Exercise 3

TMA4300 Computer Intensive Statistical Models

Mads Adrian Simonsen, William Scott Grundeland Olsen

23 april, 2021

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

We consider here the AR(2) model specified by the relation

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

where e_t are i.i.d. random variables with zero mean and constant variance, for $t = 3, \ldots, T$. In our case we analyze the data in `data3A$x`, which is of length $T = 100$, and has the form as in Figure 1.

```
x <- data3A$x
TT <- length(x)
plot(1:TT, x, xlab = "t", ylab = "x", type = "l")
```

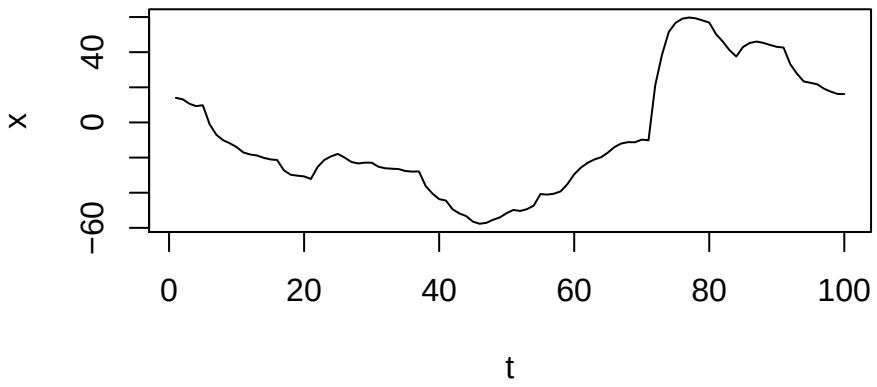

Figure 1: Time seres data.

The minimizers $\hat{\boldsymbol{\beta}}_{\text{LS}}$ and $\hat{\boldsymbol{\beta}}_{\text{LA}}$ are obtained by minimizing the sum of squared residuals (LS) and the sum of absolute residuals (LA), respectively. That is, we minimize

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

with respect to $\boldsymbol{\beta} = \begin{bmatrix} \beta_1 & \beta_2 \end{bmatrix}^\top$, respectively. We can then define the estimated residuals $\hat{e}_t = x_t - \hat{\beta}_1 x_{t-1} + \hat{\beta}_2 x_{t-2}$, for $t = 3, \ldots, T$, and if we let $\bar{e}$ be the mean of these, $\hat{\varepsilon}_t = \hat{e}_t - \bar{e}$ has mean zero.

Subproblem 1.

In the following code block we generate $B = 2000$ bootstrap samples of the residuals, containing T elements randomly sampled from the estimated residuals with replacement. From the sampled residuals the AR(2)

sequence is resampled for each $b = 1, \ldots, B$, where the initial values x_τ and $x_{\tau+1}$ are chosen randomly for each new process.

```
set.seed(269)

# Estimates for the betas:
betas <- ARp.beta.est(x, 2)
# Corresponding residuals:
res_LS <- ARp.resid(x, betas$LS)
res_LA <- ARp.resid(x, betas$LA)

# Bootstrap:
B <- 2000   # Number of bootstrap samples
n <- length(res_LS)   # = length(res_LA) is the size of resampling
# Initialize to store samples:
boot_beta_LS <- matrix(NA, nrow = 2, ncol = B)
boot_beta_LA <- matrix(NA, nrow = 2, ncol = B)

for(b in 1:B) {
  # Sample the residuals:
  sample_LS <- sample(res_LS, n, replace = TRUE)
  sample_LA <- sample(res_LA, n, replace = TRUE)
  # Calculate AR(2) sequence and betas:
  x_LS <- ARp.filter(x[rep(sample(TT-1, 1), 2) + c(0, 1)], betas$LS, sample_LS)
  x_LA <- ARp.filter(x[rep(sample(TT-1, 1), 2) + c(0, 1)], betas$LA, sample_LA)
  beta_boot_LS <- ARp.beta.est(x_LS, 2)$LS
  beta_boot_LA <- ARp.beta.est(x_LA, 2)$LA
  # Append betas to the bootstrap matrices:
  boot_beta_LS[, b] <- beta_boot_LS
  boot_beta_LA[, b] <- beta_boot_LA
}
```

We can then find the estimated variances of $\hat{\boldsymbol{\beta}}_{\text{LS}}$ and $\hat{\boldsymbol{\beta}}_{\text{LA}}$, and also their bias. The results are shown in the following code block.

```
# Estimated variances of beta1 and beta2 for LS and LA:
rowVars(boot_beta_LS)
```

```
## [1] 0.005443422 0.005278484
```

```
rowVars(boot_beta_LA)
```

```
## [1] 0.0003736724 0.0003706076
```

```
# Estimated bias of beta1 and beta2 for LS and LA:
rowMeans(boot_beta_LS) - betas$LS
```

```
## [1] -0.013089553  0.007577976
```

```
rowMeans(boot_beta_LA) - betas$LA
```

```
## [1] -0.001853485  0.001311611
```

From this we see that the estimated variance of $\hat{\boldsymbol{\beta}}_{\text{LA}}$ is smaller than that of $\hat{\boldsymbol{\beta}}_{\text{LS}}$, and this is also true for the estimated bias. This suggests that the LS estimator is not optimal for the AR(2) process.

Subproblem 2.

In this part we want to compute a 95% prediction interval for x_{101} based on the LS and the LA estimators, using the bootstrapped time series and parameter estimates obtained earlier. For this we use `boot_res_LS` and `boot_res_LA`, as we found in the last part. Using this we can estimate

$$x_{101} = \hat{\beta}_1 x_{100} + \hat{\beta}_2 x_{99} + \hat{\varepsilon}_{101},$$

where $\hat{\boldsymbol{\beta}}$ is either $\hat{\boldsymbol{\beta}}_{\text{LS}}$ or $\hat{\boldsymbol{\beta}}_{\text{LA}}$, and $\hat{\varepsilon}_{101}$ is sampled at random from the residual sample. We can use the residuals as samples for $\hat{\varepsilon}_{101}$ because they are assumed to be independent identically distributed. We do this in the following code block for the LS and the LA estimators.

```
set.seed(26)

# Finding x_101 for the two estimators:
x_101_LS <- t(boot_beta_LS[, sample(B, replace = TRUE)]) %*% x[TT - 0:1] +
  sample(res_LS, B, replace = TRUE)
x_101_LA <- t(boot_beta_LA[, sample(B, replace = TRUE)]) %*% x[TT - 0:1] +
  sample(res_LA, B, replace = TRUE)

# The prediction intervals for x_101 for LS and LA:
pred_int <- rbind(
  LS = quantile(x_101_LS, c(0.025, 0.975)),
  LA = quantile(x_101_LA, c(0.025, 0.975))
  )

pred_int
```

```
##        2.5%    97.5%
## LS 7.401362 23.08991
## LA 7.282280 23.28162
```

We see that the 95% prediction interval for x_{101} using the LS estimator is 15.69, while for the LA estimator it is 16. That is, they are almost equally wide. The prediction interval is also shown in Figure 2. We conclude that both models perform equally well on predicting the next data point, despite the fact that $\widehat{\text{Var}}[\hat{\boldsymbol{\beta}}_{\text{LA}}] < \widehat{\text{Var}}[\hat{\boldsymbol{\beta}}_{\text{LS}}]$.

```
plot(1:TT, x, xlab = "t", ylab = "x", type = "l")
lines(c(TT-1, TT+3), rep(pred_int["LS", 1], 2), type = "l", col = 2, lwd = 2)
lines(c(TT-1, TT+3), rep(pred_int["LS", 2], 2), type = "l", col = 2, lwd = 2)
lines(c(TT-1, TT+3), rep(pred_int["LA", 1], 2), type = "l", col = 4, lty = 2, lwd = 1.5)
lines(c(TT-1, TT+3), rep(pred_int["LA", 2], 2), type = "l", col = 4, lty = 2, lwd = 1.5)
legend("topleft", legend = c("LS", "LA"), col = c(2, 4), lty = c(1, 2), lwd = 1.5)
```

Problem B: Permutation test

Bilirubin is a breakdown product of hemoglobin, which is a principal component of red blood cells. If the liver has suffered degeneration, the decomposition of hemoglobin is elevated, or the gall bladder has been destroyed, large amounts of bilirubin can accumulate in the blood, leading to jaundice, which is a yellowish or greenish pigmentation in the skin.

We will look at a data set taken from Jørgensen (1993). It contains measurements of the concentration of bilirubin (mg/dL) in blood samples taken from an three young men, shown in Table 1.

We will use the F-statistic perform a permutation test.

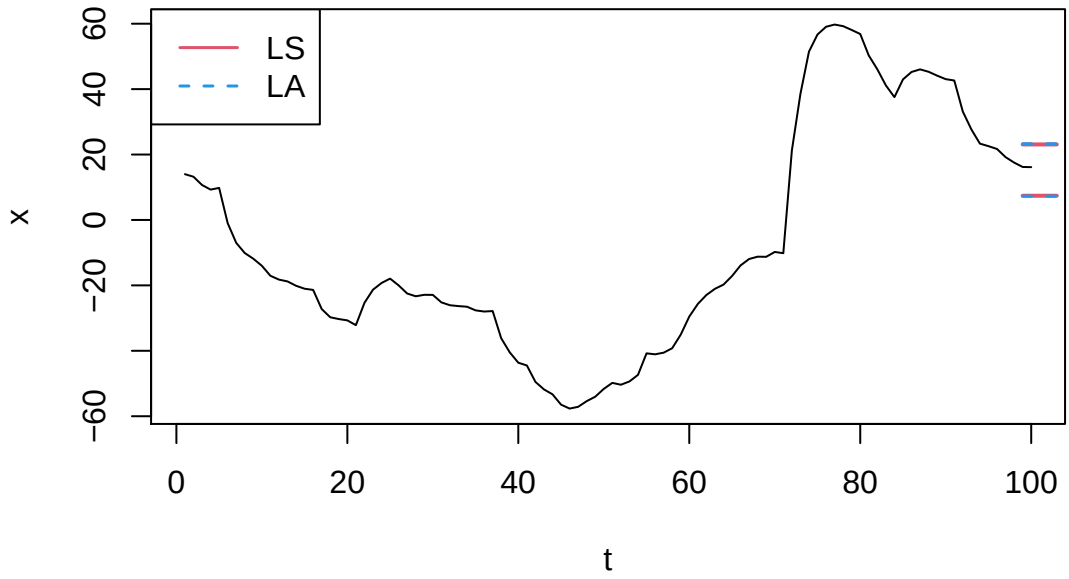

Figure 2: Time seres data with the 95% prediction intervals for x_{101} using the LS estimator in red and using the LA estimator in blue.

Table 1: The measurements of bilirubin in three individuals.

Individual	Concentration (mg/dL)										
1	0.14	0.20	0.23	0.27	0.27	0.34	0.41	0.41	0.55	0.61	0.66
2	0.20	0.27	0.32	0.34	0.34	0.38	0.41	0.41	0.48	0.55	NA
3	0.32	0.41	0.41	0.55	0.55	0.62	0.71	0.91	NA	NA	NA

Subproblem 1.

We set up the following regression model

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

where n_i are the number of measurements from individual i, and $\epsilon_{ij} \sim \text{Normal}(0, \sigma^2)$. We want to test if the concentration of bilirubin of the three young men are significantly different, so we set up the following hypothesis test

$$H_0 : \beta_1 = \beta_2 = \beta_3 \quad \text{against} \quad H_1 : \exists\, i, j \in \{1, 2, 3\} : \beta_i \neq \beta_j.$$

A boxplot of the log measurements for each individual is shown in Figure 3, where we see that individual 1 and individual 2 have similar median measurements, albeit individual 2 having a smaller spread. Individual 3 stands out from the two others.

```
bilirubin %>%
  ggplot(aes(pers, log(meas))) +
  geom_boxplot() +
  theme_minimal()
```

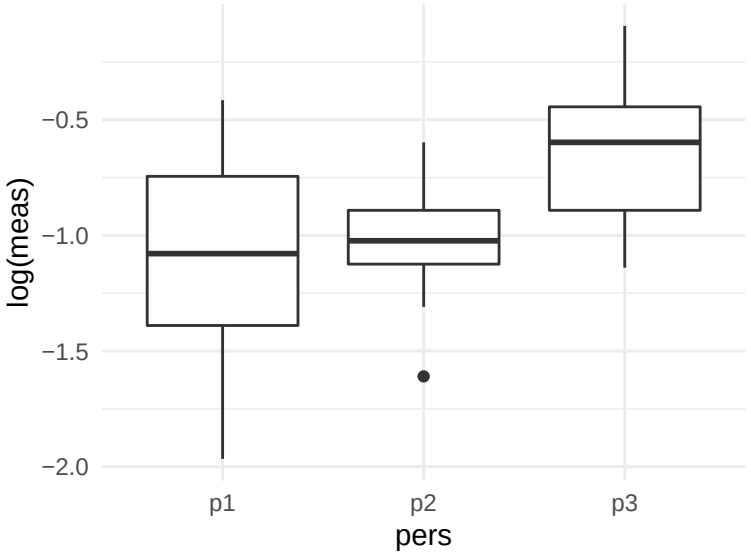

Figure 3: Box plot showing the log concentration of biliburin from the three young men.

```
(lm_fit_summary <- summary(lm(log(meas) ~ pers, bilirubin)))
```

```
##
## 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
```

```
Fval <- lm_fit_summary$fstatistic[1]
```

The summary output shows the F-statistic on 2 and 26 degrees of freedom with the respective p-value of 0.03946. With a significance level of 5%, we reject the null hypothesis and say there is evidence of the individuals having different concentrations of bilirubin.

Subproblem 2.

We create a permutation test function, where we shuffle the individual labels to the data and re-fit a linear model according to (1) and return the F-statistic.

```
# Generates permutation of the data, fitting a linear model and returning the F statistic
# data: dataframe consisting of two variables, response and covariate
permTest <- function(data) {
  # gives formula log(y) ~ x, where y is reshuffled
  perm_formula <- update(formula(data), sample(log(.)) ~ .)
  # returns F statistic
  summary(lm(perm_formula, data))$fstatistic[1]
}
```

Subproblem 3.

We now perform the permutation test with a sample size of 999. A histogram of the F-statistics is shown in Figure 4. We see it fits well under the curve of the theoretical density of the F distribution.

```
set.seed(42)
Fvals <- replicate(n = 999, permTest(bilirubin))

tibble(Fvals = Fvals, Fval = Fval) %>%
  ggplot(aes(Fvals)) +
  geom_histogram(
    mapping = aes(y = after_stat(density)),
    breaks  = seq(0, max(Fvals), by=0.2)
    ) +
  geom_function(
    mapping = aes(color = "Theoretical density"),
    fun     = df,
    n       = 1001,
    args    = list(df1 = 2, df2 = 26)
  ) +
  geom_vline(aes(xintercept = Fval, color = "F-value")) +
  theme_minimal()
```

The calculated p-value from the permutation test is given below.

```
p_val <- mean(Fvals > Fval)
print(sprintf("p-value =  %.5f", p_val))
```

```
## [1] "p-value =  0.04004"
```

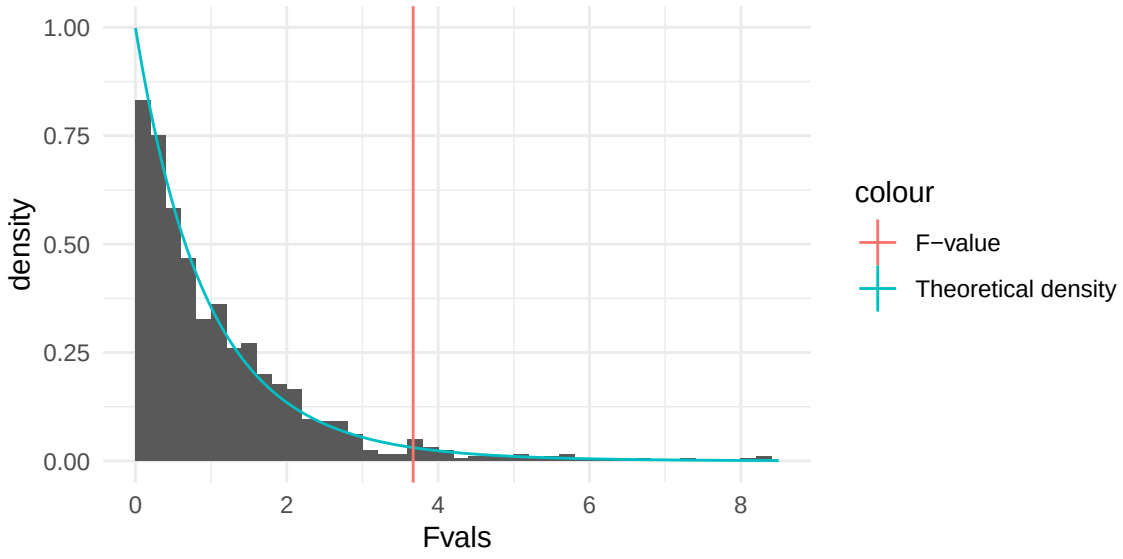

Figure 4: Normalized histogram of F-statistic from 999 permutations of the `bilirubin.txt` data together with the theoretical F distribution on 2 and 26 degrees of freedom. The vertical line is the F-statistic of the original model.

This value corresponds well with the p-value from the summary output of the original model. Again, with a significance level of 5% we reject the null hypothesis and say there is evidence of the three young men having different levels of bilirubin.

Problem C: The EM-algorithm and bootstrapping

Let $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ be independent random variables, with $x_i \sim$ Exponential(λ_0) and $y_i \sim$ Exponential(λ_1), for $i = 1, \ldots n$. Assume that we do not observe $(x_i, y_i), i = 1, \ldots, n$, directly. Instead, we observe

$$\begin{aligned} z_i &= \max(x_i, y_i), \quad \text{for} \quad i = 1, \ldots, n, \\ u_i &= I(x_i \geq y_i), \quad \text{for} \quad i = 1, \ldots, n, \end{aligned} \tag{2}$$

where $I(\cdot) \in \{0, 1\}$ is the indicator function. A histogram of the observed data is shown in Figure 5.

Based on the observed values $(z_i, u_i), i = 1, \ldots, n$, we will use the EM-algorithm to find the MLE for (λ_0, λ_1).

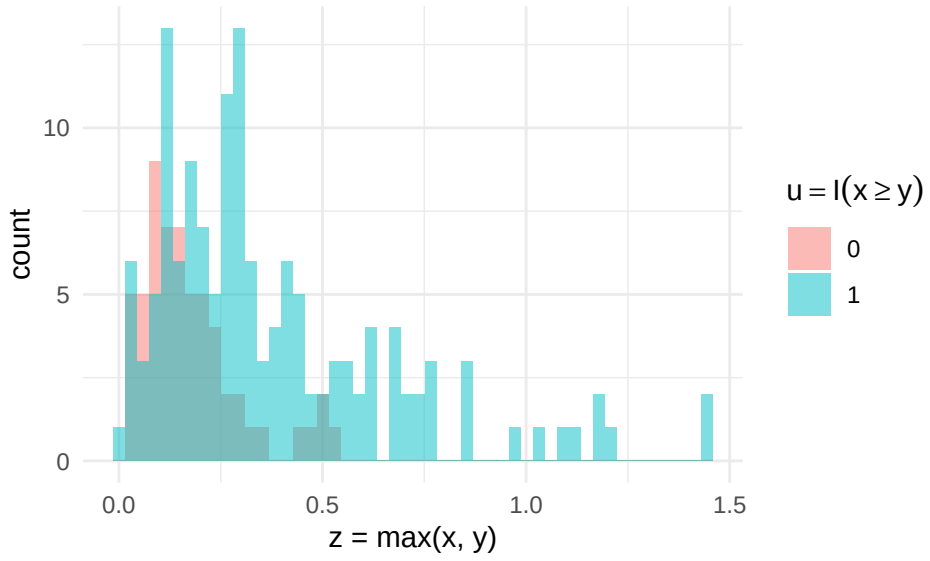

Figure 5: The observed data

Subproblem 1.

Let $\mathbf{x} = (x_1, \ldots, x_n), \mathbf{y} = (y_1, \ldots, y_n), \mathbf{z} = (z_1, \ldots, z_n)$ and $\mathbf{u} = (u_1, \ldots, u_n)$. For the E-step we compute the expectation of the joint log likelihood for the complete data conditioned on the observed data.

$$
\begin{aligned}
Q(\lambda_0, \lambda_1 \mid \lambda_0^{(t)}, \lambda_1^{(t)}) &= \mathrm{E}\left[\log \mathcal{L}(\lambda_0, \lambda_1 \mid \mathbf{X}, \mathbf{Y}) \mid \mathbf{z}, \mathbf{u}, \lambda_0^{(t)}, \lambda_1^{(t)}\right] \\
&= \mathrm{E}\left[\log f(\mathbf{X}, \mathbf{Y} \mid \lambda_0, \lambda_1) \mid \mathbf{z}, \mathbf{u}, \lambda_0^{(t)}, \lambda_1^{(t)}\right] \\
&= \mathrm{E}\left[\log\left(\prod_{i=1}^{n} f_X(X_i \mid \lambda_0) \cdot f_Y(Y_i \mid \lambda_1)\right) \mid \mathbf{z}, \mathbf{u}, \lambda_0^{(t)}, \lambda_1^{(t)}\right] \\
&= \mathrm{E}\left[\sum_{i=1}^{n} \log\left(\lambda_0 \mathrm{e}^{-\lambda_0 X_i} \cdot \lambda_1 \mathrm{e}^{-\lambda_1 Y_i}\right) \mid \mathbf{z}, \mathbf{u}, \lambda_0^{(t)}, \lambda_1^{(t)}\right] \\
&= \mathrm{E}\left[\sum_{i=1}^{n} (\log \lambda_0 + \log \lambda_1 - \lambda_0 X_i - \lambda_1 Y_i) \mid \mathbf{z}, \mathbf{u}, \lambda_0^{(t)}, \lambda_1^{(t)}\right] \\
&= n(\log \lambda_0 + \log \lambda_1) \\
&\quad - \lambda_0 \sum_{i=1}^{n} \mathrm{E}\left[X_i \mid \mathbf{z}, \mathbf{u}, \lambda_0^{(t)}, \lambda_1^{(t)}\right] \\
&\quad - \lambda_1 \sum_{i=1}^{n} \mathrm{E}\left[Y_i \mid \mathbf{z}, \mathbf{u}, \lambda_0^{(t)}, \lambda_1^{(t)}\right].
\end{aligned}
$$

To evaluate the expectations in the last equality, we first do some simplifications. For $X_i, i = 1, \ldots, n$, we have the following:

$$\begin{aligned}
\mathrm{E}\left[X_i \mid \mathbf{z}, \mathbf{u}, \lambda_0^{(t)}, \lambda_1^{(t)}\right] &= \mathrm{E}\left[X_i \mid \max(X_i, Y_i) = z_i, I(X_i \geq Y_i) = u_i, \lambda_0 = \lambda_0^{(t)}, \lambda_1 = \lambda_1^{(t)}\right] \\
&= u_i \,\mathrm{E}\left[X_i \mid \max(X_i, Y_i) = z_i, X_i \geq Y_i, \lambda_0 = \lambda_0^{(t)}, \lambda_1 = \lambda_1^{(t)}\right] \\
&\quad + (1 - u_i)\,\mathrm{E}\left[X_i \mid \max(X_i, Y_i) = z_i, X_i < Y_i, \lambda_0 = \lambda_0^{(t)}, \lambda_1 = \lambda_1^{(t)}\right] \qquad (3) \\
&= u_i \,\mathrm{E}\left[X_i \mid X_i = z_i\right] + (1 - u_i)\,\mathrm{E}\left[X_i \mid X_i < z_i, \lambda_0 = \lambda_0^{(t)}\right] \\
&= u_i z_i + (1 - u_i)\,\mathrm{E}\left[X_i \mid X_i < z_i, \lambda_0 = \lambda_0^{(t)}\right].
\end{aligned}$$

The expectation in the last equality is found by first computing the conditional cdf of $[X \mid X < z, \lambda]$, omitting the subscripts for computational convenience, as we will use the result for $Y_i, i = 1, \ldots, n$ as well.

$$\begin{aligned}
F(x \mid X < z, \lambda) &= \Pr(X < x \mid X < z, \lambda) \\
&= \frac{\Pr(X < x, X < z \mid \lambda)}{\Pr(X < z \mid \lambda)} \\
&= \frac{F_X(\min(x, z) \mid \lambda)}{F_X(z \mid \lambda)},
\end{aligned}$$

where $F_X(\cdot \mid \lambda) \sim \text{Exponential}(\lambda)$. The conditional pdf is then given by

$$f(x \mid X < z, \lambda) = \frac{d}{dx}\frac{F_X(\min(x, z) \mid \lambda)}{F_X(z \mid \lambda)} = \frac{f_X(x \mid \lambda)}{F_X(z \mid \lambda)}, \quad 0 < x < z,$$

giving the conditional expectation

$$\begin{aligned}
\mathrm{E}\left[X \mid X < z, \lambda\right] &= \int_0^z x f(x \mid X < z, \lambda)\,\mathrm{d}x \\
&= \int_0^z x \frac{f_X(x \mid \lambda)}{F_X(z \mid \lambda)}\,\mathrm{d}x \\
&= \int_0^z x \frac{\lambda \mathrm{e}^{-\lambda x}}{1 - \mathrm{e}^{-\lambda z}}\,\mathrm{d}x \\
&= \frac{1}{1 - \mathrm{e}^{-\lambda x}} \int_0^z \lambda x \mathrm{e}^{-\lambda x}\,\mathrm{d}x \\
&= \frac{1}{1 - \mathrm{e}^{-\lambda x}} \left(\frac{1}{\lambda}(1 - \mathrm{e}^{-\lambda x} - \lambda z \mathrm{e}^{-\lambda z})\right) \\
&= \frac{1}{\lambda} - \frac{z \mathrm{e}^{-\lambda z}}{1 - \mathrm{e}^{-\lambda z}} \\
&= \frac{1}{\lambda} - \frac{z}{\exp\{\lambda z\} - 1}.
\end{aligned}$$

Inserting this result into Expression (3) yields

$$\mathrm{E}\left[X_i \mid \mathbf{z}, \mathbf{u}, \lambda_0^{(t)}, \lambda_1^{(t)}\right] = u_i z_i + (1 - u_i)\left(\frac{1}{\lambda_0^{(t)}} - \frac{z_i}{\exp\{\lambda_0^{(t)} z_i\} - 1}\right). \qquad (4)$$

Similarly for $Y_i, i = 1, \ldots, n$, we have that

$$\begin{aligned}
\mathrm{E}\left[Y_i \mid \mathbf{z}, \mathbf{u}, \lambda_0^{(t)}, \lambda_1^{(t)}\right] &= \mathrm{E}\left[Y_i \mid \max(X_i, Y_i) = z_i, I(X_i \geq Y_i) = u_i, \lambda_0 = \lambda_0^{(t)}, \lambda_1 = \lambda_1^{(t)}\right] \\
&= (1 - u_i)\,\mathrm{E}\left[Y_i \mid \max(X_i, Y_i) = z_i, X_i < Y_i, \lambda_0 = \lambda_0^{(t)}, \lambda_1 = \lambda_1^{(t)}\right] \\
&\quad + u_i\,\mathrm{E}\left[X_i \mid \max(X_i, Y_i) = z_i, X_i \geq Y_i, \lambda_0 = \lambda_0^{(t)}, \lambda_1 = \lambda_1^{(t)}\right] \\
&= (1 - u_i)\,\mathrm{E}\left[Y_i \mid Y_i = z_i\right] + u_i\,\mathrm{E}\left[Y_i \mid Y_i \leq z_i, \lambda_1 = \lambda_1^{(t)}\right] \\
&= (1 - u_i)z_i + u_i\left(\frac{1}{\lambda_1^{(t)}} - \frac{z_i}{\exp\{\lambda_1^{(t)} z_i\} - 1}\right).
\end{aligned} \tag{5}$$

Thus, by substituting Expression (4) and (5) into the log-likelihood for the complete data $\mathbf{x}, \mathbf{y}$, conditional on the observed data $\mathbf{z}, \mathbf{u}$, we get

$$\begin{aligned}
Q(\lambda_0, \lambda_1 \mid \lambda_0^{(t)}, \lambda_1^{(t)}) &= n(\log\lambda_0 + \log\lambda_1) \\
&\quad - \lambda_0 \sum_{i=1}^{n}\left[u_i z_i + (1 - u_i)\left(\frac{1}{\lambda_0^{(t)}} - \frac{z_i}{\exp\{\lambda_0^{(t)} z_i\} - 1}\right)\right] \\
&\quad - \lambda_1 \sum_{i=1}^{n}\left[(1 - u_i)z_i + u_i\left(\frac{1}{\lambda_1^{(t)}} - \frac{z_i}{\exp\{\lambda_1^{(t)} z_i\} - 1}\right)\right].
\end{aligned} \tag{6}$$

Subproblem 2.

The M-step of the EM-algorithm is to maximize $Q(\lambda_0, \lambda_1 \mid \lambda_0^{(t)}, \lambda_1^{(t)})$ with respect to (λ_0, λ_1), and set the $(\lambda_0^{(t+1)}, \lambda_1^{(t+1)})$ equal to the maximizer of Q. We find the maximizer by taking the partial derivatives of Expression (6), setting them equal to zero. We have

$$\begin{aligned}
\frac{\partial}{\partial\lambda_0} Q(\lambda_0, \lambda_1 \mid \lambda_0^{(t)}, \lambda_1^{(t)}) &= \frac{n}{\lambda_0} - \sum_{i=1}^{n}\left[u_i z_i + (1 - u_i)\left(\frac{1}{\lambda_0^{(t)}} - \frac{z_i}{\exp\{\lambda_0^{(t)} z_i\} - 1}\right)\right] = 0, \\
\frac{\partial}{\partial\lambda_1} Q(\lambda_0, \lambda_1 \mid \lambda_0^{(t)}, \lambda_1^{(t)}) &= \frac{n}{\lambda_1} - \sum_{i=1}^{n}\left[(1 - u_i)z_i + u_i\left(\frac{1}{\lambda_1^{(t)}} - \frac{z_i}{\exp\{\lambda_1^{(t)} z_i\} - 1}\right)\right] = 0,
\end{aligned}$$

which gives the maximizer

$$\begin{pmatrix} \lambda_0^{(t+1)} \\ \lambda_1^{(t+1)} \end{pmatrix} = \begin{pmatrix} n \Big/ \sum_{i=1}^{n}\left[u_i z_i + (1 - u_i)\left(\frac{1}{\lambda_0^{(t)}} - \frac{z_i}{\exp\{\lambda_0^{(t)} z_i\} - 1}\right)\right] \\ n \Big/ \sum_{i=1}^{n}\left[(1 - u_i)z_i + u_i\left(\frac{1}{\lambda_1^{(t)}} - \frac{z_i}{\exp\{\lambda_1^{(t)} z_i\} - 1}\right)\right] \end{pmatrix}. \tag{7}$$

We then implement the EM-algorithm with the convergence criterion that

$$\left\|\boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^{(t)}\right\|_2 < \epsilon = 10^{-10}, \quad \boldsymbol{\lambda}^{(t)} = (\lambda_0^{(t)}, \lambda_1^{(t)}).$$

```
# Computes the expectation of the log-likelihood of the complete data given the
# observational data
# lambda:      (lambda_0^{(t)}, lambda_1^{(t)})
# lambda_next: (lambda_0^{(t + 1)}, lambda_1^{(t + 1)})
# z:           observed data, max(x, y)
# u:           observed data, I(x >= y)
```

```
Q_func <- function(lambda, lambda_next, z, u) {
  length(z)*(log(lambda_next[1]) + log(lambda_next[2])) -
    lambda_next[1]*sum(u*z + (1 - u)*(1/lambda[1] - z/(exp(lambda[1]*z) - 1))) -
    lambda_next[2]*sum((1 - u)*z + u*(1/lambda[2] - z/(exp(lambda[2]*z) - 1)))
}

# Computes the MLE of lambda_0 and lambda_1 given the observed data using the EM-algorithm
# Additionally, it returns the list of recorded log-likelihood values from the iterations
# z:       observed data, max(x, y)
# u:       observed data, I(x >= y)
# lambda:  starting values (lambda_0^{(0)}, lambda_1^{(0)})
# itermax: maximum number of iterations
# tol:     convergence tolerance value
EM <- function(z, u, lambda, itermax = 300, tol = 1e-10) {
  lambda_next <- numeric(2)
  log_liks <- numeric(0)
  for (i in 1:itermax) {
    # update (lambda_0^{(t + 1)}, lambda_1^{(t + 1)})
    lambda_next[1] <- 1 / mean(u*z + (1 - u)*(1/lambda[1] - z/(exp(lambda[1]*z) - 1)))
    lambda_next[2] <- 1 / mean((1 - u)*z + u*(1/lambda[2] - z/(exp(lambda[2]*z) - 1)))
    log_liks <- c(log_liks, Q_func(lambda, lambda_next, z, u))  # add Q-value
    # check convergence
    if(norm(lambda_next - lambda, type = "2") < tol) {
      break
    }
    lambda <- lambda_next  # update (lambda_0^{(t)}, lambda_1^{(t)})
  }
  list(lambda = lambda_next, log_liks = log_liks)
}
```

Since we only see the maximum of x and y, we expect that $\mathrm{E}[X] < \mathrm{E}[Z \mid U = 1]$ and $\mathrm{E}[Y] < \mathrm{E}[Z \mid U = 0]$.

```
cat(sprintf("mean of z given u = 1: %.1f\nmean of z given u = 0: %.1f",
            mean(z[u == 1]), mean(z[u == 0])))
```

```
## mean of z given u = 1: 0.4
## mean of z given u = 0: 0.2
```

We then use the starting values $(\lambda_0^{(0)}, \lambda_1^{(0)}) = (1/0.4, 1/0.2) = (2.5, 5)$, expecting the final estimates $(\hat{\lambda}_0, \hat{\lambda}_1)$ to be larger.

```
EM_result <- EM(z, u, c(2.5, 5), tol = 1e-10)
cat(sprintf("Estimate of lambda_%d: %.5f\n", 0:1, EM_result$lambda))
```

```
## Estimate of lambda_0: 3.46573
##  Estimate of lambda_1: 9.35321
```

These estimates $(\hat{\lambda}_0, \hat{\lambda}_1)$ are slightly higher than the initial values $(\lambda_0^{(0)}, \lambda_1^{(0)})$ as expected.

A convergence plot of the $Q(\cdot)$-function is seen in Figure 6. We see that it converges very fast. After 6 iterations, it stabilizes, while making small adjustments as we had set a quite strict convergence criterion.

```
tibble(index = 1:length(EM_result$log_liks), log_liks = EM_result$log_liks) %>%
  ggplot(aes(index, log_liks)) +
  geom_point() +
  labs(x = "iteration", y = expression(Q(lambda^{t+1}*"|"*lambda^t))) +
  theme_minimal()
```

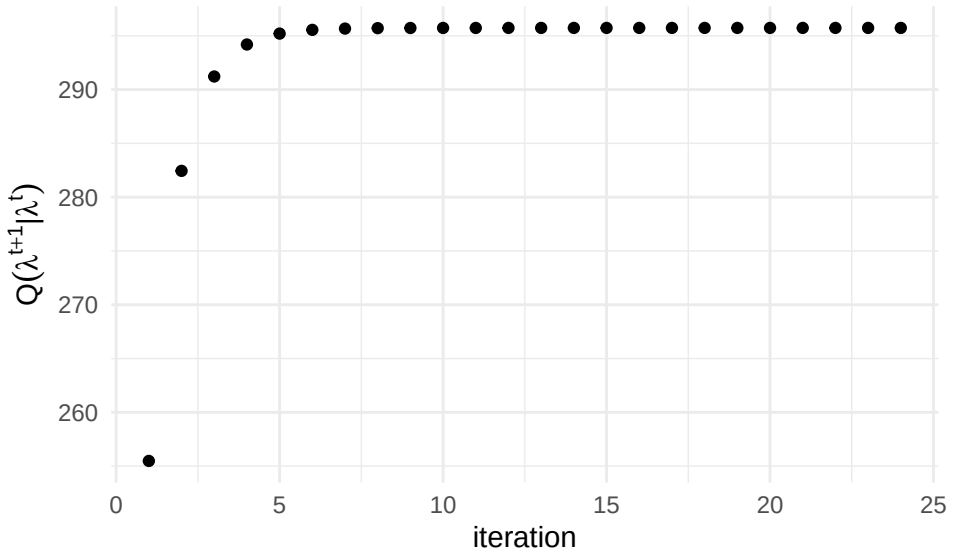

Figure 6: Convergence plot of estimated expectation of the joint log likelihood for the complete data conditional on the observed data.

Subproblem 3.

To get an idea of the biasedness and confidence in the estimated values, we perform a parametric bootstrap. The procedure is shown in Algorithm 1.

Algorithm 1: Parametric bootstrapping for inference of $(\hat{\lambda}_0, \hat{\lambda}_1)$

Result: Standard deviations, biases and correlation of $\hat{\lambda}_0, \hat{\lambda}_1$

Set $(\lambda_0^{(0)}, \lambda_1^{(0)}) \leftarrow (\hat{\lambda}_0, \hat{\lambda}_1)$;
Set $n \leftarrow 200$;
Initialize $\boldsymbol{\lambda}$-List of length B;
for $i \leftarrow 1$ **to** B **do**
    Draw $x_1^*, \ldots, x_n^* \sim \text{Exponential}(\hat{\lambda}_0)$;
    Draw $y_1^*, \ldots, y_n^* \sim \text{Exponential}(\hat{\lambda}_1)$;
    Initialize n-vectors $\mathbf{z}^*$ and $\mathbf{u}^*$;
    for $j \leftarrow 1$ **to** n **do**
        $z_j^* \leftarrow \max(x_j^*, y_j^*)$;
        $u_j^* \leftarrow I(x_j^* \geq y_j^*)$;
    end
    $(\hat{\lambda}_0^*, \hat{\lambda}_1^*) \leftarrow \text{EM-algorithm}\Big(\mathbf{z}^*, \mathbf{u}^*, (\lambda_0^{(0)}, \lambda_1^{(0)})\Big)$;
    $\boldsymbol{\lambda}$-list$[i] \leftarrow (\hat{\lambda}_0^*, \hat{\lambda}_1^*)$;
end
Compute sample standard deviation $(\hat{\sigma}_{\lambda_0}, \hat{\sigma}_{\lambda_1})$ from $\boldsymbol{\lambda}$-list;
Compute sample mean $(\hat{\mu}_{\lambda_0}, \hat{\mu}_{\lambda_1})$ from $\boldsymbol{\lambda}$-list;
$\text{bias}_{(\hat{\lambda}_0, \hat{\lambda}_1)} \leftarrow (\hat{\mu}_{\lambda_0}, \hat{\mu}_{\lambda_1}) - (\hat{\lambda}_0, \hat{\lambda}_1)$;
Compute sample correlation $\hat{\rho}$ from $\boldsymbol{\lambda}$-list;

We implement this algorithm below.

```
# performs a parametric bootstrap of the estimated parameters, and returns an estimate of
# the standard deviation, bias, and correlation of (hat_lambda_0, hat_lambda_1)
# lamba_hat: estimated parameters from the EM-algorithm
```

```r
# B:         number of bootstrap samples
# itermax:   maximum iterations in the EM-algorithm
# tol:       convergence tolerance in the EM-algorithm
param_boot <- function(lambda_hat, B = 10000, itermax = 300, tol = 1e-5) {
  n <- 200  # number of observations
  lambda_list <- matrix(numeric(2*B), nrow = B)
  for (i in 1:B) {
    x_star <- rexp(n, lambda_hat[1])  # draw x*
    y_star <- rexp(n, lambda_hat[2])  # draw y*
    z_star <- pmax(x_star, y_star)    # compute z = max(x, y)
    u_star <- 1*(x_star >= y_star)    # compute u = I(x >= y)
    lambda_star <- EM(z_star, u_star, lambda_hat, itermax, tol)$lambda  # get estimate
    lambda_list[i,] <- lambda_star                                      # add estimate
  }
  sigma_hat <- apply(lambda_list, 2, sd)              # compute sample standard deviation
  mu_hat <- apply(lambda_list, 2, mean)               # compute sample mean
  bias <- mu_hat - lambda_hat                         # compute the bias
  rho_hat <- cor(lambda_list[, 1], lambda_list[, 2])  # compute the sample correlation
  list(sample_sd = sigma_hat, bias = bias, sample_cor = rho_hat)
}
```

We get the following estimates of $\mathrm{SD}[\hat{\lambda}_0], \mathrm{SD}[\hat{\lambda}_1], \mathrm{Bias}[\hat{\lambda}_0], \mathrm{Bias}[\hat{\lambda}_1]$ and $\mathrm{Corr}[\hat{\lambda}_0, \hat{\lambda}_1]$.

```r
set.seed(93)
result <- param_boot(EM_result$lambda)
result
```

```
## $sample_sd
## [1] 0.2503350 0.8459601
##
## $bias
## [1] 0.02245412 0.10254685
##
## $sample_cor
## [1] -0.01906166
```

The standard deviations are of order 10^{-1}, giving us a fair confidence in the estimated parameters. The estimated correlation is very weak, practically speaking it is zero, which is expected as there is no reason for the parameters to be related in any way, as all the computations of the two values are done separately. However, we notice that there is a small positive bias, which means that the ML estimators are larger than their expected values. As we are interested in the true values of the parameters λ_0 and λ_1, we would prefer to go for a bias-corrected estimate.

Subproblem 4.

We wish to find an analytic formula for $f_{Z_i,U_i}(z_i, u_i \mid \lambda_0, \lambda_1)$. Omitting the indices for Z_i, U_i, X_i and Y_i for computational convenience, we get

$$\begin{aligned}
f_{Z,U}(z, u \mid \lambda_0, \lambda_1) &= \Pr(\max(X, Y) = z, I(X \geq Y) = u \mid \lambda_0, \lambda_1) \\
&= u \Pr(\max(X, Y) = z, X \geq Y \mid \lambda_0, \lambda_1) + (1 - u) \Pr(\max(X, Y) = z, Y > X \mid \lambda_0, \lambda_1) \\
&= u \Pr(X = z, X \geq Y \mid \lambda_0, \lambda_1) + (1 - u) \Pr(Y = z, Y > X \mid \lambda_0, \lambda_1) \\
&= u f_X(z \mid \lambda_0, \lambda_1) \Pr(X \geq Y \mid X = z, \lambda_0, \lambda_1) \\
&\quad + (1 - u) f_Y(z \mid \lambda_0, \lambda_1) \Pr(Y > X \mid Y = z, \lambda_0, \lambda_1) \\
&= u f_X(z \mid \lambda_0, \lambda_1) F_Y(z \mid \lambda_0, \lambda_1) + (1 - u) f_Y(z \mid \lambda_0, \lambda_1) F_X(x \mid \lambda_0, \lambda_1) \\
&= u \lambda_0 \mathrm{e}^{-\lambda_0 z}(1 - \mathrm{e}^{-\lambda_1 z}) + (1 - u)\lambda_1 \mathrm{e}^{-\lambda_1 z}(1 - \mathrm{e}^{-\lambda_0 z}).
\end{aligned}$$

We also want to find the maximum likelihood estimators $\hat{\lambda}_0$ and $\hat{\lambda}_1$. The likelihood function is given by

$$L(\lambda_0, \lambda_1 \mid \mathbf{z}, \mathbf{u}) = \prod_{i=1}^{n} f_{Z_i, U_i}(z_i, u_i \mid \lambda_0, \lambda_1) = \prod_{i:\, u_i=0} \lambda_1 \mathrm{e}^{-\lambda_1 z_i}(1 - \mathrm{e}^{-\lambda_0 z_i}) \prod_{i:\, u_i=1} \lambda_0 \mathrm{e}^{-\lambda_0 z_i}(1 - \mathrm{e}^{-\lambda_1 z_i}).$$

The log-likelihood function is then

$$\ell(\lambda_0, \lambda_1 \mid \mathbf{z}, \mathbf{u}) = \sum_{i:\, u_i=0} \ln[\lambda_1 \mathrm{e}^{-\lambda_1 z_i}(1 - \mathrm{e}^{-\lambda_0 z_i})] + \sum_{i:\, u_i=1} \ln[\lambda_0 \mathrm{e}^{-\lambda_0 z_i}(1 - \mathrm{e}^{-\lambda_1 z_i})].$$

We optimize this in the following code block using the `optim()` function in `R`.

```
# Defining the log-likelihood function. Note that it has to be defined
# as the negative of what was discussed earlier to make optim() find
# the maximum value, and not the minimum:
log_likelihood <- function(lambdas, z, u) {
  u0s <- which(u == 0)  # Indices where u = 0
  u1s <- which(u == 1)  # Indices where u = 1
  ll <- sum(log(lambdas[2] * exp(-lambdas[2]*z[u0s]) * (1 - exp(-lambdas[1]*z[u0s])))) +
        sum(log(lambdas[1] * exp(-lambdas[1]*z[u1s]) * (1 - exp(-lambdas[2]*z[u1s]))))
  return(-ll)
}

# Doing the optimization:
MLE <- optim(par = c(1, 1), fn = log_likelihood, z = z, u = u)
MLE$par
```

```
## [1] 3.465890 9.351103
```

We see then that the maximum likelihood estimators are $\hat{\boldsymbol{\lambda}}_{\text{MLE}} = \begin{bmatrix} 3.465890 & 9.351103 \end{bmatrix}^\top$. The absolute difference between the MLE and the EM estimators are shown below.

```
abs(MLE$par - EM_result$lambda)
```

```
## [1] 0.0001550136 0.0021115130
```

Se we see that the difference between the two estimators are small.

There are some advantages in optimizing the likelihood directly compared to using the EM algorithm. One of the advantages is that it is generally more efficient, as the EM algorithm can be very slow, especially depending on the "amount" of missing data. Also for MLE, only one optimization is necessary, while for the EM algorithm, the parameterized function may need to be optimized iteratively. In addition to this, when optimizing the likelihood one can derive the Hessian at the curvature, which then can give the standard errors without having to use a bootstrapping.

References

Jørgensen, Bent. 1993. *The Theory of Linear Models*. New York: Chapman & Hall.