TMA4300 Computer Intensive Statistical Methods Exercise 3, Spring 2023, Group 8

August Arnstad, Johan Sæbø

Contents


```
source("probAhelp.R")
source("probAdata.R")
```

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

In this exercise we are going to use a non-Gaussian time-series, which contains a sequence of length T = 100, and compare two different parameter estimators. We are considering an AR(2) model which is defined as

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

where e_t are iid random variable with zero mean and constant variance. The least sum of squared residuals (LS) and least sum of absolute residuals (LA) are obtained by minimizing the following loss functions with respect to $\boldsymbol{\beta}$:

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

The minimisers are denoted as $\boldsymbol{\beta_{LA}}$ and $\boldsymbol{\beta_{LS}}$ and the estimated residuals are defined as $\hat{e_t} = x_t - \hat{\beta_1} x_{t-1} + \hat{\beta_2} x_{t-2}$ for $t = 3, 4, ..., T$ with $\bar{e}$ being the mean. The $\hat{e}$ is re-centered by defining $\hat{\epsilon_t} = \hat{e_t} - \bar{e}$

Task A1

In this task, we are going to use the residual resampling bootstrap method to evaluate the relative performance of the two parameter estimators by estimating the variance and bias of the two estimators.

```
set.seed(123)

x <- data3A$x
T_n <- length(x)
AR2_coeff = ARp.beta.est(data3A$x, 2)
residuals_LS <- ARp.resid(x, AR2_coeff$LS)
residuals_LA <- ARp.resid(x, AR2_coeff$LA)


B <- 1500
sampels_beta_LS <- matrix(nrow = 2, ncol = B)
sampels_beta_LA <- matrix(nrow = 2, ncol = B)
for (i in 1:B) {
    start_index = ceiling(runif(1) * 99)
    x_0 = data3A$x[start_index:(start_index + 1)]

    # resample residuals
    residuals_LS_resample <- sample(x = residuals_LS, size = T_n, replace = T)
    residuals_LA_resample <- sample(x = residuals_LA, size = T_n, replace = T)

    # using ARp.filter to calculate AR(2)
    x_LS <- ARp.filter(x_0, AR2_coeff$LS, residuals_LS_resample)
    x_LA <- ARp.filter(x_0, AR2_coeff$LA, residuals_LA_resample)

    # estimate parameters
    sampels_beta_LS[, i] <- ARp.beta.est(x_LS, 2)$LS
    sampels_beta_LA[, i] <- ARp.beta.est(x_LA, 2)$LA
}


# Bias and variances
beta_LS_mean = c(mean(sampels_beta_LS[1, ]), mean(sampels_beta_LS[2, ]))
beta_LA_mean = c(mean(sampels_beta_LA[1, ]), mean(sampels_beta_LA[2, ]))
beta_LS_bias = beta_LS_mean - AR2_coeff$LS
beta_LA_bias = beta_LA_mean - AR2_coeff$LA
beta_LS_var = c(var(sampels_beta_LS[1, ]), var(sampels_beta_LS[2, ]))
beta_LA_var = c(var(sampels_beta_LA[1, ]), var(sampels_beta_LA[2, ]))


result <- data.frame(Paramater = c("Bias", "Variance"), beta_1_LS = c(beta_LS_bias[1],
    beta_LS_var[1]), beta_2_LS = c(beta_LS_bias[2], beta_LS_var[2]), beta_1_LA = c(beta_LA_bias[1],
```

```
    beta_LA_var[1]), beta_2_LA = c(beta_LA_bias[2], beta_LA_var[2]))


result[, -c(1)] = round(result[, -c(1)], digits = 5)
result
```

```
##   Paramater beta_1_LS beta_2_LS beta_1_LA beta_2_LA
## 1      Bias  -0.01406   0.00828  -0.00249   0.00196
## 2  Variance   0.00524   0.00514   0.00040   0.00039
```

The table shows that $\hat{\boldsymbol{\beta}}_{\boldsymbol{LA}}$ has smaller bias and variance than $\hat{\boldsymbol{\beta}}_{\boldsymbol{LS}}$. So even though the LS estimator is optimal for Gaussian AR(p) processes, it is not optimal in this case.

Task A2

In this task, we are going to compute a 95% prediction interval for x_{101} based on both estimators, i.e.

$$x_{101} = \hat{\beta}_1 x_{100} + \hat{\beta}_2 x_{99} + \hat{\epsilon}_t$$

two times using $\hat{\boldsymbol{\beta}}_{\boldsymbol{LA}}$ and $\hat{\boldsymbol{\beta}}_{\boldsymbol{LS}}$.

```
set.seed(111)
x_101_LS <- t(sampels_beta_LA[, sample(B, replace = TRUE)]) %*% x[T_n - 0:1] + sample(residuals_LS,
    B, replace = TRUE)
x_101_LA <- t(sampels_beta_LA[, sample(B, replace = TRUE)]) %*% x[T_n - 0:1] + sample(residuals_LA,
    B, replace = TRUE)
pred <- rbind(LS = quantile(x_101_LS, c(0.025, 0.975)), LA = quantile(x_101_LA, c(0.025,
    0.975)))

pred
```

```
##        2.5%    97.5%
## LS 7.304069 23.17049
## LA 7.258786 23.26917
```

Both $\hat{\boldsymbol{\beta}}_{\boldsymbol{LA}}$ and $\hat{\boldsymbol{\beta}}_{\boldsymbol{LS}}$ gave similar CI, even though there were a significant difference in bias and variance.

Problem B: Permutation Test

Bilirubin is a breakdown product of haemoglobin, which is a principal component of red blood cells. If the liver has suffered degeneration, if the decomposition of haemoglobin is elevated, or if the gall bladder has been destroyed, large amounts of bilirubin can accumulate in the blood, leading to jaundice. This data is taken from Jørgensen (1993) and contains the measurement of the concentration of bilirubin (mg/dL) in blood samples taken from three young men.

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

```
##   meas pers
## 1 0.14   p1
## 2 0.20   p1
## 3 0.23   p1
## 4 0.27   p1
## 5 0.27   p1
## 6 0.34   p1
```

The "meas"-columns gives the concentrations of bilirubin (mg/dL), and the "pers"-column denotes which person the measurement belongs to.

Task B1

To inspect the data one can take the logarithms of the concentrations for each individual for a box plot.

```
library(ggplot2)
ggplot(data = bilirubin, aes(x = pers, y = log(meas), fill = pers)) + geom_boxplot() +
    labs(x = "Person", y = "log(meas)") + ylim(c(-2, 0))
```

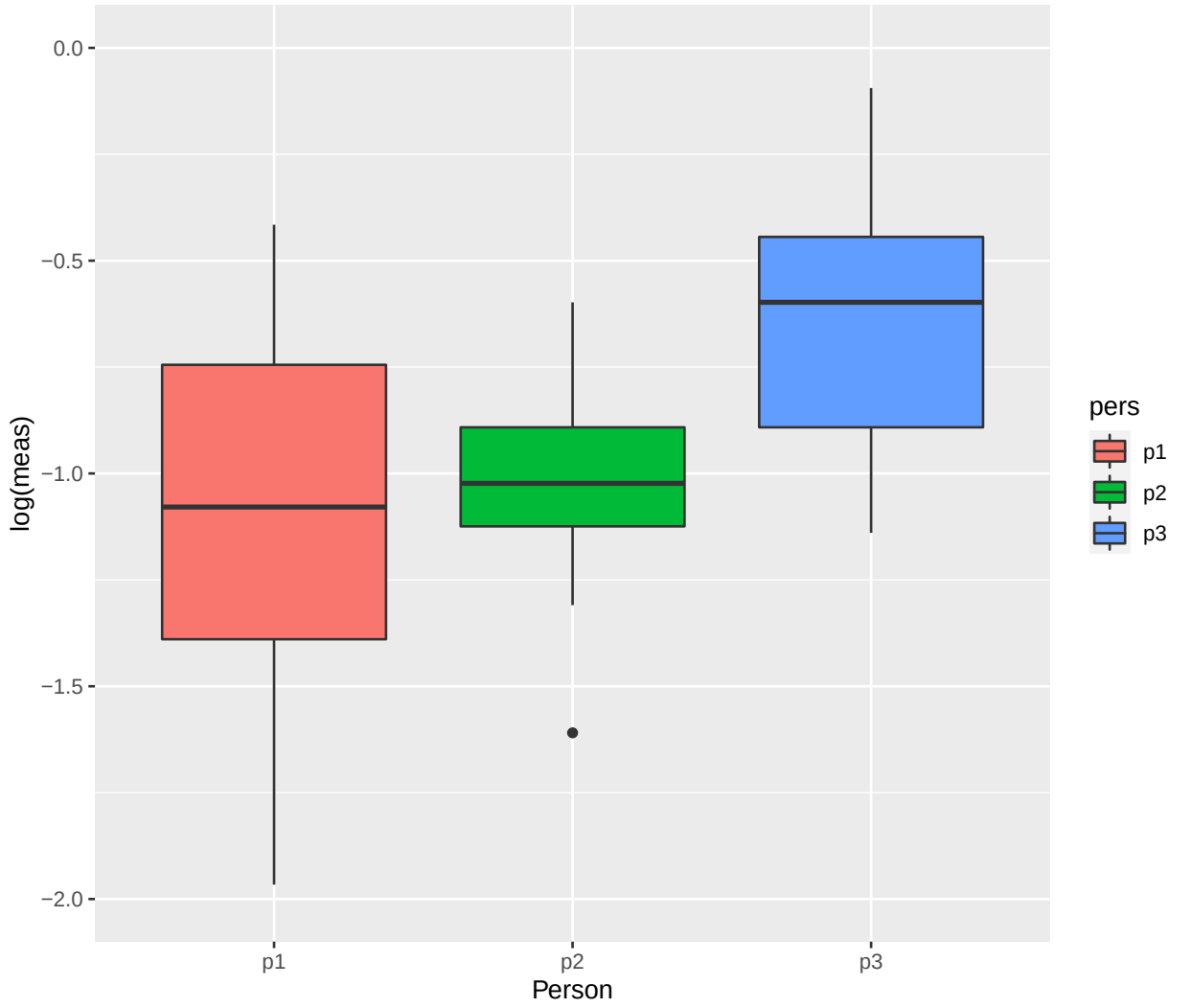

Figure 1: A boxplot of the logarithms of the concentrations of bilirubin for three young men.

Then one can fit the regression model

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

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

```
# Fitting the linear regression model
mod = lm(log(meas) ~ pers, data = bilirubin)
mod_sum = summary(mod)
Fval = mod_sum$fstatistic[1]  #Saving the the F-statistic value fro later
mod_sum
```

```
##
## Call:
## lm(formula = log(meas) ~ pers, data = bilirubin)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -0.87215 -0.26246  0.03131  0.20236  0.67844
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.09396    0.11747  -9.312 9.15e-10 ***
## persp2       0.06412    0.17023   0.377   0.7095
## persp3       0.46482    0.18104   2.568   0.0163 *
## ---
## 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:  0.2201, Adjusted R-squared:  0.1602
## F-statistic:  3.67 on 2 and 26 DF,  p-value: 0.03946
```

Using the F-test to test the null hypothesis that $H_0 : \beta_1 = \beta_2 = \beta_3$, the output shows a F-statistic of 3.67 and a p-value of 0.03946. Given a normal significance level of 0.05, the p-value is lower and therefore the hypothesis of $\beta_1 = \beta_2 = \beta_3$ is rejected.

Task B2

Writing a function *permTest*() which generates a permutation of the data between the three individuals, fits the regression model (1) and returns the value of the F-statistic for testing the null hypothesis

$$H_0 : \beta_1 = \beta_2 = \beta_3$$

```
set.seed(111)
permTest = function(df) {
    df$pers = sample(bilirubin$pers, size = length(bilirubin$pers), replace = FALSE)  #Generate sample
    mod = lm(log(meas) ~ pers, data = df)  #The log regression model
    return(summary(mod)$fstatistic[1])  #Return F-statistic
}
```

Task B3

```
set.seed(111)
n = 999  #Number of samples
Fvals = c(rep(0, n))  #Init space

for (i in 1:n) {
    Fvals[i] = permTest(bilirubin)  #Add 999 generated F-values
}

counter = c(rep(0, n))  #init a list of larger F-values than theoretical

# Check if generated F-val is bigger than theoretical value, if so: =1
for (i in 1:n) {
    if (Fvals[i] > Fval) {
        counter[i] = 1
    }
```

```
}
# The p-value, number of F-val greater than the theoretic value / number of
# samples
p = sum(counter)/n
cat("p-value:", p)
```

```
## p-value: 0.03503504
```

By using this sample for the p-value for Fval, one get $p = 0.03504$. This is a tiny bit smaller than the previous p-value of 0.03946. Both p-values suggest to reject the H_0 hypothesis given a significance level of 0.05. The statistical evidence suggest that there is a difference in level of bilirubin for the three men.

```
set.seed(111)
Fvals = data.frame(x = Fvals)
df1 = as.numeric(mod_sum$fstatistic[2])
df2 = as.numeric(mod_sum$fstatistic[3])
ggplot(data = Fvals, aes(x = x)) + geom_histogram(aes(y = after_stat(density)), bins = 100,
    color = "black") + labs(x = "F-value", y = "Density") + geom_vline(xintercept = as.numeric(Fval),
    color = "blue") + geom_function(fun = df, args = c(df1, df2), color = "red")
```

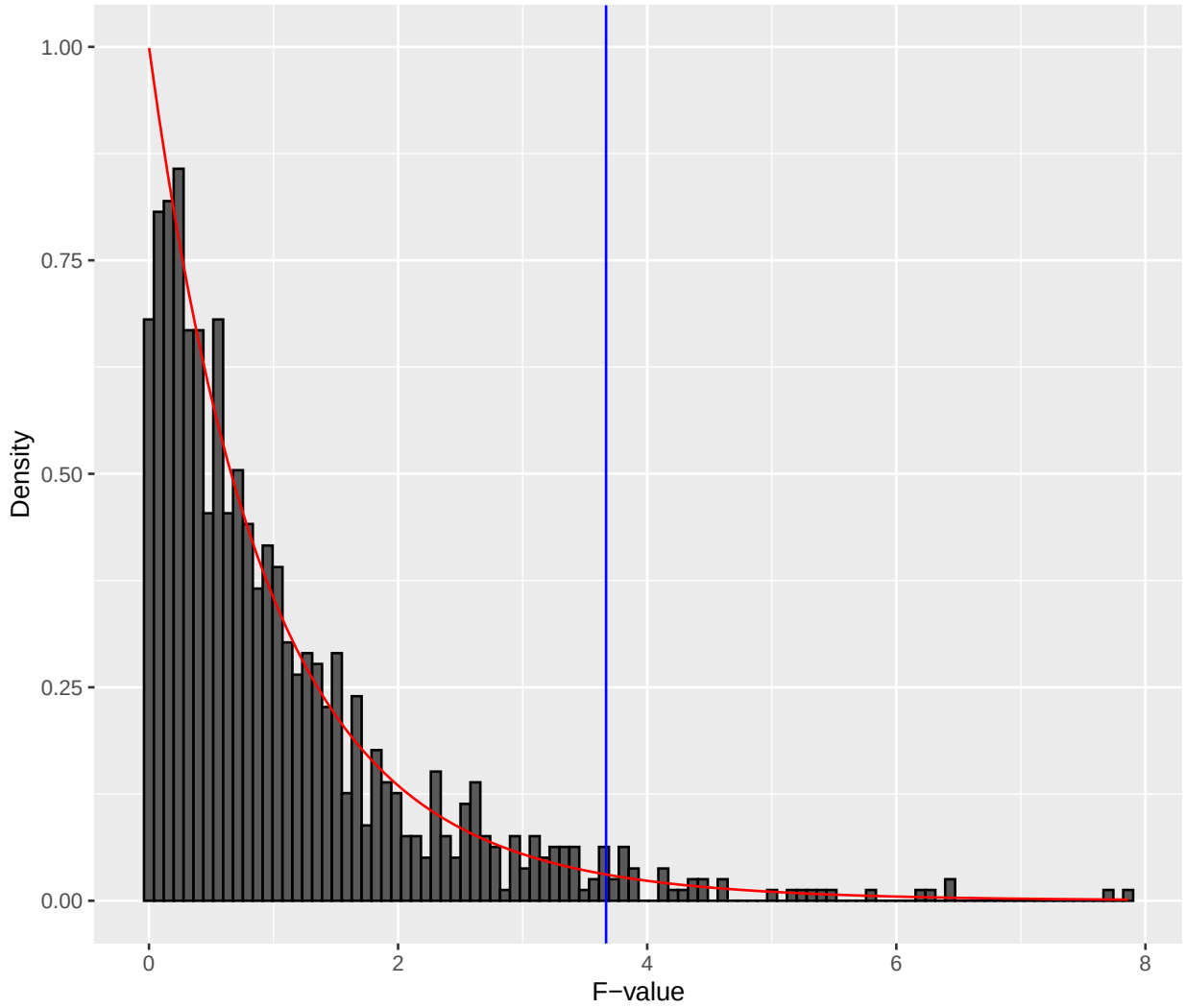

Figure 2: Histogram of the generated and theoretic F-distribution. The blue line is the estimated F-value from task B2.

Problem C: The EM algorithm and Bootstrapping

We have two independent random vectors $\mathbf{x}$ and $\mathbf{y}$ both elements in $\mathbb{R}^n$ with each $x_i \sim \exp(\lambda_0)$ independently and $y_i \sim \exp(\lambda_1)$ independently where $i = 1, ..., n$. In this task the objective is to find the maximum likelihood estimate of the quantities λ_0 and λ_1 from the Expectation Maximization(EM) algorithm based on the information we obtain by observing

$$z_i = \max(x_i, y_i) \qquad \forall i = 1, ..., n u_i = \mathbb{I}_{x_i \geq y_i} \qquad \forall i = 1, ..., n$$

from the unobserved variables $\mathbf{x}$ and $\mathbf{x}$

Task C1

As the expectation step in the EM algorithm, we find the expectation of the log-likelihood function given the data $\mathbf{z}, \mathbf{u}$ and some initial guess or the previous iteration of λ_0 and λ_1. We must now find the log-likelihood.

The random vectors are independent and both exponentially distributed. Hence, we obtain easily the joint distribution

$$f(\mathbf{x},\ \mathbf{y}|\lambda_0,\lambda_1) = f_X(\mathbf{x}|\lambda_0)f_Y(\mathbf{y}|\lambda_1) = \Pi_{i=1}^n \lambda_0 e^{-\lambda_0 X_i}\lambda_1 e^{-\lambda_1 Y_i} = \lambda_0 e^{-\lambda_0 \sum_{i=1}^n X_i}\lambda_1 e^{-\lambda_1 \sum_{i=1}^n Y_i}$$

from this we take the logarithm to obtain

$$\ln\left(f(\mathbf{x},\ \mathbf{y}|\lambda_0,\lambda_1)\right) = n\ln(\lambda_0) + n\ln(\lambda_1) - \lambda_0\sum_{i=1}^n X_i - \lambda_1\sum_{i=1}^n Y_i$$

for observations $\mathbf{x}$ and $\mathbf{y}$ this is infact the log-likelihood of λ_0 and λ_1 given $\mathbf{x,\ y}$. Now we must find the expectation of this and we proceed as follows

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

where everything follows from linearity of expectation and $(\lambda_0^{(t)},\lambda_1^{(t)})$ corresponds to the value of (λ_0,λ_1) at the previous iteration or the initial guess. It remains to find expressions for $\mathbb{E}[x_i|\lambda_0^{(t)},\lambda_1^{(t)},u_i,z_i]$ and $\mathbb{E}[y_i|\lambda_0^{(t)},\lambda_1^{(t)},u_i,z_i]$, and we can now take into account that we are given u. This gives us two cases for both x_i and y_i based on what the logical value is$(u_i \in 0,1)$, which we can handle separately. The derivation of this expectation is extremely similar for X_i and Y_i so we will first do it for X_i and then we can easily obtain the expectation for Y_i as well.

$$\mathbb{E}[X_i|\lambda_0^{(t)},\lambda_1^{(t)},u_i,z_i] = \begin{cases} z_i & u_i = 1 \\ \mathbb{E}[X_i|\lambda_0^{(t)},\lambda_1^{(t)},X_i = x_i < z_i] & u_i = 0 \end{cases}$$

which can also be written as

$$\mathbb{E}[X_i|\lambda_0^{(t)},\lambda_1^{(t)},u_i,z_i] = z_i u_i + (1-u_i)\mathbb{E}[X_i|\lambda_0^{(t)},\lambda_1^{(t)},X_i = x_i < z_i]$$

When one considers the expression

$$\mathbb{E}[X_i|\lambda_0^{(t)},\lambda_1^{(t)},X_i = x_i < z_i] = \int_0^{z_i} x_i f_{X_i}(x_i|\lambda_0^{(t)},\lambda_1^{(t)},x_i < z_i)dx_i$$

the problem is to determine $f_{X_i}(x_i|\lambda_0^{(t)},\lambda_1^{(t)},x_i < z_i)$. We go about it as follows

$$f_{X_i}(x_i|\lambda_0^{(t)},\lambda_1^{(t)},X_i = x_i < z_i) = \frac{f_{X_i<z_i}(x_i < z_i|\lambda_0^{(t)},\lambda_1^{(t)},X_i = x_i)f_{X_i}(x_i)}{\int f_{X_i<z_i}(x_i < z_i|X_i = x_i)f_X(x_i)dx_i} = \frac{\mathbb{I}_{X_i<z_i}f_{X_i}(x_i)}{\int_0^\infty \mathbb{I}_{X_i<z_i}f_{X_i}(x_i)}$$

Now, since this is the case when $u_i = 0$, it will always be the case that $X_i < z_i$. Consequently the indicator function in the numerator will always take the value 1 and the indicator function in the denominator will take the value 1 up to z_i, which can be incorporated by changing the integral boundaries. This results in

$$f_{X_i}(x_i|\lambda_0^{(t)},\lambda_1^{(t)},X_i = x_i < z_i) = \frac{f_{X_i}(x_i)}{\int_0^{z_i} f_{X_i}(x_i)} = \frac{\lambda_0^{(t)}e^{-\lambda_0^{(t)}x_i}}{1-e^{-\lambda_0^{(t)}z_i}}$$

i.e. a truncated distribution of X_i such that $X_i < z_i$. Going further one can compute the previous integral, that is

$$\mathbb{E}[x_i|\lambda_0^{(t)},\lambda_1^{(t)},X_i=x_i<z_i]=\int_0^{z_i} x_i f_X(x_i|\lambda_0^{(t)},\lambda_1^{(t)},X_i=x_i<z_i)dx_i=\int_0^{z_i} x_i\frac{\lambda_0^{(t)}e^{-\lambda_0^{(t)}x_i}}{1-e^{-\lambda_0^{(t)}z_i}}dx_i$$

which gives

$$\mathbb{E}[x_i|\lambda_0^{(t)},\lambda_1^{(t)},X_i=x_i<z_i]=\frac{\lambda_0^{(t)}}{1-e^{-\lambda_0^{(t)}z_i}}\frac{1-e^{\lambda_0^{(t)}z_i}(\lambda_0^{(t)}z_i+1)}{\lambda_0^{(t)}}=\frac{1}{\lambda_0^{(t)}}-\frac{z}{e^{\lambda_0^{(t)}z_i}-1}$$

Now for y_i the only difference is that

$$\mathbb{E}[Y_i|\lambda_0^{(t)},\lambda_1^{(t)},u_i,z_i]=(1-u_i)z_i+u_i\mathbb{E}[Y_i|\lambda_0^{(t)},\lambda_1^{(t)},z_i]=\begin{cases}\mathbb{E}[Y_i|\lambda_0^{(t)},\lambda_1^{(t)},Y_i=y_i<z_i] & u_i=1\\ z_i & u_i=0\end{cases}$$

which by the same derivation ultimately yields

$$\mathbb{E}[Y_i|\lambda_0^{(t)},\lambda_1^{(t)},Y_i=y_i<z_i]=\frac{\lambda_1^{(t)}}{1-e^{-\lambda_1^{(t)}z_i}}\frac{1-e^{\lambda_1^{(t)}z_i}(\lambda_1^{(t)}z_i+1)}{\lambda_1^{(t)}}=\frac{1}{\lambda_1^{(t)}}-\frac{z}{e^{\lambda_1^{(t)}z_i}-1}$$

Going back to the task at hand, inserting what we have now asserted, we get

$$\begin{aligned}\mathbb{E}[ln\left(f(\mathbf{x},\,\mathbf{y}|\lambda_0,\lambda_1)\right)|\lambda_0^{(t)},\lambda_1^{(t)},\mathbf{u},\,\mathbf{z}] &= n\ln(\lambda_0)+n\ln(\lambda_1)\\ &-\lambda_0\sum_{i=1}^{n}\left[z_iu_i+(1-u_i)\left(\frac{1}{\lambda_0^{(t)}}-\frac{z}{e^{\lambda_0^{(t)}z_i}-1}\right)\right]\\ &-\lambda_1\sum_{i=1}^{n}\left[(1-u_i)z_i+u_i\left(\frac{1}{\lambda_1^{(t)}}-\frac{z}{e^{\lambda_1^{(t)}z_i}-1}\right)\right]\end{aligned}\tag{2}$$

which is what we wanted to show.

Task C2

The previous task is the expectation step of the EM-algorithm. The maximization step is now to maximize this expectation, with respect to the parameters λ_0 and λ_1. We will do this by differentiating the expectation, equating to zero and use a fix point iteration (recursion) based on an initial guess $(\lambda_0^{(t)]},\lambda_1^{(t)]})$. Firstly,

$$\frac{d}{d\lambda_0}\mathbb{E}[ln\left(f(\mathbf{x},\,\mathbf{y}|\lambda_0,\lambda_1)\right)|\lambda_0^{(t)},\lambda_1^{(t)},\mathbf{u},\,\mathbf{z}]=\frac{n}{\lambda_0}-\sum_{i=1}^{n}\left[z_iu_i+(1-u_i)\left(\frac{1}{\lambda_0^{(t)}}-\frac{z}{e^{\lambda_0^{(t)}z_i}-1}\right)\right]=0$$

and

$$\frac{d}{d\lambda_1}\mathbb{E}[ln\left(f(\mathbf{x},\,\mathbf{y}|\lambda_0,\lambda_1)\right)|\lambda_0^{(t)},\lambda_1^{(t)},\mathbf{u},\,\mathbf{z}]=\frac{n}{\lambda_1}-\sum_{i=1}^{n}\left[(1-u_i)z_i+u_i\left(\frac{1}{\lambda_1^{(t)}}-\frac{z}{e^{\lambda_1^{(t)}z_i}-1}\right)\right]=0$$

which we rearranging to get the iteration

$$\hat{\lambda_0}=\frac{n}{\sum_{i=1}^{n}\left[z_iu_i+(1-u_i)\left(\frac{1}{\lambda_0^{(t)}}-\frac{z}{e^{\lambda_0^{(t)}z_i}-1}\right)\right]},\qquad \hat{\lambda_1}=\frac{n}{\sum_{i=1}^{n}\left[(1-u_i)z_i+u_i\left(\frac{1}{\lambda_1^{(t)}}-\frac{z}{e^{\lambda_1^{(t)}z_i}-1}\right)\right]}$$

With the iteration obtained, we are ready to implement the EM algorithm. A generic convergence criteria is set to be when the expectation of the current and previous iterations are small(10^{-8}). As the expectation in this case is just a number, this should be a sufficient criteria. The implementation follows

```r
set.seed(1234)
u = read.delim("u.txt")[, 1]
z = read.delim("z.txt")[, 1]

n = length(u)
```

```r
set.seed(1234)
expectation <- function(lambda_0, lambda_1, u, z, n) {
    return(n * (log(lambda_0) + log(lambda_1)) - lambda_0 * sum(u * z + (1 - u) *
        (1/lambda_0 - z/(exp(lambda_0 * z) - 1))) - lambda_1 * sum((1 - u) * z +
        u * (1/lambda_1 - z/(exp(lambda_1 * z) - 1))))
}

EM_algorithm <- function(lambda_0_init, lambda_1_init, u, z, n, crit = 1e-08) {

    lambda_0 = lambda_0_init
    lambda_1 = lambda_1_init
    lambda_0_list = c(lambda_0_init)
    lambda_1_list = c(lambda_1_init)

    expected_prev = 1e+10
    expected_curr = expectation(lambda_0, lambda_1, u, z, n)

    while ((abs(expected_curr - expected_prev) > crit)) {

        expected_prev = expected_curr
        lambda_0 = n/(sum(u * z + (1 - u) * (1/lambda_0 - z/(exp(lambda_0 * z) -
            1))))
        lambda_1 = n/(sum((1 - u) * z + u * (1/lambda_1 - z/(exp(lambda_1 * z) -
            1))))

        lambda_0_list = c(lambda_0_list, lambda_0)
        lambda_1_list = c(lambda_1_list, lambda_1)

        expected_curr = expectation(tail(lambda_0, n = 1), tail(lambda_1, n = 1),
            u, z, n)
    }

    return(list(lambda_0 = lambda_0_list, lambda_1 = lambda_1_list, n = n))
}

EM_run = EM_algorithm(1, 1, u, z, n)
lambda_0_est <- tail(EM_run$lambda_0, n = 1)
lambda_1_est <- tail(EM_run$lambda_1, n = 1)


cat("Estimate of lambda_0:", lambda_0_est, "\n")
```

```
## Estimate of lambda_0: 3.463089
```

```r
cat("Estimate of lambda_1:", lambda_1_est, "\n")
```

```
## Estimate of lambda_1: 9.334877
```

```r
par(mfrow = c(1, 1), mar = c(5, 4, 4, 0.8))
plot(EM_run$lambda_1, type = "l", xlab = "Iterations", ylab = expression(lambda),
    col = "magenta", lwd = 3)
lines(EM_run$lambda_0, col = "cyan", lwd = 3)
legend("bottomright", c(expression(lambda[1]), expression(lambda[0])), col = c("magenta",
    "cyan"), lty = 1, lwd = 3)
mtext("Convergence plot for EM algorithm", side = 3, line = -2, outer = TRUE)
```

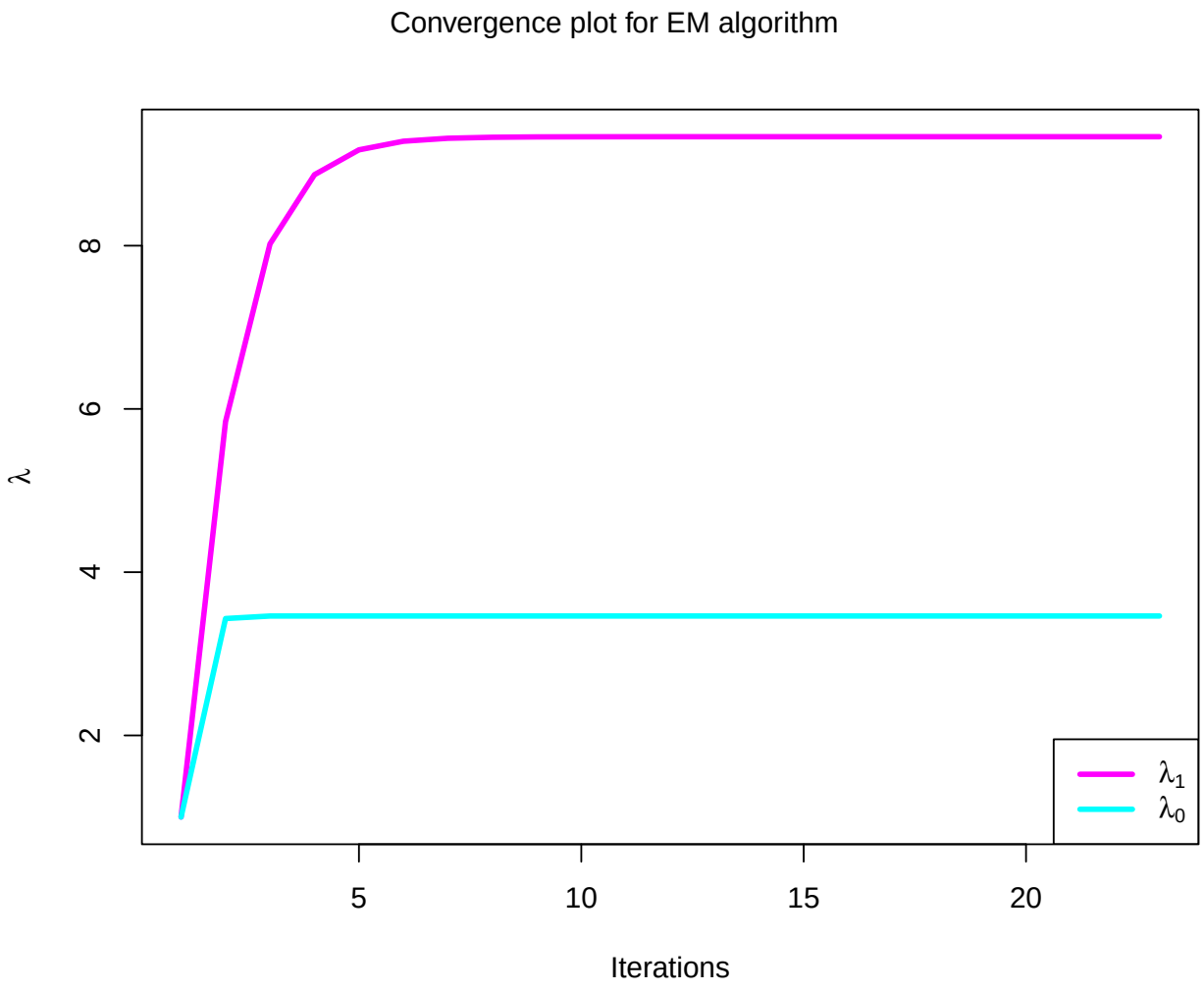

Figure 3: Convergence plot of the estimates of λ_0 and λ_1

The obtained estimates are $\lambda_0 = 3.4631$ and $\lambda_1 = 9.3349$. We see from the convergence plot that the algorithm terminates very quickly and does not need many iterations to become stable. The choice of inital values do not affect the algorithms efficiency if they are reasonable. To empirically check the results, we can realize some values of x and y and the compare the z-values from these to the ones we have observed as below

```r
set.seed(1234)

x_rand = rexp(1e+05, rate = lambda_0_est)
y_rand = rexp(1e+05, rate = lambda_1_est)
z_rand = ifelse(x_rand >= y_rand, x_rand, y_rand)

hist(z, freq = F, breaks = 20, col = "cyan", xlab = "z", ylab = "f(z)", main = "")
z_draws = hist(z_rand, breaks = 100, plot = F)
lines(z_draws$mids, z_draws$density, col = "magenta", lwd = 3)
legend("topright", c("Observed z", "Drawn z with estimated lambdas"), col = c("cyan",
    "magenta"), lty = 1)
```

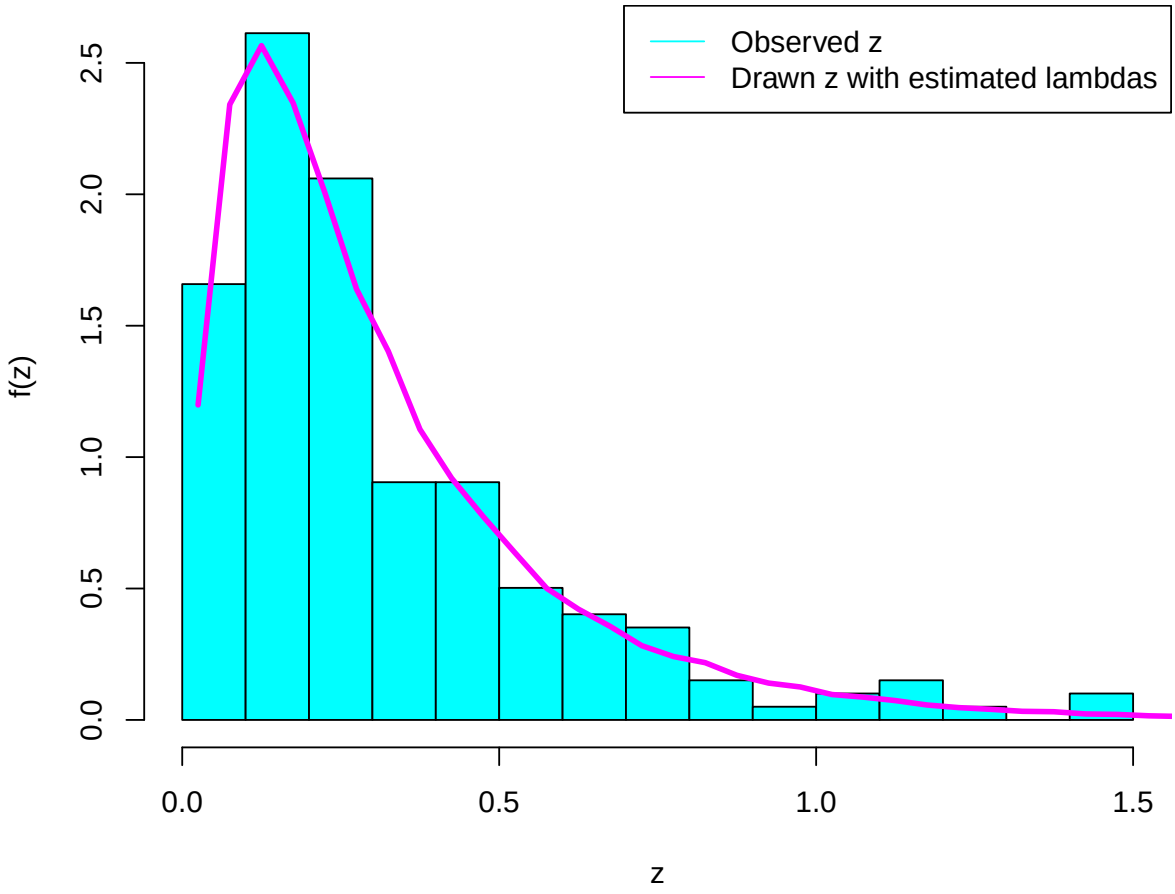

Figure 4: Histogram of the observed z data, with the distribution of drawn z samples from the calculated parameters λ_0, λ_1

The histogram gives supports the parameter values we found, and it is reasonable to assume that our data was in fact observed from variables x and y distributed as above.

Task C3

We now use the bootstrap method to extract some properties of our estimates $(\hat{\lambda_0}, \hat{\lambda_1})$, such as their standard deviation, biases and correlation the correlation between them. A pseudocode for the bootstrap algorithm is presented below

```
Extract data
Choose initial values of lambda_0 and lambda_1
for (b in 1:B){
  sample sets u_b and z_b of the same length as u and z with replacement from the datasets u and z.
  obtain parameter estimates (lambda_0, lambda_1) from the function EM_algorithm
  store them advantageously for later use
}

Calculate the properties of interest form the sample:
  standard deviation for each lambda of samples
  bias of samples for each lambda of samples
  correlation between lambdas
```

With the pseudocode in mind an implementation of bootstrapping and extracting the properties of interest is found below. We use 5000 bootstrap samples which might even be a bit excessive, but the algorithm is fast so we can afford it.

```
set.seed(1234)

B = 5000
lambda_0_samps = c()
lambda_1_samps = c()

for (b in 1:B) {
    index <- sample(1:n, size = n, replace = TRUE)
    u_hat = u[index]
    z_hat = z[index]
    EM_boot <- EM_algorithm(1, 1, u_hat, z_hat, n)
    lambda_0_samps = c(lambda_0_samps, tail(EM_boot$lambda_0, n = 1))
    lambda_1_samps = c(lambda_1_samps, tail(EM_boot$lambda_1, n = 1))
}


mean_0 = mean(lambda_0_samps)
std_0 = sd(lambda_0_samps)
bias_0 = mean_0 - lambda_0_est

mean_1 = mean(lambda_1_samps)
std_1 = sd(lambda_1_samps)
bias_1 = mean_1 - lambda_1_est

ci_0 = quantile(lambda_0_samps, probs = c(0.025, 0.975))
ci_1 = quantile(lambda_1_samps, probs = c(0.025, 0.975))

par(mfrow = c(2, 1), mar = c(5, 3, 3, 0.8))
plot(lambda_0_samps, col = "magenta", cex = 0.7, xlab = "Bootstrap sample", ylab = expression(lambda[0]))
abline(h = mean_0, col = "yellow", lwd = 3)
plot(lambda_1_samps, col = "cyan", cex = 0.7, xlab = "Bootstrap sample", ylab = expression(lambda[1]))
```

```
abline(h = mean_1, col = "yellow", lwd = 3)
legend("top", inset = c(0, -0.5), xpd = TRUE, c("mean", expression(lambda[0]), expression(lambda[1])),
    col = c("yellow", "magenta", "cyan"), lty = 1, lwd = 3, cex = 0.7)
```

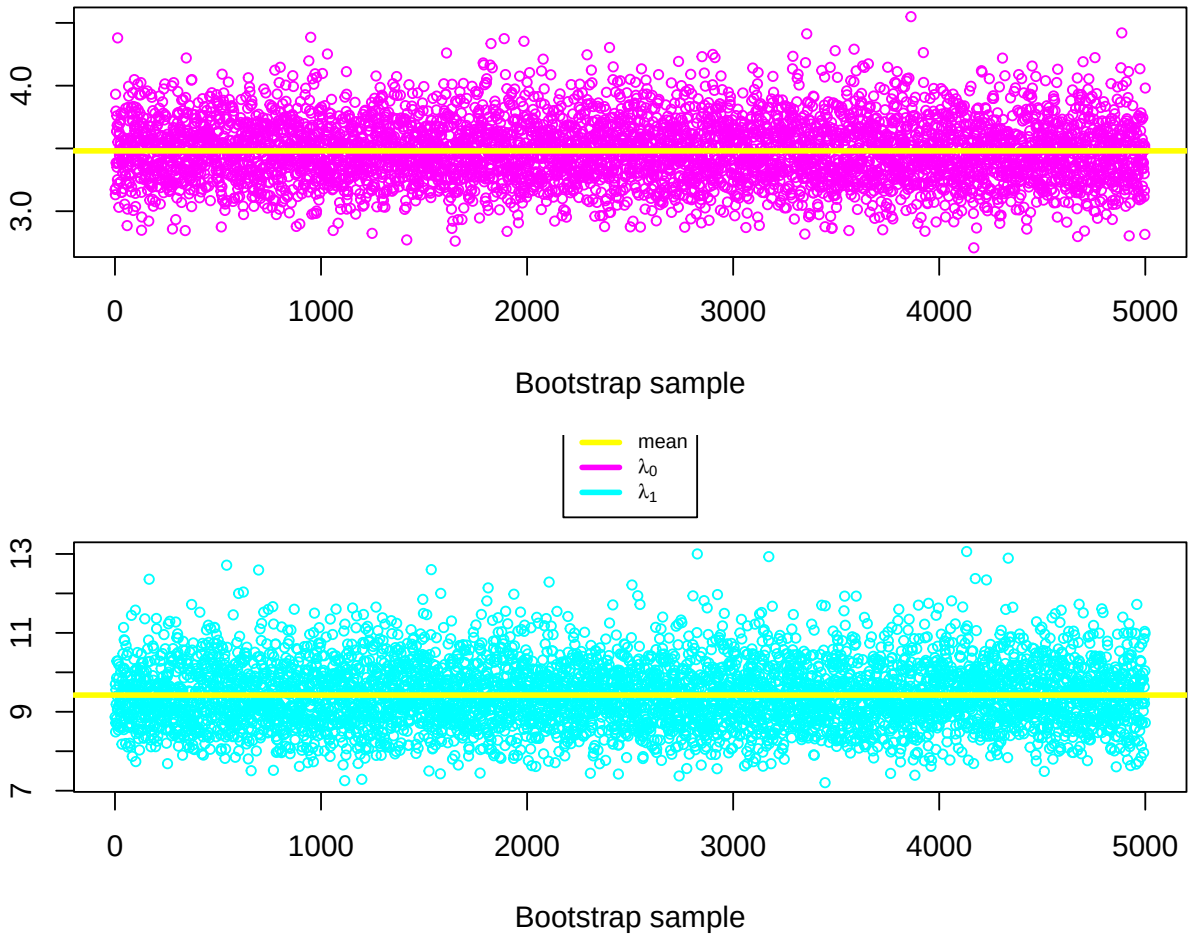

Figure 5: Bootstrap samples of λ_0 and λ_1 with the corresponding mean

```
par(mfrow = c(1, 1), mar = c(5, 4, 4, 0.8))
plot(lambda_0_samps, lambda_1_samps, col = "blue", xlim = c(1, 8), ylim = c(5, 14),
    xlab = expression(lambda[0]), ylab = expression(lambda[1]), main = "Scatterplot to check for correla
```

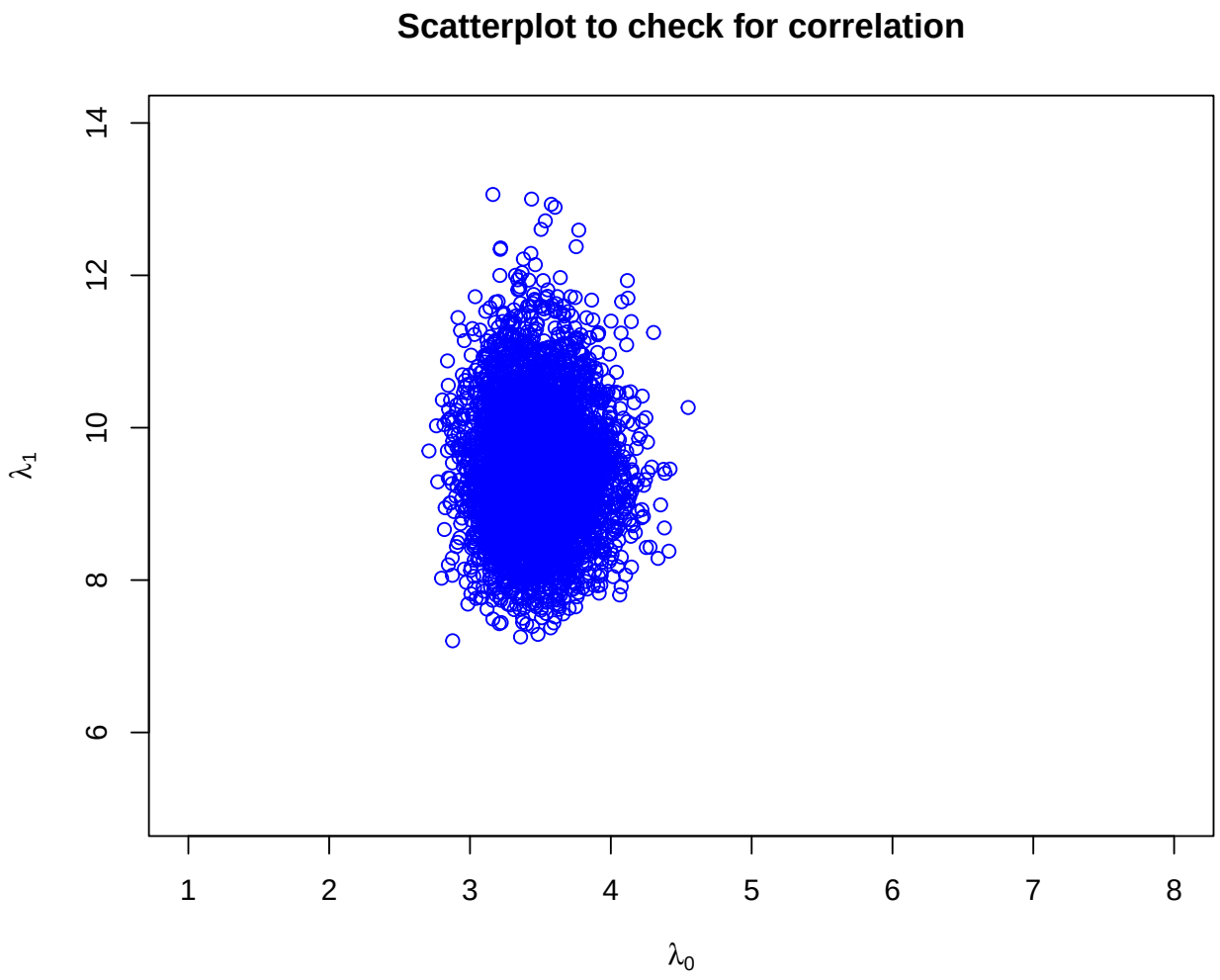

Figure 6: Scatterplot of λ_0 and λ_1

```
hist(lambda_0_samps, xlim = c(2, 14), col = "magenta", breaks = 40, ylim = c(1, 600),
    main = "Histogram of Bootstrap samples", xlab = "Samples")
abline(v = mean_0, col = "yellow")
abline(v = ci_0[1], col = "green")
abline(v = ci_0[2], col = "green")
hist(lambda_1_samps, add = T, col = "cyan", breaks = 40)
abline(v = mean(lambda_1_samps), col = "yellow", lwd = 4)
abline(v = ci_1[1], col = "green", lwd = 2)
abline(v = ci_1[2], col = "green", lwd = 2)
legend("topright", c("mean", "95% CI", expression(lambda[0]), expression(lambda[1])),
    col = c("yellow", "green", "magenta", "cyan"), lty = 1, lwd = 3)
```

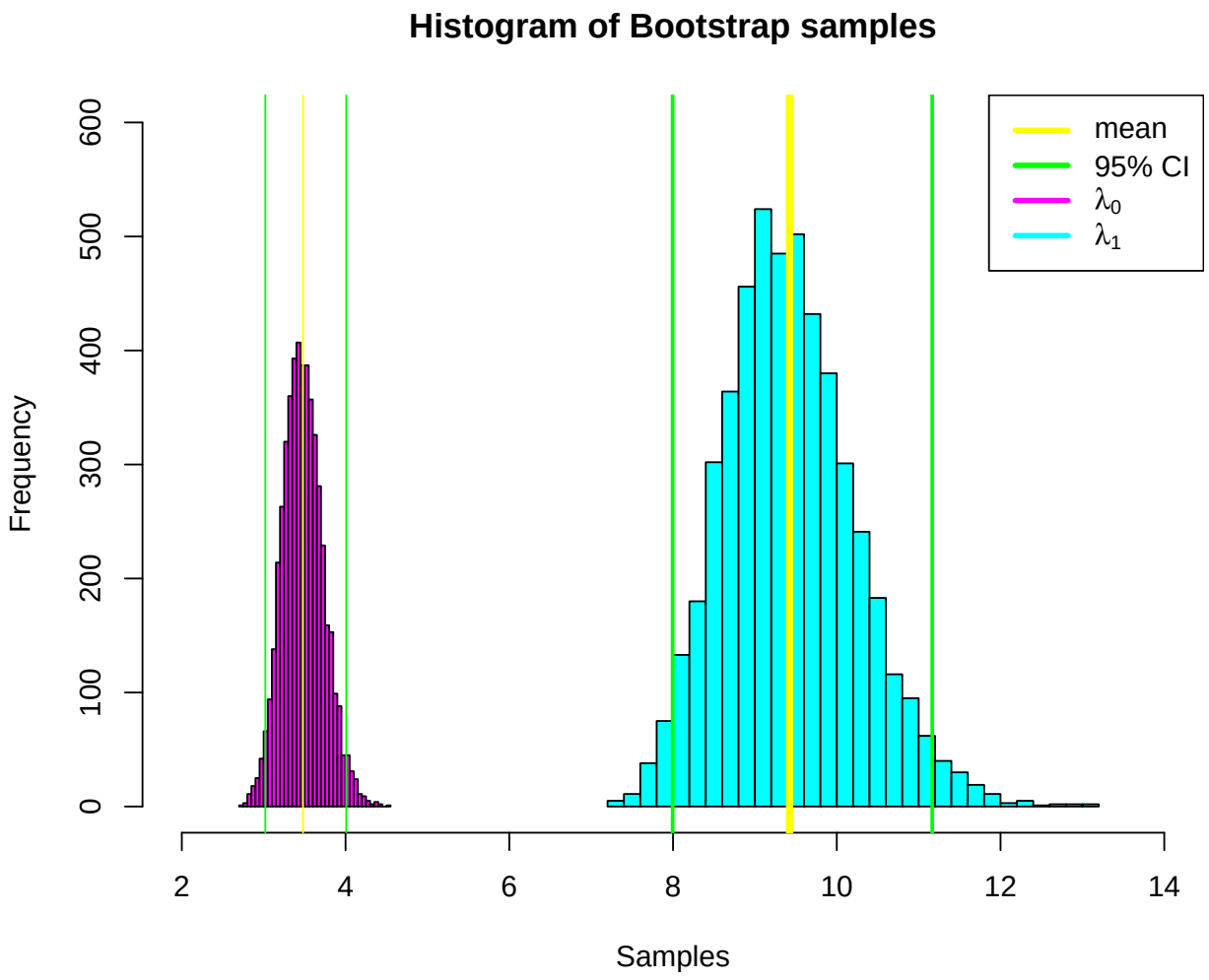

Figure 7: Histogram of λ_0 and λ_1 from the bootstrap plotted with a CI and the means

```
lambda_tab <- matrix(rep(2, times = 6), ncol = 3, byrow = TRUE)
colnames(lambda_tab) <- c("Mean", "Std", "Bias")
rownames(lambda_tab) <- c("lambda_0", "lambda_1")
lambda_tab[1, 1] = mean_0
lambda_tab[1, 2] = std_0
lambda_tab[1, 3] = bias_0
lambda_tab[2, 1] = mean_1
lambda_tab[2, 2] = std_1
lambda_tab[2, 3] = bias_1

lambda_tab <- as.table(lambda_tab)
lambda_tab
```

```
##                Mean        Std       Bias
## lambda_0 3.48066579 0.24868022 0.01757682
## lambda_1 9.42321008 0.80469496 0.08833317
```

```
cat("Correlation between lambda_0 and lambda_1: ", cor(lambda_0_samps, lambda_1_samps),
    "\n")
```

```
## Correlation between lambda_0 and lambda_1:  -0.009598811
```

From the bootstrap we obtain the mean of the estimators to be $(\lambda_0, \lambda_1) = (3.4807, 9.4232)$ with a standard deviation of $\sigma(\lambda_0) = 0.2487$ and $\sigma(\lambda_1) = 0.8047$. The biases, $\text{Bias}(\lambda_0) = 0.0176$ and $\text{Bias}(\lambda_1) = 0.0883$, of the estimators are approximately ten times smaller than the standard deviations and if one aims at lowering the bias even more it would most likely cause the variance to go up. Because of this it seems reasonable to accept the small bias to be able to keep the variance low. The correlation of the λ's are very small, something that is also reflected in the scatterplot.

Task C4

The question now is if it is possible to derive an analytical expression for $f_{U_i,Z_i}(u_i, z_i|\lambda_0, \lambda_1)$. We move forward in a similar manner as before, by considering the only two values of the logical variable u_i. If $u_i = 1$ then $X_i \geq Y_i$ and $X_i = z_i$ so we consider both cases separately. The cumulative distribution is fairly easy for exponential distributions and takes the form

$$\begin{aligned}
\mathbb{P}(Z_i \leq z_i, U_i = 1|\lambda_0, \lambda_1) &= \lambda_0\lambda_1 \int_0^{z_i}\int_0^{x_i} e^{-\lambda_0 x_i} e^{-\lambda_1 y_i} dy_i dx_i \\
&= \lambda_0\lambda_1 \int_0^{z_i} e^{-\lambda_0 x_i} \frac{1 - e^{-\lambda_1 x_i}}{\lambda_1} dx_i \\
&= \lambda_0 \int_0^{z_i} e^{-\lambda_0 x_i} - e^{-x_i(\lambda_0+\lambda_1)} dx_i \\
&= \lambda_0 \left(\frac{1 - e^{-\lambda_0 z_i}}{\lambda_0} + \frac{e^{-z_i(\lambda_0+\lambda_1)} - 1}{\lambda_0 + \lambda_1} \right) \\
&= \frac{\lambda_1}{\lambda_0 + \lambda_1} - e^{-\lambda_0 z_i} + \frac{\lambda_0}{\lambda_0 + \lambda_1} e^{-z_i(\lambda_0+\lambda_1)}
\end{aligned} \tag{3}$$

To obtain the density we simply differentiate this with respect to z_i, resulting in

$$\begin{aligned}
f_{Z_i}(Z_i = z_i, U_i = 1|\lambda_0, \lambda_1) &= \frac{d}{dz_i}\mathbb{P}(Z_i \leq z_i, U_i = 1|\lambda_0, \lambda_1) \\
&= \frac{d}{dz_i}\left(\frac{\lambda_1}{\lambda_0 + \lambda_1} - e^{-\lambda_0 z_i} + \frac{\lambda_0}{\lambda_0 + \lambda_1} e^{-z_i(\lambda_0+\lambda_1)} \right) \\
&= \lambda_0 e^{-\lambda_0 z_i} - \lambda_0 e^{-z_i(\lambda_0+\lambda_1)} \\
&= \lambda_0 e^{-\lambda_0 z_i}(1 - e^{-\lambda_1 z_i})
\end{aligned} \tag{4}$$

When considering $u_i = 0$ one quickly realizes that the same derivation holds only with the roles of λ_0 and λ_1 switched, as we have seen in earlier derivations. This results in

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

and thus we have covered all possible values for U_i. The complete probability function is thus as follows

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

With this in hand, we can find the joint distribution by recalling the independence and observing

$$f(\mathbf{z},\ \mathbf{u}|\lambda_0,\lambda_1) = \Pi_{i=1}^n \lambda_0 e^{-\lambda_0 z_i}(1-e^{-\lambda_1 z_i})\mathbb{I}_{u_i=1}\Pi_{i=1}^n \lambda_1 e^{-\lambda_1 z_i}(1-e^{-\lambda_0 z_i})\mathbb{I}_{u_i=0}$$

As we did in task C1) we take the logarithm of this, conserving the maximum by monotonicity, and arrive at

$$\begin{aligned}\ln(f(\mathbf{z},\ \mathbf{u}|\lambda_0,\lambda_1)) = n_1\ln(\lambda_0) + n_0\ln(\lambda_1) \\ - \sum_{i=1}^n \left[\lambda_0 z_i - \ln(1-e^{-\lambda_1 z_i})\right]\mathbb{I}_{u_i=1} \\ - \sum_{i=1}^n \left[\lambda_1 z_i - \ln(1-e^{-\lambda_0 z_i})\right]\mathbb{I}_{u_i=0}\end{aligned} \tag{5}$$

where n_1 and n_0 are the number of occurences where $u_i = 1$ and $u_i = 0$ respectively. As before, this corresponds to the log-likelihood function, which we want to maximize. Proceeding in the usual fashion we differentiate with respect to λ_0 and λ_1 to inspect if the system of equations are analytically solvable

$$\frac{d}{d\lambda_0}\ln(f(\mathbf{z},\ \mathbf{u}|\lambda_0,\lambda_1)) = \frac{n_1}{\lambda_0} - \sum_{i=1}^n z_i\mathbb{I}_{u_i=1} + \frac{z_i}{e^{\lambda_0 z_i}-1}\mathbb{I}_{u_i=0} = 0 \tag{6}$$

$$\frac{d}{d\lambda_1}\ln(f(\mathbf{z},\ \mathbf{u}|\lambda_0,\lambda_1)) = \frac{n_0}{\lambda_1^{(t)}} - \sum_{i=1}^n z_i\mathbb{I}_{u_i=0} + \frac{z_i}{e^{\lambda_1^{(t)} z_i}-1}\mathbb{I}_{u_i=1} = 0 \tag{7}$$

In the previous part, we chose to do a fix point iteration of these expressions, but in this case we are asked to find it analytically. That is, we must find the the $\lambda_0^{(t)}$ and $\lambda_1^{(t)}$ satisfying the above expressions. We state without proof that this set of equations do not have an analytically answer, so the log-likelihood is optimizied numerically below

```
log_likely <- function(lambda) {
    lambda_0 = lambda[1]
    lambda_1 = lambda[2]
    index_1 = which(u == 1)
    index_0 = which(u == 0)
    n_1 = length(index_1)
    n_0 = n - length(index_1)

    return(n_1 * log(lambda_0) + n_0 * log(lambda_1) - sum(lambda_0 * z[index_1] -
        log(1 - exp(-lambda_1 * z[index_1]))) - sum(lambda_1 * z[index_0] - log(1 -
        exp(-lambda_0 * z[index_0]))))
}


max_lambda = optim(par = c(lambda_0_est, lambda_1_est), fn = log_likely, control = list(fnscale = -1))

lambda_0_max = max_lambda$par[1]
lambda_1_max = max_lambda$par[2]

cat("Numerically optimized value of lambda_0:", lambda_0_max, "\n")
```

```
## Numerically optimized value of lambda_0: 3.463089
```

```r
cat("Numerically optimized value of lambda_1:", lambda_1_max, "\n")
```

```
## Numerically optimized value of lambda_1: 9.334877
```

From the numerical optimization we arrive at the exact same answer, namely $(\lambda_0, \lambda_1) = (3.4631, 9.3349)$. The advantages of the numerical optimization is of course that it requires less implementation and that it comes with more features, such as the Hessian and different maximization methods. The EM-algorithm needs to iterate one parameter at a time and does not in general provide extra features, even though runtime was not at the essence in this case. On the other hand, if the expressions were to be more complex, the simplicity of the EM-algorithm makes it elegant and the numerical optimization might not even be an option if an analytic expression for the maximization function does not exist. Also, implementing an algorithm from scratch might open up for some more flexibility than the numerical optimization which is from the outside more of a blackbox optimization. In our case, neither of the two methods proved especially difficult to implement.