Assignment-2

Bidit Sadhukhan

Table of contents

Info

- **Name:** *Bidit Sadhukhan*
- **Reg.No:** *B2230022*
- **University:** *RKMVERI*
- **Subject:** *Econometrics*

Question-01

```r
# Importing the libraries
library(knitr)
library(Metrics)
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
library(vars)
```

```
Loading required package: MASS

Attaching package: 'MASS'

The following object is masked from 'package:dplyr':

    select

Loading required package: strucchange

Loading required package: zoo

Attaching package: 'zoo'

The following objects are masked from 'package:base':

    as.Date, as.Date.numeric
```

```
Loading required package: sandwich

Loading required package: urca

Loading required package: lmtest
```

- Making the Dataframe

```r
# Define the data
time_points <- 1:12
actual_values <- c(3, 7, -4, -6, 1, 9, -3, -7, 1, 9, -3, -7)
forecast_1 <- c(2, 4, -3, -4, 0, 12, -6, -4, 4, 15, -1, -11)
forecast_2 <- c(7, 9, -8, -10, 7, 9, -8, -10, 7, 9, -8, -10)

# Create a dataframe
data <- data.frame(
  time_points,
  actual_values,
  forecast_1,
  forecast_2,row.names = NULL
)

# Print the dataframe
# Convert to kable table
kable(data, caption = "Data Table")
```

Table 1: Data Table

time_points	actual_values	forecast_1	forecast_2
1	3	2	7
2	7	4	9
3	-4	-3	-8
4	-6	-4	-10
5	1	0	7
6	9	12	9
7	-3	-6	-8
8	-7	-4	-10
9	1	4	7
10	9	15	9
11	-3	-1	-8
12	-7	-11	-10

1(a) Which forecasting method is better and why?

```r
# Calculate bias and MSE for forecast 1
forecast_1_bias <- mean(data$actual_values - data$forecast_1)
forecast_1_mse <- mean((data$actual_values - data$forecast_1)^2)

# Calculate bias and MSE for forecast 2
forecast_2_bias <- mean(data$actual_values - data$forecast_2)
forecast_2_mse <- mean((data$actual_values - data$forecast_2)^2)

# Calculate bias and MSE for each forecast
comp <- data.frame(
  Forecast = c("1", "2"),
  Bias = c(forecast_1_bias, forecast_2_bias),
  MSE = c(forecast_1_mse, forecast_2_mse)
)

# Convert to kable table
kable(comp, caption = "Bias and MSE for Forecast 1 and Forecast 2")
```

Table 2: Bias and MSE for Forecast 1 and Forecast 2

Forecast	Bias	MSE
1	-0.6666667	9
2	0.5000000	16

Bias:

- **Forecast 1:** Bias is **negative**, indicating the forecast consistently underestimates the actual values.
- **Forecast 2:** Bias is **positive**, meaning the forecast overestimates the actual values.

In general, a **bias close to zero is desirable**. While a negative bias can be compensated for by adding a constant to the forecast, a positive bias might lead to inaccurate predictions and potential issues depending on the context.

MSE:

- **Forecast 1:** MSE is **9**, indicating a lower average squared error.
- **Forecast 2:** MSE is **16**, showing a higher average squared error.

MSE measures the average squared difference between the actual and predicted values, representing the average prediction error. Therefore, a **lower MSE indicates a better fit to the actual values**.

Decision:

- Therefore, the best forecast depends on specific priorities:
  - **If minimizing bias is main concern:** Choose forecast 2.
  - **If minimizing MSE is main concern:** Choose forecast 1.

1(b) Compute all the forecast errors for both the methods.

- **MAE:** Mean Absolute Error
- **MSE:** Mean Squared Error
- **RMSE:** Root Mean Squared Error
- **MAPE:** Mean Absolute Percentage Error
- **SMAPE:** Symmetric Mean Absolute Percentage Error

```
# Calculate error measures
error_measures <- data.frame(
  Measure = c("MAE", "MSE", "RMSE", "MAPE", "SMAPE"),
  Forecast1 = c(mae(actual_values, forecast_1),
                 mse(actual_values, forecast_1),
                 rmse(actual_values, forecast_1),
                 mape(actual_values, forecast_1),
                 smape(actual_values, forecast_1)),
  Forecast2 = c(mae(actual_values, forecast_2),
                 mse(actual_values, forecast_2),
                 rmse(actual_values, forecast_2),
                 mape(actual_values, forecast_2),
                 smape(actual_values, forecast_2))
)

# Convert to kable table
kable(error_measures, caption = "Error Measures for Forecast 1 and Forecast 2")
```

Table 3: Error Measures for Forecast 1 and Forecast 2

Measure	Forecast1	Forecast2
MAE	2.6666667	3.5000000
MSE	9.0000000	16.0000000
RMSE	3.0000000	4.0000000
MAPE	0.7509921	1.6230159
SMAPE	0.6894541	0.6450609

1(c) Compute the forecast errors only using the last four observations.

```
# Define the last four observations
last_four_time_points <- tail(time_points, 4)
last_four_actual_values <- tail(actual_values, 4)
last_four_forecast_1_values <- tail(forecast_1, 4)
last_four_forecast_2_values <- tail(forecast_2, 4)

# Create a new data frame with forecast errors
last_four_errors_data <- data.frame(
  time_points = last_four_time_points,
  actual_values = last_four_actual_values,
  forecast_1 = last_four_forecast_1_values,
  forecast_2 = last_four_forecast_2_values,
  row.names = NULL
)

# Print the data frame
# Convert to kable table
kable(last_four_errors_data, caption = "Data for Forecast 1 and Forecast 2 (Last Four Obse
```

Table 4: Data for Forecast 1 and Forecast 2 (Last Four Observations)

time_points	actual_values	forecast_1	forecast_2
9	1	4	7
10	9	15	9
11	-3	-1	-8
12	-7	-11	-10

```
# Calculate error measures for last four observations
error_measures_last_four <- data.frame(
  Measure = c("MAE", "MSE", "RMSE", "MAPE", "SMAPE"),
  Forecast1 = c(
    mae(last_four_actual_values, last_four_forecast_1_values),
    mse(last_four_actual_values, last_four_forecast_1_values),
    rmse(last_four_actual_values, last_four_forecast_1_values),
    mape(last_four_actual_values, last_four_forecast_1_values),
    smape(last_four_actual_values, last_four_forecast_1_values)
  ),
  Forecast2 = c(
    mae(last_four_actual_values, last_four_forecast_2_values),
    mse(last_four_actual_values, last_four_forecast_2_values),
    rmse(last_four_actual_values, last_four_forecast_2_values),
    mape(last_four_actual_values, last_four_forecast_2_values),
    smape(last_four_actual_values, last_four_forecast_2_values)
  )
)

# Calculate bias and MSE
bias_forecast1_last_four <- abs(mean(last_four_actual_values - last_four_forecast_1_values
mse_forecast1_last_four <- mean((last_four_actual_values - last_four_forecast_1_values)^2)
bias_forecast2_last_four <- abs(mean(last_four_actual_values - last_four_forecast_2_values
mse_forecast2_last_four <- mean((last_four_actual_values - last_four_forecast_2_values)^2)

# Add bias and MSE to error_measures_last_four data frame
error_measures_last_four <- rbind(
  error_measures_last_four,
  data.frame(
    Measure = c("Bias", "MSE"),
    Forecast1 = c(bias_forecast1_last_four, mse_forecast1_last_four),
    Forecast2 = c(bias_forecast2_last_four, mse_forecast2_last_four)
  )
)

# Print the error measures
kable(error_measures_last_four, caption = "Error Measures for Forecast 1 and Forecast 2 (L
```

Table 5: Error Measures for Forecast 1 and Forecast 2 (Last Four Observations)

Measure	Forecast1	Forecast2
MAE	3.7500000	3.500000
MSE	16.2500000	17.500000
RMSE	4.0311289	4.183300
MAPE	1.2261905	2.023810
SMAPE	0.7861111	0.690508
Bias	1.7500000	0.500000
MSE	16.2500000	17.500000

1(d) Forecast the values for t = 1, 2, · · · , 12 using a linear trend model, and compare it with the previous two methods.

```
# Define time points
time_points <- 1:12

# Fit linear trend model
lm_fit <- lm(actual_values ~ time_points)

# Forecast values
forecast_linear <- predict(lm_fit, newdata = data.frame(time_points = 1:12))

# Compare forecasts
comparison_data <- data.frame(
  Time = 1:12,
  Actual = actual_values,
  Forecast1 = forecast_1,
  Forecast2 = forecast_2,
  ForecastLinear = forecast_linear
)
kable(comparison_data, caption = "Data with linear forecast")
```

Table 6: Data with linear forecast

Time	Actual	Forecast1	Forecast2	ForecastLinear
1	3	2	7	2.1153846
2	7	4	9	1.7307692
3	-4	-3	-8	1.3461538
4	-6	-4	-10	0.9615385

Time	Actual	Forecast1	Forecast2	ForecastLinear
5	1	0	7	0.5769231
6	9	12	9	0.1923077
7	-3	-6	-8	-0.1923077
8	-7	-4	-10	-0.5769231
9	1	4	7	-0.9615385
10	9	15	9	-1.3461538
11	-3	-1	-8	-1.7307692
12	-7	-11	-10	-2.1153846

```
# Calculate bias
bias_forecast1 <- mean(comparison_data$Actual - comparison_data$Forecast1)
bias_forecast2 <- mean(comparison_data$Actual - comparison_data$Forecast2)
bias_linear <- mean(comparison_data$Actual - comparison_data$ForecastLinear)

# Calculate variance
variance_forecast1 <- var(comparison_data$Actual - comparison_data$Forecast1)
variance_forecast2 <- var(comparison_data$Actual - comparison_data$Forecast2)
variance_linear <- var(comparison_data$Actual - comparison_data$ForecastLinear)

# Create data frame with bias and variance
error_measures <- data.frame(
  Measure = c("Bias", "Variance"),
  Forecast1 = c(bias_forecast1, variance_forecast1),
  Forecast2 = c(bias_forecast2, variance_forecast2),
  Linear = c(bias_linear, variance_linear)
)

# Print kable table
kable(error_measures, caption = "Bias and Variance of Forecast Methods")
```

Table 7: Bias and Variance of Forecast Methods

Measure	Forecast1	Forecast2	Linear
Bias	-0.6666667	0.50000	0.00000
Variance	9.3333333	17.18182	33.53147

Linear trend model has higher MSE than other 2 forecasts but it has bias zero.

1.(e) Suppose you want to use simple exponential smoothing for forecasting. What will be the optimal value of smoothing parameter if assumes only two values 0.1 or 0.3.

```
simple_exponential_smoothing <- function(series, alpha) {
n <- length(series)
forecast_values <- numeric(n)
forecast_values[1] <- series[1]
for (i in 2:n) {
forecast_values[i] <- alpha * series[i - 1] + (1 - alpha) * forecast_values[i - 1]
}
return(forecast_values)
}
```

```
# Define data and smoothing parameters
data <- actual_values
alpha_values <- c(0.1, 0.3)

# Initialize empty lists for storing errors
mse_errors <- list()

# Loop through each smoothing parameter
for (alpha in alpha_values) {
  # Initialize forecast and error variables
  forecast <- c(data[1])
  errors <- vector("numeric", length(data))

  # Calculate forecasts for remaining data points
  for (i in 2:length(data)) {
    forecast[i] <- alpha * data[i-1] + (1 - alpha) * forecast[i-1]
    errors[i] <- data[i] - forecast[i]
  }

  # Calculate mean squared error
  mse_error <- mean(errors^2)

  # Store MSE error
  mse_errors[[paste0("alpha = ", alpha)]] <- mse_error
}

# Find the alpha with the minimum MSE
```

```r
min_mse_index <- which.min(unlist(mse_errors))
optimal_alpha <- names(mse_errors)[min_mse_index]
```

- The optimal Alpha value and the MSE value of the optimal alpha.

```r
# Print the results
print(optimal_alpha)
```

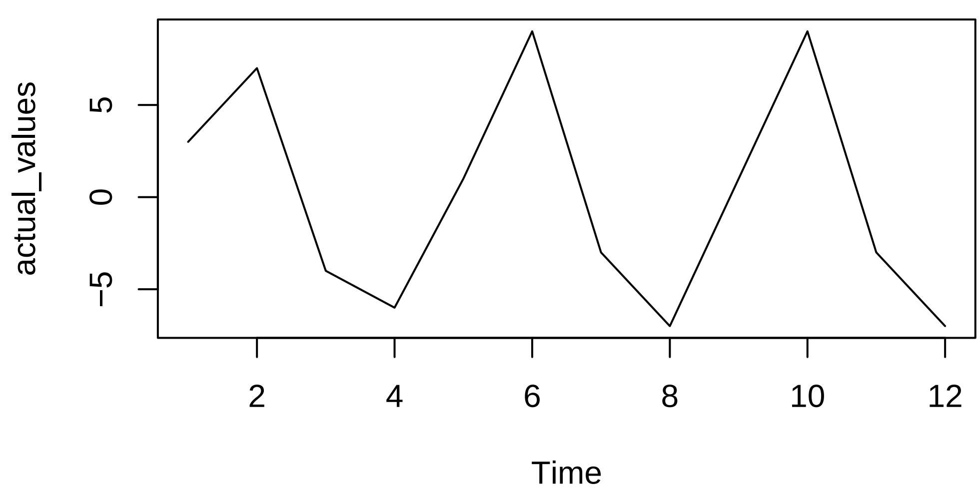
```
[1] "alpha = 0.1"
```

```r
print(mse_errors[[optimal_alpha]])
```

```
[1] 38.71599
```

1(f) What will be the value of S24, where St denotes the seasonality of yt at t?

```r
ts.plot(actual_values)
```



```r
# Define the period
period <- 4

detrendised_values=actual_values-forecast_linear
# seasonality
seasonal_component <- function(series, d, t) {
s <- numeric(d)
for (i in 1:d) {
s[i] <- mean(series[seq(i,length(series),d)])
}
if (t%%d != 0) {
return(s[t%%d])
}
else return(s[d])
}

# Calculate S24
S24 <- seasonal_component(detrendised_values,4,24)

# Print S24
print(paste("S24:", S24))
```

```
[1] "S24: -6.08974358974359"
```

1(g) Forecast the values t = 1, 2, · · · , 12 using an AR(4) model, and compare it with Method 1 and Method 2.

```r
# Detrending and deseasonalizing
deseasonalised_detrendised_values <- detrendised_values - sapply(1:4, function(x) seasonal

# Fit AR(4) model
ar_4_model <- arima(deseasonalised_detrendised_values, order = c(4, 0, 0))

# Print AR(4) model coefficients
print(ar_4_model$coef)
```

```
      ar1        ar2        ar3        ar4  intercept
1.0835933 -0.8155993  0.5530546 -0.3969140  0.1818662
```

```r
# Forecast AR(4) values
estimated_ar_4_values <- numeric(12)
estimated_ar_4_values[1:4] <- deseasonalised_detrendised_values[1:4]

for (i in 5:12) {
  estimated_ar_4_values[i] <- sum(c(ar_4_model$coef) * c(rev(estimated_ar_4_values[seq(i -
}

# Add trend and seasonality to obtain final forecasts
final_ar_4_values <- estimated_ar_4_values + forecast_linear + sapply(1:4, function(x) sea

# Print final AR(4) values
print(final_ar_4_values)
```

```
        1         2         3         4         5         6         7         8
 3.000000  7.000000 -4.000000 -6.000000  1.195522  8.636005 -2.046133 -5.251398
        9        10        11        12
 1.148205  7.519468 -4.464859 -8.170914
```

```r
# Create a data frame with actual and forecast values
comparison_data <- data.frame(
  Time = 1:12,
  Actual = actual_values,
  Forecast1 = forecast_1,
  Forecast2 = forecast_2,
  ForecastAR4 = final_ar_4_values
)

# Print kable table
kable(comparison_data, caption = "Actual vs. AR(4) Forecast Values")
```

Table 8: Actual vs. AR(4) Forecast Values

Time	Actual	Forecast1	Forecast2	ForecastAR4
1	3	2	7	3.000000
2	7	4	9	7.000000
3	-4	-3	-8	-4.000000
4	-6	-4	-10	-6.000000
5	1	0	7	1.195522
6	9	12	9	8.636005

Time	Actual	Forecast1	Forecast2	ForecastAR4
7	-3	-6	-8	-2.046133
8	-7	-4	-10	-5.251398
9	1	4	7	1.148205
10	9	15	9	7.519468
11	-3	-1	-8	-4.464859
12	-7	-11	-10	-8.170914

```r
# Calculate MSE
mse <- mse(actual_values, final_ar_4_values)
print(paste0("MSE for AR(4) model:", mse))
```

```
[1] "MSE for AR(4) model:0.822415095068545"
```

```r
# Calculate bias
bias_ar4 <- mean(comparison_data$Actual - comparison_data$ForecastAR4)

# Calculate variance
variance_ar4 <- var(comparison_data$ForecastAR4 - comparison_data$Actual)

# Create data frame with bias and variance
error_measures <- data.frame(
  Measure = c("Bias", "Variance"),
  Forecast1 = c(forecast_1_bias, forecast_1_mse),
  Forecast2 = c(forecast_2_bias, forecast_2_mse),
  AR4 = c(bias_ar4, variance_ar4)
)

# Print kable table
kable(error_measures, caption = "Bias and Variance of Forecast Methods")
```

Table 9: Bias and Variance of Forecast Methods

Measure	Forecast1	Forecast2	AR4
Bias	-0.6666667	0.5	0.1195087
Variance	9.0000000	16.0	0.8815994

AR(4) is giving lower MSE and bias than the other methods , so AR(4) is better.

Q.2) $y_t = \beta_1 \sin(2\pi\omega t) + \beta_2 \cos(2\pi\omega t) + \varepsilon_t + 0.5\,\varepsilon_{t-1}$

$\varepsilon_t \sim WN(0, \sigma^2)$

ω, β_1, β_2 are constants.

Mean:

$$E(y_t) = E[\beta_1 \sin(2\pi\omega t) + \beta_2 \cos(2\pi\omega t) + \varepsilon_t + 0.5\,\varepsilon_{t-1}]$$

$$= \beta_1 \sin(2\pi\omega t) + \beta_2 \cos(2\pi\omega t) + E(\varepsilon_t + 0.5\,\varepsilon_{t-1})$$

[$\because$ Due to linearity of Expectation. and the first two terms are constant]

$$= \beta_1 \sin(2\pi\omega t) + \beta_2 \cos(2\pi\omega t) + E(\varepsilon_t) + 0.5\,E(\varepsilon_{t-1})$$

Again, $\varepsilon_t \sim WN(0, \sigma^2)$

$\therefore E(\varepsilon_t) = E(\varepsilon_{t-1}) = 0$.

So $E(y_t) = \beta_1 \sin(2\pi\omega t) + \beta_2 \cos(2\pi\omega t)$

Variance:-

$$V(y_t) = V[\underbrace{\beta_1 \sin(2\pi\omega t) + \beta_2 \cos(2\pi\omega t)}_{\text{non stochastic}} + \varepsilon_t + 0.5\,\varepsilon_{t-1}]$$

$$= V(\varepsilon_t + 0.5\,\varepsilon_{t-1}) \quad [\because Var(a+X) = Var(X)]$$

$$= V(\varepsilon_t) + 0.5^2\,V(\varepsilon_{t-1}) + 2 \times 0.5 \times Cov(\varepsilon_t, \varepsilon_{t-1})$$

$$= \sigma^2 + 0.25\sigma^2 = 1.25\,\sigma^2 \quad [\because \varepsilon_t \sim WN(0,\sigma^2) \Rightarrow Cov(\varepsilon_t, \varepsilon_{t-1}) = 0]$$

$$\gamma(1) = Cov(y_{t+1}, y_t)$$

$$= Cov(a + \varepsilon_{t+1} + 0.5\,\varepsilon_t,\; b + \varepsilon_t + 0.5\,\varepsilon_{t-1}) \quad [a, b \text{ constants}]$$

$$= Cov(a,b) + Cov(a, \varepsilon_t + 0.5\,\varepsilon_{t-1}) + Cov(\varepsilon_{t+1} + 0.5\,\varepsilon_t, b)$$

$$+ Cov(\varepsilon_{t+1} + 0.5\,\varepsilon_t,\; \varepsilon_t + 0.5\,\varepsilon_{t-1})$$

[$\because$ By the distributive property of Covariance]

$= Cov(\varepsilon_{t+1} + 0.5\varepsilon_t, \varepsilon_t + 0.5\varepsilon_{t-1})$ [$\because$ $Cov(a, x) = 0$ as a is const]

$= Cov(\varepsilon_{t+1}, \varepsilon_t) + 0.5\, Cov(\varepsilon_t, \varepsilon_t) + 0.5\, Cov(\varepsilon_{t+1}, \varepsilon_{t-1})$
$+ 0.5^2\, Cov(\varepsilon_t, \varepsilon_{t-1})$

$= 0.5\,\sigma^2$ [$\because$ $Cov(\varepsilon_t, \varepsilon_{t+h}) = 0 \ \forall h \in \mathbb{Z}$]

$$\rho(2) = \frac{\gamma(2)}{\gamma(0)}.$$

$\gamma(2) = Cov(Y_{t+2}, Y_t).$

$= Cov(c + \varepsilon_{t+2} + 0.5\varepsilon_{t+1}, b + \varepsilon_t + 0.5\varepsilon_{t-1})$

$= Cov(\varepsilon_{t+2} + 0.5\varepsilon_{t+1}, \varepsilon_t + 0.5\varepsilon_{t-1})$

$= Cov(\varepsilon_{t+2}, \varepsilon_t) + 0.5\, Cov(\varepsilon_{t+1}, \varepsilon_t) + 0.5\, Cov(\varepsilon_{t+2}, \varepsilon_{t-1})$
$+ 0.5^2\, Cov(\varepsilon_{t+1}, \varepsilon_{t-1}).$

$= 0.$

So $\rho(2) = \frac{0}{\gamma(0)} = 0$ [$\gamma(0) = Var(Y_t) \neq 0$]

3) x_t, y_t are two stationary process.

$z_t = x_t + y_t.$

(i) $E(Z_t) = \cancel{E(X_t + X_t)}\ E(X_t + Y_t)$
$= E(X_t) + E(Y_t)$

[independent of t as X_t, Y_t stationary]

$\odot$ $E(Z_t^2) = E[(X_t + Y_t)^2] = E[X_t^2] + E[Y_t^2] + 2E[X_t, Y_t]$

If $E[X_t, Y_t] < \infty$

then $E[Z_t^2] < \infty$

$X_t ; Y_t$ are stationary process.

$\odot$ $Cov(Z_{t+h}, Z_t) = Cov(x_{t+h} + y_{t+h}, x_t + y_t)$

$= Cov(x_{t+h}, x_t) + Cov(y_{t+h}, x_t) + Cov(x_{t+h}, y_t)$

$+ Cov(y_{t+h}, y_t)$.

It is independent of t if $Cov(x_{t+h}, y_t)$ and $Cov(y_{t+h}, y_t)$ are independent of t.

If x_t and Y_{t+h} are independent then

$Cov(x_t, y_{t+h}) = 0$.

$\Rightarrow Cov(Z_{t+h}, Z_t)$ is independent of t.

So Z_t will be stationary based on finite $E[X_t, Y_t]$ with t. and independence of $Cov(x_t, Y_{t+h})$ $\forall\, h \in \mathbb{Z}$

If x_t, y_t independent, then Z_t is stationary.