# TMA4300 Exercise 3

Elsie Backen Tandberg, Maja Mathiassen, Florian Beiser

23.04.2021

## Problem A: Comparing AR(2) parameter estimators using resampling of residuals

Given a non-Gaussian time-series $\mathrm{x} = (x_t)_{t=1}^T$ of length $T = 100$, an AR(2) model is fitted to the data via least square and least absolute residuals, respectively, yielding two estimated parameters $\beta_1, \beta_2$ each.

The AR(2) model is specified by the relation

$$x_t = \beta_1 x_{t-1} + \beta_2 x_{t-2} + e_t$$

where $e_t$ is some iid noise with zero mean and constant variance.

The least sum of squared residual method (LS) estimates the two parameters of the AR(2) model (note that the parameters are independent of time) using the loss function

$$Q_{LS}(\beta_1, \beta_2 | x) = \sum_{t=3}^{T} (x_t - \beta_1 x_{t-1} - \beta_2 x_{t-2})^2$$

and minimizing $Q_{LS}$ with respect to $\beta_1, \beta_2$, resulting in a parameter estimate $\hat{\beta}_{LS}(\mathrm{x}) = (\hat{\beta}_{1,LS}, \hat{\beta}_{2,LS})$. In contrast, the least sum of absolute residual method (LA) using the loss function

$$Q_{LA}(\beta_1, \beta_2 | x) = \sum_{t=3}^{T} |x_t - \beta_1 x_{t-1} - \beta_2 x_{t-2}|$$

and the minimizing arguments of $Q_{LA}$ are denoted as the parameter estimate $\hat{\beta}_{LA}(\mathrm{x}) = (\hat{\beta}_{1,LA}, \hat{\beta}_{2,LA})$.

The respective estimated residuals $\hat{e} = (x_t - \hat{\beta}_1 x_{t-1} - \hat{\beta}_2 x_{t-2})_{t=3}^T$ are re-centered. (Note that there are no residuals for the first 2 time points.)

```
# The code requires that the extra files are downloaded from the course webpage
# and stored in the same folder as this Rmd
source("probAhelp.R")
source("probAdata.R")

# Observed timeseries data
x = data3A$x

# Parameter estimates via LS and LA
betas = ARp.beta.est(x, 2)
beta_LS = betas$LS
beta_LA = betas$LA

# Re-centered residuals for LS and LA
e_LS = ARp.resid(x, beta_LS)
e_LA = ARp.resid(x, beta_LA)
```

## 1: Residual resampling bootstrap method

We use the residual resampling bootstrap method with $B = 1500$ bootstrap samples to evaluate the performance of the LS in comparison to the LA estimator. Those samples utilize pseudo-time series which start at a random but consecutive time pair of the observed time series and is then constructed using the AR(2) model with resampled residuals. The bootstrap samples are the generated by estimating the parameters $\beta_1, \beta_2$ on those pseudo-time series with LS and LA, respectively.

Based on those bootstrap samples the variances and biases of the LS and LA method are estimated, respectively. Using the bootstrap samples $(\mathrm{x}^b)_{b=1}^B$, the variance is then estimated by $\hat{Var}(\hat{\beta}) = \frac{1}{B-1}\sum_{b=1}^B(\hat{\beta}(\mathrm{x}^b) - \bar{\hat{\beta}})$ where $\bar{\beta} = \frac{1}{B}\sum_{b=1}^B \hat{\beta}(\mathrm{x}^b)$ is the empirical mean. Note that $\hat{\beta}$ is the "true" value for the empirical distribution function of the observed time series, wherewith the bias is calculated as $\hat{bias}(\hat{\beta}) = \bar{\beta} - \hat{\beta}$. We take these formulas for each of the two components of $\beta$ separately. For the estimation of $\hat{\beta}$ the LS and LA approach are used, respectively.

```
set.seed(0421)
# Input
B = 1500 # number of bootstrap samples
T = 100  # length of time series
p = 2

# Bookkeeping
bootstrap_betas_LS = matrix(0, ncol=p, nrow=B)
bootstrap_betas_LA = matrix(0, ncol=p, nrow=B)

# Bootstrapping for LS and LA in one (using same realisations)
for (b in 1:B){
  # Construct random but consecutive inital values
  idx0 = sample(T-p+1, 1)
  x0 = x[idx0:(idx0+p-1)]
  # Resample residuals
  idxs = sample(T-p, replace=TRUE)
  e_LS_b = e_LS[idxs]
  e_LA_b = e_LA[idxs]
  # pseudo response
  x_LS_b = ARp.filter(x0, beta_LS, e_LS_b)
  x_LA_b = ARp.filter(x0, beta_LA, e_LA_b)
  # bootstrap estimate
  beta_LS_b = ARp.beta.est(x_LS_b, p)$LS
  beta_LA_b = ARp.beta.est(x_LA_b, p)$LA
  # Storing beta estimate
  bootstrap_betas_LS[b,] = beta_LS_b
  bootstrap_betas_LA[b,] = beta_LA_b
}

# Estimate for variances
var_LS = apply(bootstrap_betas_LS,2,var)
print(paste("The variance for beta_1 in the LS estimator is ", var_LS[1]))
```

```
## [1] "The variance for beta_1 in the LS estimator is  0.00576085680444579"
```

```
print(paste("The variance for beta_2 in the LS estimator is ", var_LS[2]))
```

```
## [1] "The variance for beta_2 in the LS estimator is  0.00574934117671368"
```

```
var_LA = apply(bootstrap_betas_LA,2,var)
print(paste("The variance for beta_1 in the LA estimator is ", var_LA[1]))
```

```
## [1] "The variance for beta_1 in the LA estimator is  0.000446665881203223"
```

```
print(paste("The variance for beta_2 in the LA estimator is ", var_LA[2]))
```

```
## [1] "The variance for beta_2 in the LA estimator is  0.000438405216657312"
```

```
# Estimate for bias
bias_LS = colMeans(bootstrap_betas_LS) - beta_LS
print(paste("The bias for beta_1 in the LS estimator is ", bias_LS[1]))
```

```
## [1] "The bias for beta_1 in the LS estimator is  -0.012496354745964"
```

```
print(paste("The bias for beta_2 in the LS estimator is ", bias_LS[2]))
```

```
## [1] "The bias for beta_2 in the LS estimator is  0.00692910229602606"
```

```
bias_LA = colMeans(bootstrap_betas_LA) - beta_LA
print(paste("The bias for beta_1 in the LA estimator is ", bias_LA[1]))
```

```
## [1] "The bias for beta_1 in the LA estimator is  -0.00242098656737344"
```

```
print(paste("The bias for beta_2 in the LA estimator is ", bias_LA[2]))
```

```
## [1] "The bias for beta_2 in the LA estimator is  0.00177590447916764"
```

For Gaussian AR(p) processes the LS estimator is optimal, but remember that the given process is non-Gaussian. Hereon the latest results, we observe that the LS estimator is no longer optimal for this problem since the LA estimator has smaller variance together with a smaller absolute bias! Hence, the LA is the preferable (better) estimator for the given problem.

## 2: Predicion i based on bootstrapping

We use the previous bootstrap samples to build a prediction interval for $t = 101$ for LS and LA, respectively. Therefore, we extend the observed timeseries $(x_t)_{t=1}^{T}$ utilizing the AR(2) by

$$x_{101}^{b} = \hat{\beta}_1^{b} x_{100} + \hat{\beta}_2^{b} x_{99} + e_{101}$$

where we plug in the previous bootstrap estimates for $\hat{\beta}^b$ and draw $e_{101}$ from the residual distribution $\hat{e}_{LS}$ and $\hat{e}_{LA}$, respectively (this procedure got clarified with Sara in the exercise session). The prediction interval is finally calculated from the quantiles of this prediction sample set.

```
# Bookkeeping
x101_LS = matrix(0, nrow=B)
x101_LA = matrix(0, nrow=B)

# Predicting x_101 for each bootstrap sample:
# x101 = b1*x100 + b2*x99 + e101
# where for b the estimates from part 1 are used
# and e101 is resampled from its empirical distribution
for (b in 1:B){
  x101_LS[b] = bootstrap_betas_LS[b,1]*x[100] + bootstrap_betas_LS[b,2]*x[99] + sample(e_LS,1)
  x101_LA[b] = bootstrap_betas_LA[b,1]*x[100] + bootstrap_betas_LA[b,2]*x[99] + sample(e_LA,1)
}

pred_interval_LS = quantile(x101_LS, probs=c(0.025,0.975))
pred_interval_LA = quantile(x101_LA, probs=c(0.025,0.975))
```

```
print(paste("The 95% prediction interval for LS is [",pred_interval_LS[1],",",pred_interval_LS[2],"]"))
```

```
## [1] "The 95% prediction interval for LS is [ 7.30600327243756 , 23.1340686615302 ]"
```

```
print(paste("The 95% prediction interval for LA is [",pred_interval_LA[1],",",pred_interval_LA[2],"]"))
```

```
## [1] "The 95% prediction interval for LA is [ 10.0851870417189 , 23.261560744565 ]"
```

The 95% prediction interval is for both estimators rather broad. The upper interval ends are both a bit above 23.0, but the lower end of LS is remarkably smaller (<7.5) than the one of LA (>10.0), such that the LA prediction interval is a way smaller in the end. (Of course the estimate for the quantile values is quite dependent on the chosen seed due to a relatively small sample size for estimating a 2.5% or 97.5% quantile.)

# Problem B: Permutation Test

In this problem we look at the concentration of bilirubin (mg/dL) in blood samples from the young men and do a permutation test.

```
bilirubin <- read.table("https://www.math.ntnu.no/emner/TMA4300/2020v/Exercise/ex3-additionalFiles/bili
```

## 1: Regression model and hypothesis test

First we use a box plot to look at the logarithm of the concentration for the three individuals.

```
#Finding the box plot
boxplot(log(meas)~pers, data=bilirubin)
```

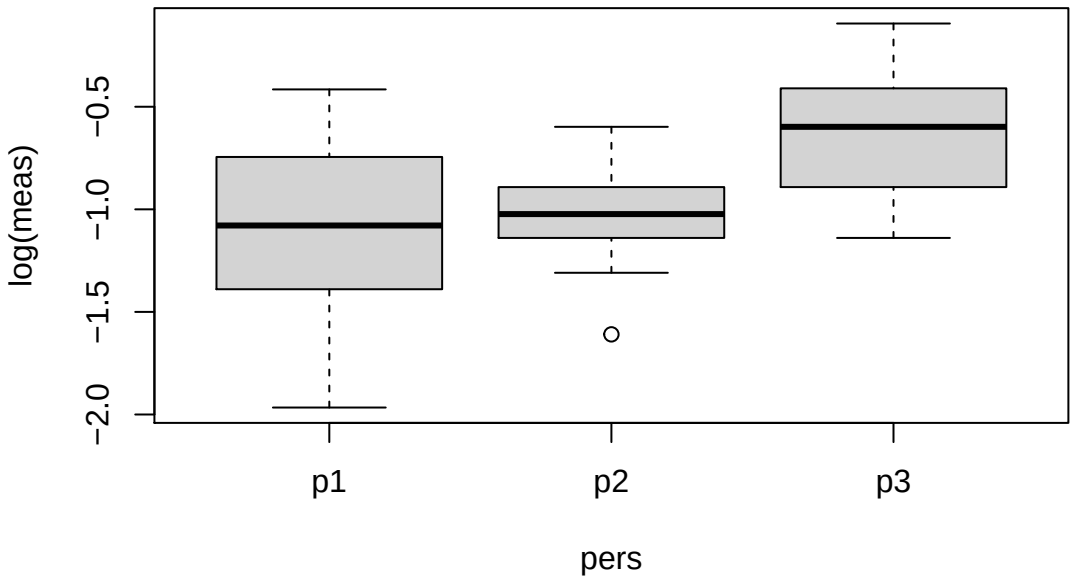


Figure 1: Box plot to inspect the logarithms of the concentrations for each individual.

From the plot in Figure 1 we can see that person 1 and person 2 have a similar median, but that person 1 has higher variability. Person 3 has higher median than the others and also a high variability.

Next we fit a regression model

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

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

```
#Fitting a linear model to the data
mod <- lm(log(meas)~pers, data=bilirubin)
#Finding the F-statistic
summ <- summary(mod)
#Saving the value of the F statistic as Fval
Fval <- summ$fstatistic[1]
#Calculating the quantile function
qf(0.95, 2, 26)
```

```
## [1] 3.369016
```

We will use the F-test to test the hypothesis that

$$H_0 : \beta_1 = \beta_2 = \beta_3 \quad \text{vs} \quad H_1 : \text{at least one is different.}$$

The F-statistic was found from the summary of the fitted linear model. $H_0$ is rejected if the F-statistic $=$ 3.6697751 is higher than 3.3690164. We see that this is true, so we reject $H_0$, concluding that there at least one of the $\beta$'s differ from the others.

## 2: Permutation of data

Now we will write a function that generates a permutation of the data between the three individuals.

```
permTest <- function(){
  #Generating one permutation of the dataset
  perm <- data.frame(meas=sample(bilirubin$meas, 29), pers=bilirubin$pers)
  #Fitting a linear model to the data
  mod <- lm(log(meas)~pers, data=perm)
  #Finding the F-statistic
  summ <- summary(mod)
  #Saving the value of the F statistic as Fval
  Fval <- summ$fstatistic[1]
  return(Fval)
}
```

## 3: Permutation test

We will generate 999 samples of the F-value by using the permutations calculated by the function permTest from B2. These samples will be used to find the p-value by counting the number of observations that are greater than the F-value found in B1. This p-value will be used to evaluate the same hypothesis test as in B1, with significance level $\alpha = 0.05$.

```
#Generating 999 samples of the F-value from the permTest function
samples <- rep(0,999)
for(i in 1:999){
  samples[i] <- permTest()
}
#Finding the p-value
pval <- sum(samples > Fval)/999
```

We found our p-value to be 0.035035, this is lower than our significance level, meaning that we reject $H_0$. So the same conclusion as in B1 is reached, where we believe that the three individuals have a differing concentration of bilirubin in their blood.

# Problem C: The EM-algorithm and bootstrapping

Here we consider $x_1, ..., x_n$ and $y_1, ..., y_n$ which are exponential distributed independent random variables with with intensity $\lambda_0$ and $\lambda_1$ respectively. We do not observe $x_1, ..., x_n$ and $y_1, ..., y_n$ directly, instead we observe $z_i = \max(x_i, y_i)$ for $i = 1, ..., n$, and $u_i = I(x_i \geq y_i)$ for $i = 1, ..., n$. Here $I(A) = 1$ if $A$ is true, and 0 otherwise. Based on the observations $(z_i, u_i)$ we will use the EM algorithm to find the maximum likelihood estimates for $(\lambda_0, \lambda_1)$.

## 1: Expected log likelihood

First the log likelihood function for the complete data $(x_i, y_i)$, $i = 1, ..., n$ is found.

$$L(x, y) = \prod_{i=1}^{n} \lambda_0 e^{-\lambda_0 x_i} \lambda_1 e^{-\lambda_1 y_i}$$

$$l(x, y) = n(\log(\lambda_0) + \log(\lambda_1)) - \lambda_0 \sum_{i=1}^{n} x_i - \lambda_1 \sum_{i=1}^{n} y_i.$$

Further we are interested in the expectation of the log likelihood

$$E[\log(f(\mathbf{x}, \mathbf{y}|\lambda_0, \lambda_1))|\mathbf{z}, \mathbf{u}, \lambda_0^{(t)}, \lambda_1^{(t)})] = n(\log(\lambda_0) + \log(\lambda_1)) - \lambda_0 \sum_{i=1}^{n} E(x_i|z_i, u_i, \lambda_0^{(t)}, \lambda_1^{(t)}) - \lambda_1 \sum_{i=1}^{n} E(y_i|z_i, u_i, \lambda_0^{(t)}, \lambda_1^{(t)}).$$

The expressions for $E(x_i|z_i, u_i, \lambda_0^{(t)}, \lambda_1^{(t)})$ and $E(y_i|z_i, u_i, \lambda_0^{(t)}, \lambda_1^{(t)})$ are dependent on which of the variables $(x_i, y_i)$ we have observed. So the pdf´s $f((x_i|z_i, u_i, \lambda_0^{(t)}, \lambda_1^{(t)}))$ and $f((y_i|z_i, u_i, \lambda_0^{(t)}, \lambda_1^{(t)}))$ can be written as two split equations depending on the value of $u_i$.

$$f(x_i|z_i, u_i, \lambda_0^{(t)}, \lambda_1^{(t)}) = \begin{cases} z_i & u_i = 1 \\ \frac{f(z_i|x_i)f(x_i)}{f(z_i)} & u_i = 0 \end{cases}$$

$$= \begin{cases} z_i & u_i = 1 \\ \frac{\lambda_0^{(t)} e^{-\lambda_0^{(t)} x_i}}{\int_0^{z_i} \lambda_0^{(t)} e^{-\lambda_0^{(t)} x_i} dx_i} & u_i = 0 \end{cases}$$

$$= \begin{cases} z_i & u_i = 1 \\ \frac{\lambda_0^{(t)} e^{-\lambda_0^{(t)} x_i}}{1 - e^{-\lambda_0^{(t)} z_i}} & u_i = 0 \end{cases}$$

$$E(x_i|z_i, u_i, \lambda_0^{(t)}, \lambda_1^{(t)}) = z_i u_i + (1 - u_i) \frac{\int_0^{z_i} x_i \lambda_0^{(t)} e^{-\lambda_0^{(t)} x_i} dx_i}{1 - e^{-\lambda_0^{(t)} x_i}}$$

The integral $\int_0^{z_i} x_i \lambda_0^{(t)} e^{-\lambda_0^{(t)} x_i} dx_i$ is found by integration by parts.

$$\begin{aligned}\int_0^{z_i} x_i \lambda_0^{(t)} e^{-\lambda_0^{(t)} x_i} dx_i &= -x_i e^{\lambda_0^{(t)} x_i}\Big|_0^{z_i} - \int_0^{z_i} -e^{-\lambda_0^{(t)} x_i} dx_i \\ &= -z_i e^{\lambda_0^{(t)} z_i} - \left[\frac{1}{\lambda_0} e^{-\lambda_0^{(t)} x_i}\right]_0^{z_i} \\ &= \frac{1}{\lambda_0^{(t)}} - \frac{z_i}{e^{\lambda_0^{(t)} z_i} - 1}\end{aligned}$$

Inserting this into the expression for the expectation we get the following equation

$$E(x_i|z_i, u_i, \lambda_0^{(t)}, \lambda_1^{(t)}) = z_i u_i + (1-u_i)\left(\frac{1}{\lambda_0^{(t)}} - \frac{z_i}{e^{\lambda_0^{(t)} z_i} - 1}\right)$$

Similar calculations are done to find $E(y_i|z_i, u_i, \lambda_0^{(t)}, \lambda_1^{(t)})$

$$E(y_i|z_i, u_i, \lambda_0^{(t)}, \lambda_1^{(t)}) = (1-u_i) z_i + u_i\left(\frac{1}{\lambda_1^{(t)}} - \frac{z_i}{e^{\lambda_1^{(t)} z_i} - 1}\right)$$

Combining the expressions for the expectation leads to the following equation

$$\begin{aligned}E[\log(f(\mathbf{x}, \mathbf{y}|\lambda_0, \lambda_1))|\mathbf{z}, \mathbf{u}, \lambda_0^{(t)}, \lambda_1^{(t)})] &= n(\log(\lambda_0) + \log(\lambda_1)) \\ &\quad - \lambda_0 \sum_{i=1}^n \left[u_i z_i + (1-u_i)\left(\frac{1}{\lambda_0^{(t)}} - \frac{z_i}{e^{\lambda_0^{(t)} z_i} - 1}\right)\right] \\ &\quad - \lambda_1 \sum_{i=1}^n \left[(1-u_i) z_i + u_i\left(\frac{1}{\lambda_1^{(t)}} - \frac{z_i}{e^{\lambda_1^{(t)} z_i} - 1}\right)\right].\end{aligned}$$

## 2: EM-algorithm

Next we want to use the formula we have found to implement an EM-algorithm that estimates the maximum likelihood of $(\lambda_0, \lambda_1)$. Therefore we want a recursion that maximizes the expression above with respect to $(\lambda_0, \lambda_1)$.

$$\begin{aligned}\frac{\partial E[\log(f(\mathbf{x}, \mathbf{y}|\lambda_0, \lambda_1))|\mathbf{z}, \mathbf{u}, \lambda_0^{(t)}, \lambda_1^{(t)})]}{\partial \lambda_0} &= \frac{n}{\lambda_0} - \sum_{i=1}^n \left[u_i z_i + (1-u_i)\left(\frac{1}{\lambda_0^{(t)}} - \frac{z_i}{e^{\lambda_0^{(t)} z_i} - 1}\right)\right] = 0 \\ \frac{\partial E[\log(f(\mathbf{x}, \mathbf{y}|\lambda_0, \lambda_1))|\mathbf{z}, \mathbf{u}, \lambda_0^{(t)}, \lambda_1^{(t)})]}{\partial \lambda_1} &= \frac{n}{\lambda_1} - \sum_{i=1}^n \left[(1-u_i) z_i + u_i\left(\frac{1}{\lambda_1^{(t)}} - \frac{z_i}{e^{\lambda_1^{(t)} z_i} - 1}\right)\right] = 0\end{aligned}$$

By solving these equations with respect to $\lambda_0$ and $\lambda_1$ respectively, we get a recursion for $(\lambda_0^{(t)}, \lambda_1^t)$,

$$\begin{aligned}\lambda_0^{(t+1)} &= n\left(\left[u_i z_i + (1-u_i)\left(\frac{1}{\lambda_0^{(t)}} - \frac{z_i}{e^{\lambda_0^{(t)} z_i} - 1}\right)\right]\right)^{-1} \\ \lambda_1^{(t+1)} &= n\left(\left[(1-u_i) z_i + u_i\left(\frac{1}{\lambda_1^{(t)}} - \frac{z_i}{e^{\lambda_1^{(t)} z_i} - 1}\right)\right]\right)^{-1}\end{aligned}$$

We run the recursion until $\|(\lambda_0^{(t+1)}, \lambda_1^{(t+1)}) - (\lambda_0^{(t)}, \lambda_1^{(t)})\|_2 < 10^{-10}$, where $\|\cdot\|_2$ is the Euclidean norm.

```
u <- read.table("https://www.math.ntnu.no/emner/TMA4300/2020v/Exercise/ex3-additionalFiles/u.txt",
                header=F)
z <- read.table("https://www.math.ntnu.no/emner/TMA4300/2020v/Exercise/ex3-additionalFiles/z.txt",
                header=F)
```

```r
#EM algorithm
EM.algorihtm <- function(lam0, lam1, u, z){
  lam0i <- lam0
  lam1i <- lam1
  n<-length(u)
  diff <- 1
  lambda0.list <- c(lam0i)
  lambda1.list <- c(lam1i)

  while(diff>1e-10){
    lamvec <- c(lam0i, lam1i)
    #Expectation step
    sum0 <- sum(u*z+(1-u)*(1/lam0i-z/(exp(lam0i*z)-1)))
    sum1 <- sum((1-u)*z+u*(1/lam1i-z/(exp(lam1i*z)-1)))
    #Q <- n*(log(lam0i)+log(lam1i))-lam0i*sum0-lam1i*sum1

    #Maximization step
    lam0i <- n/sum0
    lam1i <- n/sum1

    #Storing the lambda values
    lambda0.list <- c(lambda0.list, lam0i)
    lambda1.list <- c(lambda1.list, lam1i)

    #Convergence criteria
    diff <- norm(cbind(lamvec[1]-lam0i, lamvec[2]-lam1i), type="2")

  }
  #print(lambda0.list)
  return(cbind(c(lambda0.list), c(lambda1.list)))
}

em1<-EM.algorihtm(0.5,0.5, u$V1, z$V1)

x <- seq(1:length(em1[,1]))
ggplot()+
  geom_point(aes(x=x, y=em1[,1], color="lambda0"))+
  geom_point(aes(x=x, y=em1[,2], color="lambda1"))  +
  labs(title = "Convergence",
       x = "Iterations", y=expression(lambda))
```

From the convergence plot in Figure 2 we see that the values for both $\lambda_0$ and $\lambda_1$ have converged. We choose the last generated values as our $\lambda$-estimates. Our estimates $\hat{\lambda_0}$= 3.465735 and $\hat{\lambda_1}$= 9.3532149.

## 3: Bootstrapping

- For i in number of iterations
  - Draw n samples from our data, $(z, u)$, with replacement
  - Run the EM-algorithm for the drawn sample
  - Store the last generated lambda values in a data frame

Using our estimated values for $\lambda_0$ and $\lambda_1$ from the bootstrap algorithm, we will calculate the mean, standard deviation and bias for $\hat{\lambda_0}$ and $\hat{\lambda_1}$. The standard deviation is calculated using $\sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\hat{\lambda}^* - E[\hat{\lambda}^*])^2}$ and the bias is found by $E[\hat{\lambda}^*] - \hat{\lambda}$, $\hat{\lambda}$ is the value found in C2.

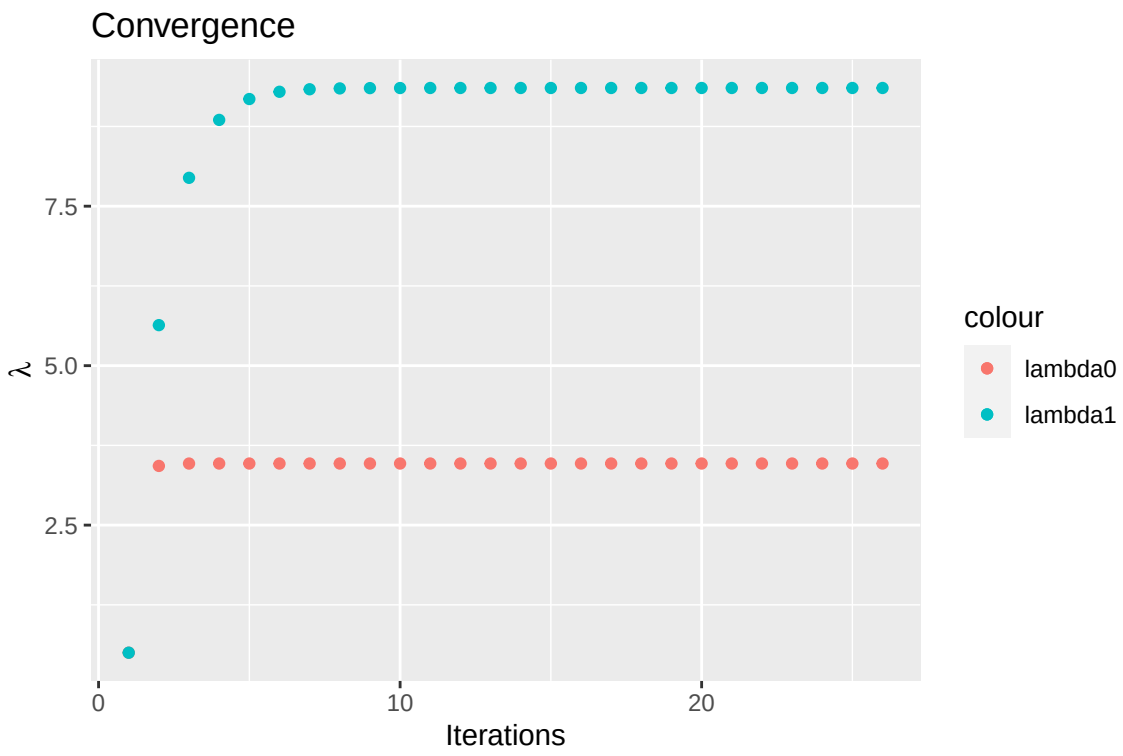


Figure 2: Convergence of $\lambda_0$ and $\lambda_1$.

```
#Bootstrapping, finding 1000 values for lambdas
u.z <- cbind("u"=u, "z"=z)
lambdas <- data.frame(lambda0 = rep(0, 1000), lambda1 = rep(0, 1000))

for(i in 1:1000){
  index <- sample(1:200,200, replace = T)
  bootstrap.sample<-u.z[index,]
  em <- EM.algorihtm(0.5,0.5, bootstrap.sample[,1], bootstrap.sample[,2])
  lambdas[i,] <- em[length(em[,1]),]
}

#Calculating the mean
means = colMeans(lambdas)
mean.lambda0 <- means[1]
mean.lambda1 <- means[2]

#Calculating the standard deviation
sd.lambda0 <- sqrt(1/(length(lambdas[,1])-1)*sum((lambdas[,1]-mean.lambda0)^2))
sd.lambda1 <- sqrt(1/(length(lambdas[,2])-1)*sum((lambdas[,2]-mean.lambda1)^2))

#Calculating the bias
bias.lambda0 <- mean.lambda0-em1[length(em1[,1]),1]
bias.lambda1 <- mean.lambda1-em1[length(em1[,1]),2]

#Calculating the correlation
corr<-cor(lambdas[,1], lambdas[,2])
```

We find that the means of the bootstrap samples are $E[\hat{\lambda_0}]$=3.4865406, and $E[\hat{\lambda_1}]$=9.4014197. The standard deviations are SD$[\hat{\lambda_0}]$=0.2434212 and SD$[\hat{\lambda_0}]$=0.7670058. The biases are bias$[\hat{\lambda_0}]$=0.0208056 and bias$[\hat{\lambda_1}]$=0.0482047. Finally, Corr$[\hat{\lambda_0}, \hat{\lambda_1}]$=-0.0020935. Observing that the standard deviation is much larger than the biases we can assume that the bias might be due to noise. This means that we prefer the maximum likelihood estimate to the bias corrected estimate. There is a low correlation between the estimators even though both estimates are calculated from the same data.

## 4: Maximum likelihood estimators

We are interested in $f_{z_i,u_i}(z_i, u_i|\lambda_0, \lambda_1)$. We first look at the situation where $u_i = 1$ which means that $z_i = x_i$ and $x_i \geq y_i$. We find the cumulative distribution of $z_i$, $F_{z_i}$.

$$\begin{aligned}
F_{Z_i,U_i}(z_i, u_i = 1|\lambda_0, \lambda_1) &= P(Z_i \leq z_i, U_i = 1) \\
&= P(Y_i \leq x_i, X_i \leq z_i) \\
&= \int_0^{z_i} \int_0^{x_i} f_{x_i,y_i}(x_i, y_i) dy_i dx_i \quad x_i \text{ and } y_i \text{ are independent} \\
&= \int_0^{z_i} \int_0^{x_i} \lambda_0 e^{-\lambda_0 x_i} \lambda_1 e^{-\lambda_1 y_i} dy_i dx_i \\
&= \int_0^{z_i} \lambda_0 e^{-\lambda_0 x_i} (1 - e^{-\lambda_1 x_i}) dx_i
\end{aligned}$$

We use the fundamental theorem of calculus

$$\begin{aligned}
f_{z_i u_i}(z_i, u_i = 1|\lambda_0, \lambda_1) &= \frac{dF_{Z_i}}{dz_i} \int_0^{z_i} \lambda_0 e^{-\lambda_0 x_i} (1 - e^{\lambda_1 x_i}) dx_i \\
&= \lambda_0 e^{-\lambda_0 z_i} (1 - e^{-\lambda_1 z_i})
\end{aligned}$$

Next we look at the situation where $u_i = 0$,

$$\begin{aligned}
F_{Z_i,U_i}(z_i, u_i = 0|\lambda_0, \lambda_1) &= P(Z_i \leq z_i, U_i = 0) \\
&= P(X_i \leq y_i, Y_i \leq z_i) \\
&= \int_0^{z_i} \int_0^{y_i} f_{x_i,y_i}(x_i, y_i) dx_i dy_i \quad x_i \text{ and } y_i \text{ are independent} \\
&= \int_0^{z_i} \int_0^{y_i} \lambda_0 e^{-\lambda_0 x_i} \lambda_1 e^{-\lambda_1 y_i} dx_i dy_i \\
&= \int_0^{z_i} \lambda_1 (1 - e^{-\lambda_0 y_i}) e^{-\lambda_1 y_i} dy_i
\end{aligned}$$

We use the fundamental theorem of calculus

$$\begin{aligned}
f_{z_i u_i}(z_i, u_i = 0|\lambda_0, \lambda_1) &= \frac{dF_{Z_i}}{dz_i} \int_0^{z_i} \lambda_1 (1 - e^{-\lambda_0 y_i}) e^{-\lambda_1 y_i} dy_i \\
&= \lambda_1 (1 - e^{-\lambda_0 z_i}) e^{-\lambda_1 z_i}
\end{aligned}$$

Next we combine the two equations $f_{z_i}(z_i, u_i = 1|\lambda_0, \lambda_1)$ and $f_{z_i}(z_i, u_i = 0|\lambda_0, \lambda_1)$.

$$f_{z_i,u_i}(z_i, u_i|\lambda_0, \lambda_1) = \begin{cases} \lambda_0 e^{-\lambda_0 z_i} (1 - e^{-\lambda_1 z_i}) & u_i = 1 \\ \lambda_1 (1 - e^{-\lambda_0 z_i}) e^{-\lambda_1 z_i} & u_i = 0 \end{cases}$$

We are going to find analytic formulas for the maximum likelihood estimators $\hat{\lambda}_0$ and $\hat{\lambda}_1$.

$$L(\lambda_0, \lambda_1) = \prod_{i=1}^{n} f_{z_i,u_i}(z_i, u_i|\lambda_0, \lambda_1)$$

$$l(\lambda_0, \lambda_1) = n_0 \ln(\lambda_1) + n_1 \ln(\lambda_0) + \sum_{i=1}^{n} \left[ u_i(\ln(1 - e^{-\lambda_1 z_i}) - \lambda_0 z_i) + (1 - u_i)(\ln(1 - e^{-\lambda_0 z_i}) - \lambda_1 z_i) \right]$$

where $n_0 = \sum_{i=1}^{n} I(u_i = 0)$ and $n_1 = \sum_{i=1}^{n} I(u_i = 1)$.

$$\frac{\partial l}{\partial \lambda_0} = \frac{n_1}{\lambda_0} + \sum_{i=1}^{n} \left[ (1 - u_i) \frac{z_i e^{-\lambda_0 z_i}}{1 - e^{-\lambda_0 z_i}} - u_i z_i \right] = 0 \frac{\partial l}{\partial \lambda_1} = \frac{n_0}{\lambda_1} + \sum_{i=1}^{n} \left[ u_i \frac{z_i e^{-\lambda_1 z_i}}{1 - e^{-\lambda_1 z_i}} - (1 - u_i) z_i \right] = 0$$

We can not find an analytic solution for the maximum likelihood estimators for $\lambda_0$ and $\lambda_1$. Instead we will compute the values using the optim-function.

```
#Finding optimized values for the lambdas
#Log likelihood function
loglike <- function(lambdavec){
  n0 <- sum(u==0)
  n1 <- sum(u)
  l <- n0*log(lambdavec[2]) + n1*log(lambdavec[1])
  l <- l + sum(u*(log(1-exp(-lambdavec[2]*z))-lambdavec[1]*z))
  l <- l + sum((1-u)*(log(1-exp(-lambdavec[1]*z))-lambdavec[2]*z))
  return(-l)
}

lambdavec <- c(0.5,0.5)
opt = optim(lambdavec, loglike, hessian=TRUE)
lambda0o <- opt$par[1]
lambda1o <- opt$par[2]
H<-opt$hessian
Hinv<-solve(H)
Hsq<-sqrt(diag(Hinv))
sd_est<-diag(Hsq)
```

The optimized values are $\hat{\lambda_0}$=3.4656736 and $\hat{\lambda_1}$=9.3556971. We see that they are close to the result obtained from the EM-algorithm.

The advantages of optimizing the likelihood directly compared to the EM algorithm is that the run time of the EM-algorithm depends on the initial values and therefore the run time might be slower than the optim-function. Additionally, the optim-function can find the Hessian for you, which can be used to estimate the standard deviation of $\hat{\lambda_0}$ and $\hat{\lambda_1}$. Then we get SD[$\hat{\lambda_0}$]=0.2465817 and SD[$\hat{\lambda_0}$]=0.815461.