KATHOLIEKE UNIVERSITEIT LEUVEN

MASTER OF STATISTICS AND DATA SCIENCE

CAPITA SELECTA IN STATISTICS [G0W36A]

Exam Report

Student:
Alexander Saines r0731904

Professor:
Prof. Kurt Barbé

June 19, 2024

Contents

1 Introduction

Time series analysis is a powerful statistical tool used for understanding and forecasting data points collected over time. The Seasonal ARIMA (AutoRegressive Integrated Moving Average) model is a widely-used technique for time series forecasting, capturing essential patterns such as trends and seasonality. This report aims to analyze various ARIMA models, both seasonal and non-seasonal, and assess their performance in forecasting time series data.

2 Objectives

The primary objectives of this assignment are:

- To simulate time series data using the `arima.sim.freq2` function.
- To fit ARIMA models using the `auto.arima()` procedure and manually specified structures.
- To forecast future data points and compare the accuracy of different models.
- To understand the implications of model parameters, poles, and zeros on the stationarity and forecast accuracy.

3 Methodology

3.1 Simulation of Data

The data simulation involves generating a time series using the `arima.sim.freq2` function. This function creates data based on specified ARIMA and seasonal components, ensuring the presence of trends and periodicity.

3.2 Model Fitting Procedures

We employ two primary methods for model fitting:

- `auto.arima()`: Automatically selects the best ARIMA model based on AIC values.
- Manually specified ARIMA and SARIMA models: These models are constructed based on the known poles and zeros of the simulated data.

3.3 Forecasting Methods

The forecasting methods include:

- Built-in `forecast` function: Provides predictions based on the fitted ARIMA model.
- `predictTS` function: Utilized to forecast and visualize future data points.

4 Model Fitting and Analysis

4.1 Exercise 1: Data Simulation using `arima.sim.freq2`

4.1.1 Simulation Parameters

In this part of the exercise, a time series is simulated by using the `arima.sim.freq2` function. The goal is to generate data that has both specific autoregressive and seasonal characteristics. Here are the parameters used for this simulation:

- **Poles:**
  - $0.9 \cdot \exp(\pm 2\pi i \cdot 0.125)$: This translates to a frequency of 0.5 cycles per day.
  - $0.8 \cdot \exp(\pm 2\pi i \cdot 0.25)$: This translates to a frequency of 1 cycle per day.
- **Zeros:**
  - $0.7 \cdot \exp(\pm 2\pi i \cdot 0.1875)$: This corresponds to a frequency of about 0.75 cycles per day.
- **Differentiator:**
  - We'll use $D = 1$ to introduce a trend into the data.
- **Seasonal Component:**
  - A pole at 0.95.
  - A zero-pair at $0.7 \cdot \exp(\pm 2\pi i \cdot 0.1)$.
  - A seasonal period $S = 6$ samples.
- **Standard Deviation:**
  - The standard deviation of the innovations (random shocks) will be set to 2.
- **Sampling Frequency:**
  - The data is sampled 4 times per day.
- **Time Unit:**
  - The unit of time for this simulation will be "daily".

Here, the parameter $S = 6$ uses the same sampling rate as our data, $fs = 4$. This applies a seasonal component with a period length of 6 samples. Given that the sampling rate is 4 samples per day, this equates to 6 hours per sample, resulting in a periodicity of 36 hours or 1.5 days.

It is important to be mindful of the dimensions in this context. When using a parameter like $0.9 \cdot \exp(\pm 2\pi i \cdot 0.125)$, the location parameter 0.125 is dimensionless. In theoretical terms:

$$z = \exp(2\pi i \frac{f}{Fs})$$

Where f is the frequency related to the pole and Fs is the sampling frequency. Both are in units of time^{-1}. This can be translated to the physical pole in days^{-1}, which results in the frequency of 1, 0.5, and 0.75 cycles per day for Poles and Zeros respectively.

4.1.2 Generated Plots and Analysis

The `arima.sim.freq2` function creates the following figures to help understand its characteristics:

- **Time Series Plot:** The graph 1 provides visualization of the simulated data over time, capturing the combined effects of the autoregressive (AR), moving average (MA), and seasonal components of the model.

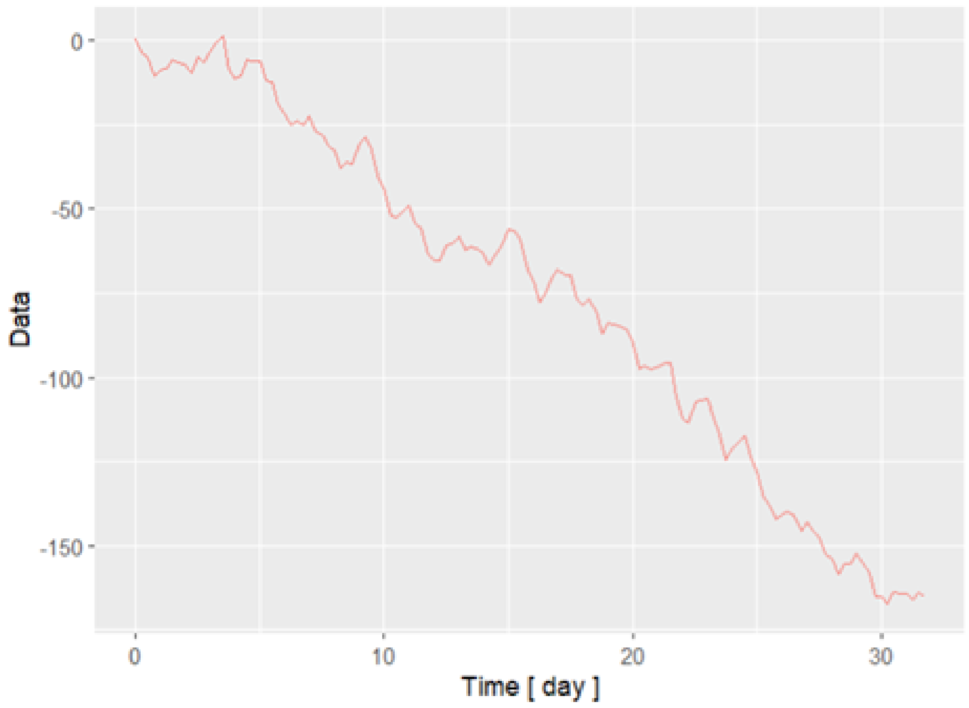


Figure 1: Simulated Time Series

The time series exhibits a clear downward trend, indicating that the data values generally decrease over time. Superimposed on this overall trend, there are periodic fluctuations, reflecting the seasonal and cyclical components of the model. Additionally, the time series shows variability and noise around the trend, which is typical for real-world data.

- **Power Spectral Density (PSD) Plot:** This plot will help us see how power is distributed across different frequencies, identifying the dominant frequencies in data. The spectral density is in the physical unit days^{-1}.

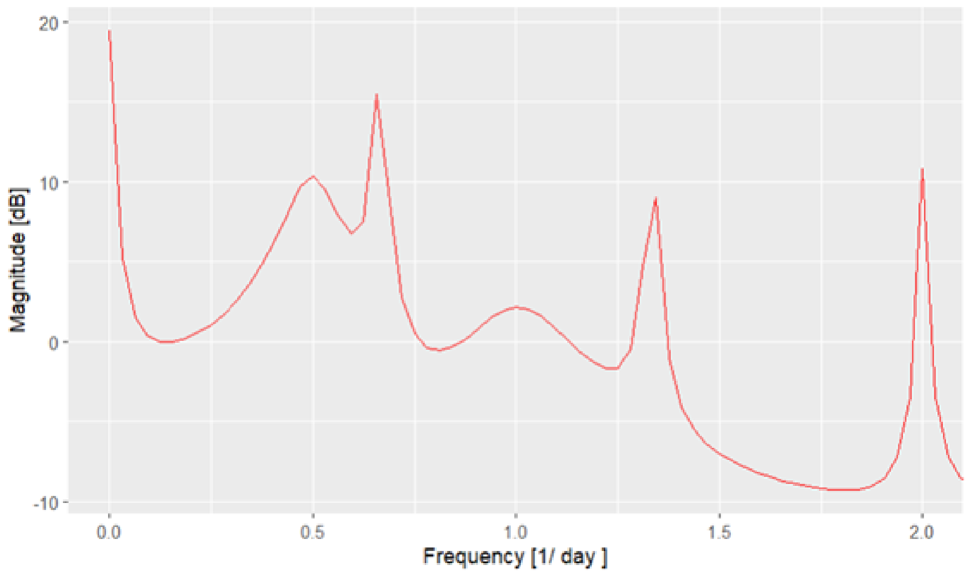


Figure 2: Power Spectral Density

The peaks at 0.5 cycles per day and 1 cycle per day represent the autoregressive (AR) poles of the model. These frequencies arise from the non-seasonal AR components (from the poles). These poles translate into periodic components.
The presence of harmonics at 0.67 cycles/day, 1.33 cycles/day, and 2.00 cycles/day can be explained by the nature of the seasonal component. Harmonics are integer multiples of the base frequency of the seasonal component. Thus:

- The 1st harmonic appears at the base frequency: 0.67 cycles per day.
- The 2nd harmonic appears at twice the base frequency: 1.33 cycles per day $(2 \cdot 0.67$ cycles/day$)$.
- The 3rd harmonic appears at three times the base frequency: 2.00 cycles per day $(3 \cdot 0.67$ cycles/day$)$.

These harmonics reflect the repeating patterns introduced by the seasonal component with a periodicity of 1.5 days.

- **Impulse Response Plot:** This plot will show how the system responds to a sudden impulse, giving us insight into the model's dynamics.

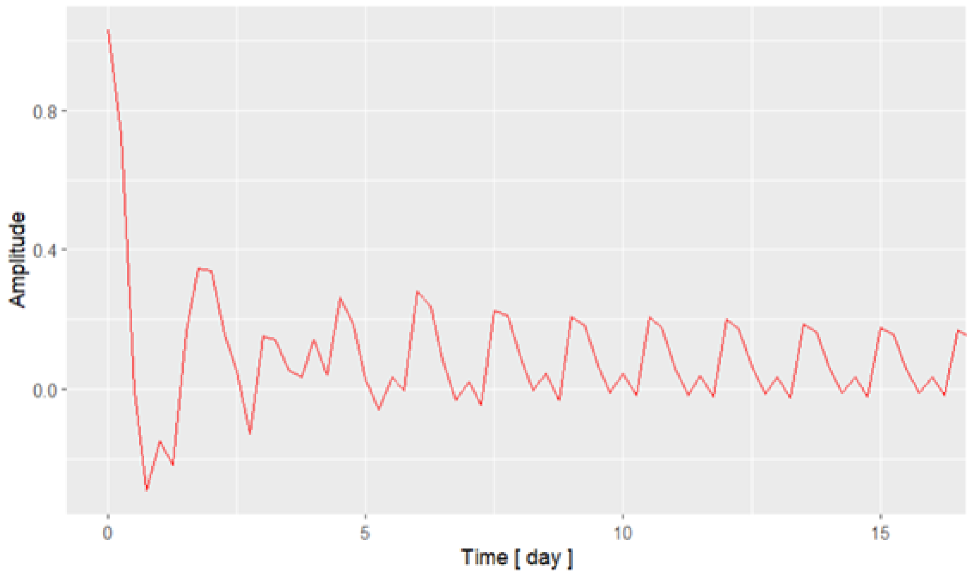


Figure 3: Impulse Response

- **Pole-Zero Configuration Plot:** This plot will show the locations of the poles and zeros in the complex plane, helping us understand the stability and behavior of the model. All these points are inside the unit circle which is accurate as the magnitude of all poles and zeros are lower than 1.

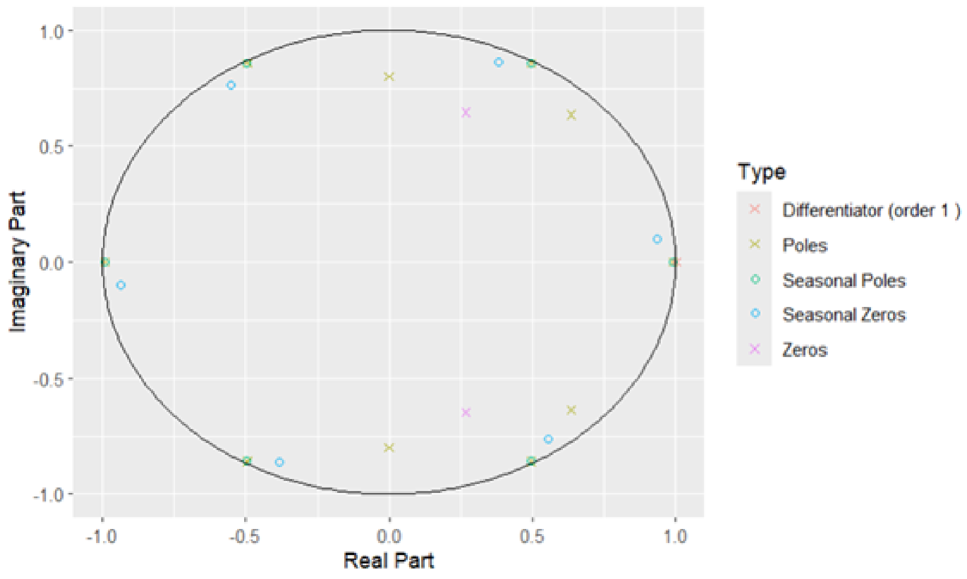


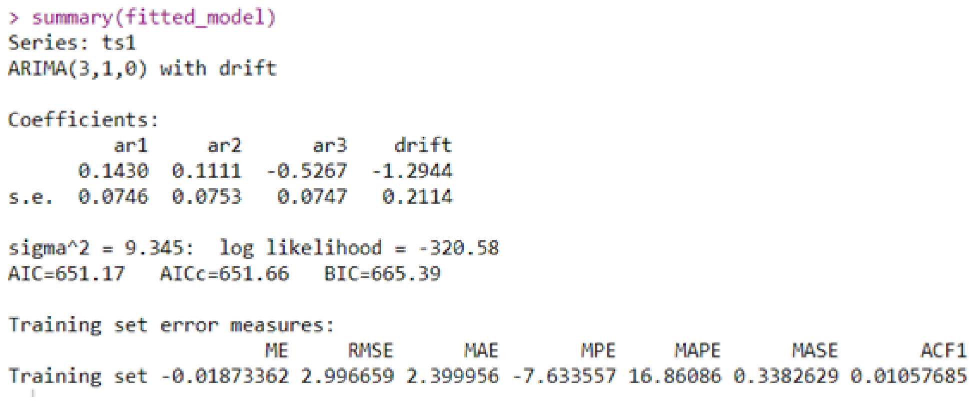
Figure 4: Pole - Zero Configuration

4.2 Exercise 2: Auto ARIMA Model

The `auto.arima()` function from the `forecast` package in R is used to model the simulated data from the previous section. The `auto.arima()` function is designed to automatically identify the best ARIMA model by selecting the optimal order parameters (p, d, q) using the Akaike Information

Criterion (AIC). The function fits various ARIMA models, comparing their performance based on AIC values, and ultimately selects the model that minimizes the AIC. The chosen model is then used to fit the data and generate forecasts [1].

For the simulated time series, the `auto.arima` function identified an ARIMA model with three autoregressive (AR) components and a drift term. However, it fails to capture the true model configuration, which includes seasonal and moving average (MA) components.



```
> summary(fitted_model)
Series: ts1
ARIMA(3,1,0) with drift

Coefficients:
         ar1     ar2      ar3    drift
      0.1430  0.1111  -0.5267  -1.2944
s.e.  0.0746  0.0753   0.0747   0.2114

sigma^2 = 9.345:  log likelihood = -320.58
AIC=651.17   AICc=651.66   BIC=665.39

Training set error measures:
                     ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
Training set -0.01873362 2.996659 2.399956 -7.633557 16.86086 0.3382629 0.01057685
```

Figure 5: Auto Arima Results

Regarding its coefficients, they are significant as they portray a pretty low standard error values.

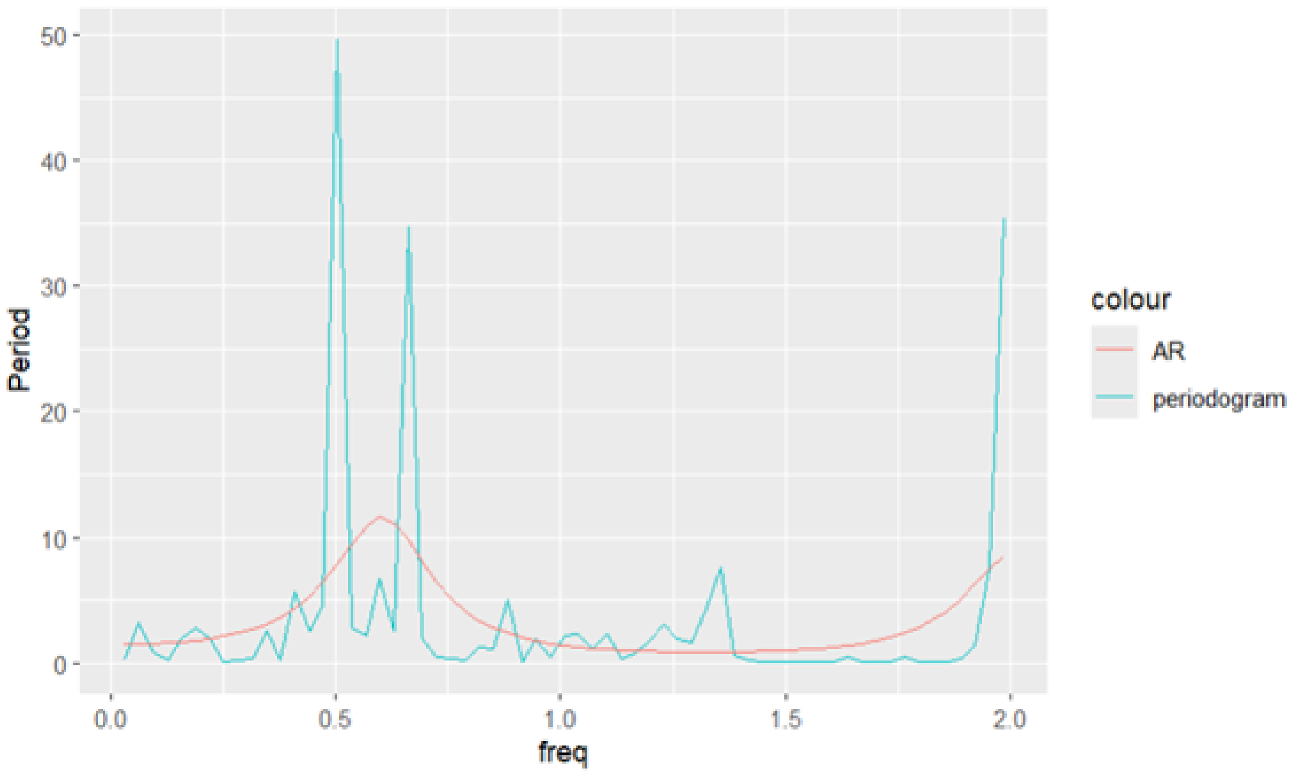


Figure 6: Spectral Density Estimation - AR order 3

The spectral density estimation graph, depicted in 6, illustrates the differences between the periodogram and AR methods for estimating the spectral density of the time series. The periodogram provides a more direct, albeit noisier, estimate of the spectral density, revealing several peaks corresponding to the dominant frequencies in the data. In contrast, the AR method offers a smoother and potentially more accurate representation of the underlying periodic components. The AR model used an order of 3, as suggested by the `auto.arima` function. However, it fails to capture some of the peaks shown in the periodogram, which are related to the frequencies of the ground truth poles and zeros explained in figure 2.

	Roots_fp	**Radius**	**Angle**
1	$0.4756210 - 6.587861 \times 10^{-1}i$	0.8125358	-0.6019112
2	$-0.8013519 - 6.941146 \times 10^{-17}i$	0.8013519	-2.0000000
3	$0.4756210 + 6.587861 \times 10^{-1}i$	0.8125358	0.6019112

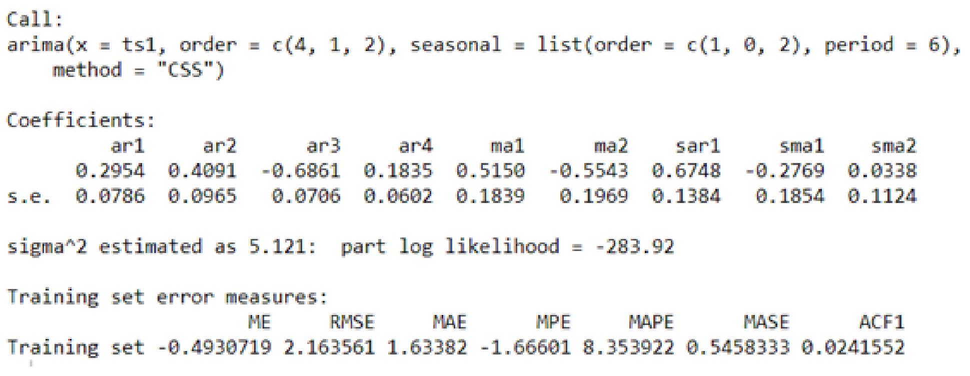
Table 1: Roots, Radius, and Angle of the ARIMA Model

When computing the angles from the roots obtained from the `auto.arima` model, they do not closely match the angles corresponding to the original poles and zeros used to simulate the time series (i.e., 0.5, 0.75, and 1 cycles per day). This discrepancy highlights the difference between the model fitted by `auto.arima` and the true underlying model used to generate the data.

4.3 Exercise 3: Manually Specified ARIMA Model

4.3.1 Model Specification and Fitting

The correct model structure includes 4 roots for the autoregressive component and 2 roots for the moving average component. Additionally, there is a drift term that accounts for the trend component. The seasonal component comprises one autoregressive root and two moving average roots, with a seasonal period length of 6 samples. The results of the model are presented below.



```
Call:
arima(x = ts1, order = c(4, 1, 2), seasonal = list(order = c(1, 0, 2), period = 6),
    method = "CSS")

Coefficients:
         ar1     ar2      ar3     ar4     ma1      ma2    sar1     sma1    sma2
      0.2954  0.4091  -0.6861  0.1835  0.5150  -0.5543  0.6748  -0.2769  0.0338
s.e.  0.0786  0.0965   0.0706  0.0602  0.1839   0.1969  0.1384   0.1854  0.1124

sigma^2 estimated as 5.121:  part log likelihood = -283.92

Training set error measures:
                     ME     RMSE     MAE      MPE     MAPE      MASE       ACF1
Training set -0.4930719 2.163561 1.63382 -1.66601 8.353922 0.5458333 0.0241552
```

Figure 7: Arima Results - correct model structure

Regarding its poles portrayed in table 2 and 3, this ARIMA model is able to accurately detect the true pole at 0.5. However, the other pole, with an angle of 2, is somewhat close to the true pole at 1. In the case of the zero proposed by this model, its angle is at zero, whereas the true angle is 0.75.

	Roots_fp	Radius	Angle
1	$0.4797486 - 5.720068 \times 10^{-1}i$	0.7465591	-0.5557006
2	$-0.9950315 + 1.769227 \times 10^{-14}i$	0.9950315	2.000000
3	$0.4797486 + 5.720068 \times 10^{-1}i$	0.7465591	0.5557006
4	$0.3309112 - 1.507086 \times 10^{-15}i$	0.3309112	-2.899389 $\times 10^{-15}$

Table 2: AR Roots, Radius, and Angle of the ARIMA Model

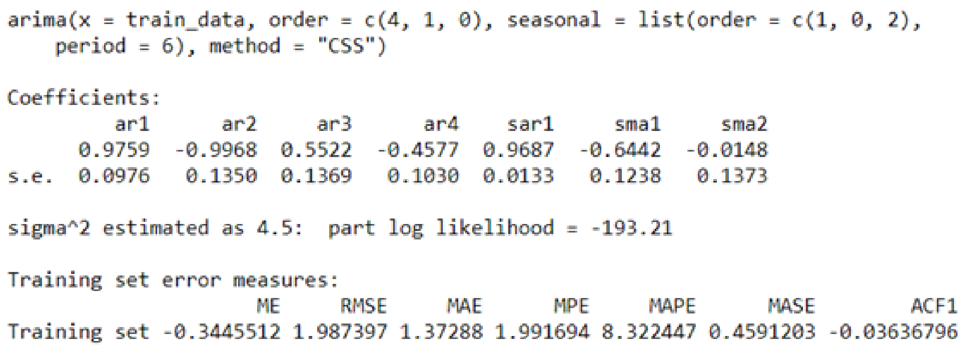
	Roots_fp	Radius	Angle
1	$-1.0452429 + 0i$	1.0452429	-2
2	$0.5302896 + 0i$	0.5302896	0

Table 3: MA Roots, Radius, and Angle of the ARIMA Model

5 Forecasting and Validation

5.1 Exercise 4: Forecasting with the First 90 Samples

In this section, a model with four AR coefficients, a drift term, and no MA coefficients was considered, while maintaining the same seasonal components as the previous model. It was not possible to run a model with the same structure proposed in the previous section, as it led to non-stationarity problems, as shown in Appendix 16.



```
arima(x = train_data, order = c(4, 1, 0), seasonal = list(order = c(1, 0, 2),
    period = 6), method = "CSS")

Coefficients:
         ar1      ar2     ar3      ar4    sar1     sma1     sma2
      0.9759  -0.9968  0.5522  -0.4577  0.9687  -0.6442  -0.0148
s.e.  0.0976   0.1350  0.1369   0.1030  0.0133   0.1238   0.1373

sigma^2 estimated as 4.5:  part log likelihood = -193.21

Training set error measures:
                     ME     RMSE     MAE      MPE     MAPE      MASE        ACF1
Training set -0.3445512 1.987397 1.37288 1.991694 8.322447 0.4591203 -0.03636796
```

Figure 8: Arima model with the first 90 Samples

This model shows a high standard error for the AR coefficients 2, 3, and 4. However, when examining the inverse characteristic roots plot below, all the inverse roots (poles and zeros) remain inside the unit circle, indicating that the process is stationary.

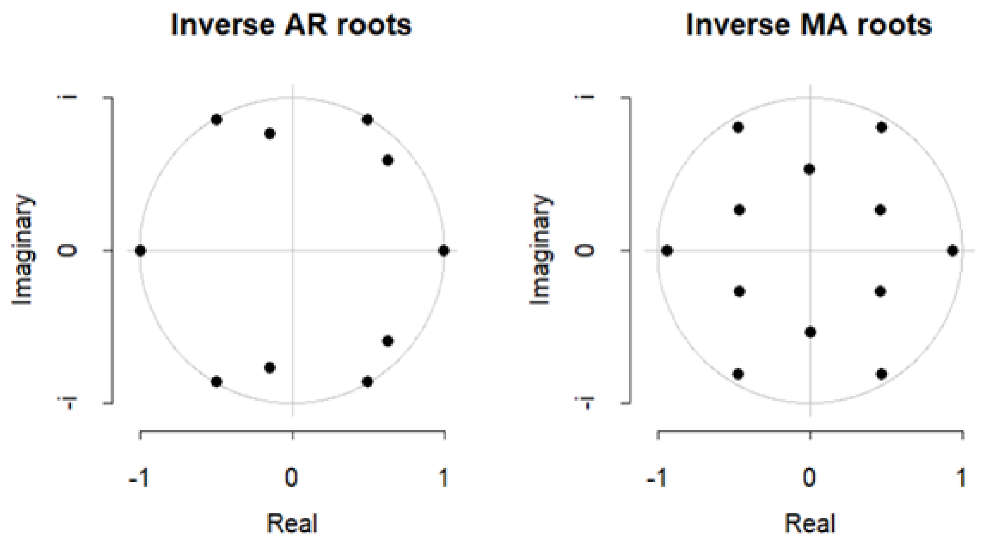


Figure 9: Inverse characteristic Roots for model in figure 8

Once the model is validated, the 38 remaining samples were forecasted by using both the built-in forecast in R and the predictTS procedures. Both procedure provides similar results as depicted in figure 10 and 11.

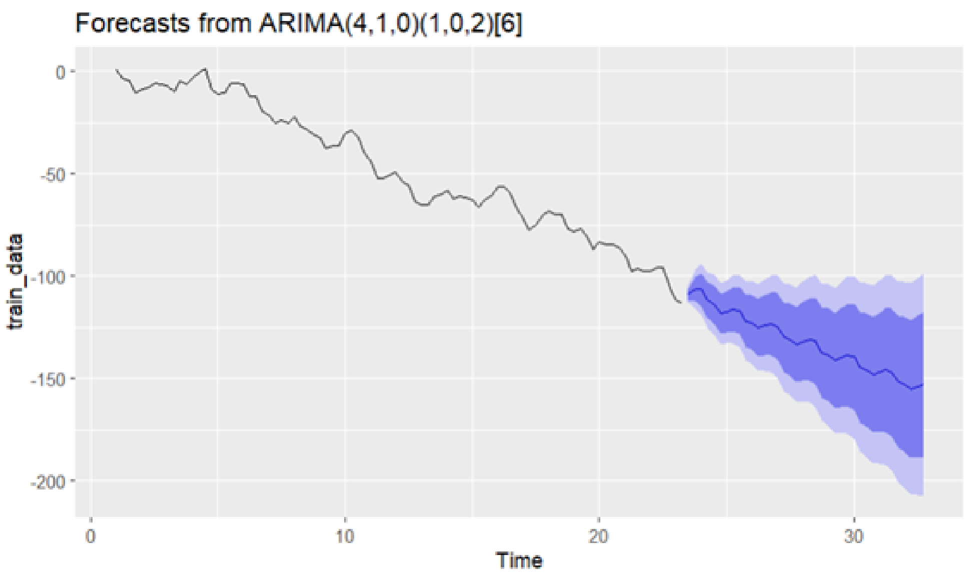


Figure 10: Forecasting using the built-in forecast in R

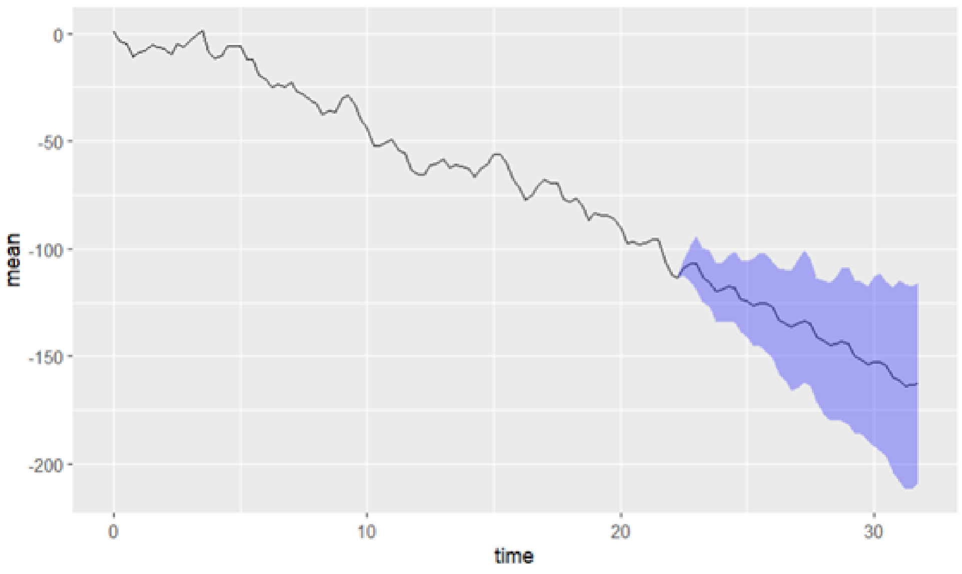
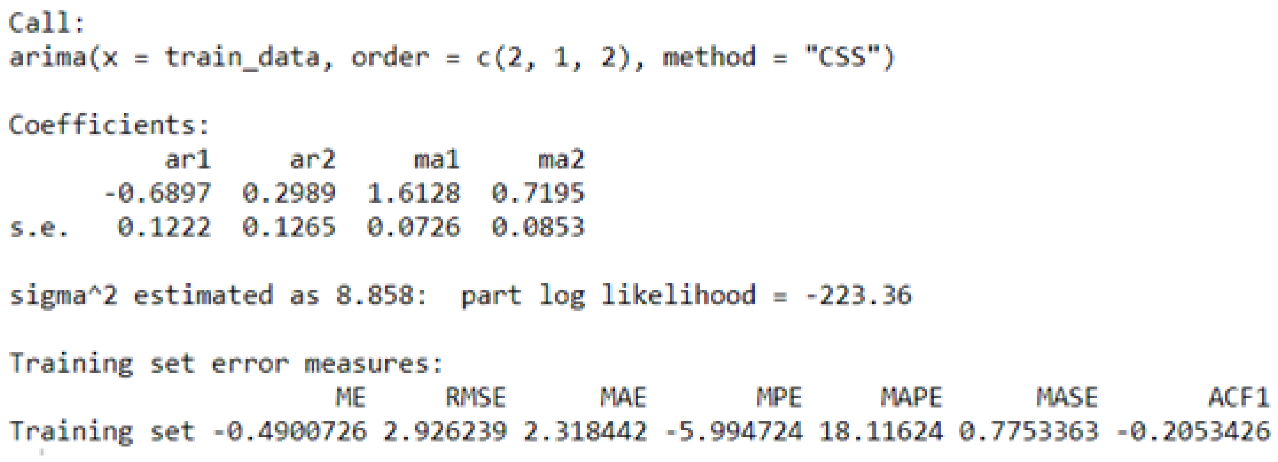


Figure 11: Forecasting using the the predictTS procedures

5.2 Exercise 5: Non-Seasonal ARIMA Model

In this section, a non-seasonal ARIMA model was implemented. The selected model included two AR components, two MA components, and a drift component, while omitting the seasonal component. The aim was to closely match the ground truth poles and zeros for the AR and MA components. However, this approach resulted in non-stationary issues, as shown in Figure 18 in the appendix. Additionally, higher-order models were considered, as illustrated in the spectrogram in Figure 17. Nonetheless, using a higher number of AR components instead of MA components can lead to overfitting or result in a model that lacks parsimony.



```
Call:
arima(x = train_data, order = c(2, 1, 2), method = "CSS")

Coefficients:
          ar1     ar2     ma1     ma2
      -0.6897  0.2989  1.6128  0.7195
s.e.   0.1222  0.1265  0.0726  0.0853

sigma^2 estimated as 8.858:  part log likelihood = -223.36

Training set error measures:
                     ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
Training set -0.4900726 2.926239 2.318442 -5.994724 18.11624 0.7753363 -0.2053426
```

Figure 12: non-seasonal Arima model

The model depicted in Figure 12 exhibits high standard errors. However, the inverse roots of these components lie inside the unit circle, as shown in Figure 13, indicating that the model has stationary properties.

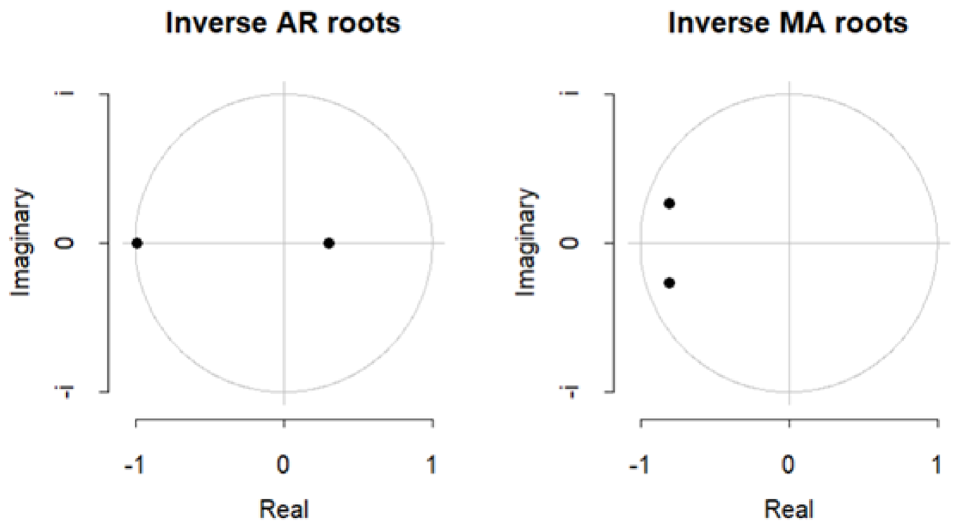


Figure 13: Inverse characteristic Roots

The angles of the poles and zeros computed for the proposed non-seasonal ARIMA model are significantly different from the original angles of the ground truth poles and zeros. Table 4 presents these computed poles and zeros.

	Roots_fp	**Radius**	**Angle**
	Poles		
1	$-0.9912468 - 1.828717 \times 10^{-24}i$	0.9912468	-75.000000
2	$0.3015474 + 1.692365 \times 10^{-25}i$	0.3015474	1.339831×10^{-23}
	Zeros		
1	$-0.8063836 - 0.2630633i$	0.848208	-67.4718
2	$-0.8063836 + 0.2630633i$	0.848208	67.4718

Table 4: Poles and Zeros of the ARIMA Model

Once the arima model was chossen, a forecasting procedure was conduncted using the built-in forecast in R and predictTS procedure. The figure 14 and 15 depcits these forecasting respectively.

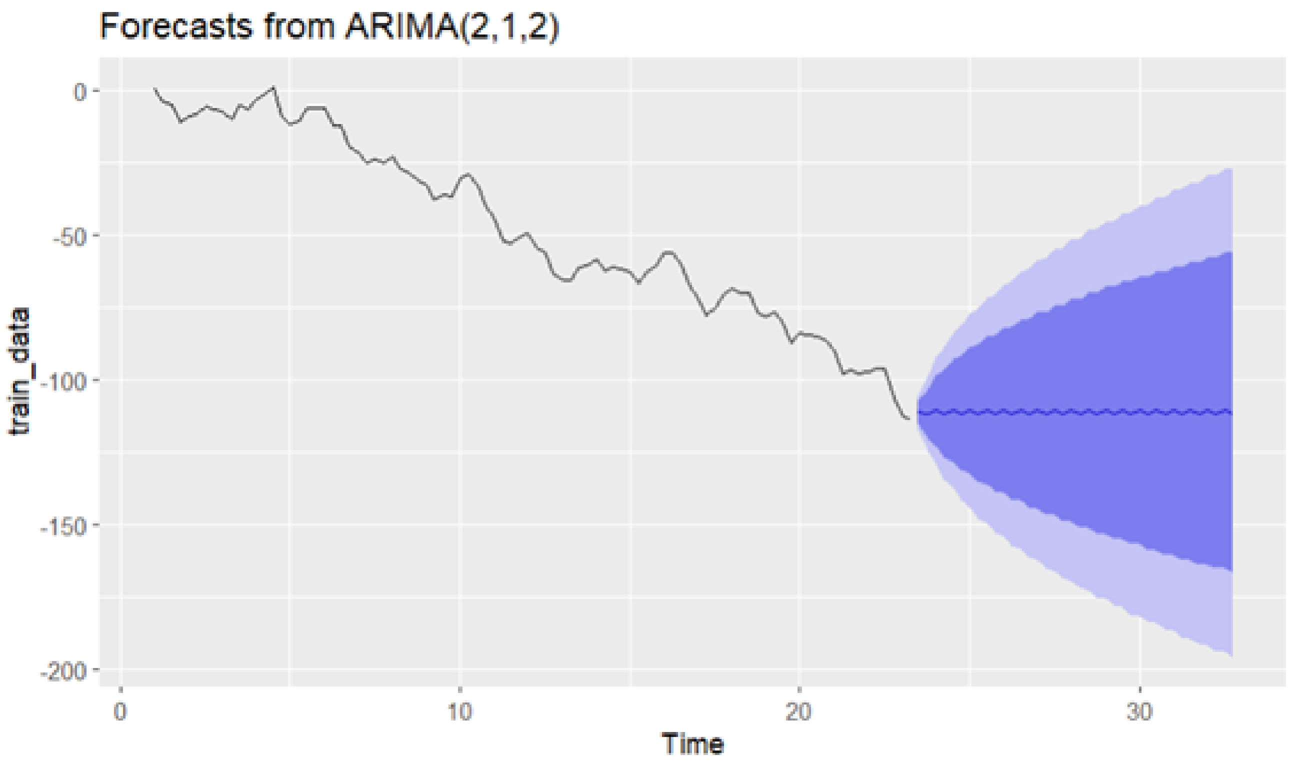


Figure 14: Forecasting using the built-in forecast in R

When using the built-in forecast function, it fails to make accurate predictions, resulting in a flat line. Despite including a drift component in the model, this method does not capture the decreasing trend, making it unconvincing as a continuation of the past data. A possible reason for this is that the built-in procedure does not account for the innovation process, which is essential for future predictions. This issue does not occur with the `predictTS` procedure, which successfully predicts the negative trend.

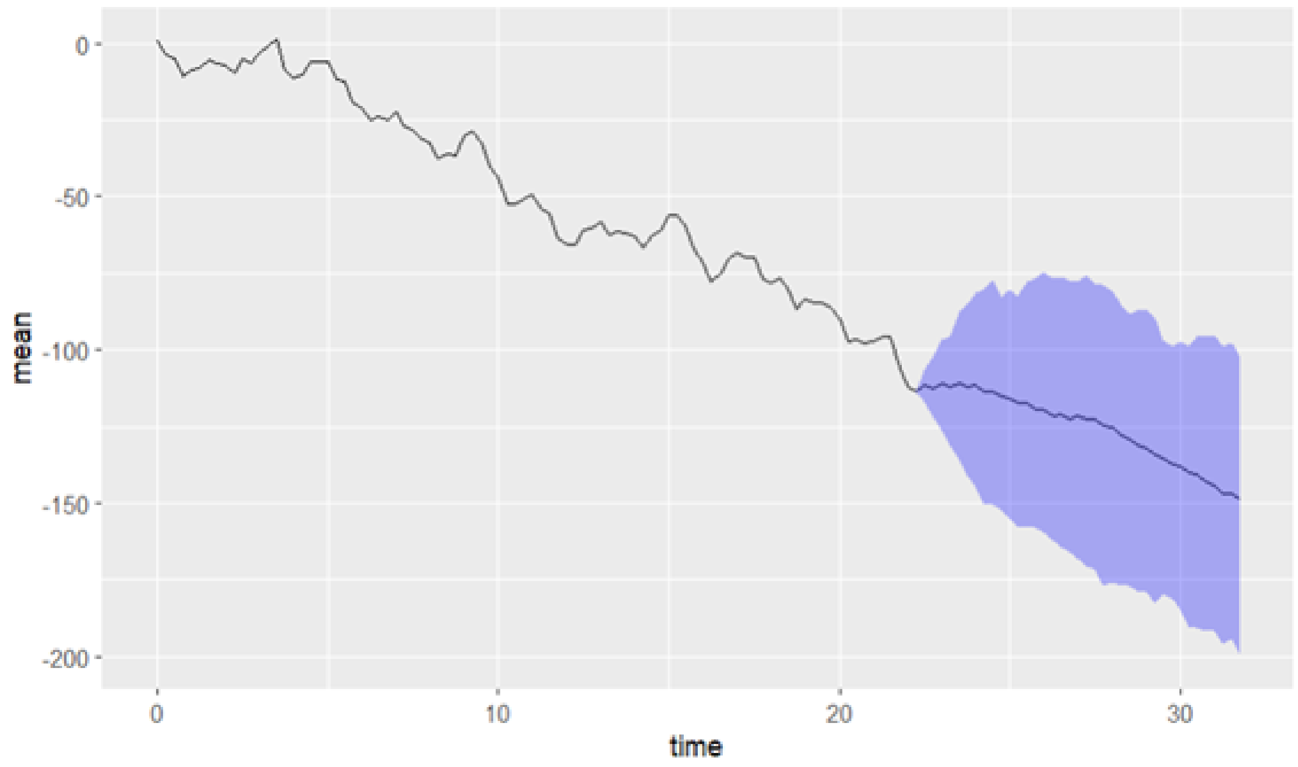


Figure 15: Forecasting using the the predictTS procedures

6 Conclusion

This report aimed to analyze various S-ARIMA models for time series forecasting, using both automatic and manually specified approaches. The main findings are summarized as follows:

- The `auto.arima` function managed to identify an ARIMA model with three AR components and a drift term. However, it failed to detect the true SARIMA process, which includes seasonal and moving average components.
- The manually specified SARIMA model, while configured with the true parameters, failed to accurately capture all the angles of the actual AR and MA components.
- Different orders for the SARIMA and ARIMA models were used in Exercises 4 and 5, respectively, as only the training dataset was considered. This led to different orders compared to when the full dataset (128 observations) was used.
- When a non-seasonal ARIMA model was used to fit the data, the built-in `forecast` function in R failed to provide accurate forecasting, resulting in a flat line and failing to capture the negative trend present in the data. In contrast, the `predictTS` function successfully captured and accurately predicted the negative trends.

References

[1] Rob J. Hyndman and Yeasmin Khandakar. Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software*, 27(3):1–22, 2008.

A Appendices

A.1 Exercise 4

SARIMA2 ¡- arima(train_data, order = c(4, 1, 2), seasonal = list(order = c(1, 0, 2), period = 6), method = ”CSS”)

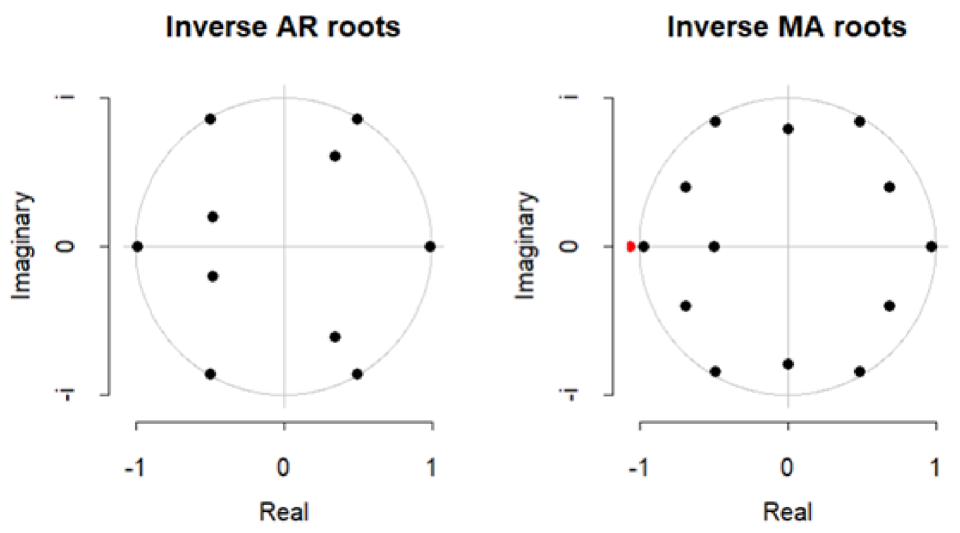


Figure 16: Inverse characteristic Roots

A.2 Exercise 5

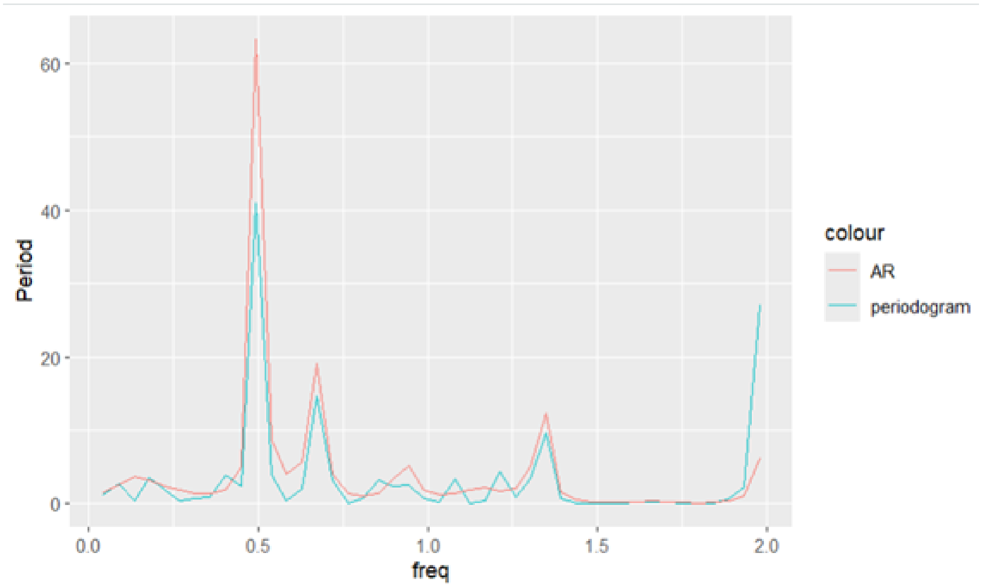


Figure 17: Enter Caption

arima_train ¡- arima(train_data, order = c(4, 1, 2), method = ”CSS”)

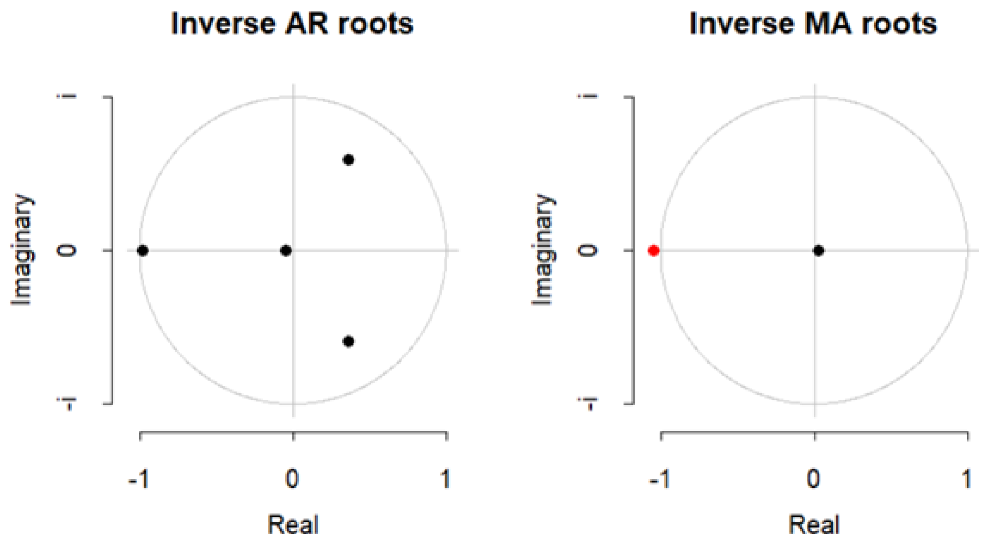


Figure 18: Enter Caption

arima_train2 ¡- arima(train_data, order = c(8, 1, 2), method = ”CSS”)

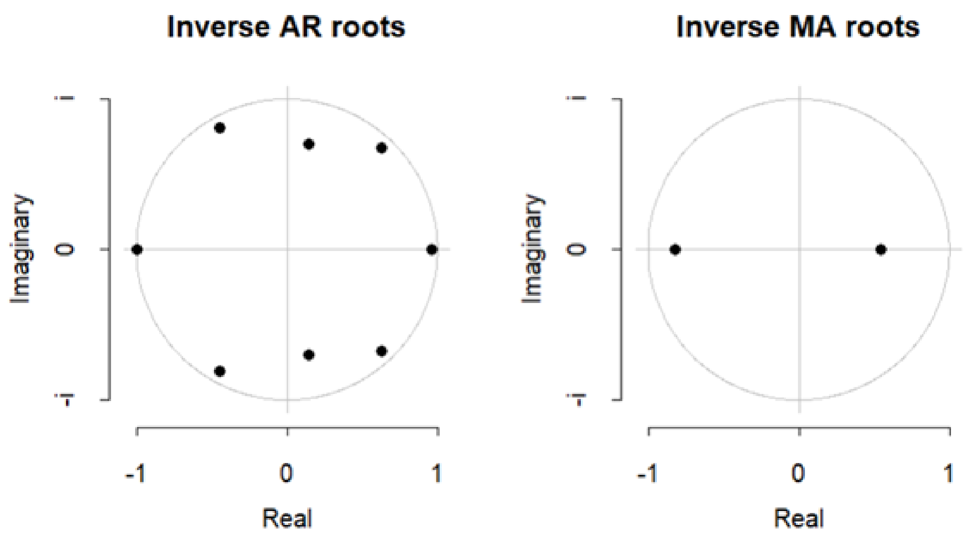


Figure 19: Enter Caption

B R Code for Data Simulation and Analysis

B.1 Exercise 1

```
#ex1
# Given parameters
n <- 128
model <- list(
  Poles = c(0.9 * exp(2 * pi * 1i * 0.125), 0.8 * exp(2 * pi * 1i * 0.25))
     ,
  Zeros = 0.7 * exp(2 * pi * 1i * 0.1875)
)
smodel <- list(
  Poles = 0.95,
  Zeros = 0.7 * exp(2 * pi * 1i * 0.1),
  S = 6
)
D <- 1
std <- 2
fs <- 4
unit <- "day"

# Simulate the data
result <- arima.sim.freq2(n, model, smodel, D, std = std, fs = fs, unit =
   unit)

# Display the plots
#Power Spectral Density (PSD) Plot
```

```
print(result$PSD)

print(result$ImpRes)

print(result$TS)

print(result$Configuration)
```

B.2 Exercise 2

```
# ex 2
library(forecast)
ts1 <- ts(result$data$TimeSeries, frequency = 4)
autoplot(ts1)

fitted_model <- auto.arima(ts1)
summary(fitted_model)

# comparing spectrum
ts1_d1 <- diff(ts1)

# Compute the spectrum using periodogram and autoregressive methods
s_period <- spectrum(ts1_d1, method = "pgram", fast = FALSE, log = "yes")
s_ar <- spectrum(ts1_d1, method = "ar", type = "b", log = "yes", n.freq =
    (length(ts1_d1) / 2) + 1, order = 3)

# Create a data frame with the spectral densities
Spectral <- data.frame(freq = s_period$freq, Period = s_period$spec, AR =
   s_ar$spec[2:((length(ts1_d1) / 2) + 1)])
plot(ggplot(Spectral, aes(x=freq)) +  geom_line(aes(y = Period, colour = "
   periodogram")) + geom_line(aes(y=AR, colour = "AR")))

#compare angle
AR1 <- ar(ts1_d1,aic = FALSE, order = 3, method = c("ols"))
A1 <- c(1,-AR1$ar)
Fs=4
AR_Model1 <-data.frame(Roots_fp = 1/polyroot(A1), Radius = abs(1/polyroot(
   A1)), Angle = Arg(1/polyroot(A1))/2/pi*Fs)
```

B.3 Exercise 3

```
#ex3
# Fit the SARIMA model to the time series
```

```
SARIMA <- arima(ts1, order = c(4, 1, 2), seasonal = list(order = c(1, 0,
   2), period = 6), method = "CSS")
summary(SARIMA)

A2 <- c(1,-SARIMA$coef[1:4])
AR_Model1 <-data.frame(Roots_fp = 1/polyroot(A2), Radius = abs(1/polyroot(
   A2)), Angle = Arg(1/polyroot(A2))/2/pi*Fs)
B2 <- c(1,SARIMA$coef[5:6])
MA_Model1 <-data.frame(Roots_fp = 1/polyroot(B2), Radius = abs(1/polyroot(
   B2)), Angle = Arg(1/polyroot(B2))/2/pi*Fs)
```

B.4 Exercise 4

```
#ex4
n <- 128

train_data <- head(ts1, 90)
test_data <- tail(ts1, 38)

# fit original structure
SARIMA2 <- arima(train_data, order = c(4, 1, 2), seasonal = list(order = c
   (1, 0, 2), period = 6), method = "CSS")
summary(SARIMA2)
par(mfrow=c(1,2))
plot(arroots(SARIMA2),main="Inverse AR roots")
plot(maroots(SARIMA2),main="Inverse MA roots")

# structure without problems in roots
SARIMA2 <- arima(train_data, order = c(4, 1, 0), seasonal = list(order = c
   (1, 0, 2), period = 6), method = "CSS")
summary(SARIMA2)
par(mfrow=c(1,2))
plot(arroots(SARIMA2),main="Inverse AR roots")
plot(maroots(SARIMA2),main="Inverse MA roots")

# Forecast the future 38 samples
forecasted_values <- forecast(SARIMA2, h = 38)

plot(autoplot(forecasted_values))
# check this graph
PredictTS(train_data,SARIMA2,100,38,4,0.05)
```

B.5 Exercise 5

```r
#ex5
# checking spectral part
# prediction
train_data <- head(ts1, 90)
test_data <- tail(ts1, 38)

# comparing spectrum
ts1_d1 <- diff(train_data)

# Compute the spectrum using periodogram and autoregressive methods order
   20 -- Overffiting
s_period <- spectrum(ts1_d1, method = "pgram", fast = FALSE, log = "yes")
s_ar <- spectrum(ts1_d1, method = "ar", type = "b", log = "yes", n.freq =
   (length(ts1_d1) / 2) + 1, order = 20)

# Create a data frame with the spectral densities
Spectral <- data.frame(freq = s_period$freq, Period = s_period$spec, AR =
   s_ar$spec[2:((length(ts1_d1) / 2) + 1)])
plot(ggplot(Spectral, aes(x=freq)) +  geom_line(aes(y = Period, colour = "
   periodogram")) + geom_line(aes(y=AR, colour = "AR"))

# no stationary process
arima_train <- arima(train_data, order = c(4, 1, 2), method = "CSS")
par(mfrow=c(1,2))
plot(arroots(arima_train),main="Inverse AR roots")
plot(maroots(arima_train),main="Inverse MA roots")

# checking with auto.arima the training dataset ARIMA(0,1,0)
fitted_model <- auto.arima(train_data)
summary(fitted_model)

# Fit the ARIMA based on unit circle (stationary process), ground truth
arima_train2 <- arima(train_data, order = c(2, 1, 2), method = "CSS")
# Summary of the fitted model
summary(arima_train2)

par(mfrow=c(1,2))
plot(arroots(arima_train2),main="Inverse AR roots")
plot(maroots(arima_train2),main="Inverse MA roots")

# checking the poles
A2 <- c(1,-arima_train2$coef[1:2])
AR_Model1 <-data.frame(Roots_fp = 1/polyroot(A2), Radius = abs(1/polyroot(
   A2)), Angle = Arg(1/polyroot(A2))/2/pi*Fs)
B2 <- c(1,arima_train2$coef[3:4])
MA_Model1 <-data.frame(Roots_fp = 1/polyroot(B2), Radius = abs(1/polyroot(
   B2)), Angle = Arg(1/polyroot(B2))/2/pi*Fs)

# Forecast the future 38 samples
forecasted_values <- forecast(arima_train2, h = 38)
plot(autoplot(forecasted_values))

# check this graph
PredictTS(train_data,arima_train2,100,38,4,0.05)
```

B.6 Other functions

```
# Compute AR roots
arroots <- function(object)
{
  if(!("Arima" %in% class(object)) &
     !("ar" %in% class(object)))
    stop("object must be of class Arima or ar")
  if("Arima" %in% class(object))
    parvec <- object$model$phi
  else
    parvec <- object$ar
  if(length(parvec) > 0)
  {
    last.nonzero <- max(which(abs(parvec) > 1e-08))
    if (last.nonzero > 0)
      return(structure(list(
        roots=polyroot(c(1,-parvec[1:last.nonzero])),
        type="AR"),
        class='armaroots'))
  }
  return(structure(list(roots=numeric(0), type="AR"),
                   class='armaroots'))
}

# Compute MA roots
maroots <- function(object)
{
  if(!("Arima" %in% class(object)))
    stop("object must be of class Arima")
  parvec <- object$model$theta
  if(length(parvec) > 0)
  {
    last.nonzero <- max(which(abs(parvec) > 1e-08))
    if (last.nonzero > 0)
      return(structure(list(
        roots=polyroot(c(1,parvec[1:last.nonzero])),
        type="MA"),
        class='armaroots'))
  }
  return(structure(list(roots=numeric(0), type="MA"),
                   class='armaroots'))
}

plot.armaroots <- function(x, xlab="Real", ylab="Imaginary",
                           main=paste("Inverse roots of", x$type,
                                      "characteristic polynomial"),
                           ...)
{
  oldpar <- par(pty='s')
  on.exit(par(oldpar))
  plot(c(-1,1), c(-1,1), xlab=xlab, ylab=ylab,
       type="n", bty="n", xaxt="n", yaxt="n", main=main, ...)
  axis(1, at=c(-1,0,1), line=0.5, tck=-0.025)
  axis(2, at=c(-1,0,1), label=c("-i","0","i"),
       line=0.5, tck=-0.025)
  circx <- seq(-1,1,l=501)
  circy <- sqrt(1-circx^2)
  lines(c(circx,circx), c(circy,-circy), col='gray')
  lines(c(-2,2), c(0,0), col='gray')
  lines(c(0,0), c(-2,2), col='gray')
  if(length(x$roots) > 0)
  {
    inside <- abs(x$roots) > 1
    points(1/x$roots[inside], pch=19, col='black')
    if(sum(!inside) > 0)
      points(1/x$roots[!inside], pch=19, col='red')
  }
}
```