bar{y}$$
$$3)\ \widehat{\beta_1} = \overline{y_{,1}} - \bar{y}$$
$$4)\ \widehat{\beta_3} = \overline{y_{,3}} - \bar{y}$$
$$5)\ \widehat{\alpha_2} = -\widehat{\alpha_1}$$
$$6)\ \widehat{\beta_2} = -\widehat{\beta_1} - \widehat{\beta_3}$$
$$7)\ \widehat{y_{22}} = \bar{\mu} + \widehat{\alpha_2} + \widehat{\beta_2}$$

8) Come back to the point 1

R code representation of the algorithm is similar. The extra part is the condition under which algorithm is continued if difference of updated y_{22} is larger than 1e-5.

```
# EM-algorithm
y <- c(5, 10, 8, 0, 7, 12)
hist <- c(0, 0, 0, 0, 0, 0, 0)
mu <- mean(y) * 6 / 5
y22 <- 1
while(abs(y[4] - y22) > 1e-5){  # Condition (dy22 < 1e-5)
   alpha1 <- (y[1] + y[3] + y[5]) / 3 - mu
   beta1 <- (y[1] + y[2]) / 2 - mu
   beta3 <- (y[5] + y[6]) / 2 - mu
   alpha2 <- -alpha1
   beta2 <- -beta1 - beta3
   y22 <- y[4]
   y[4] <- mu +alpha2 + beta2
   rbind(hist, c(mu, alpha1, alpha2, beta1, beta2, beta3,
   y[4])) -> hist
   mu <- mean(y)
}
```

```
# The last 8 computed values of the parameters
hist[25:length(hist[,1]), 1:6]
[,1]       [,2]      [,3]       [,4]      [,5]       [,6]
 9.166598 -2.499932 2.499932 -1.666598 1.333197 0.3334016
 9.166621 -2.499954 2.499954 -1.666621 1.333242 0.3333789
 9.166636 -2.499970 2.499970 -1.666636 1.333273 0.3333637
 9.166646 -2.499980 2.499980 -1.666646 1.333293 0.3333536
 9.166653 -2.499987 2.499987 -1.666653 1.333306 0.3333468
 9.166658 -2.499991 2.499991 -1.666658 1.333315 0.3333423
 9.166661 -2.499994 2.499994 -1.666661 1.333321 0.3333393
 9.166663 -2.499996 2.499996 -1.666663 1.333325 0.3333373
# The last 8 computed values of y22
hist[25:length(hist[,1]),7]
12.99973  12.99982  12.99988  12.99992  12.99995  12.99996
12.99998 12.99998
```

In order to validate the results we use the least square estimator.

$$\mathrm{A} = (X^T X)^{-1} X^T Y$$

where:

$$Y = \begin{pmatrix} y_{11} \\ y_{21} \\ y_{12} \\ y_{13} \\ y_{23} \end{pmatrix} = \begin{pmatrix} 5 \\ 10 \\ 8 \\ 7 \\ 12 \end{pmatrix}, \qquad X = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & -1 & 1 & 0 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 0 & 1 \\ 1 & -1 & 0 & 1 \end{pmatrix}$$

Code representation of the method is as follows.

```
# Validation of the EM-algorithm
Y = matrix(c(5, 10, 8, 7, 12), ncol = 1)
X = matrix(c(1, 1, 1, 1, 1,
             1,-1, 1, 1,-1,
             1, 1,-1, 0, 0,
             0, 0,-1, 1, 1), nrow = 5)
solve(t(X)%*%X)%*%(t(X)%*%Y)
          [,1]
 [1,]  9.1666667
 [2,] -2.5000000
 [3,] -1.6666667
 [4,]  0.3333333
```

As we see the values are the same.

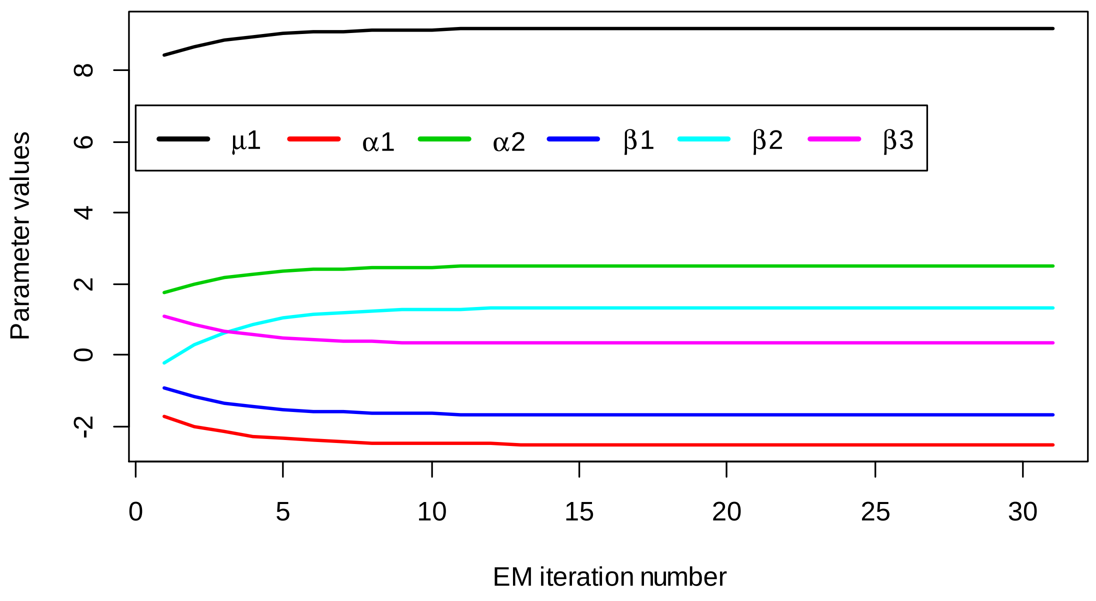
Estimated variance (unbiased estimator):

$$\hat{\sigma}^2 = \frac{1}{4}\sum_{ij}\left(y_{ij} - \hat{\mu} - \widehat{\alpha_\iota} - \widehat{\beta_J}\right)^2$$

Substituting known values into the above equation, estimation of variance is derived. It is found to be zero. Thus ϵ_{ij} are iid random variables with variance equal zero $\hat{\sigma}^2 = 0$.

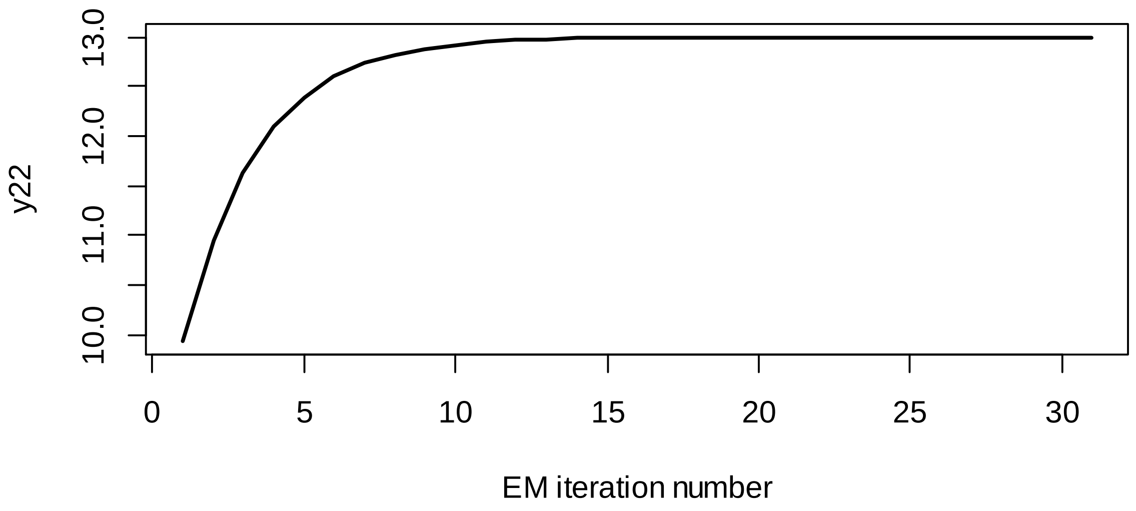
2. Plots o the estimated parameters $\mu, \alpha_1, \alpha_2, \beta_1, \beta_2, \beta_3$ and the value of y_{22} as the function of the iteration number are enclosed bellow together with the code.

```
# Plotting routine (parameters values)
yrange<-range(c(hist[,1],  hist[,2],  hist[,3],  hist[,4],
hist[,5], hist[,6]))
plot(hist[2:length(hist[,1]),  1],  type="l",  ylim=yrange,
col=1,  lwd  =  2,  xlab  =  "EM  iteration  number",  ylab  =
"Parameter values")
lines(hist[2:length(hist[,1]), 2], col=2, lwd = 2)
lines(hist[2:length(hist[,1]), 3], col=3, lwd = 2)
lines(hist[2:length(hist[,1]), 4], col=4, lwd = 2)
lines(hist[2:length(hist[,1]), 5], col=5, lwd = 2)
lines(hist[2:length(hist[,1]), 6], col=6, lwd = 2)
legend(0,  6,  lty  =  c(1,1),lwd=c(3,3),col=c(1,2,3,4,5,6),
c(expression(mu*"1"), expression(alpha*"1"),
expression(alpha*"2"),expression(beta*"1"),expression(beta
*"2"),expression(beta*"3")), horiz = TRUE)
```



```
# Plotting routine (y22 value)

plot(hist[2:length(hist[,1]), 7], type="l", col=1, lwd =
2, xlab = "EM iteration number", ylab = "y22")
```


y22
EM iteration number