Exercise 3

TMA4300

Erling Fause Steen, Christian Oppegård Moen

Spring 2022

Contents

Introduction

Problem A

In this problem we will analyze a dataset which contain a sequence of length $T = 100$ of a non-Gaussian time-series displayed in Figure 1, for which we will compare two different parameter estimators. Consider the autoregressive model of order 2 (AR(2)) specified by the relation

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

where e_t are independent and identically distributed (iid) variables with zero mean and constant variance. Also, consider the loss functions with respect to $\boldsymbol{\beta} = [\beta_1, \beta_2]^T$ given by

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

Then, the least sum residuals (LS) and least sum of absolute residuals (LA) are obtained by minimizing $Q_{LS}(\boldsymbol{x})$ and $Q_{LA}(\boldsymbol{x})$ respectively. We denote the minimisers (our two different parameter estimators) by $\hat{\boldsymbol{\beta}}_{LS}$ and $\hat{\boldsymbol{\beta}}_{LA}$, and define the estimated residuals by $\hat{e}_t = x_t - \hat{\beta}_1 x_{t-1} - \hat{\beta}_2 x_{t-2}$ for $t = 3, ..., T$ with mean $\bar{e}$.

```
x = data3A$x
plot(x, type = "l", xlab = "t", ylab = "x")
```

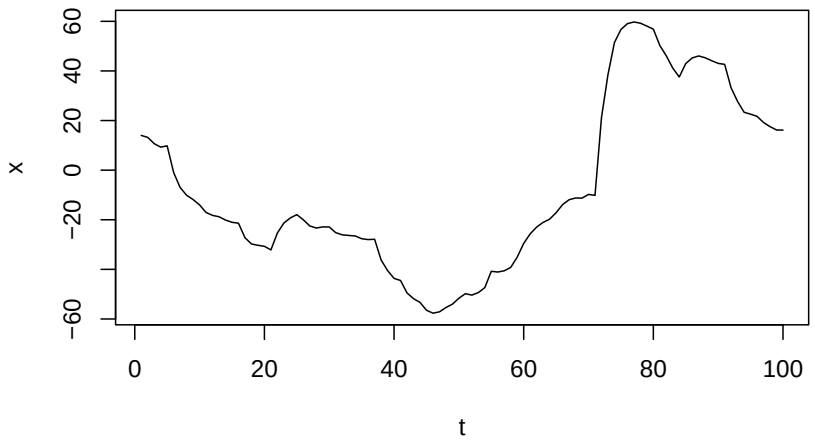

Figure 1: The non-Gaussian time-series.

1

Now we will use the residual resampling bootstrap method to evaluate the relative performance of the two parameter estimators, $\hat{\boldsymbol{\beta}}_{LS}$ and $\hat{\boldsymbol{\beta}}_{LA}$, which are calculated by the given function `ARp.beta.est` in `R`. We

consider the variance and bias of the estimators $\hat{\boldsymbol{\beta}}^*_{LS}$ and $\hat{\boldsymbol{\beta}}^*_{LA}$ obtained by minimizing $Q_{LS}(\boldsymbol{x}^*)$ and $Q_{LA}(\boldsymbol{x}^*)$ respectively, for pseudo data $\boldsymbol{x}^*$. The full method is shown in Algorithm 1. In Table 1 the variance of $\hat{\boldsymbol{\beta}}^*_{LA}$ is lower than that of $\hat{\boldsymbol{\beta}}^*_{LS}$, and the absolute value of the bias is lower as well. This suggest that the LS estimator is not optimal for this non-Gaussian time-series.

Algorithm 1

for LS and LA estimator of $\boldsymbol{\beta}$ **do**
  for $t = 3, ..., n$ **do**
    Define estimated residuals $\hat{e}_t$.
  end for
  for $b = 1, ..., B$ **do**
    Resample $n + 1$ pseudo residuals $\hat{\boldsymbol{e}}^*_b$ from the estimated residuals $\hat{\boldsymbol{e}}_b$ with replacement.
    Generate pseudo data $\boldsymbol{x}^*_b$.
    Estimate $\hat{\boldsymbol{\beta}}^*_b$.
  end for
end for

```
detach("package:dplyr", unload = T)
rsBoot = function(B, x, p = 2) {
    T = length(x)
    # Estimate beta
    beta.hat = ARp.beta.est(x, p)
    # calculate observed residuals of AR(p) seq
    e.LS.observed = ARp.resid(x, beta.hat$LS)
    e.LA.observed = ARp.resid(x, beta.hat$LA)
    # Allocate memory
    beta.LS.star = matrix(nrow = B, ncol = 2)
    beta.LA.star = matrix(nrow = B, ncol = 2)

    # Bootstrap
    for (b in 1:B) {
        # Resample from observed residuals to yield pseudo innovations
        e.LS.star = sample(e.LS.observed, size = T, replace = TRUE)
        e.LA.star = sample(e.LA.observed, size = T, replace = TRUE)
        # Generate pseudo data (Timeseries based on sampled residuals and
        # beta.hat)
        i = sample(T - 1, 1)
        x.LS.star = ARp.filter(x[c(i, i + 1)], beta.hat$LS, e.LS.star)
        x.LA.star = ARp.filter(x[c(i, i + 1)], beta.hat$LA, e.LA.star)
        # compute beta star
        beta.LS.star[b, ] = ARp.beta.est(x.LS.star, 2)$LS
        beta.LA.star[b, ] = ARp.beta.est(x.LA.star, 2)$LA
    }
    return(list(beta.hat = beta.hat, beta.LS = beta.LS.star, beta.LA = beta.LA.star,
        e.LS = e.LS.observed, e.LA = e.LA.observed, x.LS = x.LS.star, x.LA = x.LA.star))
}

B = 1500
set.seed(420)
boot = rsBoot(B, x)

# Compute variance and bias
beta.var = rbind(apply(boot$beta.LS, 2, var), apply(boot$beta.LA, 2, var))
```

Table 1: Variance and bias for $\hat{\beta}_1^*$ and $\hat{\beta}_2^*$.

	Var_1	Var_2	bias_1	bias_2
LS	0.005451	0.005240	-0.012071	0.005988
LA	0.000362	0.000358	-0.001311	0.000829

Table 2: Prediction interval of x_{101}.

	2.5%	97.5%
LS	10.01	23.34
LA	7.21	23.67

```
beta.bias = rbind(apply(boot$beta.LS, 2, mean) - boot$beta.hat$LS, apply(boot$beta.LA,
    2, mean) - boot$beta.hat$LA)
beta.vb = cbind(beta.var, beta.bias)
rownames(beta.vb) = c("LS", "LA")

kable(round(beta.vb, 6), caption = "Variance and bias for $\\hat{\\beta_1^*}$ and $\\hat{\\beta_2^*}$."
    format = "latex") %>%
    kable_styling() %>%
    add_header_above(c(" ", "Var$_{1}$", "Var$_{2}$", "bias$_{1}$", "bias$_{2}$"),
        escape = FALSE)
```

2

Next, we will compute a 95% prediction interval for x_{101} for both the LS and the LA estimator. That is, we bootstrap sample B times

$$x_{101} = \beta_1^* x_{100} + \beta_2^* x_{99} + \hat{e}_i$$

where β_1^* and β_2^* are sampled from all 1500 bootstrap samples $\hat{\boldsymbol{\beta}}^*$ found in section A1, and $\hat{e}_i$ is sampled from the estimated residuals $\hat{e}_t$ for $t = 2, ..., 100$. The prediction intervals are found in Table 2.

```
set.seed(420)
# Bootstrap x_101 using beta.star and e.observed
x_101.LS = vector(mode = "double", length = B)
x_101.LA = vector(mode = "double", length = B)
for (b in 1:B) {
    # i.e = sample(98, 1)
    i.beta = sample(1500, 1)
    # x_101.LS[b] = boot$beta.LS[i.beta,] %*% x[c(100,99)] + boot$e.LS[i.e]
    # x_101.LA[b] = boot$beta.LS[i.beta,] %*% x[c(100,99)] + boot$e.LA[i.e]
    x_101.LS[b] = boot$beta.LS[i.beta, ] %*% x[c(100, 99)] + sample(boot$e.LS, 1)
    x_101.LA[b] = boot$beta.LS[i.beta, ] %*% x[c(100, 99)] + sample(boot$e.LA, 1)
}

pi.boot = rbind(LS = quantile(x_101.LS, probs = c(0.025, 0.975)), LA = quantile(x_101.LA,
    probs = c(0.025, 0.975)))
kable(round(pi.boot, 2), caption = "Prediction interval of $x_{101}$.")
```

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	
3	0.32	0.41	0.41	0.55	0.55	0.62	0.71	0.91			

Table 3: Concentration of bilirubin in three different individuals.

Problem B

We will investigate the concentration of bilirubin (mg/dL), which is a breakdown of haemoglobin, in blood samples taken from three young men shown in Table 3.

1

The logarithms of the concentration for each individual are displayed in the boxplot in Figure 2. We see that the mean logarithm of bilirubin is smallest for individual 1 and highest for individual 2. The variation is smallest for individual 2 while individual 1 have slightly larger standard deviation than individual 3.

```
ylim = c(-2.1, 0)
bilirubin <- read.table("./additionalFiles/bilirubin.txt", header = T)
boxplot(log(meas) ~ pers, data = as.data.frame(bilirubin), xlab = "Person", ylab = "log(meas)",
    ylim = ylim)
```

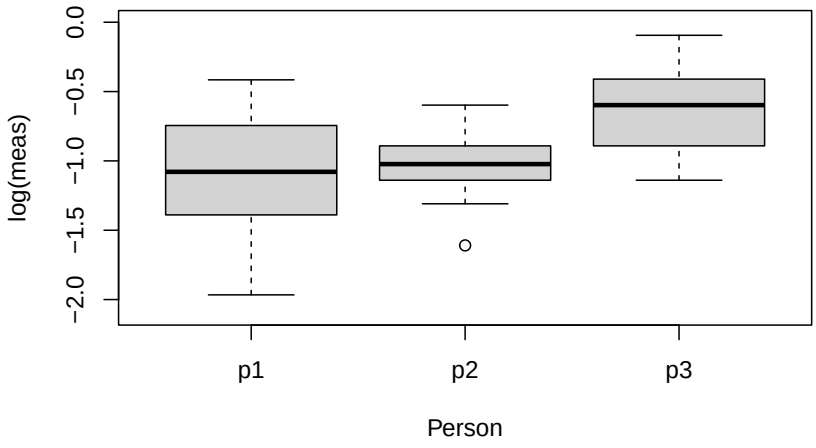

Figure 2: The logarithm of the concentration of bilirubin (`meas`) for each individual.

```
fitB1 = lm(log(meas) ~ pers, data = bilirubin)
fitB1.Fstat = summary(fitB1)$fstatistic
Fval = fitB1.Fstat["value"]
Pval = pf(Fval, fitB1.Fstat[2], fitB1.Fstat[3], lower.tail = F)
```

Let $\log Y_{ij} = \log(\texttt{meas}_{ij})$ for individual $i = 1, 2, 3$ and observation $j = 1, ..., n_i$, where $n_1 = 11$, $n_2 = 10$ and

$n_3 = 8$. In the chunk above we fit the regression model

$$\log Y_{ij} = \beta_i + \varepsilon_{ij}, \tag{1}$$

where $\varepsilon_{ij} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$. Then, the F-statistic on $3-1$ and $29-3$ degrees of freedom of the hypothesis that $\beta_1 = \beta_2 = \beta_3$ is 3.67 with p-value 0.03946. With the F-statistic we are investigating whether the bilirubin of each individual are significantly equally distributed. Judging from the p-value we reject the hypothesis on a $\alpha = 0.05$-level. That is, on a $\alpha = 0.05$-level, the individuals do not share the same concentration of bilirubin.

2

To further investigate the individuals distribution of bilirubin concentration we consider a permutation test where the idea is that shuffling the labels will not change the joint distribution of the data. That is, we assume that the distributions are equal so that shuffling the order of the data to generate bootstrap samples should be valid. This is done by first randomly assigning bilirubin values, `meas`, to the three individuals. then we generate $B = 999$ bootstrap samples. Implementation of the permutation and computation of the F-statistic is seen in the chunk below.

```
permTest = function(data = bilirubin) {
    bilirubin$perm = bilirubin$meas[sample(29, 29)]
    return(summary(lm(log(perm) ~ pers, data = bilirubin))$fstatistic["value"])
}
```

3

We generate $B = 999$ samples of the F-statistic, `Fvals`, using the function `permTest`. Then we compute the p-value defined by `Pval`$= (\texttt{Fvals} \geq \texttt{Fval})/B$, where `Fval` is the F-statistic found in section B1.

```
set.seed(420)
B = 999
Fvals = vector(mode = "double", length = 999)
for (b in 1:B) {
    Fvals[b] = permTest()
}
Pval.perm = sum(Fvals >= Fval)/B
```

The p-value of the 999 F-statistics for this run was 0.03704 which also reject the hypothesis on a $\alpha = 0.05$-level. Comparing it to the p-value from section B1 we see that they are almost equal, with the value from the permutation test being slightly lower.