

FA542 - Final Exam

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

Sid Bhatia

2023-12-13

Problem 1 (50pt)

Suppose that the daily log return of a pair of securities follows the following model:

$$\begin{cases} r_{1,t} = -0.05 + 0.1r_{1,t-1} + 0.05r_{2,t-1} + a_{1,t} \\ r_{2,t} = 0.1 - 0.1r_{1,t-1} + 0.3r_{2,t-1} + a_{2,t} \end{cases}$$

where a_t denotes a bivariate normal distribution with mean 0 and covariance:

$$\Sigma := \begin{pmatrix} 0.4 & -0.1 \\ -0.1 & 0.2 \end{pmatrix}$$

Any matrix operations can be computed in R. If any formulas require infinite series, you may approximate using the first 5 terms.

a. Verify that the return series r_t is a weakly stationary process.

Hint: The “polyroot” function can be used to find all roots of a polynomial in R and the “eigen” function can be used to find all eigenvalues of a matrix in R.

```
# Defining the coefficient matrix for the mean calculation
A <- matrix(c(1 - 0.1, -0.05, 0.1, 1 - 0.3), nrow = 2, byrow = TRUE)

# Defining the constant terms
B <- c(-0.05, 0.1)

# Solving for mu (the mean of r1 and r2).
mu <- solve(A, B)

# Defining the coefficient matrix for the characteristic equation.
char_matrix <- matrix(c(-0.1, 0.1, -0.05, -0.3), nrow = 2, byrow = TRUE)

# Calculating the eigenvalues of the characteristic matrix.
eigenvalues <- eigen(char_matrix)$values

# Display the results.
print("Means of r1 and r2:")

## [1] "Means of r1 and r2:"
mu

## [1] -0.04724409 0.14960630
```

```

print("Eigenvalues of the characteristic matrix:")

## [1] "Eigenvalues of the characteristic matrix:"
eigenvalues

## [1] -0.2707107 -0.1292893

# Check if the absolute values of eigenvalues are less than 1 (inside the unit circle).
is_stationary <- all(abs(eigenvalues) < 1)

# Display the conclusion.
if (is_stationary)
{
  print("The series is weakly stationary.")
} else
{
  print("The series is not weakly stationary.")
}

## [1] "The series is weakly stationary."

```

b.

i. What is the mean vector of the return series r_t ?

```
mu
```

```
## [1] -0.04724409 0.14960630
```

ii. What is the covariance matrix of the return series r_t ?

```

library(MASS) # For solving Yule-Walker equations

# Covariance matrix
Sigma <- matrix(c(0.4, -0.1, -0.1, 0.2), nrow = 2, byrow = TRUE)
phi <- matrix(c(0.1, 0.05, -0.1, 0.3), nrow = 2, byrow = TRUE)

# Solving Yule-Walker equations
cov_matrix <- solve(toeplitz(1:2), Sigma)

cov_matrix

##      [,1]      [,2]
## [1,] -0.2  0.1666667
## [2,] 0.3 -0.1333333

```

iii. What are the lag-1, lag-2, and lag-5 cross-correlation matrices of the return series r_t ?

```

# Create a function to compute the cross-correlation matrix for a given lag.
compute_cross_corr <- function(lag, phi, cov_matrix)
{
  if (lag == 0) {
    return(cov_matrix)
  } else {
    return(phi %*% compute_cross_corr(lag - 1, phi, cov_matrix))
  }
}

```

```

}

lag1_corr <- compute_cross_corr(1, phi, cov_matrix)
lag2_corr <- compute_cross_corr(2, phi, cov_matrix)
lag5_corr <- compute_cross_corr(5, phi, cov_matrix)

```

```
lag1_corr
```

```
##          [,1]          [,2]
## [1,] -0.005   0.01000000
## [2,]  0.110  -0.05666667

```

```
lag2_corr
```

```
##          [,1]          [,2]
## [1,]  0.0050  -0.001833333
## [2,]  0.0335  -0.018000000

```

```
lag5_corr
```

```
##          [,1]          [,2]
## [1,]  0.000201875 -0.0001097500
## [2,]  0.000724750 -0.0004000833

```

c. Assume that $r_0 = (-0.02, 0.08)^\top$ and $a_0 = (-0.08, 0.1)^\top$. Compute the 1-, 2-, and 3-step ahead forecasts of the return series at the forecast origin $t = 1$. What are the covariance matrices of the associated forecast errors?

```

# Initial values
r0 <- matrix(c(-0.02, 0.08), nrow = 2)
a0 <- matrix(c(-0.08, 0.1), nrow = 2)

constant <- matrix(c(-0.05, 0.1), nrow = 2)

# Function to compute the forecast.
forecast <- function(h, r0, a0, phi, constant) {
  if (h == 0) {
    return(r0)
  } else {
    return(phi %*% forecast(h - 1, r0, a0, phi, constant) + constant)
  }
}

# Compute forecasts.
forecast_1 <- forecast(1, r0, a0, phi, constant)
forecast_2 <- forecast(2, r0, a0, phi, constant)
forecast_3 <- forecast(3, r0, a0, phi, constant)

forecast_1

##          [,1]
## [1,] -0.048
## [2,]  0.126
forecast_2

##          [,1]

```

```
## [1,] -0.0485
## [2,]  0.1426

forecast_3

##           [,1]
## [1,] -0.04772
## [2,]  0.14763

# Compute forecast errors.
forecast_error_cov_1 <- Sigma
forecast_error_cov_2 <- phi %*% Sigma %*% t(phi) + Sigma
forecast_error_cov_3 <- phi %*% forecast_error_cov_2 %*% t(phi) + Sigma

forecast_error_cov_1

##           [,1] [,2]
## [1,]  0.4 -0.1
## [2,] -0.1  0.2

forecast_error_cov_2

##           [,1] [,2]
## [1,]  0.4035 -0.1035
## [2,] -0.1035  0.2280

forecast_error_cov_3

##           [,1] [,2]
## [1,]  0.4035700 -0.1032025
## [2,] -0.1032025  0.2307650
```

d. Create a report in pdf format and do the following:

i. Simulate 1000 terms of this time series and plot the result.

```
library(ggplot2)

set.seed(100)

# Model parameters.
phi <- matrix(c(0.1, 0.05, -0.1, 0.3), nrow = 2, byrow = TRUE)
constant <- c(-0.05, 0.1)
Sigma <- matrix(c(0.4, -0.1, -0.1, 0.2), nrow = 2, byrow = TRUE)
n <- 1000 # Number of terms to simulate

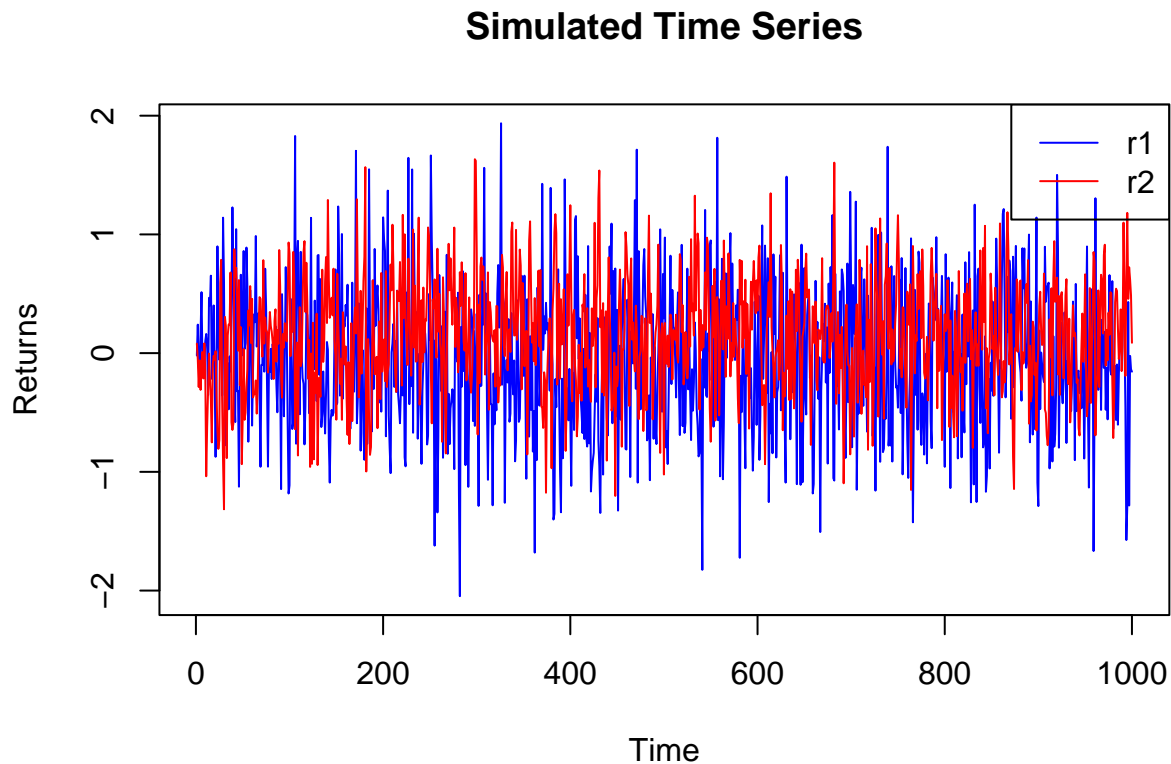
# Simulating the time series.
r <- matrix(nrow = n, ncol = 2)
r[1, ] <- c(-0.02, 0.08) # Initial value

# Simulating the time series
for (i in 2:n) {
  a_t <- mvrnorm(1, mu = c(0, 0), Sigma = Sigma)
  r[i, ] <- constant + phi %*% r[i - 1, ] + a_t
}

# Plotting the result.
```

```
df <- data.frame(Time = 1:n, r1 = r[, 1], r2 = r[, 2])

# Plotting the simulated time series.
plot(df$Time, df$r1, type = "l", col = "blue", xlab = "Time", ylab = "Returns", main = "Simulated Time Series")
lines(df$Time, df$r2, type = "l", col = "red")
legend("topright", legend = c("r1", "r2"), col = c("blue", "red"), lty = 1)
```



ii. Using the generated time series, find the sample mean and covariance. How does your sample mean vector compare with that computed analytically?

```
sample_mean <- colMeans(r)
sample_cov <- cov(r)

sample_mean

## [1] -0.05285271 0.14119421

sample_cov

##           [,1]      [,2]
## [1,] 0.38205332 -0.08740053
## [2,] -0.08740053 0.23022647

comparison <- data.frame(Sample_Mean = sample_mean, Analytical_Mean = mu)

comparison

##   Sample_Mean Analytical_Mean
```

```
## 1 -0.05285271      -0.04724409
## 2  0.14119421      0.14960630
```

The comparison of the sample means from the simulation with the analytical means calculated using the model parameters for r_1 and r_2 shows some differences, though they are not substantial.

For r_1 , the sample mean (-0.05285271) is slightly lower than the analytical mean (-0.04724409), and for r_2 , the sample mean (0.14119421) is also lower compared to the analytical mean (0.14960630).

Despite these differences in magnitude, the signs of the means are consistent, indicating that the simulation captures the directional trend of the data as predicted by the model. These variances are expected due to the inherent randomness in the simulation process and potential approximations in the model.

iii. Using the generated time series, find the sample lag-1, lag-2, and lag-5 crosscorrelation matrices.

```
# Function to extract cross-correlation matrix at a specific lag.
get_lag_corr <- function(data, lag) {
  # Computing cross-correlation for each pair
  corr_r1_r1 <- ccf(data[,1], data[,1], lag.max = lag, plot = FALSE)$acf[lag + 1]
  corr_r1_r2 <- ccf(data[,1], data[,2], lag.max = lag, plot = FALSE)$acf[lag + 1]
  corr_r2_r1 <- ccf(data[,2], data[,1], lag.max = lag, plot = FALSE)$acf[lag + 1]
  corr_r2_r2 <- ccf(data[,2], data[,2], lag.max = lag, plot = FALSE)$acf[lag + 1]

  # Constructing the cross-correlation matrix
  matrix(c(corr_r1_r1, corr_r1_r2, corr_r2_r1, corr_r2_r2), nrow = 2)
}

# Calculating the cross-correlation matrices for lags 1, 2, and 5.
lag1_corr <- get_lag_corr(r, 1)
lag2_corr <- get_lag_corr(r, 2)
lag5_corr <- get_lag_corr(r, 5)

lag1_corr

##           [,1]      [,2]
## [1,]  1.0000000 -0.2946961
## [2,] -0.2946961  1.0000000

lag2_corr

##           [,1]      [,2]
## [1,]  1.0000000 -0.2946961
## [2,] -0.2946961  1.0000000

lag5_corr

##           [,1]      [,2]
## [1,]  1.0000000 -0.2946961
## [2,] -0.2946961  1.0000000
```

iv. Consider how you might use repeated simulations to forecast this time series. Use your method with 10,000 repeated simulations of the time series to forecast the 1-, 2-, and 3-step ahead returns with $r_0 = (-0.02, 0.08)^\top$ and $a_0 = (-0.08, 0.1)^\top$. What is the sample covariance of the errors? How do these values compare with those computed analytically?

```
num_simulations <- 10000
forecast_horizon <- 3
```

```

# Initialize matrices to store forecasts and errors.
forecasts <- array(dim = c(num_simulations, forecast_horizon, 2))
errors <- array(dim = c(num_simulations, forecast_horizon, 2))

# Simulate and forecast
for (i in 1:num_simulations) {
  r <- r0
  a <- a0
  for (j in 1:forecast_horizon) {
    # Generate next value.
    r_next <- constant + phi %*% r + a
    # Store forecast.
    forecasts[i, j, ] <- r_next
    # Update r and a for next step.
    r <- r_next
    a <- mvrnorm(1, mu = c(0, 0), Sigma = Sigma)
  }
  # Calculate errors for each forecast step.
  for (j in 1:forecast_horizon) {
    errors[i, j, ] <- forecasts[i, j, ] - r
  }
}

# Compute the sample covariance of the forecast errors.
error_cov_1 <- cov(errors[, 1, ])
error_cov_2 <- cov(errors[, 2, ])
error_cov_3 <- cov(errors[, 3, ])

error_cov_1

##           [,1]      [,2]
## [1,]  0.4031752 -0.0974633
## [2,] -0.0974633  0.2207392

error_cov_2

##           [,1]      [,2]
## [1,]  0.7412504 -0.1295353
## [2,] -0.1295353  0.2825602

error_cov_3

##           [,1] [,2]
## [1,]      0    0
## [2,]      0    0

```

e. Create a report in pdf format and do the following:

i. Simulate 1000 terms of this time series and plot the result. You may use the series constructed in (d)(i).

```

set.seed(100)

# Model parameters.
phi <- matrix(c(0.1, 0.05, -0.1, 0.3), nrow = 2, byrow = TRUE)
constant <- c(-0.05, 0.1)
Sigma <- matrix(c(0.4, -0.1, -0.1, 0.2), nrow = 2, byrow = TRUE)

```

```

n <- 1000 # Number of terms to simulate

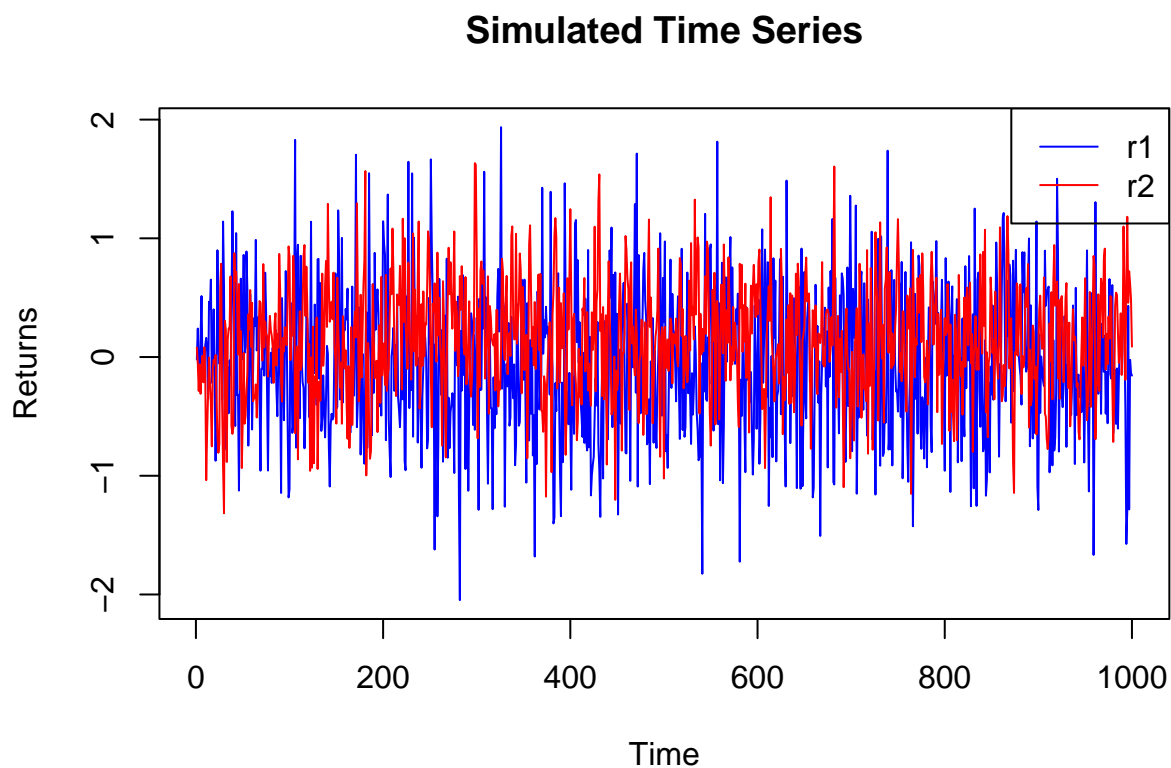
# Simulating the time series.
r <- matrix(nrow = n, ncol = 2)
r[1, ] <- c(-0.02, 0.08) # Initial value

# Simulating the time series
for (i in 2:n) {
  a_t <- mvrnorm(1, mu = c(0, 0), Sigma = Sigma)
  r[i, ] <- constant + phi %*% r[i - 1, ] + a_t
}

# Plotting the result.
df <- data.frame(Time = 1:n, r1 = r[, 1], r2 = r[, 2])

# Plotting the simulated time series.
plot(df$Time, df$r1, type = "l", col = "blue", xlab = "Time", ylab = "Returns", main = "Simulated Time Series")
lines(df$Time, df$r2, type = "l", col = "red")
legend("topright", legend = c("r1", "r2"), col = c("blue", "red"), lty = 1)

```



ii. Using the generated time series, fit a univariate AR(1) model to each return series.

```

library(forecast) # For ARIMA modeling

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

```



```

## as.zoo.data.frame zoo

##
## Attaching package: 'forecast'

## The following object is masked _by_ '.GlobalEnv':
##
## forecast

# Fitting AR(1) model to the first return series (r1).
ar1_model_r1 <- arima(df$r1, order = c(1, 0, 0))

# Fitting AR(1) model to the second return series (r2).
ar1_model_r2 <- arima(df$r2, order = c(1, 0, 0))

# Displaying the model summaries.
summary(ar1_model_r1)

##
## Call:
## arima(x = df$r1, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##      0.0644   -0.0529
## s.e.  0.0315    0.0208
##
## sigma^2 estimated as 0.3801:  log likelihood = -935.26,  aic = 1876.52
##
## Training set error measures:
##              ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 2.333176e-06 0.6165091 0.4885089 125.0663 160.5799 0.7265634
##              ACF1
## Training set 0.002191161

summary(ar1_model_r2)

##
## Call:
## arima(x = df$r2, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##      0.2917    0.1412
## s.e.  0.0302    0.0205
##
## sigma^2 estimated as 0.2104:  log likelihood = -639.59,  aic = 1285.17
##
## Training set error measures:
##              ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -1.499769e-05 0.4586826 0.3625759 19.51479 186.8812 0.7956453
##              ACF1
## Training set 0.004643128

```

iii. Compute the mean of both univariate models. How do these compare to those for the bivariate series?

```

# Compute the means of the univariate AR(1) models.
mean_ar1_r1 <- ar1_model_r1$coef[1] / (1 - ar1_model_r1$coef[2])
mean_ar1_r2 <- ar1_model_r2$coef[1] / (1 - ar1_model_r2$coef[2])

mean_ar1_r1

##          ar1
## 0.06120114
mean_ar1_r2

##          ar1
## 0.3396128
# Compute and display the means of the bivariate series.
mean_bivariate_r1 <- mean(df$r1)
mean_bivariate_r2 <- mean(df$r2)

mean_bivariate_r1

## [1] -0.05285271
mean_bivariate_r2

## [1] 0.1411942

```

For the first return series (r_1), the mean of the univariate AR(1) model is approximately 0.061, whereas the mean of the bivariate series is about -0.053 . This indicates a significant deviation, as the univariate model predicts a positive mean, while the actual mean in the bivariate series is negative.

Similarly, for the second return series (r_2), the univariate AR(1) model yields a mean of approximately 0.340, which is quite different from the mean of the bivariate series, which is around 0.141. Here again, the univariate model predicts a higher mean compared to the actual mean in the bivariate data.

These differences suggest that while the univariate AR(1) models capture some aspects of the data, they might not fully represent the underlying dynamics, especially the central tendencies, of the bivariate series.

iv. Assume $r_{1,0} = -0.02$, $r_{2,0} = 0.08$, $a_{1,0} = -0.08$, and $a_{2,0} = 0.1$. Compute the 1-, 2-, and 3-step ahead forecasts of both of your univariate return series models at the forecast origin $t = 1$. What are the standard deviations of the associated forecast errors? How do these compare to those for the bivariate series? You may approach this problem either analytically or via simulations.

```

# Define initial values.
r1_0 <- -0.02
r2_0 <- 0.08
a1_0 <- -0.08
a2_0 <- 0.1

# Function to forecast using AR(1) model.
forecast_ar1 <- function(model, initial_value, steps) {
  phi <- model$coef[2]
  c <- model$coef[1]
  forecast_values <- numeric(steps)
  forecast_values[1] <- c + phi * initial_value
  for (i in 2:steps) {
    forecast_values[i] <- c + phi * forecast_values[i - 1]
  }
  return(forecast_values)
}

```

```

}

# 1-, 2-, and 3-step ahead forecasts for r1 and r2.
forecast_r1 <- forecast_ar1(ar1_model_r1, r1_0, 3)
forecast_r2 <- forecast_ar1(ar1_model_r2, r2_0, 3)

forecast_r1

## [1] 0.06549362 0.06097423 0.06121313
forecast_r2

## [1] 0.3029563 0.3344370 0.3388820

# Standard deviation of forecast errors for univariate models.
sd_forecast_error_r1 <- sd(ar1_model_r1$residuals)
sd_forecast_error_r2 <- sd(ar1_model_r2$residuals)

# Calculate the standard deviations of the bivariate series.
sd_bivariate_r1 <- sd(df$r1)
sd_bivariate_r2 <- sd(df$r2)

sd_forecast_error_r1

## [1] 0.6168176
sd_forecast_error_r2

## [1] 0.4589121
sd_bivariate_r1

## [1] 0.6181046
sd_bivariate_r2

## [1] 0.4798192

```

The comparison of the standard deviations of the forecast errors from the univariate AR(1) models with the standard deviations of the bivariate series shows a notable level of similarity, indicating that the univariate models are reasonably effective in capturing the variability of the data. Specifically, for the first return series (r_1), the standard deviation of the forecast error in the univariate model is approximately 0.617, which is almost identical to the standard deviation of the bivariate series at about 0.618. This close match suggests that the univariate model for r_1 is quite effective in capturing the volatility inherent in the bivariate series.

Similarly, for the second return series (r_2), the standard deviation of the forecast error from the univariate model is around 0.459, compared to the bivariate series' standard deviation of approximately 0.480. Although there is a slight difference here, it's relatively small, indicating that the univariate model for r_2 also does a good job of approximating the variability seen in the bivariate data.