FA542 - Homework #2

I pledge my honor that I have abided by the Stevens Honor System.

Sid Bhatia

2023-10-02

Problem #1

Suppose that the daily log return of a security follows the AR(2) model:

$$r_t = 0.1 - 0.5r_{t-2} + a_t$$

where a_t is a Gaussian white noise series with mean zero and variance 0.2.

i.

$$\mathbb{E}[r_t] = \mu = \frac{\phi_0}{1 - \phi_1 - \phi_2}$$

In this case, $\phi_0 = 0.1$, $\phi_1 = 0$, and $\phi_2 = -0.5$.

```r
phi_0 <- 0.1
phi_1 <- 0
phi_2 <- -0.5

mu_rt <- phi_0 / (1 - phi_1 - phi_2)
mu_rt
```

```
## [1] 0.06666667
```

$$\text{Var}(r_t) = \gamma(0) = \frac{\sigma_a^2}{1 - \phi_1^2 - \phi_2^2}$$

In this case, $\sigma_a^2 = 0.2$.

```r
error_variance <- 0.2

gamma_0 <- error_variance / (1 - phi_1^2 - phi_2^2)
var_rt <- gamma_0

var_rt
```

```
## [1] 0.2666667
```

ii. For a stationary AR(2) series r_t, we have $\rho_0 = 1$.

$$\rho_1 = \frac{\phi_1}{1 - \phi_2}$$

$$\rho_l = \phi_1 * \rho_{l-1} + \phi_2 * \rho_{l-2}$$

```
rho_0 <- 1

rho_1 <- phi_1 / (1 - phi_2)
rho_1
```

```
## [1] 0
```

```
rho_2 <- phi_1 * rho_1 + phi_2 * rho_0
rho_2
```

```
## [1] -0.5
```

iii. For a 1-step ahead forecast r_{101}, it is defined as the following:

$$r_{101} = 0.1 - 0.5 * r_{99} + a_{101}$$

Given that a_{101} is a Gaussian white noise with $\mu = 0$ and $\sigma_a^2 = 0.2$:

$$r_{101} = 0.1 - 0.5(0.05) + a_{101} = 0.075 + a_{101}$$

If we take the conditional expectation of this, we have end up with the following:

$$\mathbb{E}[r_{101}|\mathcal{F}] = 0.075 + \mathbb{E}[a_{101}] = 0.075$$

.

```
# Given values
r_100 <- 0.2
r_99 <- 0.05
variance_a <- 0.2

# 1-Step Ahead Forecast (r_101)
forecast_1_step <- 0.1 - 0.5 * r_99 + rnorm(1, mean = 0, sd = sqrt(variance_a))
r_101 <- 0.075 + forecast_1_step

r_101
```

```
## [1] 0.7029006
```

For a 2-step ahead forecast r_{102}, it is defined as the following:

$$r_{102} = 0.1 - 0.5 * r_{100} + a_{102}$$

Given that a_{102} is a Gaussian white noise with $\mu = 0$ and $\sigma_a^2 = 0.2$:

$$r_{102} = 0.1 - 0.5(0.2) + a_{102} = 0.0 + a_{102}$$

Similarly, if we compute the condiitonal expectation, we have the following:

$$\mathbb{E}[r_{102}|\mathcal{F}] = 0.0 + \mathbb{E}[a_{102}] = 0.0$$

.

```r
# 2-Step Ahead Forecast (r_102)
forecast_2_step <- 0.1 - 0.5 * r_100 + rnorm(1, mean = 0, sd = sqrt(variance_a))
r_102 <- forecast_2_step
r_102
```

```
## [1] -0.1717736
```

The standard deviation of the forecast error at time $n + m$ is:

$$SE(x_{n+m}^n - x_{n+m}) = \sqrt{\hat{\sigma}_w^2 \sum_{j=0}^{m-1} \phi_j^2}$$

When forecasting $m = 1$ time past the end of the series, the SE of the forecast is:

$$SE(x_{n+1}^n - x_{n+1}) = \sqrt{\hat{\sigma}_w^2(1)}$$

When forecasting $m = 2$ time past the end of the series, the SE of the forecast is:

$$SE(x_{n+2}^n - x_{n+2}) = \sqrt{\hat{\sigma}_w^2(1 + \phi_1^2)}$$

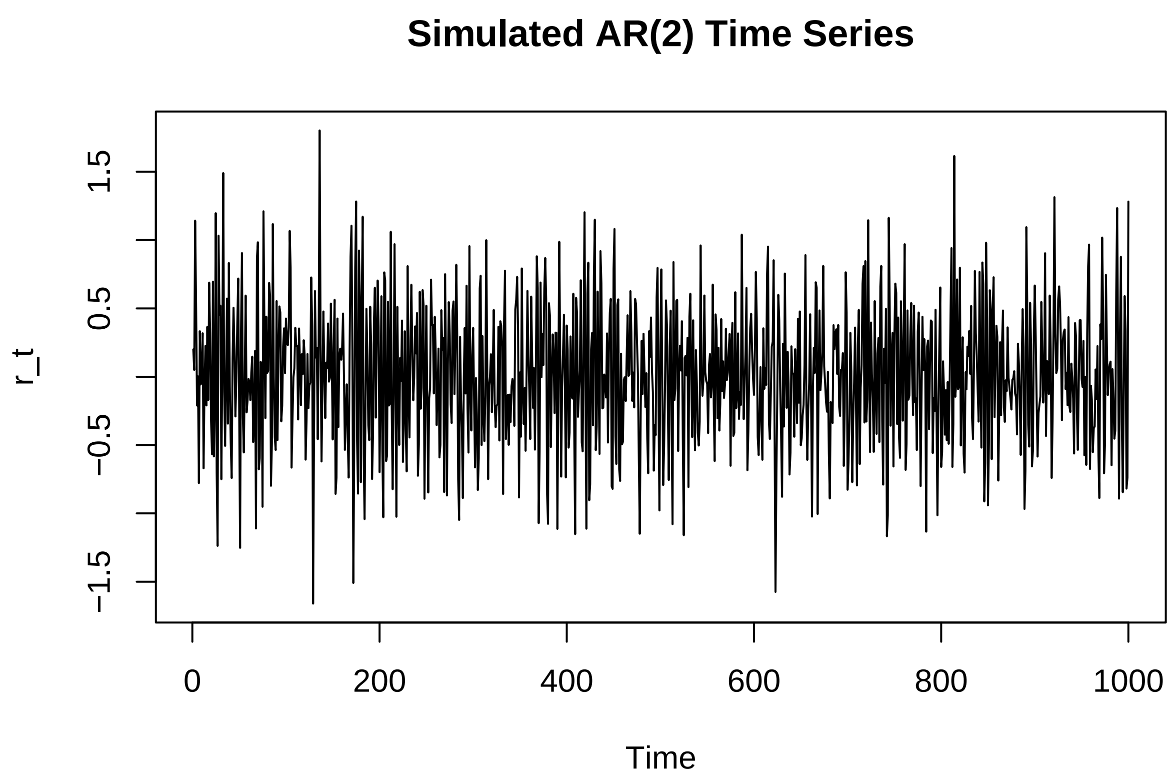
In this case, the associated standard deviations of the forecast errors are $\sqrt{0.2}$ which is:

```r
sqrt(variance_a)
```

```
## [1] 0.4472136
```

iv.

```r

# Set the number of terms to simulate
n <- 1000

# Define the AR(2) coefficients (phi_0, phi_1, phi_2)
phi_0 <- 0.1
phi_1 <- 0
phi_2 <- -0.5

# Set the variance of the Gaussian white noise series (a_t)
variance_a <- 0.2

# Create an ARIMA model representing an AR(2) process
arima_order <- c(2, 0, 0)

# Simulate the time series using arima.sim()
simulated_data <- arima.sim(model = list(ar = c(phi_1, phi_2)), n = n, innov = rnorm(n, mean = 0, sd = 

# Set the initial values manually
simulated_data[1] <- 0.2
simulated_data[2] <- 0.05

# Plot the simulated AR(2) time series
plot(simulated_data, type = 'l', main = 'Simulated AR(2) Time Series', xlab = 'Time', ylab = 'r_t')
```



a.

```
# Calculate sample mean and variance from the generated time series
sample_mean <- mean(simulated_data)
sample_variance <- var(simulated_data)

# Analytical mean and variance (based on the AR(2) model)
analytical_mean <- phi_0 / (1 - phi_1 - phi_2)
analytical_variance <- (variance_a) / (1 - phi_1^2 - phi_2^2)

# Print the results
cat("Sample Mean:", sample_mean, "\n")
```

b.

```
## Sample Mean: 0.01033351
```

```
cat("Sample Variance:", sample_variance, "\n")
```

```
## Sample Variance: 0.2574268
```

```
cat("Analytical Mean:", analytical_mean, "\n")
```

```
## Analytical Mean: 0.06666667
```

```
cat("Analytical Variance:", analytical_variance, "\n")
```

```
## Analytical Variance: 0.2666667
```

```r
# Calculate the sample ACF from the generated time series
sample_acf <- acf(simulated_data, lag.max = 2, plot = FALSE)$acf

cat("Sample Lag-1 ACF:", sample_acf[2], "\n")
```

c.

```
## Sample Lag-1 ACF: 0.01772399
```

```r
cat("Sample Lag-2 ACF:", sample_acf[3], "\n")
```

```
## Sample Lag-2 ACF: -0.4773361
```

```r
cat("Analytical Lag-1 ACF:", rho_1, "\n")
```

```
## Analytical Lag-1 ACF: 0
```

```r
cat("Analytical Lag-2 ACF:", rho_2, "\n")
```

```
## Analytical Lag-2 ACF: -0.5
```

```r
# Set the number of repeated simulations
n_simulations <- 1000

# Define the forecast origin values
forecast_origin <- c(0.2, 0.05)

# Initialize a matrix to store forecast results
forecasts <- matrix(NA, nrow = n_simulations, ncol = 2)

# Simulate and forecast for each simulation
for (i in 1:n_simulations) {
  # Simulate a new AR(2) time series
  simulated_data <- arima.sim(model = list(ar = c(phi_1, phi_2)), n = n, innov = rnorm(n, mean = 0, sd

  # Set the initial values manually
  simulated_data[1] <- forecast_origin[1]
  simulated_data[2] <- forecast_origin[2]

  # Forecast 1-step ahead
  forecast_1_step <- simulated_data[1] * phi_1 + simulated_data[2] * phi_2

  # Forecast 2-step ahead
  forecast_2_step <- forecast_1_step * phi_1 + simulated_data[1] * phi_2

  # Store the forecasts
  forecasts[i, 1] <- forecast_1_step
  forecasts[i, 2] <- forecast_2_step
}

# Calculate the sample standard deviation of the forecasts
sample_std_dev_1_step <- sd(forecasts[, 1])
sample_std_dev_2_step <- sd(forecasts[, 2])
```

```r
# Calculate the analytical standard deviations for forecasts
analytical_std_dev_1_step <- sqrt(variance_a)
analytical_std_dev_2_step <- sqrt(variance_a)

# Print the results
cat("Sample Standard Deviation of 1-Step Ahead Forecasts:", sample_std_dev_1_step, "\n")
```

d.

```
## Sample Standard Deviation of 1-Step Ahead Forecasts: 0
```

```r
cat("Sample Standard Deviation of 2-Step Ahead Forecasts:", sample_std_dev_2_step, "\n")
```

```
## Sample Standard Deviation of 2-Step Ahead Forecasts: 0
```

```r
cat("Analytical 1-Step Ahead Forecast:", analytical_std_dev_1_step, "\n")
```

```
## Analytical 1-Step Ahead Forecast: 0.4472136
```

```r
cat("Analytical 2-Step Ahead Forecast:", analytical_std_dev_2_step, "\n")
```

```
## Analytical 2-Step Ahead Forecast: 0.4472136
```

Problem #2

Suppose that the simple return of a monthly bond index follows the MA(1) model:

$$R_t = a_t - 0.1a_{t-1}$$

where a_t is a Gaussian white noise series with mean zero and variance 0.01.

i.

$$\mathbb{E}[R_t] = \mu = 0$$

$$\text{Var}(R_t) = \sigma_a^2(1 + \theta_1^2)$$

where $\theta_1 = -0.1$.

Therefore:

```r
mu_Rt <- 0
theta_1 <- -0.1
variance_a_Rt <- 0.01

variance_Rt <- variance_a_Rt * (1 + theta_1^2)
variance_Rt
```

```
## [1] 0.0101
```

```r
mu_Rt
```

```
## [1] 0
```

```r
variance_Rt
```

```
## [1] 0.0101
```

ii.

$$\rho_1 = \frac{\theta_1}{1+\theta_1^2}$$

$$\rho_2 = \frac{\theta_2}{1+\theta_1^2+\theta_2^2}$$

```
rho_1_Rt <- theta_1 / (1 + theta_1^2)
rho_1_Rt
```

```
## [1] -0.0990099
```

```
theta_2 <- 0
rho_2_Rt <- theta_2 / (1 + theta_1^2 + theta_2^2)
rho_2_Rt
```

```
## [1] 0
```

iii. For a 1-step ahead forecast R_{101}, it is defined as the following:

$$R_{101} = a_{101} - 0.1 * a_{100}$$

Given that a_{101} is a Gaussian white noise with $\mu = 0$ and $\sigma_a^2 = 0.01$:

$$R_{101} = a_{101} - 0.1(0.01) = a_{101} - 0.001$$

If we take the conditional expectation of this, we have end up with the following:

$$\mathbb{E}[R_{101}|\mathcal{F}] = \mathbb{E}[a_{101}] - 0.001 = 0.001$$

.

For a 2-step ahead forecast R_{102}, it is defined as the following:

$$R_{102} = a_{102} - 0.1 * a_{101}$$

Given that a_{102} is a Gaussian white noise with $\mu = 0$ and $\sigma_a^2 = 0.2$:

$$R_{102} = a_{102} - 0.1 * a_{101}$$

Similarly, if we compute the condiitonal expectation, we have the following:

$$\mathbb{E}[R_{102}|\mathcal{F}] = 0.0$$

.

```
# Given values
a_100 <- 0.01
a_101 <- rnorm(1, mean = 0, sd = sqrt(0.01))  # Simulate a_101 as it's not given

# 1-Step Ahead Forecast (R_101)
forecast_1_step <- a_101 - 0.1 * a_100

a_102 <- rnorm(1, mean = 0, sd = sqrt(0.01)) # Simulate a_102 as it's not given
```

```r
# 2-Step Ahead Forecast (R_102)
forecast_2_step <- a_102 - 0.1 * a_101

# Calculate standard deviations of forecast errors
std_dev_1_step <- sqrt(0.01)
std_dev_2_step <- sqrt(0.01 * (1 + theta_1^2))

# Print the results
cat("1-Step Ahead Forecast (R_101):", forecast_1_step, "\n")
```

```
## 1-Step Ahead Forecast (R_101): 0.02960806
```

```r
cat("2-Step Ahead Forecast (R_102):", forecast_2_step, "\n")
```

```
## 2-Step Ahead Forecast (R_102): 0.0009520318
```

```r
cat("Standard Deviation of 1-Step Ahead Forecast Error (e_101):", std_dev_1_step, "\n")
```

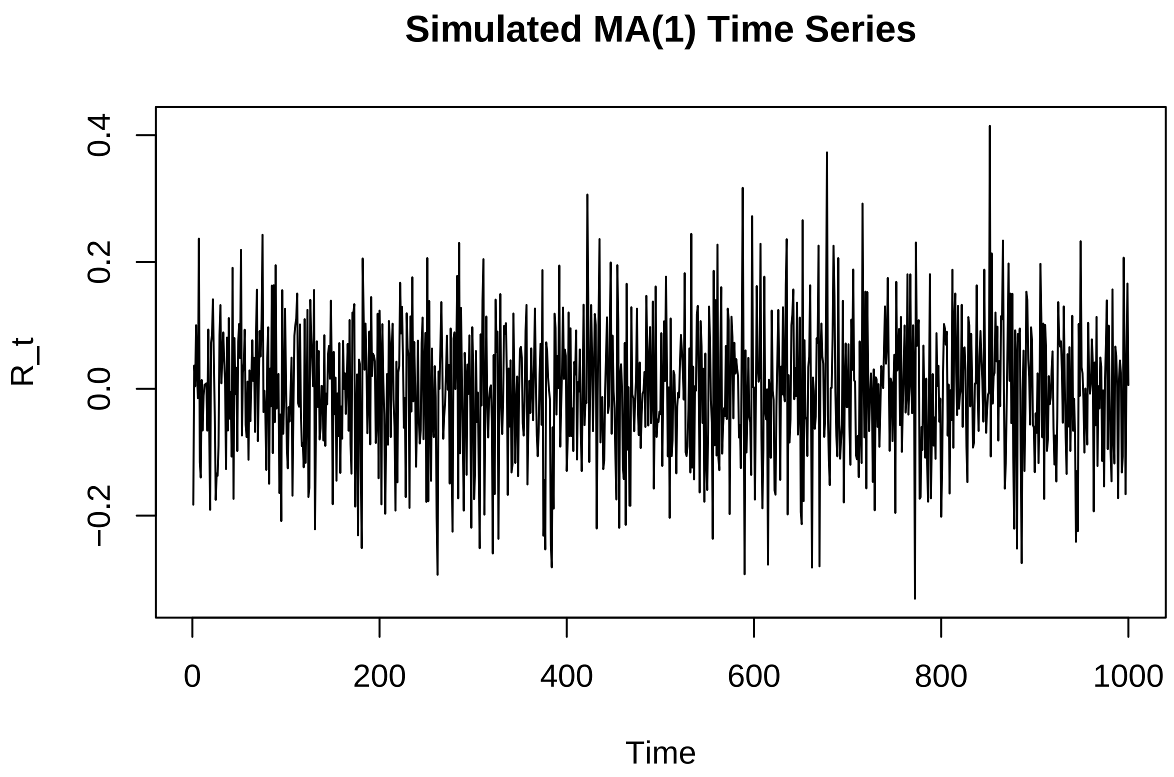
```
## Standard Deviation of 1-Step Ahead Forecast Error (e_101): 0.1
```

```r
cat("Standard Deviation of 2-Step Ahead Forecast Error (e_102):", std_dev_2_step, "\n")
```

```
## Standard Deviation of 2-Step Ahead Forecast Error (e_102): 0.1004988
```

iv.

```r
# Number of time periods
n <- 1000

# Set the parameters of the MA(1) model
order_ma <- c(0, 0, 1)  # ARIMA order (p, d, q)
ma_coefs <- -0.1        # MA(1) coefficient

# Variance of the white noise series a_t
variance_a <- 0.01

# Simulate the MA(1) time series
simulated_data_Rt <- arima.sim(model = list(order = order_ma, ma = ma_coefs), n = n, innov = rnorm(n, me

# Plot the simulated time series
plot(simulated_data_Rt, type = "l", main = "Simulated MA(1) Time Series", xlab = "Time", ylab = "R_t")
```



a.

```r
# Calculate sample mean and variance
sample_mean_Rt <- mean(simulated_data_Rt)
sample_variance_Rt <- var(simulated_data_Rt)

# Analytical mean and variance
analytical_mean_Rt <- 0
analytical_variance_Rt <- variance_a * (1 + theta_1^2)

# Print the results
cat("Sample Mean:", sample_mean_Rt, "\n")
```

b.

```
## Sample Mean: -0.002023778
```

```r
cat("Sample Variance:", sample_variance_Rt, "\n")
```

```
## Sample Variance: 0.01153689
```

```r
cat("Analytical Mean:", analytical_mean_Rt, "\n")
```

```
## Analytical Mean: 0
```

```r
cat("Analytical Variance:", analytical_variance_Rt, "\n")
```

```
## Analytical Variance: 0.0101
```

```r
# Calculate sample ACF
acf_result <- acf(simulated_data_Rt, lag.max = 2, plot = FALSE)

# Analytical ACF
analytical_acf_1 <- theta_1 / (1 + theta_1^2)  # Analytical ACF at lag 1
analytical_acf_2 <- 0  # Analytical ACF at lag 2 for MA(1)

# Print the results
cat("Sample Lag-1 ACF:", acf_result$acf[2], "\n")
```

c.

```
## Sample Lag-1 ACF: -0.08106657
```

```r
cat("Sample Lag-2 ACF:", acf_result$acf[3], "\n")
```

```
## Sample Lag-2 ACF: -0.07569482
```

```r
cat("Analytical Lag-1 ACF:", analytical_acf_1, "\n")
```

```
## Analytical Lag-1 ACF: -0.0990099
```

```r
cat("Analytical Lag-2 ACF:", analytical_acf_2, "\n")
```

```
## Analytical Lag-2 ACF: 0
```

```r
# Number of time periods
n <- 1000

# Number of simulations
n_simulations <- 1000

# Fixed value for a_t
a_fixed <- 0.01

# Initialize vectors to store forecasts
forecasts_1_step <- numeric(n_simulations)
forecasts_2_step <- numeric(n_simulations)

# Simulate the MA(1) time series multiple times
for (sim in 1:n_simulations) {
  # Simulate white noise series with the fixed value a_fixed
  a <- rep(a_fixed, n)

  # Simulate the MA(1) time series
  simulated_data <- arima.sim(model = list(order = order_ma, ma = ma_coefs), n = n, innov = a)

  # Calculate 1-step ahead forecast (R_{t+1})
  forecasts_1_step[sim] <- simulated_data[1]

  # Calculate 2-step ahead forecast (R_{t+2})
  forecasts_2_step[sim] <- simulated_data[2]
}
```

```r
# Calculate sample standard deviations of forecasts
std_dev_1_step <- sd(forecasts_1_step)
std_dev_2_step <- sd(forecasts_2_step)

# Analytical standard deviations
analytical_std_dev_1_step <- sqrt(a_fixed)  # Analytical std dev for 1-step ahead
analytical_std_dev_2_step <- sqrt(a_fixed * (1 + theta_1^2))  # Analytical std dev for 2-step ahead

# Print the results
cat("Sample Standard Deviation of 1-Step Ahead Forecasts:", std_dev_1_step, "\n")
```

d.

```
## Sample Standard Deviation of 1-Step Ahead Forecasts: 0.1019233
```

```r
cat("Sample Standard Deviation of 2-Step Ahead Forecasts:", std_dev_2_step, "\n")
```

```
## Sample Standard Deviation of 2-Step Ahead Forecasts: 0
```

```r
cat("Analytical Standard Deviation of 1-Step Ahead Forecasts:", analytical_std_dev_1_step, "\n")
```

```
## Analytical Standard Deviation of 1-Step Ahead Forecasts: 0.1
```

```r
cat("Analytical Standard Deviation of 2-Step Ahead Forecasts:", analytical_std_dev_2_step, "\n")
```

```
## Analytical Standard Deviation of 2-Step Ahead Forecasts: 0.1004988
```

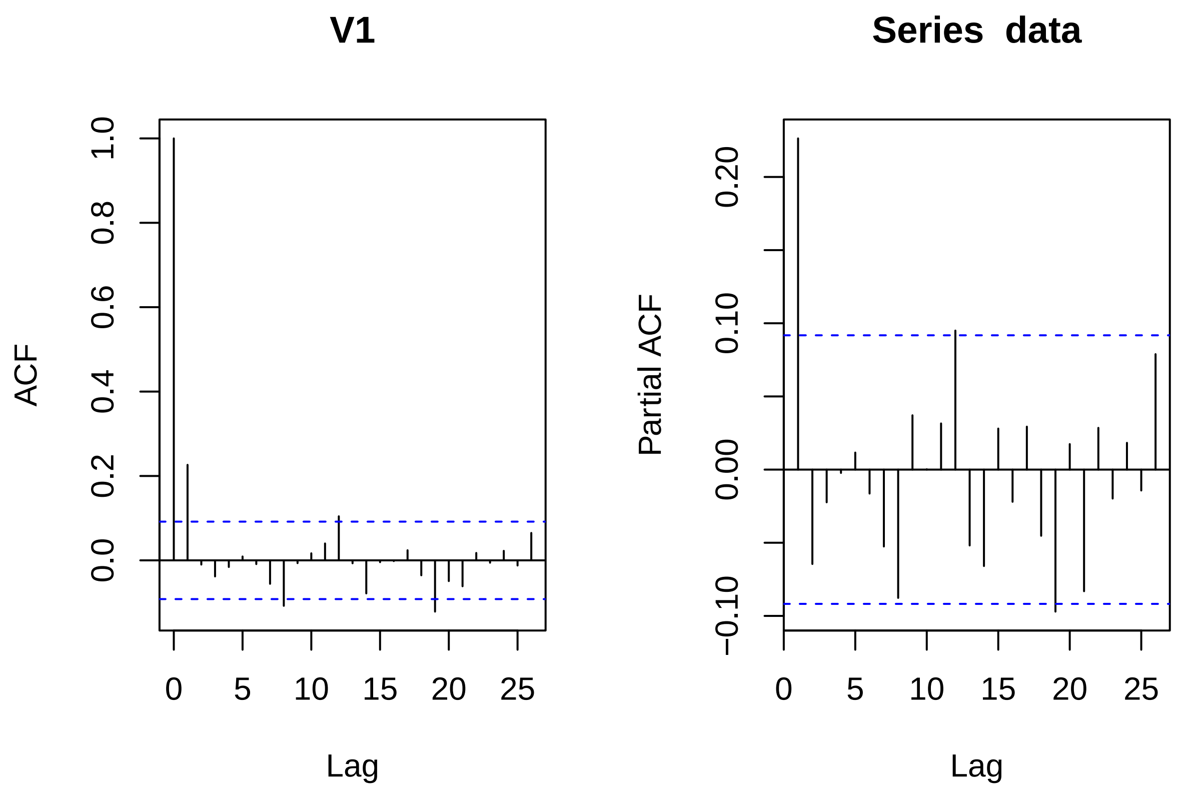
Problem #3

```r
# Load in data
data <- read.table("C:/Users/sbhatia2/My Drive/University/Academics/Semester V/FA542 - Time Series with

# Fit an AR model with lag order determined by AIC
ar_order <- ar(data, aic = TRUE)

# Summary of the AR model (AR(1))
summary(ar_order)
```

i.

```
##              Length Class  Mode
## order          1    -none- numeric
## ar             1    -none- numeric
## var.pred       1    -none- numeric
## x.mean         1    -none- numeric
## aic           27    -none- numeric
## n.used         1    -none- numeric
## n.obs          1    -none- numeric
## order.max      1    -none- numeric
## partialacf    26    -none- numeric
## resid        456    -none- numeric
## method         1    -none- character
## series         1    -none- character
## frequency      1    -none- numeric
## call           3    -none- call
## asy.var.coef   1    -none- numeric
```

```r
# Create a combined ACF and PACF plot
par(mfrow = c(1, 2))  # Set up a 1x2 grid for plotting
acf(data)
pacf(data)
```



```r
ar_model <- arima(data, order = c(1, 0, 0))
ar_model
```

```
##
## Call:
## arima(x = data, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.2267     1.0626
## s.e.  0.0456     0.3297
##
## sigma^2 estimated as 29.68:  log likelihood = -1420.11,  aic = 2846.22
```

As such, the fitted model is the following:

$$r_t = 1.0626 + 0.2267r_{t-1}$$

```r
# Perform model selection for MA order (e.g., from 0 to a maximum order)
max_ma_order <- 10
```

```r
best_order <- NULL
lowest_aic <- Inf

for (p in 0:max_ma_order) {
  ma_order <- arima(data, order = c(0, 0, p))
  aic <- AIC(ma_order)

  cat("MA(", p, ") AIC:", aic, "\n")

  if (aic < lowest_aic) {
    best_order <- p
    lowest_aic <- aic
  }
}
```

ii.

```
## MA( 0 ) AIC: 2868.238
## MA( 1 ) AIC: 2844.73
## MA( 2 ) AIC: 2846.713
## MA( 3 ) AIC: 2848.278
## MA( 4 ) AIC: 2850.042
## MA( 5 ) AIC: 2852.041
## MA( 6 ) AIC: 2854.041
## MA( 7 ) AIC: 2856.017
## MA( 8 ) AIC: 2851.92
## MA( 9 ) AIC: 2853.908
## MA( 10 ) AIC: 2855.385
```

```r
cat("Lowest AIC:", lowest_aic, "\n")  # Print the lowest AIC (MA(1))
```

```
## Lowest AIC: 2844.73
```

```r
ma_model <- arima(data, order = c(0, 0, 1))
ma_model
```

```
##
## Call:
## arima(x = data, order = c(0, 0, 1))
##
## Coefficients:
##          ma1  intercept
##       0.2385     1.0605
## s.e.  0.0449     0.3153
##
## sigma^2 estimated as 29.59:  log likelihood = -1419.37,  aic = 2844.73
```

As such, the fitted model is the following:

$$R_t = 1.0605 + 0.2385a_{t-1}$$

```r
library(forecast)
```

iii.

```
## Registered S3 method overwritten by 'quantmod':
```

```
##   method             from
##   as.zoo.data.frame zoo
```

```
# 1-Step Ahead Forecast for AR Model
ar_1step_forecast <- forecast(ar_model, h = 1)
cat("AR 1-Step Ahead Forecast:", ar_1step_forecast$mean[1], "\n")
```

```
## AR 1-Step Ahead Forecast: 2.601682
```

```
# 2-Step Ahead Forecast for AR Model
ar_2step_forecast <- forecast(ar_model, h = 2)
cat("AR 2-Step Ahead Forecast:", ar_2step_forecast$mean[2], "\n")
```

```
## AR 2-Step Ahead Forecast: 1.411453
```

```
# 1-Step Ahead Forecast for MA Model
ma_1step_forecast <- forecast(ma_model, h = 1)
cat("MA 1-Step Ahead Forecast:", ma_1step_forecast$mean[1], "\n")
```

```
## MA 1-Step Ahead Forecast: 2.250303
```

```
# 2-Step Ahead Forecast for MA Model
ma_2step_forecast <- forecast(ma_model, h = 2)
cat("MA 2-Step Ahead Forecast:", ma_2step_forecast$mean[2], "\n")
```

```
## MA 2-Step Ahead Forecast: 1.060512
```

```

# Compare AIC and BIC
cat("AR Model AIC:", AIC(ar_model), "\n")
```

iv.

```
## AR Model AIC: 2846.221
```

```
cat("AR Model BIC:", BIC(ar_model), "\n")
```

```
## AR Model BIC: 2858.588
```

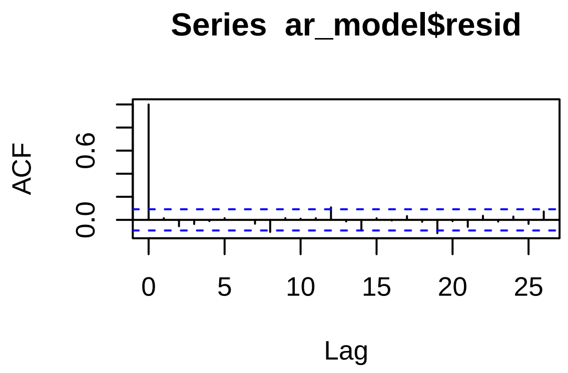
```
cat("MA Model AIC:", AIC(ma_model), "\n")
```

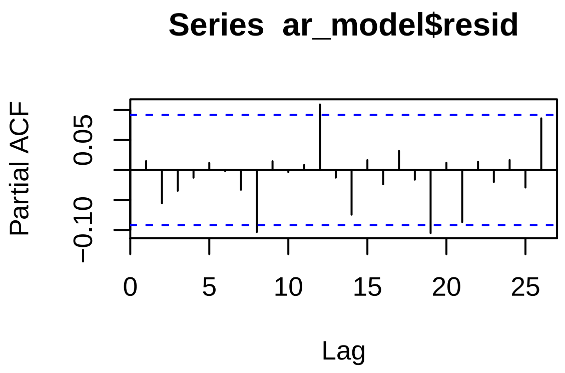
```
## MA Model AIC: 2844.73
```

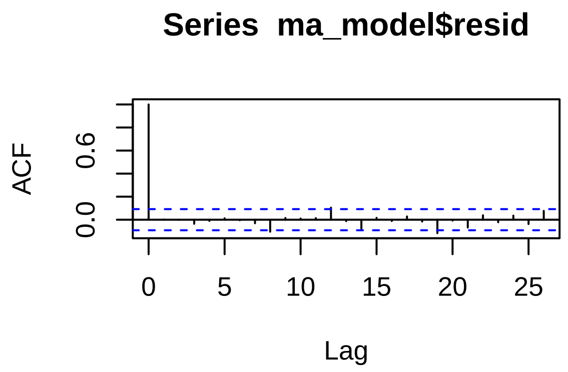
```
cat("MA Model BIC:", BIC(ma_model), "\n")
```

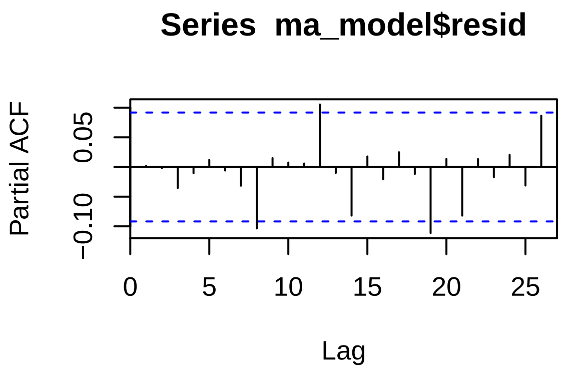
```
## MA Model BIC: 2857.098
```

```
# Compare residual diagnostics (ACF and PACF plots)
par(mfrow = c(2, 2))
acf(ar_model$resid)
pacf(ar_model$resid)
acf(ma_model$resid)
pacf(ma_model$resid)
```









As we can see, the MA model has lower AIC and BIC respectively, implying that is better representative of the data.

Problem #4

```
# Read in the data and separate into respective columns
problem_4_data <- read.table("C:/Users/sbhatia2/My Drive/University/Academics/Semester V/FA542 - Time S
colnames(problem_4_data)
```

i.

```
## [1] "DATE" "AAA"
```

```
head(problem_4_data)
```

```
##         DATE  AAA
## 1 1962-01-01 4.42
## 2 1962-02-01 4.42
## 3 1962-03-01 4.39
## 4 1962-04-01 4.33
## 5 1962-05-01 4.28
## 6 1962-06-01 4.28
```

```
library(e1071)

# Calculate sample mean
```

```r
mean_yield <- mean(problem_4_data$AAA)

# Calculate standard deviation
std_deviation_yield <- sd(problem_4_data$AAA)

# Calculate skewness
skewness_yield <- skewness(problem_4_data$AAA)

# Calculate excess kurtosis
excess_kurtosis_yield <- kurtosis(problem_4_data$AAA)

# Display summary statistics
cat("Sample Mean:", mean_yield, "\n")
```

ii.

```
## Sample Mean: 6.991056
```

```r
cat("Standard Deviation:", std_deviation_yield, "\n")
```

```
## Standard Deviation: 2.713106
```

```r
cat("Skewness:", skewness_yield, "\n")
```

```
## Skewness: 0.6954361
```

```r
cat("Excess Kurtosis:", excess_kurtosis_yield, "\n")
```

```
## Excess Kurtosis: 0.246882
```

```r

library(tseries)

# Check for stationarity
adf_test_result <- adf.test(problem_4_data$AAA)
adf_test_result$p.value
```

iii.

```
## [1] 0.5291745
```

Since the p-value is greater than 0.05, we fail to reject the null hypothesis that this time series data is not stationary at the 95% confidence level.

As such, we will difference the data and see if that is stationary.

```r
differenced_data <- diff(problem_4_data$AAA, differences = 1)
adf.test(differenced_data)$p.value
```

```
## Warning in adf.test(differenced_data): p-value smaller than printed p-value
```

```
## [1] 0.01
```

Since the p-value is less than 0.05, we reject the null hypothesis that this difference ($d = 1$) time series data is not stationary at the 95% confidence level.

As such, we accept the alternative hypothesis that this difference time series data is stationary.

```r
# Grid search for ARIMA orders
best_aic <- Inf
best_order <- c(0, 0, 0)
```

```r
for (p in 0:10) {
  for (d in 1:1) {
    for (q in 0:10) {
      current_order <- c(p, d, q)
      current_aic <- AIC(arima(problem_4_data$AAA, order = current_order))

      if (current_aic < best_aic) {
        best_aic <- current_aic
        best_order <- current_order
      }
    }
  }
}
```

```
## Warning in arima(problem_4_data$AAA, order = current_order): possible
## convergence problem: optim gave code = 1

## Warning in arima(problem_4_data$AAA, order = current_order): possible
## convergence problem: optim gave code = 1

## Warning in log(s2): NaNs produced

## Warning in arima(problem_4_data$AAA, order = current_order): possible
## convergence problem: optim gave code = 1

## Warning in arima(problem_4_data$AAA, order = current_order): possible
## convergence problem: optim gave code = 1

## Warning in arima(problem_4_data$AAA, order = current_order): possible
## convergence problem: optim gave code = 1

## Warning in arima(problem_4_data$AAA, order = current_order): possible
## convergence problem: optim gave code = 1

## Warning in log(s2): NaNs produced

## Warning in arima(problem_4_data$AAA, order = current_order): possible
## convergence problem: optim gave code = 1

## Warning in arima(problem_4_data$AAA, order = current_order): possible
## convergence problem: optim gave code = 1

## Warning in arima(problem_4_data$AAA, order = current_order): possible
## convergence problem: optim gave code = 1

## Warning in arima(problem_4_data$AAA, order = current_order): possible
## convergence problem: optim gave code = 1

## Warning in arima(problem_4_data$AAA, order = current_order): possible
## convergence problem: optim gave code = 1

## Warning in arima(problem_4_data$AAA, order = current_order): possible
## convergence problem: optim gave code = 1

## Warning in arima(problem_4_data$AAA, order = current_order): possible
## convergence problem: optim gave code = 1

## Warning in log(s2): NaNs produced

## Warning in arima(problem_4_data$AAA, order = current_order): possible
## convergence problem: optim gave code = 1

## Warning in arima(problem_4_data$AAA, order = current_order): possible
## convergence problem: optim gave code = 1
```

```r
cat("Best ARIMA Order (p, d, q):", best_order, "\n")
```

```
## Best ARIMA Order (p, d, q): 5 1 10
```

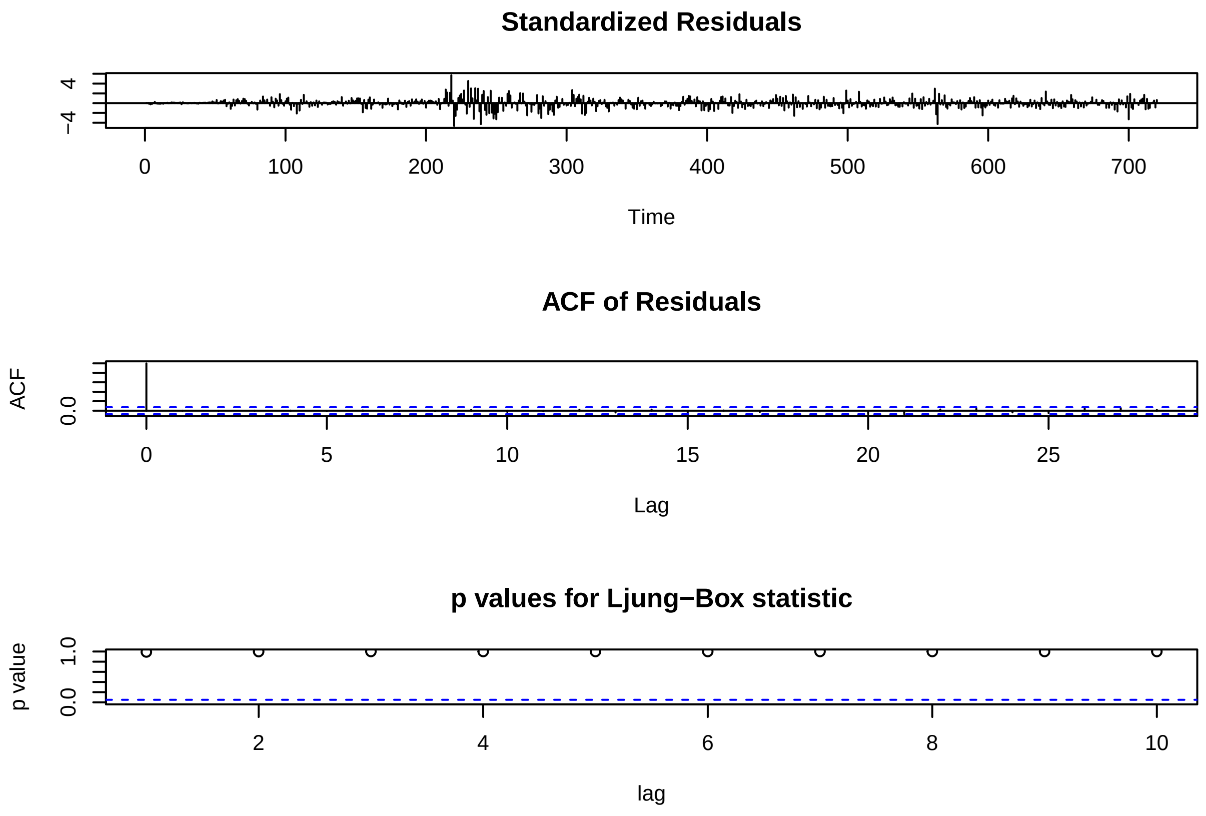
```r
cat("Best AIC:", best_aic, "\n")
```

```
## Best AIC: -244.2574
```

```r
# Fit best ARIMA model
best_arima_model <- arima(problem_4_data$AAA, order = best_order)
```

```
## Warning in arima(problem_4_data$AAA, order = best_order): possible convergence
## problem: optim gave code = 1
```

```r
# Model diagnostics
tsdiag(best_arima_model)
```



```r
# Out-of-Sample Forecasting (split data into train and validation sets)
n_train <- floor(0.8 * length(problem_4_data$AAA))
train_data <- problem_4_data$AAA[1:n_train]
```

```r
validation_data <- problem_4_data$AAA[(n_train + 1):length(problem_4_data$AAA)]

best_forecast <- forecast(best_arima_model, h = length(validation_data))

# Calculate MAE and RMSE for validation
mae <- mean(abs(best_forecast$mean - validation_data))
rmse <- sqrt(mean((best_forecast$mean - validation_data)^2))

cat("Mean Absolute Error (MAE):", mae, "\n")
```

```
## Mean Absolute Error (MAE): 1.159337
```

```r
cat("Root Mean Squared Error (RMSE):", rmse, "\n")
```

```
## Root Mean Squared Error (RMSE): 1.326411
```

As we can see, the ARIMA$(5, 1, 10)$ did well with its respective AIC at -244.2574, MAE at 1.159, and RMSE at 1.326.