Time Series Analysis

Assignment 2

AUTHORS

Elli Georgiou (s223408)
Dimitris Voukatas (s230148)
Dimitris Sousounis (s230118)

March 15, 2023

Question 2.1: Stability

We have the following process:

$$X_t = 0.8X_{t-1} + \epsilon_t + 0.8\epsilon_{t-1} - 0.5\epsilon_{t-2} \tag{1}$$

where ϵ_t is a white noise process with $\sigma = 0.4$.

Q2.1.1: Stationary and invertible

To examine whether the process is stationary we need to find the roots of the polynomial $\phi(z^{-1})$ and if the roots lie within the unit circle then the process is stationary. Furthermore, regarding the invertibility of the process we will locate the roots of the polynomial $\theta(z^{-1})$ and similarly if they lie within the unit circle then the process is invertible.

Generally, an ARMA(p, q) process can be written in the following form using the backward operator B:

$$\varphi(B)X_t = \theta(B)\varepsilon_t \tag{2}$$

where the $\varphi(B)$ and $\theta(B)$ are polynomials of order p and q, respectively and for the given X_t process these are the following :

$$\varphi(B) = 1 - 0.8B \quad \text{and} \quad \theta(B) = 1 + 0.8B - 0.5B^2 \tag{3}$$

Therefore the polynomial $\varphi(z^{-1})$ is:

$$\varphi(z^{-1}) = 1 - 0.8z^{-1} \tag{4}$$

The polynomial that is described by equation (4) has one root $z = 0.8$, which lies within the unit circle. Therefore, the process X_t is **stationary**.

As for the polynomial $\theta(z^{-1})$ that is:

$$\theta(z^{-1}) = 1 + 0.8z^{-1} - 0.5z^{-2} \tag{5}$$

The polynomial that is described by equation (5) has two roots $z_1 = 0.4124$ and $z_2 = -1.212$, which lies outside the unit circle. Therefore, the process X_t is not **invertible.**.

Q2.1.2: Second order moment representation

To analyze the second order moment representation of the process is to calculate the mean value and the auto-covariance function of the process.

Since the process is stationary, the mean value $\mathbb{E}[X_t]$ is time invariant. Thus $\mathbb{E}[X_t] = \mathbb{E}[X_{t-1}]$. Also, since the mean of ϵ_t is zero for all t, as $\{e_t\}$ is white noise, it follows:

$$\mu(t) = \mathbb{E}[X_t] = 0.8\mathbb{E}[X_{t-1}] + \mathbb{E}[\varepsilon_t] + 0.8\mathbb{E}[\varepsilon_{t-1}] - 0.5\mathbb{E}[\varepsilon_{t-2}] \Leftrightarrow$$
$$0.2\mathbb{E}[X_t] = 0 \Leftrightarrow \mathbb{E}[X_t] = 0$$

To calculate the auto-covariance function, first we have to calculate the correlation between the white noise and the process with the following difference equation:

$$\gamma_{\epsilon X}(k) + \phi_1\gamma_{\epsilon X}(k-1) + ... + \phi_p\gamma_{\epsilon X}(k-p) = \theta_k\sigma_\epsilon^2, \quad k = 0, 1, ... \tag{6}$$

In our case, for $p = 1$ and $q = 2$, the equation takes the form

$$\gamma_{\epsilon X}(k) + \phi_1\gamma_{\epsilon X}(k-1) = \theta_k\sigma_\epsilon^2, \quad k = 0, 1, ... \Leftrightarrow$$
$$k = 0 \Rightarrow \gamma_{\epsilon X}(0) = \theta_0\sigma_\epsilon^2$$
$$k = 1 \Rightarrow \gamma_{\epsilon X}(1) + \phi_1\gamma_{\epsilon X}(0) = \theta_1\sigma_\epsilon^2$$
$$k = 2 \Rightarrow \gamma_{\epsilon X}(2) + \phi_1\gamma_{\epsilon X}(1) = \theta_2\sigma_\epsilon^2$$
$$k \geq 3 \Rightarrow \gamma_{\epsilon X}(k) = -\phi_1\gamma_{\epsilon X}(k-1)$$

Now by solving the above linear system, the auto-covariance function can be found by solving the following system

$$\gamma(0) + \phi_1\gamma(1) = \theta\gamma_{\epsilon X}(0) + \theta_1\gamma_{\epsilon X}(1) + \theta_2\gamma_{\epsilon X}(2)$$
$$\gamma(1) + \phi_1\gamma(0) = \theta\gamma_{\epsilon X}(0) + \theta_2\gamma_{\epsilon X}(1)$$
$$\gamma(2) + \phi_1\gamma(1) = \theta_2\gamma_{\epsilon X}(0)$$
$$\gamma(k) + \phi_1\gamma(k-1) = 0$$

In Table 1, the values of the auto-covariance and auto-correlation function are shown for $k = 0, 1, \ldots, 5$. $(\rho(k) = \gamma(k)/\gamma(0))$

k	0	1	2	3	4	5
$\gamma(k)$	0.84	0.67	0.46	0.37	0.29	0.23
$\rho(k)$	1	0.8	0.54	0.43	0.35	0.27

Table 1: The auto-covariance and auto-correlation function.

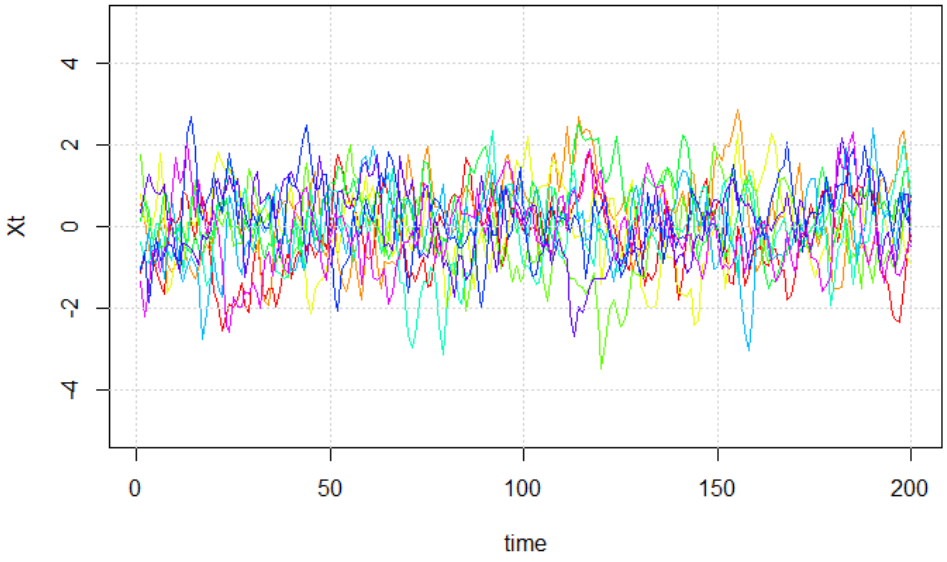


Figure 1: 10 realisations with 200 observations from the process X_t

Q2.1.3: 10 realisations

The following plot (Figure 1) illustrates 10 realizations of the X_t process, each with 200 observations. It can be observed that every realization is fluctuating around 0 with relatively small deviation from others, which is expected as process X_t is stationary.

Q2.1.4: ACF for 10 realisations

Auto-correlation refers to the correlation between a time series (signal) and a delayed version of itself, and figure 2 demonstrates the correlation coefficient as a function of the lag, serving as a graphical representation of auto-correlation. We notice that the ACF always equal one for a lag of zero, which makes sense because the signal is always perfectly correlated with itself. It is also observable that for lag equal to 1 and onward the ACFs are decaying exponentially and, for higher lags the majority of the estimations of the ACF are within the confidence intervals but there are exceptions.

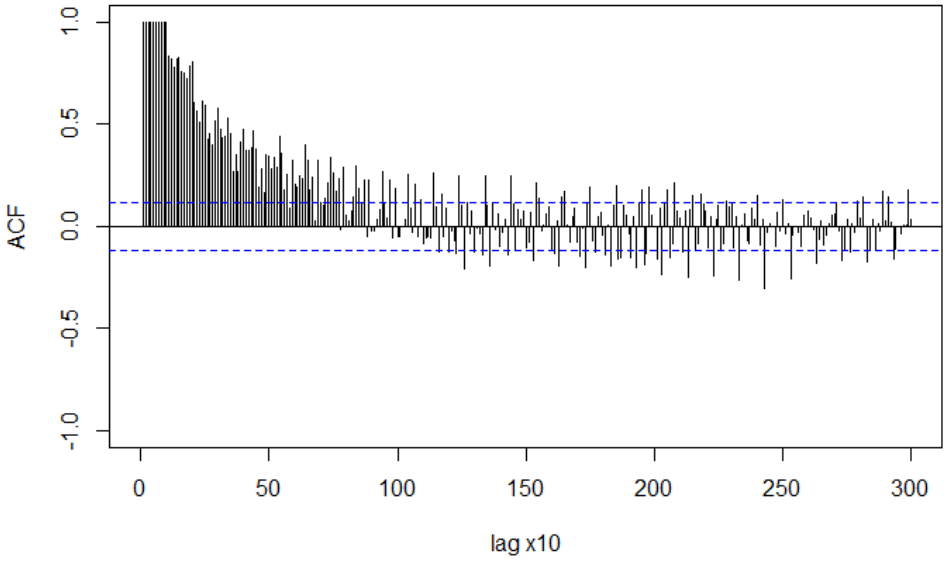


Figure 2: The ACF of 10 realisations with 200 observations from the process X_t

Q2.1.5: PACF for 10 realisations

The PACF figure 3 presents similar results as well as a damped sine function for lag 1 onwards.

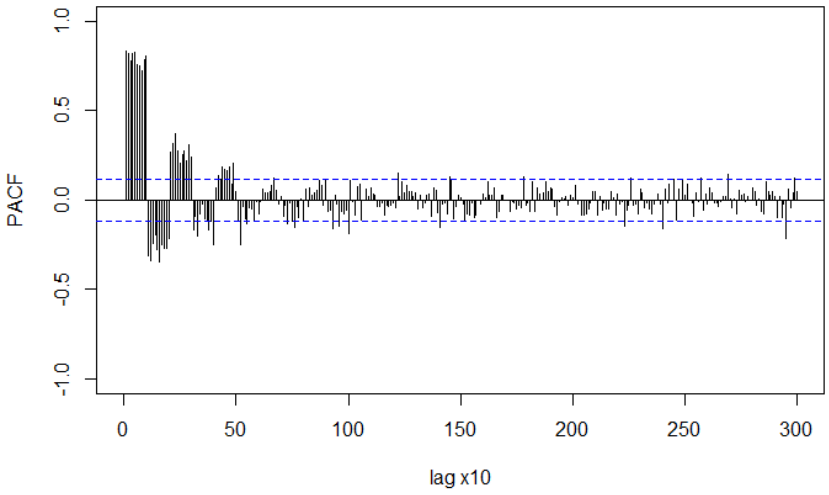


Figure 3: The PACF of 10 realisations with 200 observations from the process X_t

Q2.1.6: Variance for 10 realisations

Realization	Variance
1	0.9105593
2	1.0225407
3	0.9806444
4	0.9008256
5	0.6344074
6	0.8396065
7	0.7447493
8	0.7697457
9	0.7687580
10	0.8911214

Table 2: Variance of each of the 10 realisations

From table 2 and the mean value of the variances of the realizations (0.846), it is clear that the analytical and numerical calculations are fairly close. The standard deviation of the mean value is also easily calculated and equal to 0.065 with which we can calculate a 97.5 % confidence interval (0.7186, 0.9914) which the variances fit nicely (for the most part).

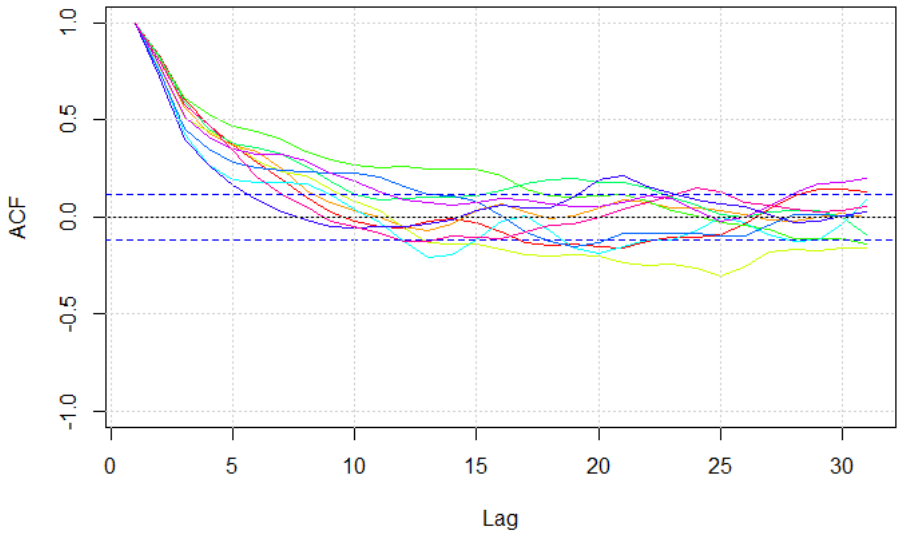
Q2.1.7: Compare and discuss

Comparing the analytical results from question 2.1.2 with the numerical results from the figure 2 in table 3, we can see that the two approaches agree with each other with only slight deviations. This is probably because the numerical variances are based on an exact number of simulations, which is 10 while the analytical variance is based on a theoretical approach. Overall, the numerical estimates provide an accurate representation of the actual variance of the ARIMA process, but they are still sensitive to some sampling fluctuation.

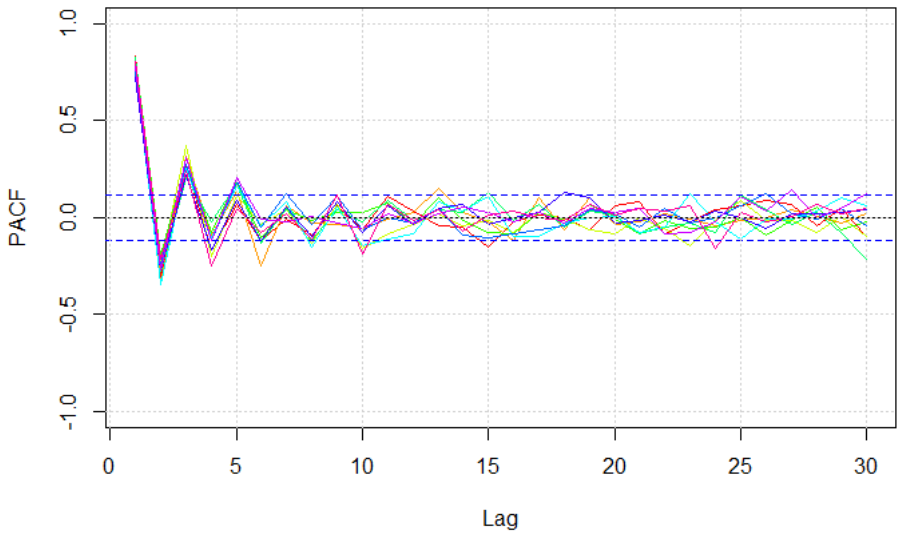
What is more noticeable and unexpected is the amount of estimations outside the confidence intervals. These values suggest the presence of significant auto-correlation in the data. By the numbers, for 10 realizations with 30 lags each and a 95 % confidence interval, we would expect 10 significant lags after lag 10. In our cases, we can count more and many of them are quite significant.

Lag	Analytical	Numerical
0	1	1
1	0.8	0.7902643
2	0.54	0.5243413
3	0.43	0.4096335
4	0.35	0.3311211

Table 3: Analytical and Numerical Results of Auto-correlation



(a) The ACF of 10 realisations with 200 observations from the process X_t



(b) The PACF of 10 realisations with 200 observations from the process X_t

Q2.2: Predicting the number of sales of apartments

Q2.2.1

Our $(2,0,0) \times (1,0,1)_4$ seasonal model includes three major components: the autoregressive component, the autoregressive seasonal component, and the moving average seasonal term. The autoregressive component is represented by a quadratic polynomial $(1-1.04B+0.2B^2)$, which provides for a nonlinear trend over time. The seasonal component is represented by a fourth-order polynomial $(1-0.86B^4)$, which represents the quarterly seasonality in the data along with the error term which is represented by the fourth-order polynomial $(1-0.42B^4)$, which allows for residual autocorrelation. The error is generated via a white-noise process with a constant variance of $\sigma^2 = 36963$. This means the errors are uncorrelated across time and have a constant variance.

$$(1 - 1.04B + 0.2B^2)(1 - 0.86B^4)(Y_t - \mu) = (1 - 0.42B^4)\epsilon_t \tag{7}$$

We use our model and the estimated parameters to create forecasts and prediction intervals for Y_t values for t = 2019Q1 and 2019Q2, as well as 95% prediction intervals. One way to do this would be to calculate the white noise of each quarter starting from the beginning of the data until we reach the values necessary for the calculation of the conditional expectation hence the predicted values, assuming that all data and white noise before the start are zero.

$$\hat{X}_{t+1|t} = \mathbb{E}[Y_{t+1}|Y_t, Y_{t-1}, ...] = 1.04X_t - 0.2X_{t-1} + 0.86X_{t-3} - 0.89X_{t-4} + 0.17X_{t-5} - 0.42\epsilon_{t-3}$$

where $X_t = Y_t - \mu$. Similarly, we can predict the sales for the next quarter

$$\hat{X}_{t+2|t} = \mathbb{E}[Y_{t+1}|Y_t, Y_{t-1}, ...] = 1.04\hat{X}_{t+1|t} - 0.2X_t + 0.86X_{t-2} - 0.89X_{t-3} + 0.17X_{t-4} - 0.42\epsilon_{t-2}$$

But this process is computationally slow so we opted out for the much faster functions in R.

Time	predicted number of sales	standard errors
2019Qtr1	2200.279	192.2816
2019Qtr2	2272.447	277.3975

Table 4: predictions of Yt for t = 2019Q1 and 2019Q2

According to the model , the number of flats sales in Denmark's capital region is predicted to grow significantly from Qtr1 to Qtr2 in 2019. However, these predictions are subject to uncertainty, and the actual number of sales might well be higher or lower than expected.

The prediction interval width shows the uncertainty regarding the point forecasts. The prediction interval for Qtr2 is wider than the prediction interval for Qtr1, showing that the forecasted value for Qtr2 is more uncertain. This is in agreement with the fact that Qtr2 has a higher standard error than Qtr1. Also, the predictions further down the time horizon are bound to be less accurate than the previous ones.

Time	Lower bound	Upper bound
2019Qtr1	1823.414	2577.144
2019Qtr2	1728.758	2816.136

Table 5: 95% prediction intervals

The prediction intervals are easily calculated like so:

$$\hat{Y}_{t+1|t} \pm \sigma_\epsilon, \quad \hat{Y}_{t+2|t} \pm \sigma_\epsilon \sqrt{1 + 1.04^2}$$

but we already started our calculations in R, we might as well finish them in R.

Q2.2.2: Actual and predicted values of Y_t

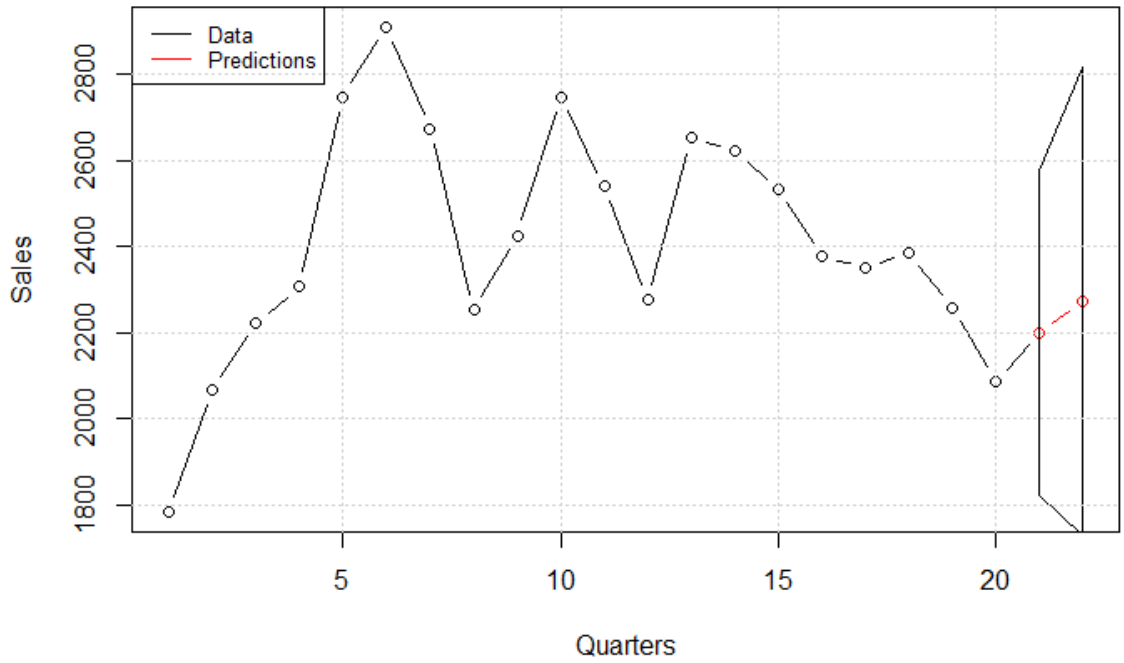


Figure 5: Actual and predicted values of Y_t

Based on Figure 5, we can conclude that the sales of apartments in Denmark's capital region have fluctuated throughout time. There were phases of significant growth, slow decline, and stability. On the other hand, the predictions show a wide range of sales values for the following two quarters.

The intervals are wider since the sample on which the forecast model is based is relatively small, which adds a greater degree of uncertainty to the predictions. Potentially, this could be improved if we contribute additional training data to our model, especially if we aim to make more accurate predictions in the long term.

We then compute the Root Mean Square Error, which is a measure of the model's accuracy; a lower RMSE number implies a better fit of the model to the data. We found that the RMSE is 136.874, indicating that the ARIMA model is not an excellent fit for our sales data, but it can still provide important insights and forecasts.

Q2.3

An AR(2) or ARMA(2,0) process is a time series model having autoregressive and no moving average elements, as described by the equation below.

$$\varphi(B)X_t = X_t - 1.5X_{t-1} + \varphi_2 X_{t-2} = \epsilon_t \tag{8}$$

where $\varphi(B)$ denotes the autoregressive operator provided by

$$\varphi(B) = 1 - \varphi_1 B - \varphi_2 B^2 \tag{9}$$

The backward shift operator (B) moves a time series backwards by one unit of time, while the absence of the moving average component in this model indicates that there are no short-term shocks incorporated into the process. The autoregressive parameter values φ_1 and φ_2 control the strength and direction of the autoregressive dependence. We must first define the values of φ_2 and σ_2 in order to simulate the four iterations of the ARMA(2,0) process. Hence, we'll take into account the following four cases

$$\begin{gathered}\varphi = 0.52\ ,\ \sigma^2 = 0.1^2 \\ \varphi = 0.98\ ,\ \sigma^2 = 0.1^2 \\ \varphi = 0.52\ ,\ \sigma^2 = 5^2 \\ \varphi = 0.98\ ,\ \sigma^2 = 5^2\end{gathered}$$

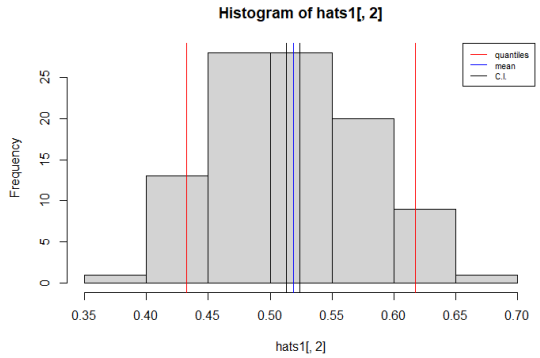
Q2.3.1: Roots of $\phi(z^{-1}) = 0$ for both values of ϕ_2

By changing the equation's order and factoring X_t we get

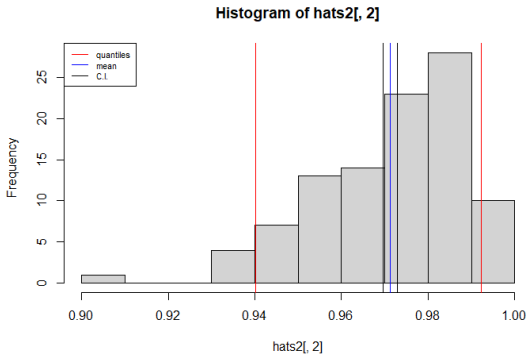
$$\varphi(B)X_t = (1 - 1.5X_{t-1} + \varphi_2 B^2)X_t = \epsilon_t \tag{10}$$

For $\phi_2 = 0.52$ the roots of the equation $\phi(z^{-1}) = 0$ are 0.95 and 0.54, both within the unit circle. For $\phi_2 = 0.98$ the roots of the equation are $0.75 \pm i0.64$, also within the unit circle. Thus the process X_t is stationary for both values of ϕ_2.

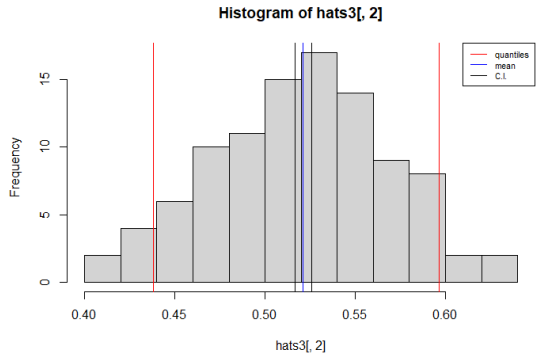
Q2.3.2: Histogram plots of the estimates of parameter ϕ_2



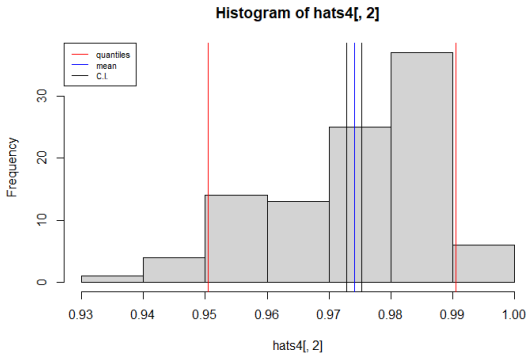
(a) Distribution of ϕ_2 estimations for sd=0.1 and $\phi_2 = 0.52$



(b) Distribution of ϕ_2 estimations for sd=0.1 and $\phi_2 = 0.98$



(c) Distribution of ϕ_2 estimations for sd=5 and $\phi_2 = 0.52$



(d) Distribution of ϕ_2 estimations for sd=5 and $\phi_2 = 0.98$

Q2.3.3: Analysis of the effect of different values of ϕ_2 on the variance/distribution of the estimated ϕ_2

φ_2	Mean	Standard deviation	Variance
$\varphi_2 = 0.52$, $\sigma^2 = 0.1^2$	0.5185	0.05026594	0.002526665
$\varphi_2 = 0.98$, $\sigma^2 = 0.1^2$	0.9713	0.01650415	0.0002723869
$\varphi_2 = 0.52$, $\sigma^2 = 5^2$	0.5212	0.05333535	0.002844660
$\varphi_2 = 0.98$, $\sigma^2 = 5^2$	0.9741	0.01371384	0.0001880694

Table 6: Analysis of the effect of different values of φ_2 and σ on the mean, the standard deviation and the variance of the estimated φ_2

The histograms (a) and (b) demonstrate that the distribution of the estimated φ_2 for $\varphi_2 = 0.52$ appears symmetric, whereas it does not for 0.98. Estimates for smaller values are

centered around the mean value and range between 0.35 and 0.70, whereas estimates for larger values range between 0.90 and 1. This trend is maintained in the histograms (c) and (d). With φ_2 equal to 0.52 and 0.98 respectively, the estimated φ_2 are in the range (0.40, 0.65) and (0.93,1). The fact that the variance of the estimated coefficient decreases as the true value grows further confirms this. The estimations are more clustered closely together for $\varphi_2 = 0.98$ and thus have a smaller deviation from the mean value.

Also, on another note, by increasing the value of the coefficient φ_2, past values have a stronger influence on current values. However, this influence makes it harder to accurately estimate the coefficient.

As it can be seen from Table 6, the mean of the estimated φ_2 values is fairly close to the true value in each case, showing that the LS estimator is effective. The standard deviation of the estimated φ_2 values, fluctuates depending on the value of φ_2 and the variance of the error term. This suggests that when the signal-to-noise ratio is low, there is more fluctuation in the estimations of φ_2.

By performing a t-test, we can additionally observe that the p-value is less than 2.2e-16, which means that it is less than 0.05. This implies that we're able to reject the null hypothesis, which states that there is no difference between the means of the two groups, and come to the conclusion that there is a significant difference between the two groups' means. The 95% confidence interval for the mean difference is (-0.4645131, -0.4409605), and indicates that we are 95% certain that the true difference in means between the two groups falls within this range.

Q2.3.4: Analysis of the effect of different values of σ on the variance/distribution of the estimated ϕ_2

Similarly, increasing the standard deviation of the white noise causes the distribution's support to decrease. However, this reduction is substantially smaller when compared to the corresponding change in distribution support in the previous case. Despite the fact that the quantitative growth of φ_2 (doubled) is significantly smaller than the increase in the standard deviation, the support shrinkage is much smaller. This indicates that the coefficient has bigger influence over the variance of the estimations than the white-noise's standard deviation.

When φ_2 is large (0.98) and the variance of the error term is small ($\sigma^2 = 0.1^2$), the standard deviation of the estimated φ_2 is relatively small (0.01650415). This means that when the signal-to-noise ratio is high, the estimations of φ_2 are more precise.

Q2.3.5: All the estimated pairs of parameters (ϕ_1, ϕ_2) for the four variations

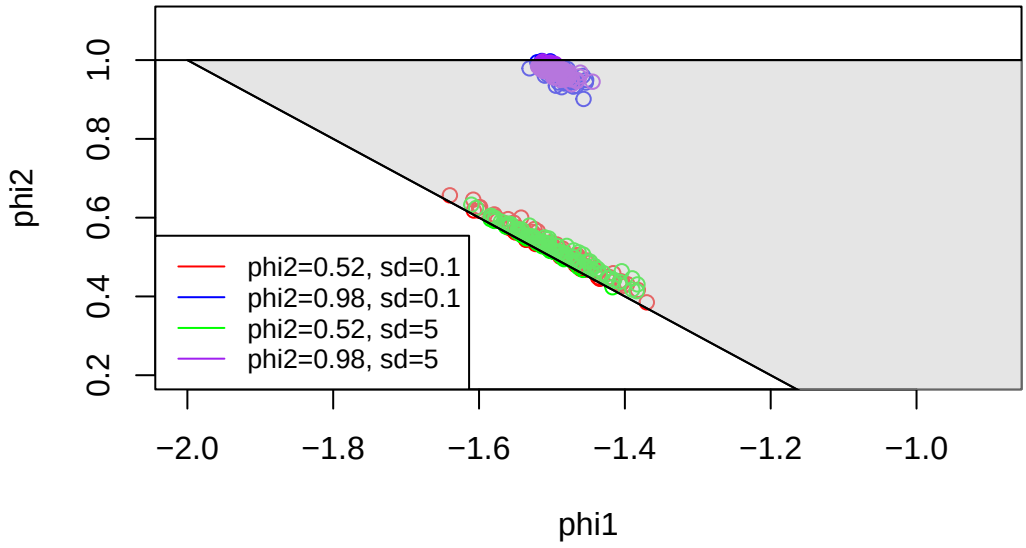


Figure 7: All the estimated pairs of parameters (φ_1, φ_2) for the four variations

Q2.3.6: Stability of the process and discussion

For the process to be stationary, the roots of the following equation must be within the unit circle

$$\varphi(z^{-1}) = z^2 + \varphi_1 z + \varphi_2 = 0$$

In other words

$$|z_1| = \frac{|-\varphi_1 + \sqrt{\varphi_1^2 - 4\varphi_2}|}{2} < 1$$
$$|z_2| = \frac{|-\varphi_1 - \sqrt{\varphi_1^2 - 4\varphi_2}|}{2} < 1$$

or

$$\varphi_2 > -\varphi_1 - 1$$

The process's stability is related to the absolute value of the AR parameter. If the absolute value is bigger than 1, the process is unstable and the time series will burst to infinity over time. If the absolute value is less than one, the process is stable, and the time series will eventually settle into a stationary distribution. As we can see from figure 7, all the estimated pairs of parameters φ_1 and φ_2 for the four variations lie within the region of stationarity that is depicted in grey. Thus, the process is always stationary.

The distribution of the fitted approximated values will be stationary if the AR(2) process is stationary. Specifically, the estimated values will have a normal distribution with a mean and variance determined by the AR(2) model parameters and the error term variance.

Last but not least, because the process is stationary, we can expect more accurate predictions in the future.

Appendix

Question 2.1

```r
rm(list=ls())
set.seed(10)

n <- 200
phi <- c(0.8)
theta <- c(0.8, -0.5)
sd <- 0.4
n.start <- 3

sim <- replicate(10, arima.sim(model = list(ar = phi, ma = theta), n,
        innov = rnorm(n, 0, sd),n.start=3,
        start.innov = rnorm(n.start, 0, sd)))

variances <- apply(sim, 2, var)
mean_var <- mean(variances)
sd_var <- sqrt(mean_var)/sqrt(n)
#the results differ slightly. Reasons: small sample size,
#make a table for different n


matplot(sim, lty=1, type="l", col=rainbow(11), ylim=c(-5,5),
        xlab="time", ylab="Xt")
grid()

lag <- 30
acf.est <- matrix(NA, nrow = 10, ncol = lag+1)
pacf.est <- matrix(NA, nrow = 10, ncol = lag)

for (i in 1:10){
  acf.est[i,] <- acf(sim[,i], lag.max = lag, plot = FALSE)$acf
  pacf.est[i,] <- pacf(sim[,i], lag.max = lag, plot = FALSE)$acf
}

# Plot ACF estimates (the  matrix acf.test is
#transposed in order to plot over
#lag and not realizations)
matplot(t(acf.est), lty=1, type="l", col=rainbow(10),
        ylim=c(-1,1),
        xlab="Lag", ylab="ACF")
abline(h=0)
```

```
abline(h=qnorm(.95)/sqrt(n), col = "blue", lty = 2, lwd = 1)
abline(h=-qnorm(.95)/sqrt(n), col = "blue", lty = 2, lwd = 1)
grid()

#Plot PACF estimates (same)
matplot(t(pacf.est), lty=1, type="l", col=rainbow(10), ylim=c(-1,1),
        xlab="Lag", ylab="PACF")
abline(h=0)
abline(h=qnorm(.95)/sqrt(n), col = "blue", lty = 2, lwd = 1)
abline(h=-qnorm(.95)/sqrt(n), col = "blue", lty = 2, lwd = 1)
grid()
#paromoiws

for (i in 1:30){
  time <- seq(1:10)+10*(i-1)
  if (i==1){
    plot(acf.est[,i]~time, type = "h", xlim = c(0,300), ylim = c(-1,1),
         ylab = "ACF", xlab = "lag x10")
  }
  else{
    lines(acf.est[,i]~time, type = "h")
  }
}
abline(h=0)
abline(h=qnorm(.95)/sqrt(n), col = "blue", lty = 2, lwd = 1)
abline(h=-qnorm(.95)/sqrt(n), col = "blue", lty = 2, lwd = 1)
#abline(v=10)
#abline(h=0.8)
#abline(v=20)
#abline(h=0.54)
#abline(v=30)
#abline(h=0.43)
#abline(v=40)
#abline(h=0.35)
#abline(v=50)

for (i in 1:30){
  time <- seq(1:10)+10*(i-1)
  if (i==1){
    plot(pacf.est[,i]~time, type = "h", xlim = c(0,300), ylim = c(-1,1),
         ylab = "PACF", xlab = "lag x10")
  }
  else{
    lines(pacf.est[,i]~time, type = "h")
```

```
  }
}
abline(h=0)
abline(h=qnorm(.95)/sqrt(n), col = "blue", lty = 2, lwd = 1)
abline(h=-qnorm(.95)/sqrt(n), col = "blue", lty = 2, lwd = 1)
```

Question 2.2

```
rm(list=ls())
set.seed(123)
library(forecast)

data <- read.table("A2_sales.txt", header = TRUE, sep = "")

wn_var <- 36963
mean <- 2070

ts_data <- ts(data$Sales, frequency = 4)

fit <- arima(ts_data, order = c(2, 0, 0),
             seasonal = list(order = c(1, 0, 1), period = 4),
       include.mean = TRUE, fixed = c(1.04, -0.2, 0.86, -0.42, mean))
fit$sigma2 <- wn_var

y_hat <- predict(fit, n.ahead = 2, newxreg = NULL, se.fit = TRUE)
forecasts <- forecast(fit, h=2, level = c(0.95))

for (i in 1:20){
  data$Quarter[i] <- i
}

data <- rbind(data, c(21, y_hat$pred[1]), c(22, y_hat$pred[2]))
data$Quarter <- as.numeric(data$Quarter)

data1 <- data[data$Quarter<21,]
data2 <- data[data$Quarter>20,]

plot(data$Quarter, data$Sales, type = "b",
     xlab = "Quarters", ylab = "Sales")
lines(data2$Quarter, data2$Sales, type = "b", col = "red")
polygon(c(data2$Quarter,rev(data2$Quarter)),
         c(forecasts$upper,rev(forecasts$lower)))
legend("topleft", legend = c("Data", "Predictions"),
       col = c("black", "red"), lty = 1, cex = 0.8)
```

```
grid()
```

Question 2.3

```
rm(list=ls())
set.seed(123)

n <- 300
phi1 <- c(1.5, -0.52)
phi2 <- c(1.5, -0.98)
sd1 <- 0.1
sd2 <- 5
n.start <- 3

# f2=0.52, sd=0.1
sim1 <- replicate(100, arima.sim(model = list(ar = phi1), n,
    innov = rnorm(n, 0, sd1), n.start=3,
    start.innov = rnorm(n.start, 0, sd1)))

# f2=0.98, sd=0.1
sim2 <- replicate(100, arima.sim(model = list(ar = phi2), n,
    innov = rnorm(n, 0, sd1), n.start=3,
    start.innov = rnorm(n.start, 0, sd1)))

# f2=0.52, sd=5
sim3 <- replicate(100, arima.sim(model = list(ar = phi1), n,
    innov = rnorm(n, 0, sd2), n.start=3,
    start.innov = rnorm(n.start, 0, sd2)))

# f2=0.98, sd=5
sim4 <- replicate(100, arima.sim(model = list(ar = phi2), n,
    innov = rnorm(n, 0, sd2), n.start=3,
    start.innov = rnorm(n.start, 0, sd2)))

hats1 <- matrix(NA, nrow = 100, ncol = 2)# f2=0.52, sd=0.1
hats2 <- matrix(NA, nrow = 100, ncol = 2)# f2=0.98, sd=0.1
hats3 <- matrix(NA, nrow = 100, ncol = 2)# f2=0.52, sd=5
hats4 <- matrix(NA, nrow = 100, ncol = 2)# f2=0.98, sd=5

for (i in 1:100){
  x1 <- cbind(-sim1[-c(1,n),i], -sim1[-c(n-1,n),i])
  y1 <- sim1[-c(1,2),i]
  hats1[i,] <- solve(t(x1) %*% x1) %*% t(x1) %*% y1
  x2 <- cbind(-sim2[-c(1,n),i], -sim2[-c(n-1,n),i])
```

```r
  y2 <- sim2[-c(1,2),i]
  hats2[i,] <- solve(t(x2) %*% x2) %*% t(x2) %*% y2
  x3 <- cbind(-sim3[-c(1,n),i], -sim3[-c(n-1,n),i])
  y3 <- sim3[-c(1,2),i]
  hats3[i,] <- solve(t(x3) %*% x3) %*% t(x3) %*% y3
  x4 <- cbind(-sim4[-c(1,n),i], -sim4[-c(n-1,n),i])
  y4 <- sim4[-c(1,2),i]
  hats4[i,] <- solve(t(x4) %*% x4) %*% t(x4) %*% y4
}
hist(hats1[,2])
abline(v = mean(hats1[,2]), col="blue")
lq1 <- quantile(hats1[,2], 0.05)
uq1 <- quantile(hats1[,2], 0.95)
abline(v = lq1, col = "red")
abline(v = uq1, col = "red")
lb1 <-  mean(hats1[,2])-qnorm(.95)*sqrt(var(hats1[,2]))/sqrt(300)
ub1 <-  mean(hats1[,2])+qnorm(.95)*sqrt(var(hats1[,2]))/sqrt(300)
abline(v = lb1)
abline(v = ub1)
legend("topright", legend=c("quantiles","mean","C.I."),
       col = c("red", "blue", "black"), lty=1, cex=0.7)

hist(hats2[,2])
abline(v = mean(hats2[,2]), col="blue")
lq2 <- quantile(hats2[,2], 0.05)
uq2 <- quantile(hats2[,2], 0.95)
abline(v = lq2, col = "red")
abline(v = uq2, col = "red")
lb2 <-  mean(hats2[,2])-qnorm(.95)*sqrt(var(hats2[,2]))/sqrt(300)
ub2 <-  mean(hats2[,2])+qnorm(.95)*sqrt(var(hats2[,2]))/sqrt(300)
abline(v = lb2)
abline(v = ub2)
legend("topleft", legend=c("quantiles","mean","C.I."),
       col = c("red", "blue", "black"), lty=1, cex=0.7)

hist(hats3[,2])
abline(v = mean(hats3[,2]), col="blue")
lq3 <- quantile(hats3[,2], 0.05)
uq3 <- quantile(hats3[,2], 0.95)
abline(v = lq3, col = "red")
abline(v = uq3, col = "red")
lb3 <-  mean(hats3[,2])-qnorm(.95)*sqrt(var(hats3[,2]))/sqrt(300)
ub3 <-  mean(hats3[,2])+qnorm(.95)*sqrt(var(hats3[,2]))/sqrt(300)
abline(v = lb3)
```

```
abline(v = ub3)
legend("topright", legend=c("quantiles","mean","C.I."),
       col = c("red", "blue", "black"), lty=1, cex=0.7)

hist(hats4[,2])
abline(v = mean(hats4[,2]), col="blue")
lq4 <- quantile(hats4[,2], 0.05)
uq4 <- quantile(hats4[,2], 0.95)
abline(v = lq4, col = "red")
abline(v = uq4, col = "red")
lb4 <-  mean(hats4[,2])-qnorm(.95)*sqrt(var(hats4[,2]))/sqrt(300)
ub4 <-  mean(hats4[,2])+qnorm(.95)*sqrt(var(hats4[,2]))/sqrt(300)
abline(v = lb4)
abline(v = ub4)
legend("topleft", legend=c("quantiles","mean","C.I."),
       col = c("red", "blue", "black"), lty=1, cex=0.7)

var1 <- sd1^2* solve(t(x1) %*% x1)# f2=0.52, sd=0.1
var2 <- sd1^2* solve(t(x2) %*% x2)# f2=0.98, sd=0.1
var3 <- sd2^2* solve(t(x3) %*% x3)# f2=0.52, sd=5
var4 <- sd2^2* solve(t(x4) %*% x4)# f2=0.98, sd=5

var_1 <- var(hats1[,2])
var_2 <- var(hats2[,2])
var_3 <- var(hats3[,2])
var_4 <- var(hats4[,2])

####t-test for 2.3.3
t.test(hats1[,2], hats2[,2])

#################
# Set up the plot
plot(hats1[,2]~hats1[,1], col="red",xlab="phi1", ylab="phi2",
     ylim=c(0.2, 1.1), xlim=c(-2,-0.9))#phi2=.52 k sd=.1
points(hats2[,2]~hats2[,1], col="blue")#phi2=.98 k sd=.1
points(hats3[,2]~hats3[,1], col="green")#phi2=.52 k sd=5
points(hats4[,2]~hats4[,1], col="purple")#phi2=.98 k sd=5

# Draw the lines and fill the area inside them
x <- seq(-2,2,length=100)
y1 <- -x-1
y2 <- rep(1, length(x))
# define transparent gray color
transparent_gray <- rgb(0.8, 0.8, 0.8, alpha=0.5)
```

```
polygon(c(x, rev(x)), c(y1, rev(y2)), col = transparent_gray)

# Draw the lines again on top of the filled area
lines(x, y1)
abline(h=1)

# Add the legend
legend("bottomleft", legend =
         c("phi2=0.52, sd=0.1","phi2=0.98, sd=0.1", "phi2=0.52,
           sd=5", "phi2=0.98, sd=5")
       , col = c("red", "blue", "green", "purple"),lty=1,cex=0.8)
```