B and C

Problem B: Permutation test

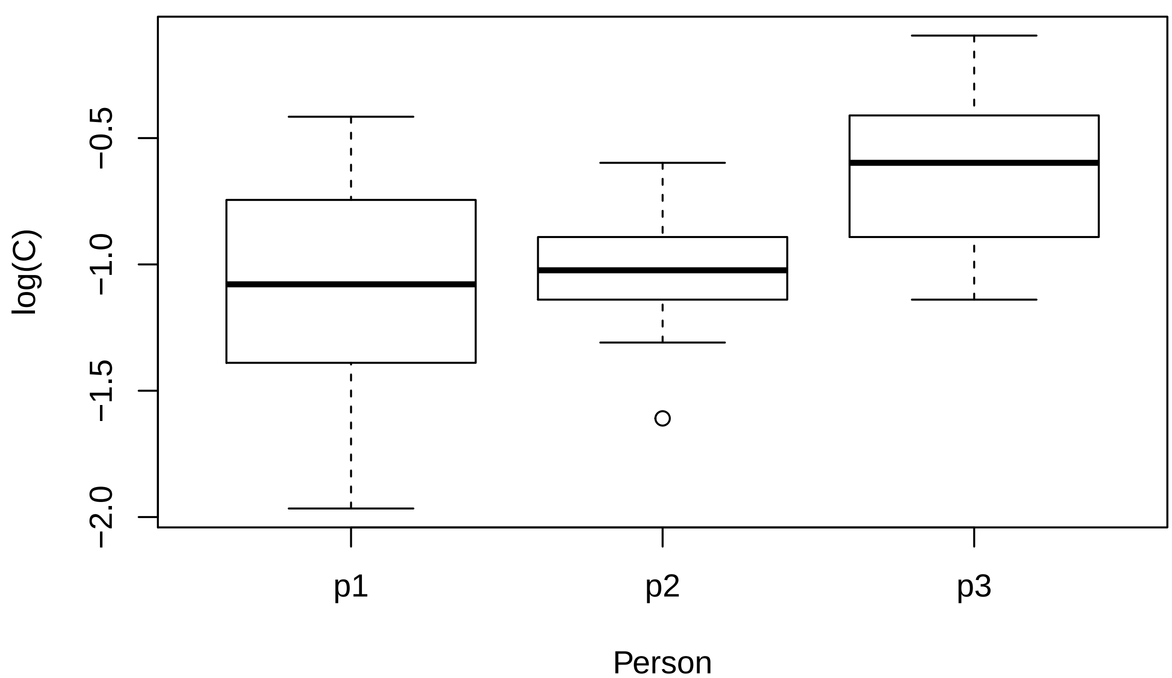
In this task, the blood consentration of bilirubin for three different men will be studied. Let the stocastic variable $\mathbf{Y}$ be the vector of bilirubin in blood samples. The total number of measurements is $n_t = 29$, with the individual number of mesurments beeing $n_1 = 11$, $n_2 = 10$ and $n_3 = 8$.

```
library(MASS)
bilirubin <- read.table("bilirubin.txt",header=T)
```

1)

We make some visual inference from the box-plot, to get some sense of the difference between the different individuals.

```
boxplot( log(meas)~ pers,data=bilirubin, xlab="Person", ylab="log(C)")
```



As seen from the above graph, individual one and individual two seem to have approximately the same median. Meanwhile, the third individual seem to have a somewhat higher median within this dataset. Furthermore, both individual one and three seem to have a bigger variability in the measurements, compared to individual two.

Next, we fit a linear regression model for the log-consentration.

$$\log(Y_{ij}) = \beta_i + \epsilon_{ij}, \quad \text{for } i = 1, 2, 3 \text{ and } j = 1, 2, \ldots, n_i$$

where $\epsilon_{ij} \sim \mathcal{N}(0, \sigma^2)$ are i.i.d. The F-test will be used on the null hypothesis $\beta_1 = \beta_2 = \beta_3$.

```
lin.fit <- lm(log(meas)~pers,data=bilirubin)
Fval <- summary(lin.fit)$fstatistic[1]
Fval
```

```
##    value
## 3.669775
```

As seen, the F-value is $Fval = 3.67$, which corresponts to a p-value of 0.039. With a significance level of 0.05, we therefore reject the null hypothesis. Thus the new claim is that atleast one of the three men have a different value of bilirubin consentration in their blood.

2)

A function, `permTest`, is written in order to perform a permutation test. The function takes a random sample from the measurements, and fits a linear regression model. The return value is the F-statistic.

```
permTest <- function(p,meas){
  n <- length(meas)
  x <- sample(meas, n)
  return(as.numeric(summary(lm(log(x) ~ p))$fstatistic[1]))
}
```

3)

The permutation test consist in sampling many F-values, in our case, sampling $n = 999$ values, using the above `permTest` function. We store the value from each iteration in a vector $\mathbf{F}$. Lastly, the p-value of interest is found by taking $\sum_{i=1}^{999} I(\mathbf{F}_i > Fval)$, where I is the indicator function.

```
n <- 999
F_vec <- numeric(n)

for (i in 1:n){
  F_vec[i] <- permTest(bilirubin$pers,bilirubin$meas)
}
#Compute pval using the indicatior function
pval <- sum(F_vec > Fval)/n
pval
```

```
## [1] 0.04204204
```

The achieved p-value is 0.042 which closely matches the p-value from the regular linear regression, and the null hypothesis is still rejected.

Problem C: The EM-algorithm and bootstrapping

Let $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ be independent random variables. Furthermore, let the x_i's and y_i's be exponentially distributed with intensity λ_0 and λ_1 respectively. We do, however, not directly measure these variables. We instead observe $z_i = \max(x_i, y_i)$ and $u_i = I(x_i \geq y_i)$ for $i = 1, \ldots, n$. Using the observed $(z_i, u_i), i = 1, \ldots, n$ the EM algorithm will be used to find the maximum likelhood estimates for (λ_0, λ_1).

1)

Since x_i and y_i are independent

$$f(\mathbf{x}, \mathbf{y}|\lambda_0,\, \lambda_1) = \prod_{i=1}^{n} f(x_i|\lambda_0) f(y_i|\lambda_1) = (\lambda_0\lambda_1)^n \exp\left(-\lambda_0 \sum_{i=1}^{n} x_i\right) \exp\left(-\lambda_1 \sum_{i=1}^{n} y_i\right).$$

The log likelihood therefore becomes

$$\ln f(\mathbf{x}, \mathbf{y}|\lambda_0,\, \lambda_1) = n\ln(\lambda_0) + n\ln(\lambda_1) - \lambda_0 \sum_{i=1}^{n} x_i - \lambda_1 \sum_{i=1}^{n} y_i.$$

The EM algorithm alternates between performing an expectation step and a maximization step, hence the name. In the first step, the expectation of the log-likelihood is evaluated with the current best estimate for the parameters. The conditional expectation is given as

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

We therefore need to evaluate $E\left[x_i|z_i, u_i, \lambda_0^{(t)},\, \lambda_1^{(t)}\right]$ and $E\left[y_i|z_i, u_i, \lambda_0^{(t)},\, \lambda_1^{(t)}\right]$. The conditional probabilities of x_i given $z_i, u_i, \lambda_0^{(t)}, \lambda_1^{(t)}$ are

$$f(x_i|z_i, u_i, \lambda_0^{(t)},\, \lambda_1^{(t)}) = \begin{cases} \dfrac{\lambda_0^{(t)} \exp(-\lambda_0^{(t)} x_i)}{1 - \exp(-\lambda_0^{(t)} z_i)}, & \text{when } u_i = 0, \\ z_i, & \text{when } u_i = 1, \end{cases}$$

and the conditional probabilities of y_i given $z_i, u_i, \lambda_0^{(t)}, \lambda_1^{(t)}$ are

$$f(y_i|z_i, u_i, \lambda_0^{(t)},\, \lambda_1^{(t)}) = \begin{cases} z_i, & \text{when } u_i = 0, \\ \dfrac{\lambda_1^{(t)} \exp(-\lambda_1^{(t)} y_i)}{1 - \exp(-\lambda_1^{(t)} z_i)}, & \text{when } u_i = 1. \end{cases}$$

We then comtupe the expected values. Firstly,

$$\begin{aligned} E\left[x_i|z_i, u_i, \lambda_0^{(t)},\, \lambda_1^{(t)}\right] &= u_i z_i + (1 - u_i) \int_0^{z_i} x_i \frac{\lambda_0^{(t)} \exp(-\lambda_0^{(t)} x_i)}{1 - \exp(-\lambda_0^{(t)} z_i)} = u_i z_i + (1 - u_i) \left(\frac{\exp(\lambda_0 z_i) - \lambda_0 z_i - 1}{\lambda_0(\exp(\lambda_0 z_i) - 1)}\right) \\ &= u_i z_i + (1 - u_i) \left(\frac{1}{\lambda_0^{(t)}} - \frac{z_i}{\exp(\lambda_0^{(t)} z_i) - 1}\right). \end{aligned}$$

Lastly

$$E\left[y_i|z_i, u_i, \lambda_0^{(t)},\, \lambda_1^{(t)}\right] = (1 - u_i) z_i + u_i \int_0^{z_i} y_i \frac{\lambda_1^{(t)} \exp(-\lambda_1^{(t)} y_i)}{1 - \exp(-\lambda_1^{(t)} z_i)} = (1 - u_i) z_i + u_i \left(\frac{1}{\lambda_1^{(t)}} - \frac{z_i}{\exp(\lambda_1^{(t)} z_i) - 1}\right).$$

Simply inserting these into the expression for $E\left[\ln f(\mathbf{x},\mathbf{y}|\lambda_0,\,\lambda_1)|\mathbf{z},\mathbf{u},\lambda_0^{(t)},\,\lambda_1^{(t)}\right]$ yields

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

as was desired.

2)

The maximization step involves maximizing the likelihood found in 1). We find the extremal values using

$$\frac{dE\left[\ln f(\mathbf{x},\mathbf{y}|\lambda_0,\,\lambda_1)|\mathbf{z},\mathbf{u},\lambda_0^{(t)},\,\lambda_1^{(t)}\right]}{d\lambda_0}=0 \qquad \text{and} \qquad \frac{dE\left[\ln f(\mathbf{x},\mathbf{y}|\lambda_0,\,\lambda_1)|\mathbf{z},\mathbf{u},\lambda_0^{(t)},\,\lambda_1^{(t)}\right]}{d\lambda_1}=0,$$

which yields the scheme

$$\lambda_0^{(t+1)}=\frac{n}{\sum_{i=1}^n\left(u_iz_i+(1-u_i)\left(\frac{1}{\lambda_0^{(t)}}-\frac{z_i}{\exp(\lambda_0^{(t)}z_i)-1}\right)\right)},$$

and

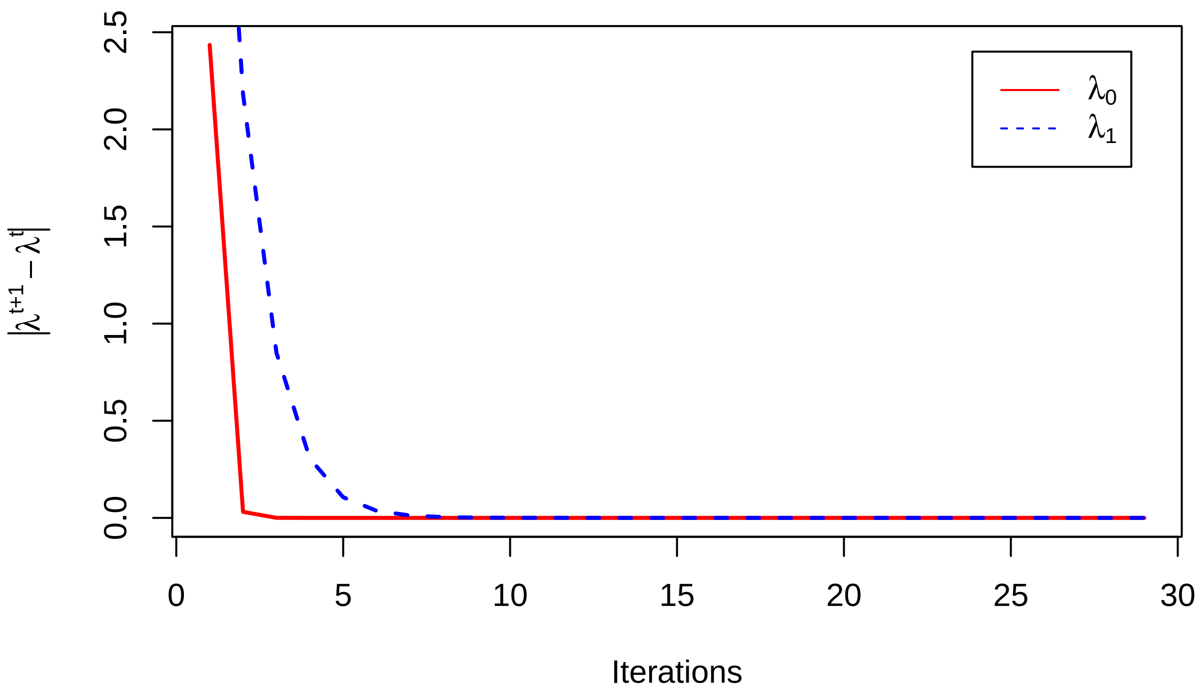
$$\lambda_1^{(t+1)}=\frac{n}{\sum_{i=1}^n\left((1-u_i)z_i+u_i\left(\frac{1}{\lambda_1^{(t)}}-\frac{z_i}{\exp(\lambda_1^{(t)}z_i)-1}\right)\right)}.$$

Below is the implementation of the EM-algorithm. The initial values of both λ_0 and λ_1 is set to one.

```r
# variables
z <- as.numeric(read.table("z.txt")[,1])
u <- as.numeric(read.table("u.txt")[,1])
n <- length(u)
iter <- 30
lambda <- matrix(0,nrow=iter,ncol=2)
lambda[1,] <- cbind(1,1)
#help function
recursion <- function(l,z,u){
  l0 <- l[1]
  l1 <- l[2]
  sum_l0 <- u %*% z + (1-u) %*% (1/l0-z/(exp(l0*z)-1))
  sum_l1 <- (1-u) %*% z + u %*% (1/l1-z/(exp(l1*z)-1))
  l0 <- n/(sum_l0)
  l1 <- n/(sum_l1)
  return(cbind(l0,l1))
}
for (i in 2:iter){
  lambda[i,] <- recursion(lambda[i-1,],z,u)
}
# Use last vector
l_EM <- lambda[iter,]
sprintf("lambda_0: %.3f,  lambda_1: %.3f", l_EM[1], l_EM[2])
```

```
## [1] "lambda_0: 3.466,  lambda_1: 9.353"
```

```r
#plot konvergence
dl0 = abs(diff(lambda[,1]))
dl1 = abs(diff(lambda[,2]))
par(mar=c(5,5,4,1)+.1)
plot(1:(iter-1), dl0, type = "l", col = "red", lwd = 2, xlab = "Iterations", ylab = expression(abs( lam
lines(1:(iter-1), dl1, col = "blue", lwd = 2,lty = 2)
legend("topright", inset = 0.05, legend = c(expression(lambda[0]), expression(lambda[1])), col = c("red
```



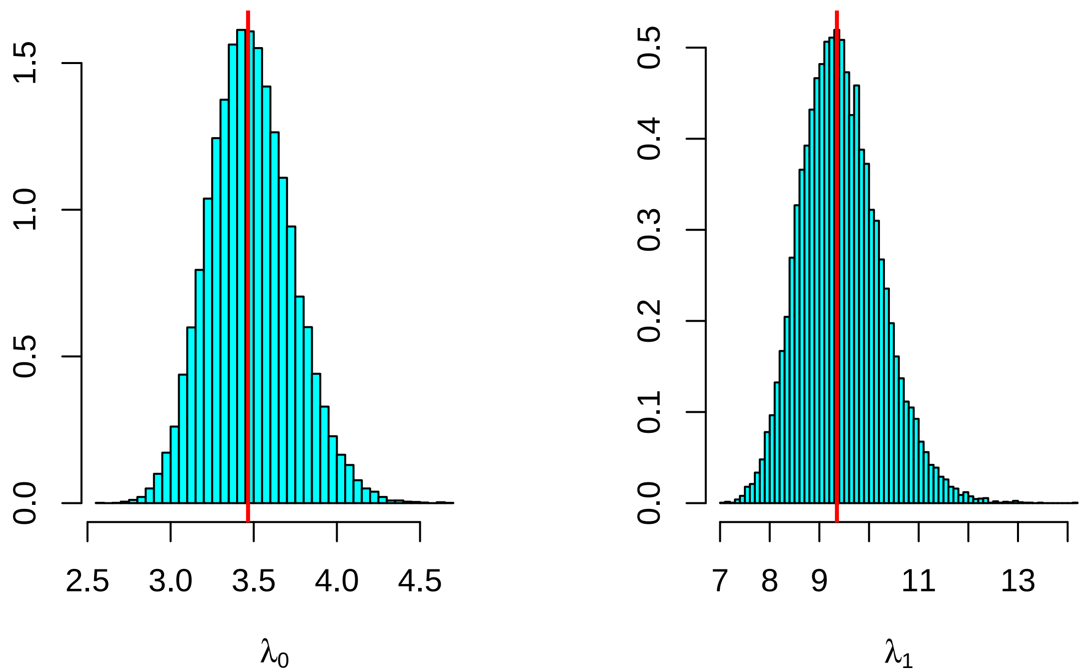
The plot above shows the convergence of both λ_0 and λ_1 in a red and blue (stripled) line respectively. On the y-axis, the value of $|\lambda_i^{t+1} - \lambda_i^t|$ is shown. The convergence is quite fast, needing less than 20 iterations to reach machine precision. ## 3) Both $\hat{\lambda}_0$ and $\hat{\lambda}_1$ will be found using bootstrap. Furthermore, the correlation $\text{Corr}[\hat{\lambda}_0, \hat{\lambda}_1]$ will be estimated. A simple pseudocode is presented as follows

$$
\begin{aligned}
&\textit{for } i \textit{ in } 1, 2, \ldots, N: \\
&\quad \textit{Sample } \tilde{z} \textit{ with replacement from } z \\
&\quad \textit{Sample } \tilde{u} \textit{ with replacement from } u \\
&\quad \textit{Compute } (\tilde{\lambda}_{0,i}, \tilde{\lambda}_{1,i}) \textit{ by using the EM-algorithm with the newly sampled } \tilde{z}_i \textit{ and } \tilde{u}_i \\
&\quad \textit{Store the result } (\tilde{\lambda}_{0,i}, \tilde{\lambda}_{1,i}) \\
&\textit{Compute the desired quantities from the stored samples}
\end{aligned}
$$

Of course, one needs to define a stopping criteria for the EM-algorithm. The two-norm is used, with tolerance 10^{-10}.

```r
N = 20000
EM <- function(z,u){
```

```
  l <- l_EM #use the global l_EM
  diff <- 1
  while (diff > 1e-10){
    l_old <- l
    l <- recursion(l,z,u)
    diff <- sqrt(sum((l-l_old)^2))
  }
  return(l)
}
l_vec <- matrix(0,nrow = N, ncol=2)
for (i in 1:N){
  s <- sample(1:n, replace=TRUE) # random permutation vector
  z_i <- z[s]
  u_i <- u[s]
  l_vec[i,] <- EM(z_i,u_i)
}
#Plot historgrams
par(mfrow=c(1,2))
truehist(l_vec[,1], xlab=expression(lambda[0]))
abline(v=l_EM[1], col="red",lwd = 2) # plot the l_EM to compare
truehist(l_vec[,2],xlab=expression(lambda[1]))
abline(v=l_EM[2],col="red", lwd = 2)
```




```
#Find quantities of interest
mean.l <- colMeans(l_vec)
sd.l <- sqrt(diag(cov(l_vec)))
```

```r
bias.l <- mean.l - l_EM
corr.l <- cor(l_vec)[1, 2]
cat(" Mean: ",mean.l,"    Standard deviation: ",sd.l,"\nBias: ",bias.l,"    Corr: ",corr.l,"\n")
```

```
##  Mean:  3.483348 9.439828     Standard deviation:  0.2486145 0.7968307
## Bias:  0.01761272 0.08661303     Corr:  0.003323225
```

The means achieved by bootstrapping differ slightly from the maximum likelihood estimate, so there is some Bias. However, the standard deviation is large, and could be a determing factor inn the error of the estimate. Because of the relatively large standard deviation and a small bias we perfer the maximum likelihood estiamate. Gettting a correlation close to zero is a good sign that the code is correctly implemented, since the variables x_i and y_i are independent.

4)

We now look to optimize the likelihood directly, instead of using the EM-algorithm. An analytical formula for $f_{Z_i,U_i}(z_i, u_i|\lambda_0, \lambda_1)$ must therefore be found. We first look at the situation $u_i = 1$, when $z_i = x_i$ and find the cumulative distribution.

$$
\begin{aligned}
F_{Z_i}(z_i|u_i = 1) &= \int_0^{z_i}\int_0^{x_i} f_{X_i}(x_i|\lambda_0)f_{Y_i}(y_i|\lambda_1)\mathrm{dy}_i\mathrm{dx}_i \\
&= \int_0^{z_i}\int_0^{x_i} \lambda_0\lambda_1 e^{-\lambda_0 x_i}e^{-\lambda_1 y_i}\mathrm{dy}_i\mathrm{dx}_i = \int_0^{z_i} \lambda_0 e^{-\lambda_0 x_i}(1-e^{-\lambda_1 y_i})\mathrm{dx}_i,
\end{aligned}
$$

which, after differentiating yields

$$
f_{Z_i}(z_i|u_i = 1) = \lambda_0 e^{-\lambda_0 z_i}(1-e^{-\lambda_1 z_i}).
$$

Likewize for $u_i = 0$.

$$
f_{Z_i}(z_i|u_i = 0) = \lambda_1 e^{-\lambda_1 z_i}(1-e^{-\lambda_0 z_i}).
$$

Thus, we have

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

This yields the log likelihood

$$
l(\lambda_0, \lambda_1|\mathbf{z}, \mathbf{u}) = n_0\ln(\lambda_1) + n_1\ln(\lambda_0) + \sum_{i,\ u_i=0}\left(\ln(1-e^{-\lambda_0 z_i}) - \lambda_1 z_i\right) + \sum_{i,\ u_i=1}\left(\ln(1-e^{-\lambda_1 z_i}) - \lambda_0 z_i\right),
$$

where $n_0 = \sum_{i=1}^n \mathrm{I}(u_i = 0)$ and $n_1 = \sum_{i=1}^n \mathrm{I}(u_i = 1)$ using the indicator function. The task now boils down to finding the maximum. We look for extremal values.

$$
\frac{\partial l(\lambda_0, \lambda_1|\mathbf{z}, \mathbf{u})}{\partial\lambda_0} = \frac{n_1}{\lambda_0} + \sum_{i,\ u_i=0}\frac{z_i e^{\lambda_0 z_i}}{e^{\lambda_0 z_i}-1} - \sum_{i,\ u_i=0} z_i = 0
$$

and

$$
\frac{\partial l(\lambda_0, \lambda_1|\mathbf{z}, \mathbf{u})}{\partial\lambda_1} = \frac{n_0}{\lambda_1} + \sum_{i,\ u_i=0}\frac{z_i e^{\lambda_0 z_i}}{e^{\lambda_0 z_i}-1} - \sum_{i,\ u_i=0} z_i = 0.
$$

The above equations are transendental. The solution could therefore be hard to find in a closed form. However, the hessian is negative definite for all λ_0 and λ_1. Therefore there exist a maxima, and it is unique. The hessian is given as

$$\nabla^2 l(\lambda_0, \lambda_1|\mathbf{z}, \mathbf{u}) = \begin{pmatrix} -\frac{n_1}{\lambda_0^2} - \sum_{i,\ u_i=0} \frac{z_i^2 e^{\lambda_0 z_i}}{(e^{\lambda_0 z_i}-1)^2} & 0 \\ 0 & -\frac{n_0}{\lambda_1^2} - \sum_{i,\ u_i=0} \frac{z_i^2 e^{\lambda_1 z_i}}{(e^{\lambda_1 z_i}-1)^2} \end{pmatrix}.$$

We use the optim function.

```
likelihood <- function(l){
  # Find the positions where u = 0 and u = 1
  pos.u0 <- which(u == 0)
  pos.u1 <- which(u == 1)
  n.0 <- length(pos.u0)
  n.1 <- length(pos.u1)
  # compute the log-likelihood
  log.likelihood <- n.0*log(l[2])+n.1*log(l[1]) +
                    sum(log(1-exp(-l[1]*z[pos.u0]))-l[2]*z[pos.u0]) +
                    sum(log(1-exp(-l[2]*z[pos.u1]))-l[1]*z[pos.u1])
  return(log.likelihood)
}
l_optimal <-optim(par=c(1,1),fn = likelihood, control=list(fnscale=-1,
      maxit=10000,reltol=1e-15))$par
l_EM; l_optimal
```

```
## [1] 3.465735 9.353215
```

```
## [1] 3.465735 9.353215
```

The results above were the same as for the EM-algorithm, up to 6 decimal places. The main advantage of optimalizing the likelihood instead of using the EM-algorithm, is ofcourse that you directly optimize the quantity you are after, instead of approximating it and then optimizing. Furthermore, the EM-algorithm can have a slow convergence rate and performance depending on the initial values.