TMA4300 Computer Intensive Statistical Methods

Exercise 3, Spring 2022

Martin Tufte, Jim Totland

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

1. In this exercise, we analyse a non-Gaussian time series of length $T = 100$ given in the data `data3A$x` and compare two different parameter estimators using bootstrapping. Let this time series be denoted as $\boldsymbol{x} = (x_1, x_2, \ldots, x_T)$. We consider an auto-regressive model of order two, AR(2), which can be expressed as

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

where $e_t \sim \text{WN}(0, \sigma_e^2)$, i.e. iid random variables with mean zero and constant variance. A plot of the time series is displayed in Figure 1.

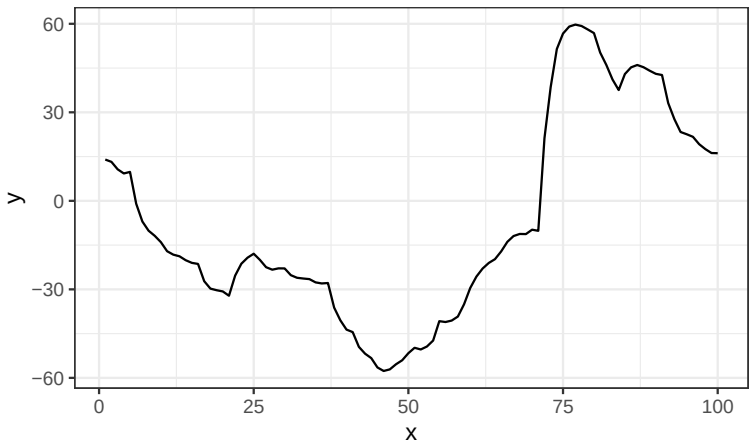


Figure 1: Plot of the time series $\boldsymbol{x}$.

The two parameter estimates for $\boldsymbol{\beta} = \begin{bmatrix}\beta_1 & \beta_2\end{bmatrix}^\top$ are based on minimizing the sum of squared residuals (LS) and the sum of absolute residuals (LA). They can be found from minimizing the following loss functions with respect to $\boldsymbol{\beta}$:

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

The two minimisers are denoted as $\widehat{\boldsymbol{\beta}}_{\text{LS}}$ and $\widehat{\boldsymbol{\beta}}_{\text{LA}}$ respectively. In addition, denote by $\hat{e}_t = x_t - \hat{\beta}_1 x_{t-1} - \hat{\beta}_2 x_{t-2}$ the estimated residuals for $t = 3, \ldots, T$ and let $\bar{e}$ be the mean of these. The estimated residuals can therefore be re-centered to have mean zero by defining $\hat{\epsilon}_t = \hat{e}_t - \bar{e}$. It is important to use the re-centered estimated residuals since we assume an AR process, which is a stationary time series with constant mean. Using the function `ARp.beta.est` from the file `probAhelp.R`, the parameter estimates are calculated as $\widehat{\boldsymbol{\beta}}_{\text{LS}} = \begin{bmatrix}1.5528 & -0.5680\end{bmatrix}^\top$ for the sum of squared residuals and $\widehat{\boldsymbol{\beta}}_{\text{LA}} = \begin{bmatrix}1.5466 & -0.5575\end{bmatrix}^\top$ for the sum of absolute residuals.

To evaluate the relative performance of each estimator, we will use residual resampling bootstrap. Denote a bootstrap sample as $\boldsymbol{x}^* = (x_1^*, x_2^*, \ldots, x_T^*)$. It is a new time series of length T generated in three steps:

1. Fit the parameter estimates to the original time series $\boldsymbol{x}$ and obtain the fitted parameters $\widehat{\boldsymbol{\beta}}$ and residuals $\widehat{\boldsymbol{\epsilon}}$.
2. Sample a bootstrap set of residuals $\epsilon_3^*, \epsilon_4^*, \cdots, \epsilon_T^*$ completely at random from $\widehat{\boldsymbol{\epsilon}}$ with replacement.
3. Pick two consecutive values x_1^* and x_2^* from $\boldsymbol{x}$ at random for initial values for the bootstrap sample $\boldsymbol{x}^*$. Use the sampled residuals to calculate $x_t^* = \beta_1 x_{t-1}^* + \beta_2 x_{t-2}^* + \epsilon_t^*$ for $t = 3, \ldots, 100$.
4. Fit the parameter estimates to $\boldsymbol{x}^*$ to obtain a bootstrap estimate $\widehat{\boldsymbol{\beta}}^*$.

Using $B = 10000$ bootstrap samples, the estimated variance of $\widehat{\boldsymbol{\beta}}_{\text{LS}}$ and $\widehat{\boldsymbol{\beta}}_{\text{LA}}$ is reported in the Table 1. The estimated variance is lower for the LA estimator, roughly one order of magnitude, than that of the LS estimator.

```
ARp.beta.bootstrap <- function(x, p, B){
  # Fit the regression model
  beta <- ARp.beta.est(x, p)
  T <- length(x)
  # Find the residuals
  resid.LS <- ARp.resid(x, beta$LS)
  resid.LA <- ARp.resid(x, beta$LA)

  # Initialize matrices to store the bootstrap estimates of beta
  betas.LS <- matrix(NA, nrow=B, ncol=p)
  betas.LA <- matrix(NA, nrow=B, ncol=p)

  for(b in 1:B){
    # Find consequetive starting values for the resampling
    idx <- sample(1:(T-p+1), 1)
    x0 <- x[idx:(idx+p-1)]

    # Resample residuals and x
    resample.resid.LS <- sample(resid.LS, replace = TRUE)
    resample.resid.LA <- sample(resid.LA, replace = TRUE)
    bootstrap.x.LS <- ARp.filter(x0, beta$LS, resample.resid.LS)
    bootstrap.x.LA <- ARp.filter(x0, beta$LA, resample.resid.LA)

    # Store new estimates for beta
    betas.LS[b,] <- ARp.beta.est(bootstrap.x.LS, p)$LS
    betas.LA[b,] <- ARp.beta.est(bootstrap.x.LA, p)$LA
  }
  return(list(LS=betas.LS, LA=betas.LA))
}

set.seed(4200)
betas <- ARp.beta.bootstrap(x=data3A$x, p=2, B=10000)

# Estimate variance of the two estimators
var.LS <- apply(betas$LS, 2, var)
var.LA <- apply(betas$LA, 2, var)
```

Next, we calculate the estimated bias of the two estimators. The ideal bootstrap estimate for the bias is estimated using the bootstrap expectation of the mean of each estimator. Then the bias equals the difference in the bootstrap mean and the parameter estimates. The bias is reported in Table 2. Also here we see that the LA estimator performs better, as it has a lower magnitude of the bias than the LS estimator. The LS estimator is known to be optimal for Gaussian AR(p) processes. However, here we see that the LA estimator performs better. This suggests that the time series $\boldsymbol{x}$ is either not an $AR(2)$-process, so the model assumptions is not reasonable or

Table 1: Estimated variance of the two estimators.

$\boldsymbol{\beta}$	Estimated variance of $\widehat{\boldsymbol{\beta}}_{\text{LS}}$	Estimated variance of $\widehat{\boldsymbol{\beta}}_{\text{LA}}$
β_1	$5.673 \cdot 10^{-3}$	$4.283 \cdot 10^{-4}$
β_2	$5.546 \cdot 10^{-3}$	$4.204 \cdot 10^{-4}$

that $\boldsymbol{x}$ is an AR(2) process, but the error terms are not Gaussian. The fact that $\boldsymbol{x}$ is non-Gaussian substantiates the latter.

```
# Estimated bias of the two estimators using the bootstrap expectation
bias.LS <- apply(betas$LS, 2, mean) - beta$LS
bias.LA <- apply(betas$LA, 2, mean) - beta$LA
```

Table 2: Estimated bias of the two estimators.

$\boldsymbol{\beta}$	Estimated bias of $\widehat{\boldsymbol{\beta}}_{\text{LS}}$	Estimated bias of $\widehat{\boldsymbol{\beta}}_{\text{LA}}$
β_1	$-1.281 \cdot 10^{-2}$	$-2.872 \cdot 10^{-3}$
β_2	$7.131 \cdot 10^{-3}$	$2.328 \cdot 10^{-3}$

2. The estimators can be used to make prediction for the next value of the time series. A 95% prediction interval can be made for x_{101} using both estimators. The variability of x_{101} must reflect both the variability of the simulated x_{101} values based on our lack of knowledge of the parameter values and about the residual distribution. A bootstrap estimate of x^*_{101} which reflects both these variabilities can be generated in three steps:

1. Sample a bootstrap time series $\boldsymbol{x}^*$ and obtain the bootstrap estimate $\widehat{\boldsymbol{\beta}}^*$.
2. Pick a random residual ϵ_{101} from $\widehat{\boldsymbol{\epsilon}}$.
3. Sample $x^*_{101} = x_{100}\widehat{\boldsymbol{\beta}}^*_1 + x_{99}\widehat{\boldsymbol{\beta}}^*_2 + \epsilon_{101}$.

Using $B = 10000$ bootstrap estimates for both parameter estimates, a 95% prediction interval is found from the lower and upper 0.025-quantiles of the bootstrap samples. The 95% prediction intervals for x_{101} using LS is $[7.344, 23.206]$ and for LA is $[7.288, 23.292]$.

```
ARp.pred.bootstrap <- function(x, p, B){
  # Fit the regression model
  beta <- ARp.beta.est(x, p)
  T <- length(x)

  # Find the residuals
  resid.LS <- ARp.resid(x, beta$LS)
  resid.LA <- ARp.resid(x, beta$LA)

  # Initialize matrices to store the bootstrap estimates of x_{T+1}
  pred.LS <- matrix(NA, nrow=B, ncol=1)
  pred.LA <- matrix(NA, nrow=B, ncol=1)

  for(b in 1:B){
    # Find consequetive starting points for the resampling
    idx <- sample(1:(T-p+1), 1, replace = TRUE)
    x0 <- x[idx:(idx+p-1)]

    # Resample residuals and x
    resample.resid.LS <- sample(resid.LS, replace = TRUE)
    resample.resid.LA <- sample(resid.LA, replace = TRUE)
```

```
    bootstrap.x.LS <- ARp.filter(x0, beta$LS, resample.resid.LS)
    bootstrap.x.LA <- ARp.filter(x0, beta$LA, resample.resid.LA)

    # Estimate bootstrap betas
    bootstrap.beta.LS <- ARp.beta.est(bootstrap.x.LS, p)$LS
    bootstrap.beta.LA <- ARp.beta.est(bootstrap.x.LA, p)$LA

    # Store new estimates for x_{T+1}
    pred.LS[b,] <- bootstrap.beta.LS %*% rev(x[(T-p+1):T]) + sample(resid.LS, 1)
    pred.LA[b,] <- bootstrap.beta.LA %*% rev(x[(T-p+1):T]) + sample(resid.LA, 1)
  }
  return(list(LS=pred.LS, LA=pred.LA))
}

set.seed(4200)
preds <- ARp.pred.bootstrap(x=data3A$x, p=2, B=10000)

quantile(preds$LS, probs = c(0.025, 0.975))
quantile(preds$LA, probs = c(0.025, 0.975))
```

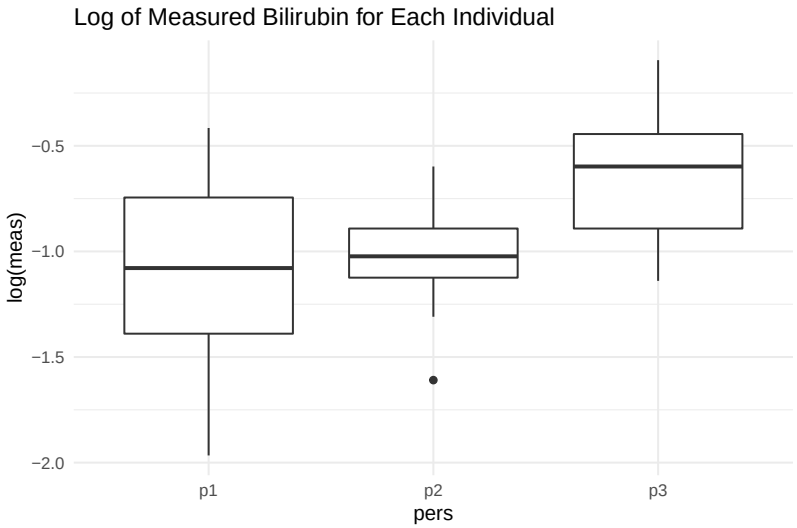


Figure 2: A box-plot for the log of the measured concentrations of bilirubin, for each of the 3 individuals. The data set is described in the introduction of **Problem B**.

Problem B: Permutation test

In this section, we will consider a dataset concerning the level of bilirubin (mg/dL) in blood samples taken from three young men. The data is taken from [1] and contains two columns: `meas`, which is the measured concentrations of bilirubin, and `pers`, which is an indicator for which individual the measurement belongs to.

1. We start by creating a box-plot for the log of `meas` for each individual. This is displayed in Figure 2. We observe that the concentrations belonging to individual 3 are quite a bit higher compared to the other two individuals. The required code is given below.

```
# Load dataset
bilirubin <- read.table("files/bilirubin.txt",header=T)

# Create box plot of logarithm of measurement
ggplot(bilirubin, aes(x = pers, y = log(meas))) +
  geom_boxplot() + xlab("pers") + ylab("log(meas)") +
  ggtitle("Log of Measured Bilirubin for Each Individual") + theme_minimal()
ggsave("figures/boxplot.pdf", height = 5, width = 5)
```

Next, we use the function `lm` in `R` to fit the model

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

where $n_1 = 11$, $n_2 = 10$ and $n_3 = 8$ and $\varepsilon_{i,j} \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$. We conduct an F-test to test the hypothesis that $\beta_1 = \beta_2 = \beta_3$ and save the resulting value as `Fval`. This is done in the code below.

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

The F-statistc evalueates to $F \approx 3.67$, and the p-value is $p \approx 0.03946$, which means that the hypothesis is rejected at a significance level of 5%.

2. We create the function `permTest()`, which generates a permutation of the data between the three individuals, then fits the model given in (1) and finally returns the F-statistic corresponding to the hypothesis test $\beta_1 = \beta_2 = \beta_3$. The implementation is given below.

```r
permTest <- function(){
  df.perm <- data.frame(bilirubin)
  df.perm$pers <- sample(bilirubin$pers, size = nrow(bilirubin),
                         replace = FALSE)
  mod <- lm(log(meas) ~ pers, data = df.perm)
  s <- summary(mod)
  return(s$fstatistic[1])
}
```

3. We perform a permutation test by generating 999 samples of the F-statistic using the `permTest()` function and plot the distribution of the samples:

```r
n <- 999 # Number of samples
F.samples <- rep(NA, n)
set.seed(4300)
for(i in 1:n){
  F.samples[i] <- permTest()
}

sum(F.samples >= Fval)/n # p-value

# Create histogram
F.right <- F.samples[which(F.samples >= Fval)]
F.left <- setdiff(F.samples, F.right)
F.df <- data.frame(F.vals = c(F.left, F.right),
                   ind = as.factor(c(rep( "< Fval", length(F.left)),
                   rep( "> Fval", length(F.right)))))
ggplot(F.df) +
  geom_histogram(aes(x = F.vals, colour = ind), binwidth = 0.1, fill = "white") +
  xlab("F-values") + ylab("Count") +
  theme_minimal() +  theme(legend.title=element_blank())
ggsave("figures/F-vals.pdf", width = 6, height = 4)
```

Figure 3 illustrates the distribution of the sampled F-values. The resulting p-value is $p \approx 0.038$, which is similar to the value found in paragraph **1.** Thus, the hypothesis that all β_i, $i = 1, 2, 3$ are equal, is rejected at a significance level of 5 %. The fact that the p-value is so similar to that of the F-test in `summary.lm` could be interpreted as a sign that the assumptions of the linear model are correct, since the p-value of the permutation test does not depend on these assumptions.

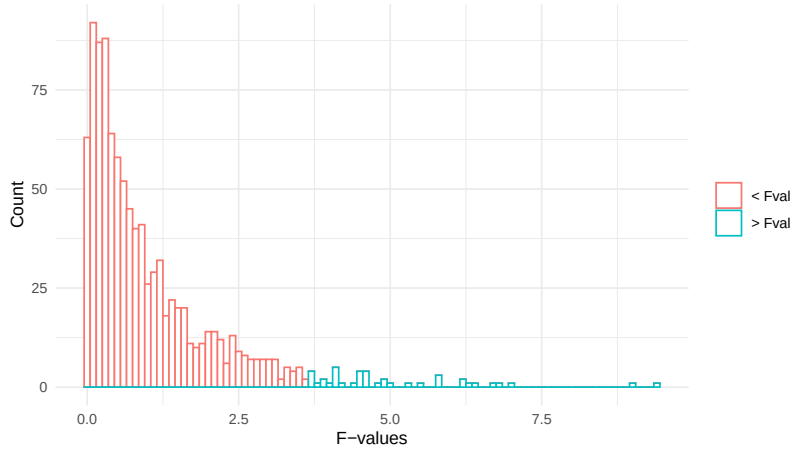


Figure 3: Histogram of the F-values generated by the permutation test.

Problem C: The EM-algorithm and bootstrapping

Let $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ be independent random variables, where $X_i \sim \text{Exp}(\lambda_0)$ and $Y_i \sim \text{Exp}(\lambda_1)$. We assume that we do not observe $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ directly, but that we observe

$$z_i = \max(x_i,\ y_i), \qquad \text{for} \quad i = 1, \ldots, n$$

and

$$u_i = I(x_i \geq y_i), \qquad \text{for} \quad i = 1, \ldots, n,$$

where $I(A) = 1$ if A is true and 0 otherwise. Based on the observed $(z_i, u_i), i = 1, \ldots, n$ we will use the EM-algorithm to find the maximum likelihood estimates for (λ_0, λ_1).

1. Define $\boldsymbol{x} = (x_1, \ldots, x_n)$ and similar definitions for $\boldsymbol{y}, \boldsymbol{z}$ and $\boldsymbol{u}$. The likelihood for the complete data $(\boldsymbol{x}, \boldsymbol{y})$ can be written as

$$\begin{aligned}
\mathcal{L}(\lambda_0, \lambda_1 \mid \boldsymbol{x}, \boldsymbol{y}) &= f_{X,Y}(\boldsymbol{x}, \boldsymbol{y} \mid \lambda_0, \lambda_1) = f_X(\boldsymbol{x} \mid \lambda_0) f_Y(\boldsymbol{y} \mid \lambda_1) \\
&= \prod_{i=1}^{n} f_X(x_i \mid \lambda_0) \prod_{i=1}^{n} f_Y(y_i \mid \lambda_1) \\
&= \prod_{i=1}^{n} \lambda_0 e^{-\lambda_0 x_i} \prod_{i=1}^{n} \lambda_1 e^{-\lambda_1 y_i} \\
&= \lambda_0^n e^{-\lambda_0 \sum_{i=1}^n x_i} \lambda_1^n e^{-\lambda_1 \sum_{i=1}^n y_i},
\end{aligned}$$

meaning that the log-likelihood becomes

$$\begin{aligned}
\ell(\lambda_0, \lambda_1 \mid \boldsymbol{x}, \boldsymbol{y}) &= \ln \mathcal{L}(\lambda_0, \lambda_1 \mid \boldsymbol{x}, \boldsymbol{y}) \\
&= n(\ln \lambda_0 + \ln \lambda_1) - \lambda_0 \sum_{i=1}^{n} x_i - \lambda_1 \sum_{i=1}^{n} y_i.
\end{aligned}$$

We want to use the EM-algorithm to find the MLE of λ_0 and λ_1. Denote by $\lambda_0^{(t)}$ and $\lambda_1^{(t)}$ the current parameter estimates at iteration t in the EM algorithm. We need to find the expected log-likelihood for $(\boldsymbol{x}, \boldsymbol{y})$ at this iteration conditioned on $\boldsymbol{z}$, $\boldsymbol{u}$, λ_0 and λ_1. This means finding an expression for

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

so we need to find expressions for $\text{E}\left[x_i \mid z_i, u_i, \lambda_0^{(t)}, \lambda_1^{(t)}\right]$ and $\text{E}\left[y_i \mid z_i, u_i, \lambda_0^{(t)}, \lambda_1^{(t)}\right]$. The derivations are similar, so we will only derive it for the first expression. Since u_i can only take the discrete values $\{0, 1\}$, we can study each case separately. If $u_i = 0$, we know that y_i is greater than x_i and $z_i = \max(x_i, y_i) = y_i$. It follows that

$$\begin{aligned}
\text{E}\left[x_i \mid z_i, u_i = 0, \lambda_0^{(t)}, \lambda_1^{(t)}\right] &= \text{E}\left[x_i \mid x_i \leq z_i, \lambda_0^{(t)}\right] \\
&= \int_0^{z_i} x_i \ \frac{f_X\left(x_i \mid \lambda_0^{(t)}\right)}{F_X\left(z_i \mid \lambda_0^{(t)}\right)} \ \text{d}x_i \\
&= \frac{1}{1 - e^{-\lambda_0^{(t)} z_i}} \int_0^{z_i} x_i \lambda_0^{(t)} e^{-\lambda_0^{(t)} x_i} \ \text{d}x_i \\
&= \frac{1}{1 - e^{-\lambda_0^{(t)} z_i}} \left(-z_i e^{-\lambda_0^{(t)} z_i} + \frac{1 - e^{-\lambda_0^{(t)} z_i}}{\lambda_0^{(t)}} \right) \\
&= \frac{1}{\lambda_0^{(t)}} - \frac{z_i}{e^{\lambda_0^{(t)} z_i} - 1}.
\end{aligned}$$

If $u_i = 1$, then x_i is greater than y_i and $z_i = \max(x_i, y_i) = x_i$, so the expectation simply becomes

$$\text{E}\left[x_i \mid z_i, u_i = 1, \lambda_0^{(t)}, \lambda_1^{(t)}\right] = z_i.$$

Combining these two cases it follows that

$$\begin{aligned}\mathrm{E}\left[x_i \mid z_i, u_i, \lambda_0^{(t)}, \lambda_1^{(t)}\right] &= I(u_i = 0)\left(\frac{1}{\lambda_0^{(t)}} - \frac{z_i}{e^{\lambda_0^{(t)} z_i} - 1}\right) + I(u_i = 1) z_i \\ &= (1 - u_i)\left(\frac{1}{\lambda_0^{(t)}} - \frac{z_i}{e^{\lambda_0^{(t)} z_i} - 1}\right) + u_i z_i.\end{aligned} \tag{2}$$

Similarly, the other expression becomes

$$\mathrm{E}\left[y_i \mid z_i, u_i, \lambda_0^{(t)}, \lambda_1^{(t)}\right] = u_i\left(\frac{1}{\lambda_1^{(t)}} - \frac{z_i}{e^{\lambda_1^{(t)} z_i} - 1}\right) + (1 - u_i) z_i. \tag{3}$$

Inserting the expected values from Equations 2 and 3, the expected log-likelihood of the complete data can be written as

$$\begin{aligned}\mathrm{E}\left[\ell(\lambda_0, \lambda_1 \mid \boldsymbol{x}, \boldsymbol{y}) \mid \boldsymbol{z}, \boldsymbol{u}, \lambda_0^{(t)}, \lambda_1^{(t)}\right] &= n\left(\ln \lambda_0 + \ln \lambda_1\right) \\ &\quad - \lambda_0 \sum_{i=1}^{n}\left[u_i z_i + (1 - u_i)\left(\frac{1}{\lambda_0^{(t)}} - \frac{z_i}{e^{\lambda_0^{(t)} z_i} - 1}\right)\right] \\ &\quad - \lambda_1 \sum_{i=1}^{n}\left[(1 - u_i) z_i + u_i\left(\frac{1}{\lambda_1^{(t)}} - \frac{z_i}{e^{\lambda_1^{(t)} z_i} - 1}\right)\right].\end{aligned} \tag{4}$$

2. Let the set of parameters be denoted as $\boldsymbol{\theta} = (\lambda_0, \lambda_1)$. For the M-step of the EM-algorithm, we need to calculate

$$\boldsymbol{\theta}^{(t+1)} = \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmax}}\, Q(\boldsymbol{\theta}),$$

where $Q(\boldsymbol{\theta}) = Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}) = \mathrm{E}\left[\ell(\boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{y}) \mid \boldsymbol{z}, \boldsymbol{u}, \boldsymbol{\theta}^{(t)}\right]$ is given in Equation 4. The range of possible parameter values are $\Theta = \mathbb{R}^2_{>0}$. The value of $\boldsymbol{\theta}$ giving the maximum value can be found from setting the gradient equal to zero, which results in solving the following set of equations:

$$\frac{\partial}{\partial \lambda_0} Q(\boldsymbol{\theta}) = 0, \qquad \frac{\partial}{\partial \lambda_1} Q(\boldsymbol{\theta}) = 0.$$

We find the solutions

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

and

$$\begin{aligned}\frac{\partial}{\partial \lambda_1} Q(\boldsymbol{\theta}) &= \frac{n}{\lambda_1} - \sum_{i=1}^{n}\left[(1 - u_i) z_i + u_i\left(\frac{1}{\lambda_1^{(t)}} - \frac{z_i}{e^{\lambda_1^{(t)} z_i} - 1}\right)\right] = 0 \\ &\Longrightarrow \\ \lambda_1 &= n\left(\sum_{i=1}^{n}\left[(1 - u_i) z_i + u_i\left(\frac{1}{\lambda_1^{(t)}} - \frac{z_i}{e^{\lambda_1^{(t)} z_i} - 1}\right)\right]\right)^{-1}.\end{aligned}$$

To check that $\boldsymbol{\theta}^{(t+1)} = (\tilde{\lambda_0}, \tilde{\lambda_1})$ is a global maximum, we evaluate the Hessian matrix,

$$H_Q(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial^2 Q(\boldsymbol{\theta})}{\partial \lambda_0^2} & \frac{\partial^2 Q(\boldsymbol{\theta})}{\partial \lambda_0 \partial \lambda_1} \\ \frac{\partial^2 Q(\boldsymbol{\theta})}{\partial \lambda_1 \partial \lambda_0} & \frac{\partial^2 Q(\boldsymbol{\theta})}{\partial \lambda_1^2} \end{bmatrix} = \begin{bmatrix} -\frac{n}{\lambda_0^2} & 0 \\ 0 & -\frac{n}{\lambda_1^2} \end{bmatrix},$$

which is negative-definite for all $\boldsymbol{\theta} \neq \mathbf{0}$. In addition, we must check that both λ_0 and λ_1 are positive to ensure that $\boldsymbol{\theta} \in \Theta$. Since both λ_0 and λ_1 are equal to the sum of non-negative terms it follows that $\boldsymbol{\theta}^{(t+1)} > 0$ as long as $\boldsymbol{\theta} > 0$. This means that $\boldsymbol{\theta} = (\tilde{\lambda_0}, \tilde{\lambda_1})$ is a global maximum for $Q(\boldsymbol{\theta})$. A recursion to find the maximum likelihood estimators is therefore given by

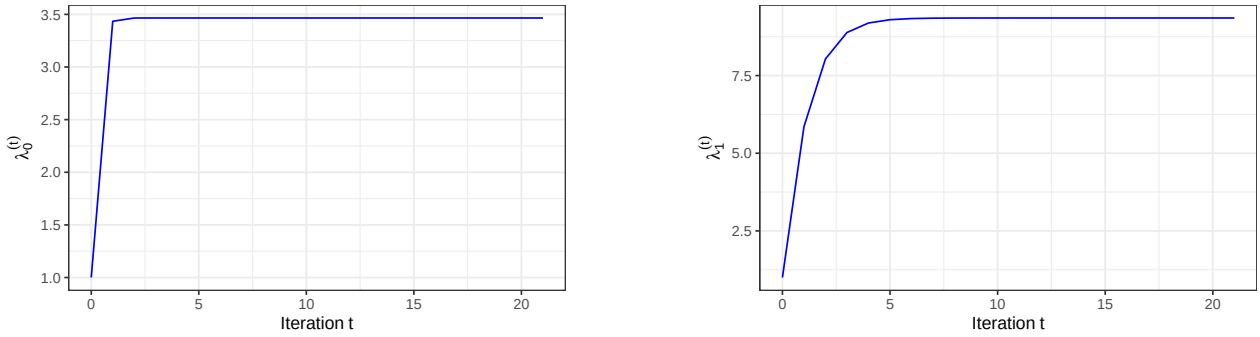
$$\lambda_0^{(t+1)} = n \left(\sum_{i=1}^{n} \left[u_i z_i + (1-u_i) \left(\frac{1}{\lambda_0^{(t)}} - \frac{z_i}{e^{\lambda_0^{(t)} z_i} - 1} \right) \right] \right)^{-1},$$

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

As a stopping criteria, we calculated the maximum absolute difference of the current parameter estimate and the parameter estimate at the next iteration by

$$\text{error}^{(t)} = \max \left(|\lambda_0^{(t+1)} - \lambda_0^{(t)}|, |\lambda_1^{(t+1)} - \lambda_1^{(t)}| \right),$$

and stopped once the error was below a set tolerance of $1.0 \cdot 10^{-8}$. Using initial values $\boldsymbol{\theta}^{(0)} = (\lambda_0^{(0)}, \lambda_1^{(0)}) = (1, 1)$, the EM-algorithm converged after 22 iterations. The final parameter estimates where $(3.466, 9.353)$. Convergence plots are displayed in Figure 4. From the plots, we see that the EM-algorithm converged very close to the actual parameter estimates after about 5 iterations.



(a) Convergence of λ_0.

(b) Convergence of λ_1.

Figure 4: Convergence of $\boldsymbol{\theta}$ with initial values $(1, 1)$.

```
# implementation of the EM algorithm
EM.alg = function(z, u, lambda0, lambda1, tol=1e-8) {
  n <- length(z)
  err <- Inf
  lambda0.iter <- c(lambda0)
  lambda1.iter <- c(lambda1)

  t <- 0
  while(err > tol){
    t <- t+1

    # add new lambda
    lambda0.iter <- c( lambda0.iter,
    n/sum(u*z + (1-u)*(1/lambda0.iter[t] - z/(exp(lambda0.iter[t]*z) - 1))) )
    lambda1.iter <- c( lambda1.iter,
    n/sum((1-u)*z + u*(1/lambda1.iter[t] - z/(exp(lambda1.iter[t]*z) - 1))) )

    # calculate maximum error
    err <- max(abs(lambda0.iter[t] - lambda0.iter[t+1]),
```

```
    abs(lambda1.iter[t] - lambda1.iter[t+1]))
  }

  return(list(lambda0=lambda0.iter, lambda1=lambda1.iter))
}

# read in the data
z = read.table('files/z.txt')[,1]
u = read.table('files/u.txt')[,1]

# find the MLE of lambda_0 and lambda_1
MLE = EM.alg(z, u, 1, 1)

print(tail(MLE$lambda0,1)) # 3.465735
print(tail(MLE$lambda1,1)) # 9.353215

df.lambda <- data.frame(x=1:length(MLE$lambda0)-1, lam0=MLE$lambda0, lam1=MLE$lambda1)

# convergence plots of the MLE estimates
convergence_lambda0 <- ggplot(df.lambda, aes(x=x)) +
  geom_line(aes(y=lam0), lwd=0.5, color='blue') + xlab(expression(Iteration~t)) +
  ylab(expression(lambda[0]^{(t)})) + theme_bw()
ggsave("./figures/convergence_lambda0.pdf", plot = convergence_lambda0, width = 5, height = 3)

convergence_lambda1 <- ggplot(df.lambda, aes(x=x)) +
  geom_line(aes(y=lam1), lwd=0.5, color='blue') + xlab(expression(Iteration~t)) +
  ylab(expression(lambda[1]^{(t)})) + theme_bw()
ggsave("./figures/convergence_lambda1.pdf", plot = convergence_lambda1, width = 5, height = 3)
```

3. Next, we want to use bootstrapping in order to estimate the standard deviations and the biases of both $\hat{\lambda}_0$ and $\hat{\lambda}_1$. The algorithm used to generate the bootstrap samples is given in 1.

Algorithm 1 Bootstrap algorithm for $(\hat{\lambda}_0, \hat{\lambda}_1)$.

Require: $B > 0$, vectors $\hat{\boldsymbol{\lambda}}_0^*$ and $\hat{\boldsymbol{\lambda}}_1^*$ with B elements.
  for $i = 1, \ldots, B$ **do**
    Uniformly Sample (z_i^*, u_i^*) from $\boldsymbol{z}$ and $\boldsymbol{u}$ n times with replacement.
    Based on these samples, calculate $(\hat{\lambda}_0^*, \hat{\lambda}_1^*)$ with the EM-algorithm.
    Store $\hat{\lambda}_0^*$ and $\hat{\lambda}_1^*$ as a bootstrap sample on the i'th index of $\hat{\boldsymbol{\lambda}}_0^*$ and $\hat{\boldsymbol{\lambda}}_1^*$, respectively.
  end for

The standard deviations and the biases computed from the bootstrap sample is given in Table 3. The corresponding R-code is given below.

```
# Bootstrapping algo. for lambda0 and lambda1
boot.lambda <- function(B, seed = 4300){
  set.seed(seed)
  n <- length(z)
  lambda0.b <- lambda1.b <- rep(NA, B)
  for(i in 1:B){
    sample.idx <- sample(1:n, replace = TRUE)
    zb <- z[sample.idx]
    ub <- u[sample.idx]
    lambdas <- EM.alg(zb, ub, 1, 1)
    lambda0.b[i] <- tail(lambdas$lambda0, 1)
```

```
    lambda1.b[i] <- tail(lambdas$lambda1, 1)
  }
  return(list(lambda0 = lambda0.b, lambda1 = lambda1.b))
}

# Run bootstrap
lambdas.b <- boot.lambda(30000)
lambda0.b <- lambdas.b$lambda0
lambda1.b <- lambdas.b$lambda1

# Find standard deviation and bias
sdev <- c(sd(lambda0.b), sd(lambda1.b))
bias <- c(mean(lambda0.b) - tail(MLE$lambda0, 1),
          mean(lambda1.b) - tail(MLE$lambda1, 1))
boot.df <- data.frame(sdev, bias)
xtable(boot.df)

# Find correlation
cor(lambdas.b$lambda0, lambdas.b$lambda1)
```

	$\widehat{\text{SD}}_B$	$\widehat{\text{bias}}_B$
$\hat{\lambda}_0$	0.2485	0.0168
$\hat{\lambda}_1$	0.8019	0.0729

Table 3: The bootstrap estimates of the standard deviations and biases of $\hat{\lambda}_0$ and $\hat{\lambda}_1$.

We observe that the standard deviations are an order of magnitude larger than the bias, which leads us to believe that the variance is the dominating source of error in the estimators. Adjusting the estimators by subtracting the the bias, i.e. correcting them for bias, will further increase the variance of the estimators. Consequently, we prefer the maximum likelihood estimates to the the bias corrected estimates of λ_0 and λ_1.

The empirical correlation from the bootstrap-samples evaluates to $\text{Corr}[\hat{\lambda}_0, \hat{\lambda}_1] \approx -0.00065$, i.e. nearly 0, which is reassuring, since we initially assumed the x_i's and y_i's to be independent.

4. As an alternative to the EM-algorithm, we want to investigate the possibility of calculating the MLEs of λ_0 and λ_1 directly, either analytically or numerically. In order to achieve this, we need to determine $f_{Z_i,U_i}(z_i, u_i \mid \lambda_0, \lambda_1)$. Hence, we consider the situation where $U_i = 1$, which gives the joint cumulative distribution

$$\begin{aligned} F_{Z_i,U_i}(z_i, u_i = 1|\lambda_0, \lambda_1) &= F_{Z_i|U_i}(z_i|u_i = 1, \lambda_0, \lambda_1) \cdot \Pr(U_i = 1) \\ &= \frac{\lambda_1}{\lambda_0 + \lambda_1} \int_0^{z_i} \int_0^{x_i} f_{X,Y}(x_i, y_i) \mathrm{d}y_i \mathrm{d}x_i \\ &= \frac{\lambda_1}{\lambda_0 + \lambda_1} \int_0^{z_i} \lambda_0 e^{-\lambda_0 x_i}(1 - e^{-\lambda_1 x_i}) \mathrm{d}x_i, \end{aligned}$$

where we have used that

$$\Pr(U_i = 1) = \Pr(X_i \geq Y_i) = \int_0^{\infty} \int_0^{x_i} f_{X,Y}(x_i, y_i) \mathrm{d}y_i \mathrm{d}x_i = \frac{\lambda_1}{\lambda_0 + \lambda_1}.$$

The fundamental theorem of calculus then tells us that

$$f_{Z_i,U_i}(z_i, u_i = 1|\lambda_0, \lambda_1) \propto \frac{\lambda_0 \lambda_1}{\lambda_0 + \lambda_1} e^{-\lambda_0 z_i}(1 - e^{-\lambda_1 z_i}). \tag{5}$$

Normalizing (5), we finally obtain

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

An analogous procedure when $U_i = 0$ gives

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

Thus, we can write the joint distribution as

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

Now, we can write up the log-likelihood. For ease of notation, we define the index sets $\mathcal{M}_0 = \{i : u_i = 0\}$ and $\mathcal{M}_1 = \{i : u_i = 1\}$. Then, the log-likelihood can be written as

$$\begin{aligned} \ell(\lambda_0, \lambda_1; \boldsymbol{z}, \boldsymbol{u}) &= \ln\left(\prod_{i=1}^{n} f_{Z_i,U_i}(z_i, u_i \mid \lambda_0, \lambda_1)\right) \\ &= |\mathcal{M}_0| \ln \lambda_1 + |\mathcal{M}_1| \ln \lambda_0 \\ &\quad + \sum_{i \in \mathcal{M}_0} \left[\ln(1 - e^{-\lambda_0 z_i}) - \lambda_1 z_i\right] + \sum_{i \in \mathcal{M}_1} \left[\ln(1 - e^{-\lambda_1 z_i}) - \lambda_0 z_i\right]. \end{aligned}$$

Furthermore, we compute the partial derivatives:

$$\begin{aligned} \frac{\partial \ell(\lambda_0, \lambda_1; \boldsymbol{z}, \boldsymbol{u})}{\partial \lambda_0} &= \frac{|\mathcal{M}_1|}{\lambda_0} - \sum_{i \in \mathcal{M}_1} z_i + \sum_{i \in \mathcal{M}_0} \frac{z_i}{e^{\lambda_0 z_i} - 1} \\ \frac{\partial \ell(\lambda_0, \lambda_1; \boldsymbol{z}, \boldsymbol{u})}{\partial \lambda_1} &= \frac{|\mathcal{M}_0|}{\lambda_1} - \sum_{i \in \mathcal{M}_0} z_i + \sum_{i \in \mathcal{M}_1} \frac{z_i}{e^{\lambda_1 z_i} - 1} \end{aligned}$$

From this, we are not able to set the equations to zero and acquire a closed form for $\hat{\lambda}_0$ and $\hat{\lambda}_1$. However, we observe that the Hessian,

$$\nabla^2 \ell(\lambda_0, \lambda_1; \boldsymbol{z}, \boldsymbol{u}) = \begin{bmatrix} -\frac{|\mathcal{M}_1|}{\lambda_0^2} - \sum_{i \in \mathcal{M}_0} \frac{z_i^2 e^{\lambda_0 z_i}}{(e^{\lambda_0 z_i} - 1)^2} & 0 \\ 0 & -\frac{|\mathcal{M}_0|}{\lambda_1^2} - \sum_{i \in \mathcal{M}_1} \frac{z_i^2 e^{\lambda_1 z_i}}{(e^{\lambda_1 z_i} - 1)^2} \end{bmatrix},$$

is negative definite, which means that the negated log-likelihood is convex. This makes the problem of finding $(\hat{\lambda}_0, \hat{\lambda}_1)$ a prime candidate for the vast arsenal of optimization techniques. We choose to use the derivative-free method which the `optim` function in `R` provides. The code is given below.

```
# Log-likelihood
log.lik <- function(lambdas){
  l0 <- lambdas[1]
  l1 <- lambdas[2]
  M0 <- which(u == 0)
  M1 <- which(u == 1)
  return(length(M0)*log(l1) + sum(log(1 - exp(-l0*z[M0])) - l1*z[M0]) +
           length(M1)*log(l0) + sum(log(1 - exp(-l1*z[M1])) - l0*z[M1]))
}

# Run optimization
mle <- optim(par = c(1,1), fn = log.lik, control = list(fnscale = -1))
mle$par
```

The result is approximately the same as for the EM-algorithm, with $(\hat{\lambda}_0, \hat{\lambda}_1) \approx (3.4659, 9.3511)$, which is what we would expect. An advantage of optimizing the likelihood directly is that there is usually some sort of guarantee on the converge rates, e.g. Newton's method has quadratic convergence 'close' to the solution, while the EM-algorithm

has no guarantee on the convergence rate, though the estimates will improve for each iteration. Furthermore, it may be difficult to set a satisfactory stopping criterion for the EM-algorithm, since it (in this case) is based on the change is parameter estimates and not in change of likelihood. Another important advantage of optimizing the likelihood directly is that (assuming that the MLEs are approximately normal and that the Hessian is available) we can acquire estimates of the standard deviations for the maximum likelihood estimates.

References

[1] Bent Jørgensen. *The Theory of Linear Models*. Chapman and Hall, 1993.