ANL557_ECA_Sallyyeo001_SallyMarcellinaYeo.docx

by SALLY MARCELLINA YEO

Submission date: 16-Apr-2024 08:35PM (UTC+0800)
Submission ID: 2351587217
File name: ANL557_ECA_Sallyyeo001_SallyMarcellinaYeo.docx (1.27M)
Word count: 5505
Character count: 33525

SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

ANL557

Applied Forecasting

END-OF-COURSE ASSIGNMENT

JAN 2024 PRESENTATION

Submission date: 16 Apr 2024

Name: Sally Marcellina Yeo
PI No.: Y2410435

Table of Contents

Question 1
Data Cleaning

The raw data requires following cleaning:

- Reshape data using pivot_longer() the year.
- Reshape data using pivot_wider() the data series.
- Remove ‘X’ from year and convert year to date type.
- Convert all columns to numeric by removing comma.
- Set empty values to NA.
- Sort the year in chronological order.

Code Snippet:

```
# Reshape the data by pivot_longer() & pivot_wider()
long_data <- raw %>%
  pivot_longer(cols = starts_with("X"), names_to = "Year", values_to = "Value")
wide_data <- long_data %>%
  pivot_wider(names_from = Data.Series, values_from = Value)

# Remove 'X' from the Year column
wide_data$Year <- sub("X", "", wide_data$Year)

# Convert all columns to numeric
wide_data <- wide_data %>%
  mutate_all(function(x) {
    x <- gsub(",", "", x)   # to remove comma
    parse_number(x, na = "na")}) # set empty values to NA

# Convert year to date type
wide_data$Year <- as.Date(paste(wide_data$Year, "-01-01", sep=""), format="%Y-%m-%d")

# Ensure the data is in ascending chronological order
wide_data <- wide_data[order(wide_data$Year), ]
```

Preparing Time Series

For this report, I will use the means daily maximum temperature to prepare the time-series from 1960 to 2022 shown in Fig. 0 and forecast the temperature in Singapore for the next 10 years. Maximum temperature is important for understanding heat-related illnesses and mortality, such as heatstroke and dehydration (Lin et al., 2021). By focusing on maximum temperature predictions, health authorities and policymakers can better plan public health interventions and develop heat action plans to protect vulnerable populations, including the elderly and children.

```
temp_max_ts
Time Series:
Start = 1960
End = 2022
Frequency = 1
 [1] 30.9 31.1 30.7 31.1 30.7 31.0 31.0 30.6 30.7
[10] 30.8 30.7 30.6 30.8 30.5 30.3 30.4 30.7 30.9
[19] 31.0 31.0 31.0 31.3 31.4 31.7 30.8 31.3 31.1
[28] 31.5 31.3 31.2 31.8 31.5 31.5 31.3 31.3 31.3
[37] 31.3 32.4 32.1 31.3 31.4 31.4 32.0 31.4 31.7
[46] 31.9 31.5 31.1 31.1 31.7 31.9 31.2 31.2 31.3
[55] 31.6 31.9 32.0 31.1 31.6 32.3 31.7 31.7 31.6
```

Fig. 0. Time Series.

Code Snippet:

```
# Convert 'Air Temperature Means Daily Maximum (Degree Celsius)' to a ts object
temp_max_ts <- ts(wide_data$`Air Temperature Means Daily Maximum (Degree Celsius)`,
                  start = min(year(wide_data$Year)),
                  end = max(year(wide_data$Year)),
                  frequency = 1) # Since the data is yearly, frequency is 1
```

Exploring Time Series

Following are the options to explore the time series:

- Plot trend line across the time series.
- Find out the p-value of the linear regression.
- Analyze the autocorrelation using Auto Correlation Function (ACF).

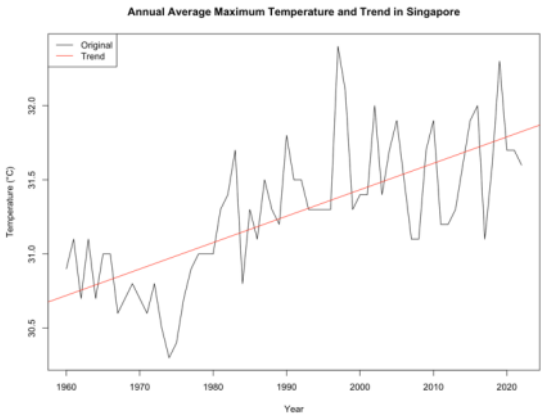


Fig. 1. Trend line.

From the trend line in Fig. 1, there is obvious upward trend in the time series. This is further confirmed using the p-value < 0.05 of linear regression and the multiple R-squared value of 0.494 in Fig. 2, concluding there is a trend in the time series and strong linear relationship. The slope coefficient of 0.01783 suggests that one unit increase in year leads to a very small increase around 0.01783 in the predicted temperature.

```
summary(temp_max_trend_model)
Call:
lm(formula = temp_max_ts ~ time_index)
Residuals:
    Min      1Q  Median      3Q     Max
-0.6699 -0.1587 -0.0513  0.2022  1.0200
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -4.22321    4.60002  -0.918    0.362
time_index   0.01783    0.00231   7.717 1.35e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 0.3335 on 61 degrees of freedom
Multiple R-squared:  0.494,     Adjusted R-squared:  0.4857
F-statistic: 59.55 on 1 and 61 DF,  p-value: 1.351e-10
```

Fig. 2. Linear Regression Summary.

From the Fig. 3 below, the ACF still shows gradual decay and high value even after lag > 3, confirming there is a trend in the time series. There are no seasonal spikes in the ACF, confirming there is no seasonality in the time series. Time series with trend component is also automatically non-stationary.

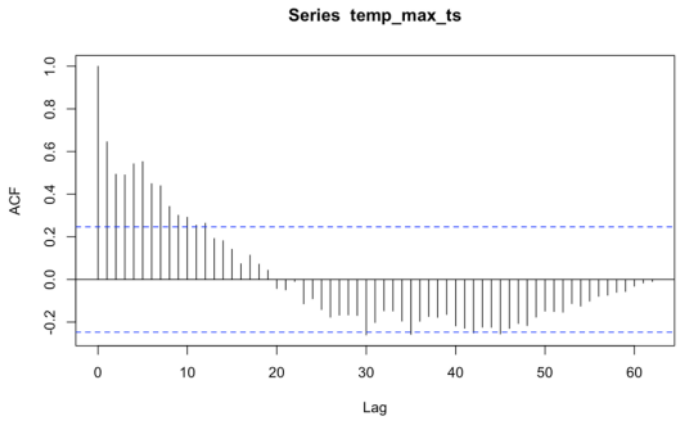


Fig. 3. ACF & PACFTest.

Choosing & Fitting Models

Summary of the time series characteristic:

- Consists of trend.
- Absence of seasonality.
- ACF shows gradual decay.
- Non-stationary.

From these characteristic of the time series, the suitable forecasting models would be the one that focus on the trend and autoregressive nature of the time series. The proposed models: Holt's linear trend model (Double Exponential Smoothing), Linear Regression and ARIMA.

The Holt's linear trend model and Linear Regression can capture linear trends in the data. To implement the ARIMA model, the time series must be transformed into stationary using differencing and log transformation. In the subsequent part of the report, the proposed forecasting models fitting, and evaluation will be discussed using common accuracy metrics such as Mean Absolute Percentage Error (MAPE), Mean Absolute Deviance (MAD), Mean Square Deviance (MSD) and Akaike Information Criterion (AIC).

Holt's Linear Trend Method

The forecasted values using Holt's Linear Trend method is shown in Fig. 4 and 5. The steady increase in the point forecasts from 2023 to 2032 suggests that the model has detected a linear trend in the historical data. To evaluate its accuracy, MAPE, MAD, MSD and AIC will be analyzed in Fig. 7.

```
> print(temp_max_ts.desm)
     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2023       31.74058 31.30797 32.17320 31.07896 32.40221
2024       31.75364 31.30641 32.20087 31.06965 32.43762
2025       31.76669 31.30529 32.22809 31.06104 32.47234
2026       31.77975 31.30460 32.25491 31.05306 32.50644
2027       31.79281 31.30427 32.28134 31.04566 32.53995
2028       31.80586 31.30430 32.30743 31.03879 32.57294
2029       31.81892 31.30464 32.33319 31.03240 32.60544
2030       31.83198 31.30529 32.35866 31.02647 32.63748
2031       31.84503 31.30621 32.38386 31.02097 32.66909
2032       31.85809 31.30738 32.40879 31.01586 32.70032
```

Fig. 4. Holt's linear trend forecast values.

```
> print(temp_max_ts.desm_fc)
Time Series:
Start = 1960
End = 2032
Frequency = 1
 [1] 30.96350 30.95993 31.00972 30.94160 30.99618 30.93160
 [7] 30.96257 30.98542 30.89743 30.85868 30.85628 30.82829
[13] 30.78141 30.79923 30.73373 30.63294 30.58474 30.62781
[19] 30.71203 30.80041 30.86565 30.91381 31.02800 31.13852
[25] 31.29874 31.18107 31.22528 31.20548 31.29573 31.30991
[31] 31.29416 31.43982 31.46870 31.49001 31.45331 31.42621
[37] 31.40620 31.39142 31.66888 31.79506 31.67847 31.61861
[43] 31.57442 31.69909 31.63382 31.66428 31.73919 31.68963
[49] 31.54816 31.44373 31.52393 31.63556 31.53445 31.45980
[55] 31.43091 31.48822 31.60917 31.72467 31.57399 31.59382
[61] 31.79198 31.78096 31.77281 31.74058 31.75364 31.76669
[67] 31.77975 31.79281 31.80586 31.81892 31.83198 31.84503
[73] 31.85809
```

Fig. 5. Holt's linear trend time series.

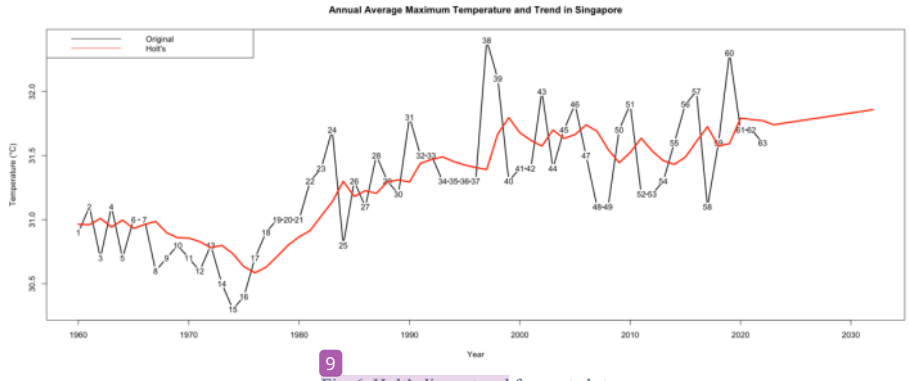


Fig. 6. Holt's linear trend forecast plot.

Code Snippet:

```
# Set forecast length to 10 yrs
fc_length = 10

# Apply Holt's linear trend method
temp_max_ts.desm <- holt(temp_max_ts, h = fc_length)

# Create time series combining the fitted value of a model with the forecasted values
temp_max_ts.desm_fc <- ts(c(temp_max_ts.desm$fitted, temp_max_ts.desm$mean), start = c(1960, 1),
frequency = 1)
print(temp_max_ts.desm_fc)

# Get the extended time periods
x_val <- time(temp_max_ts.desm_fc)

# Append the exact number of NA as the forecast length
y_val <- c(wide_data$`Air Temperature Means Daily Maximum (Degree Celsius)`, rep(NA, fc_length))

# Plot using the extended series
plot(x = x_val, y = y_val, type = "l", xlab = "Year", ylab = "Temperature (°C)", lwd = 2,
    main = "Annual Average Maximum Temperature and Trend in Singapore")

# Overlay the double exponential smoothing to the plot
lines(x = x_val, y = temp_max_ts.desm_fc, col = "red", lwd = 3, type = "l")
legend("topleft", legend = c("Original", "Holt's"), col = c("black", "red"), lty = 1)
```

Holt's Linear Trend Method Evaluation

```
> # Print the metrics
> print(paste("MAPE_holt:", MAPE_holt))
[1] "MAPE_holt: 0.845606672437578"
> print(paste("MAD_holt:", MAD_holt))
[1] "MAD_holt: 0.265236023309835"
> print(paste("MSD_holt:", MSD_holt))
[1] "MSD_holt: 0.10671831495861"
> print(paste("AIC_holt:", AIC_holt))
[1] "AIC_holt: 130.051051125273"
```

Fig. 7. Accuracy Metric results.

A MAPE of 0.85% < 1% is exceptionally low, suggesting that, on average, the model's forecasts are within 1% of the actual temperatures. This suggests a high level of accuracy in the model's predictions. A MAD of approximately 0.265 means that the model's forecasts deviate from the actual temperatures by an average of 0.265 degrees. This is a relatively small error, indicating good model accuracy, especially in the context of weather forecasting where temperatures can vary widely. An MSD of approximately 0.107 suggests that the model's errors, on average, are not large, contributing to an overall effective model performance. AIC is a measure used to compare models, with a lower AIC indicating a better fit to the data, considering the complexity of the model. The AIC value will be compared at the Models Evaluation part of the report.

Code snippet for calculating the accuracy metric:

```
# Compute the Mean Absolute Percentage Error (MAPE)
MAPE_holt <- mean(abs(temp_max_ts.desm$residuals) / wide_data$`Air Temperature Means Daily Maximum (Degree Celsius)`) * 100
# Compute the Mean Absolute Deviance (or Error) (MAD)
MAD_holt <- mean(abs(temp_max_ts.desm$residuals))
# Compute the Mean Square Deviance (or Error) (MSD)
MSD_holt <- mean(temp_max_ts.desm$residuals ^ 2)
# Extract the Akaike Information Criterion (AIC)
AIC_holt <- temp_max_ts.desm$model$aic
```

Linear Regression

Following are the forecasted plot and values using Linear Regression model as shown in Fig. 8 and Fig. 9. The model forecasts gradual increase in the maximum temperature every year. This increment reflects the linear trend identified by the regression, suggesting a consistent rise in maximum temperatures over time. The distribution and autocorrelation of the residuals will be analyzed to evaluate the model's accuracy using Anova test, QQ plot, Shapiro test and Durbin Watson test. The test results are shown in Fig. 10, 11, 12, 13 respectively.

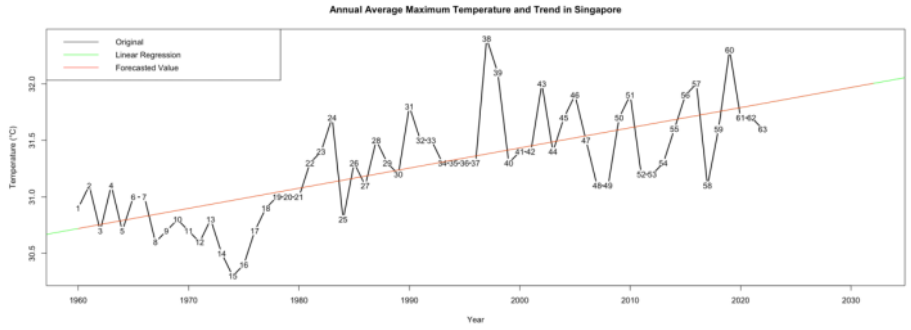


Fig. 8. Linear Regression forecast plot.

```
23 nt(temp_max_ts.linear_fc)
Time Series:
Start = 1960
End = 2032
Frequency = 1
      1       2       3       4       5       6       7       8       9
30.72034 30.73817 30.75599 30.77382 30.79165 30.80948 30.82731 30.84514 30.86296
     10      11      12      13      14      15      16      17      18
30.88079 30.89862 30.91645 30.93428 30.95211 30.96993 30.98776 31.00559 31.02342
     19      20      21      22      23      24      25      26      27
31.04125 31.05908 31.07690 31.09473 31.11256 31.13039 31.14822 31.16605 31.18387
     28      29      30      31      32      33      34      35      36
31.20170 31.21953 31.23736 31.25519 31.27302 31.29084 31.30867 31.32650 31.34433
     37      38      39      40      41      42      43      44      45
31.36216 31.37999 31.39781 31.41564 31.43347 31.45130 31.46913 31.48696 31.50478
     46      47      48      49      50      51      52      53      54
31.52261 31.54044 31.55827 31.57610 31.59393 31.61175 31.62958 31.64741 31.66524
     55      56      57      58      59      60      61      62      63
31.68307 31.70090 31.71872 31.73655 31.75438 31.77221 31.79004 31.80787 31.82569
     64      65      66      67      68      69      70      71      72
31.84352 31.86135 31.87918 31.89701 31.91484 31.93266 31.95049 31.96832 31.98615
     73
32.00398
```

Fig. 9 Linear Regression forecasted values.

Code Snippet:

```
# Create a data frame with years and temperature values
years <- seq(1960, 2022)  # Sequence from start year to end year
temp_max_df <- data.frame(year = years, temp = temp_max_ts)

# Fitting a linear model
linear_model <- lm(temp ~ year, data = temp_max_df)
summary(linear_model)  # View model summary for details

# Create a new data frame that includes both historical and future years
all_years <- data.frame(year = c(1960:2022, 2023:2032))

# Combining fitted with forecasted values
temp_max_df.linear_fc <- predict(linear_model, newdata = all_years)
all_years$temp <- temp_max_df.linear_fc

# Convert to time series object
temp_max_ts.linear_fc <- ts(temp_max_df.linear_fc, start = 1960, end = 2032, frequency = 1)
# Print the forecasted values
print(temp_max_ts.linear_fc)

# Get the extended time periods & append the exact number of NA as the forecast length
x_val <- time(temp_max_ts.linear_fc)
y_val <- c(wide_data$`Air Temperature Means Daily Maximum (Degree Celsius)`, rep(NA, fc_length))

# Plot using the extended series
plot(x = x_val, y = y_val, type = "l", xlab = "Year", ylab = "Temperature (°C)", lwd = 2,
     main = "Annual Average Maximum Temperature and Trend in Singapore")

#abline(linear_model, col = "green") # Plot linear model fit

# Plot forecasted values
lines(x = x_val, y = temp_max_ts.linear_fc, col = "red", lty = 1, type ="l")
legend("topleft", legend = c("Original", "Linear Regression", "Forecasted Value"), col = c("black", "green", "red"), lty
= 1)
```

Linear Regression Evaluation

```
> anova(linear_model)
Analysis of Variance Table

Response: temp
          Df Sum Sq Mean Sq F value    Pr(>F)
year       1 6.6214  6.6214   59.55 1.351e-10 ***
Residuals 61 6.7827  0.1112
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Fig. 10. Anova Test of Linear Regression forecasted values.

From the Anova test result in Fig. 10 above, the p-value is less than 0.001 indicating that there is statistically significant effect of “year” and “temp”. The F value is large at 59.55, suggesting that “year” is a strong predictor of temperature. The W value of 0.97428 in the Shapiro test result in Fig. 12 below suggests that the residuals of the linear model are quite close to being normally distributed. The p-value is 0.2093, which is greater than 0.05. This means that there is not enough statistical evidence to reject the null hypothesis, and it can be assumed that the residuals of the linear regression model are normally distributed. That means the assumption of normality for the residuals has not been violated.

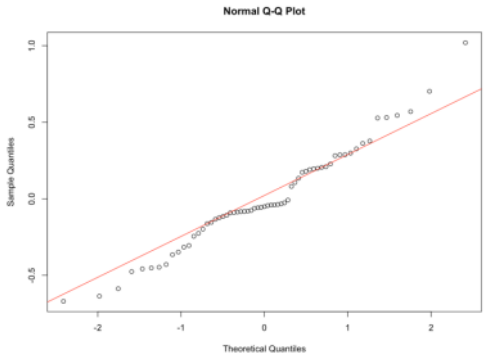


Fig. 11. QQ Plot for Linear Regression model residuals and the QQ Line.

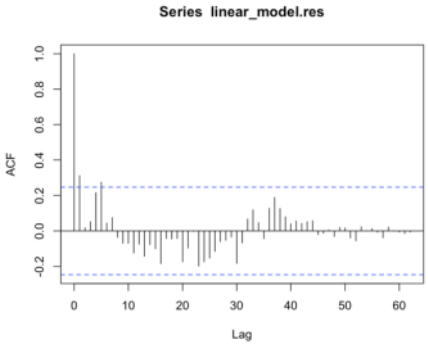
```
> shapiro.test(linear_model.res)

        Shapiro-Wilk normality test

data:  linear_model.res
W = 0.97428, p-value = 0.2093
```

Fig. 12. Shapiro test result.

The ACF cut-off pattern in Fig. 13 after the first lag suggests that the residuals are largely independent at time lags beyond the first. The Durbin Watson Test in Fig.13 shows a D-W statistics of 1.362993 < 2 but not close to 0. This means there is certain degree of positive correlation in the residuals. The p-value = 0.004 means the null hypothesis of no autocorrelation should be rejected, and there is significant positive autocorrelation in the residuals. Linear regression also assumes that the residuals are independent of each other. The autocorrelation violates this key assumption, suggesting that forecasting with linear regression model must be done in cautious as it can cause biased estimates of regression coefficients, making the model unreliable for understanding the impact of an independent variable on the dependent variable.




```
> durbinWatsonTest(linear_model)
 lag Autocorrelation D-W Statistic p-value
   1       0.312369       1.362993   0.004
 Alternative hypothesis: rho != 0
```

Fig. 13. ACF of residuals and Durbin Watson test result.

ARIMA

Since the time series is identified as non-stationary, we need to perform differencing to make it stationary. The cut-off pattern after first lag of ACF plot in Fig. 14 confirms that the time series is stationary after differencing. However, Fig. 15 shows that the variance increases over time, therefore we need to try log transformation.

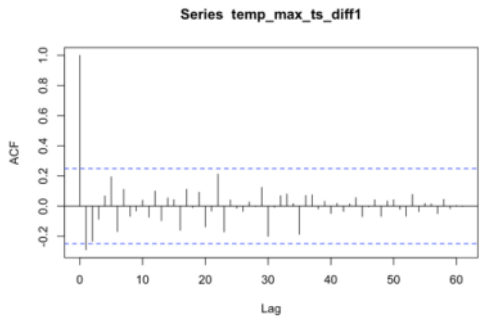


Fig. 14. ACF after differencing.

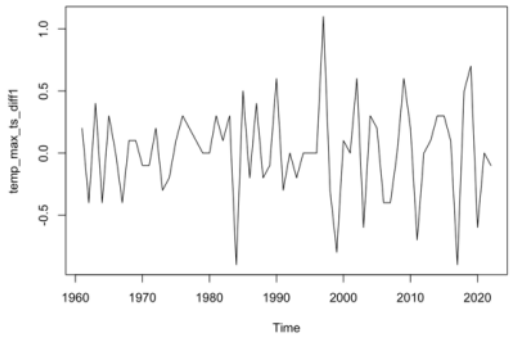


Fig. 15. Time series after differencing.

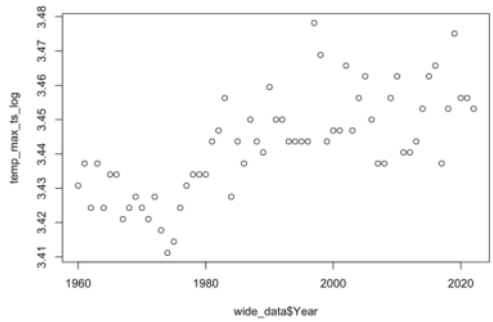


Fig. 16. Time series after log transformation.

After log transformation, the variance and means are now showing an increase over time in Fig. 16. Therefore, we still need to perform differencing on the log transformed time series.

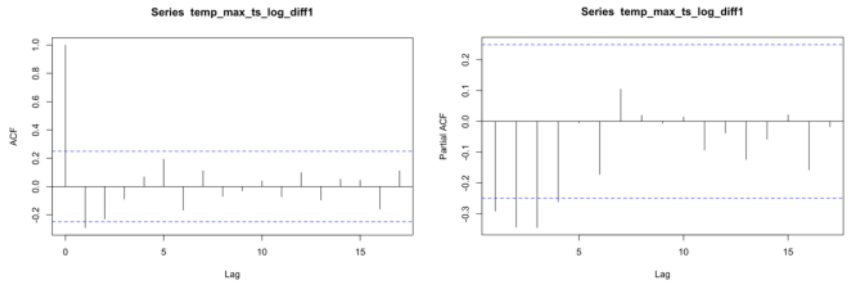


Fig. 17. ACF and PACF after differencing the log transformed.

After performing log transformation and differencing, Fig. 17 shows the final ACF is a cut-off pattern and PACF is exponential decay pattern. This conclude that we need to use Moving Average (MA).

Moving Average (MA) of ARIMA Evaluation

Following is the comparison of the p-value test between MA(1) and MA(2). The p-value of MA(1) shows a more significant p-value than MA(2). Therefore MA(1) is more suitable. The ACF of the residuals in Fig. 18 also shows cut off after lag 0, suggesting the model is safe. Additionally, the Shapiro test of p-value = 0.4378 > 0.05, W value of 0.98 close to 1 and QQ plot in Fig. 19 also confirms that the model does not violate the normal distribution assumption of the residuals.

Result of MA(1)

```
> pt(ttest_ma1, df = length(ma_process.ma1) - length(param_ma1), lower.tail
= FALSE) * 2
       ma1    intercept
5.522074e-06 2.326294e-01
```

Result of MA(2)

```
> pt(ttest_ma2, df = length(ma_process.ma2) - length(param_ma2), lower.tail =
FALSE) * 2
       ma1         ma2    intercept
0.0009131808 0.1291124905 0.0854869038
```

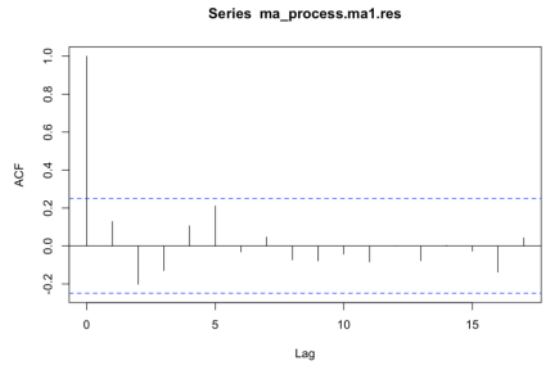


Fig. 18. ACF of residuals in ARIMA MA(1).

```
> shapiro.test(ma_process.ma1.res)

        Shapiro-Wilk normality test

data:  ma_process.ma1.res
W = 0.98073, p-value = 0.4378
```

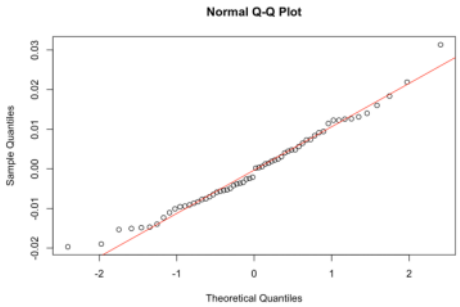


Fig. 19. Shapiro Test & QQ plot of residuals of ARIMA MA(1).

After getting the forecasted value, the values need to be reversed to its original scale. Following Fig. 20 are the final forecasted values in original scale, highlighted in red. It is very similar to the mean of the historical data which is 31.27302. Finally, Fig. 21 shows the forecast plot with rising temperature represented by red line.

```
> print(ma1_original_scale_fc)
Time Series:
Start = 1960
End = 2032
Frequency = 1
 [1] 30.90000 31.10000 30.70000 31.10000 30.70000 31.00000 31.00000 30.60000 30.70000 30.80000 30.70000
30.60000
[13] 30.80000 30.50000 30.30000 30.40000 30.70000 30.90000 31.00000 31.00000 31.00000 31.30000 31.40000
31.70000
[25] 30.80000 31.30000 31.10000 31.50000 31.30000 31.20000 31.80000 31.50000 31.50000 31.30000 31.30000
31.30000
[37] 31.30000 32.40000 32.10000 31.30000 31.40000 31.40000 32.00000 31.40000 31.70000 31.90000 31.50000
31.10000
[49] 31.10000 31.70000 31.90000 31.20000 31.20000 31.30000 31.60000 31.90000 32.00000 31.10000 31.60000
32.30000
[61] 31.70000 31.70000 31.60000 31.74560 31.76005 31.77451 31.78898 31.80345 31.81793 31.83242 31.84691
31.86142
[73] 31.87592
> mean(temp_max_df$temp)
[1] 31.27302
```

Fig. 20. ARIMA MA(1) forecasted values.

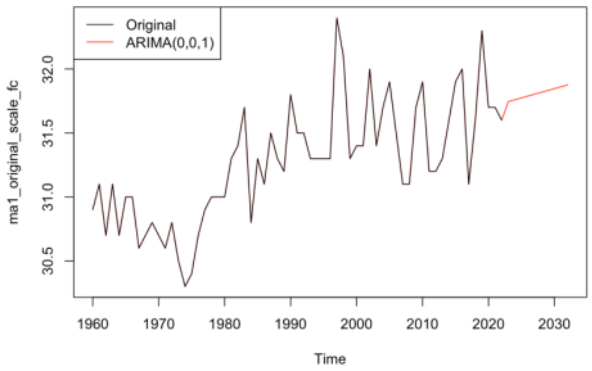


Fig. 21. Arima MA(1) forecast plot.

Code Snippet for reversing to original scale:

```
# Forecast using MA(1)
fc_length = 10
ma1_fc = predict(ma_process.ma1, n.ahead = fc_length)

# Reverse the differencing
fcval <- c(temp_max_ts_log_diff1, ma1_fc$pred) # Combine historical & forecasted
fcval <- ts(fcval, start = c(1960), frequency = 1)

# Reverse the log transformation
newval <- c(temp_max_ts_log[1], temp_max_ts_log[1] + cumsum(fcval))
newval <- ts(newval, start = c(1960), frequency = 1)

ma1_original_scale_fc = exp(newval)
print(ma1_original_scale_fc)
```

Models Evaluation

Model	MAPE	MAD	MSD	AIC
Holt's Linear Trend	0.8456067	0.265236	0.1067183	130.0511
Linear Regression	0.8119677	0.254446	0.1076616	44.37423
ARIMA MA(1)	0.8426017	0.264614	0.1088727	44.15

Table 1. Models Evaluation Summary.

Comparing the accuracy metrics above, Holt's Linear method seems to be the least effective having the highest MAPE and MAD. It also has higher AIC compared to the Linear Regression. Overall, Linear Regression and ARIMA are similar and likely to be more suitable for forecasting the temperature in Singapore.

Looking at Fig. 9 and Fig. 20, both forecasting models show gradual increase in temperature over the decade. In short-term future the temperature is ranging around mid-31 degrees. Although the increment is slow, but a steady rise could have implication on the energy consumption for cooling (Chidiac et al., 2022). Over the medium term, the temperatures gradually reaching 32 degrees. This persistent increase indicating that the warming trend is not expected to come down soon.

We should be prepared for increased demand for cooling, potential impact on water resources. Adaptation strategies need to be implemented to manage the inevitable rise in temperature. This could include developing more green spaces, improving building code to increase energy efficiency and heat resistance, reducing reduce greenhouse gas emissions, and putting water conservation measures in action.

Word count: 1399

Question 2

Preparing Time Series

Below is the time series and Fig. 1 shows a diffuse spread of data points around the regression line. This could imply a weak relationship between the rainfall time series and the temperature time series.

```
> print(temp_max_ts)
Time Series:
Start = 1960
End = 2022
Frequency = 1
 [1] 30.9 31.1 30.7 31.1 30.7 31.0 31.0 30.6 30.7 30.8 30.7 30.6 30.8 30.5 30.3 30.4
[17] 30.7 30.9 31.0 31.0 31.0 31.3 31.4 31.7 30.8 31.3 31.1 31.5 31.3 31.2 31.8 31.5
[33] 31.5 31.3 31.3 31.3 31.3 32.4 32.1 31.3 31.4 31.4 32.0 31.4 31.7 31.9 31.5 31.1
[49] 31.1 31.7 31.9 31.2 31.2 31.3 31.6 31.9 32.0 31.1 31.6 32.3 31.7 31.7 31.6
> print(rainfall_ts)
Time Series:
Start = 1960
End = 2022
Frequency = 1
 [1] 1569.6 1817.6 2293.2 1824.1 2833.2 1857.2 2486.9 2918.8 2084.2 2297.1 2283.7
[12] 1613.5 1806.9 2956.0 2066.4 1925.4 2166.6 1774.7 2766.0 2168.1 2325.7 1555.8
[23] 1749.6 2080.8 2686.7 1483.9 2536.1 2102.8 2598.6 2463.2 1523.8 1877.0 2260.8
[34] 2168.7 1941.8 2332.6 2418.0 1118.9 2623.1 2134.0 2370.5 2783.1 1748.9 2391.2
[45] 2136.4 1930.7 2753.2 2886.2 2325.1 1920.9 2075.1 2524.2 2159.9 2748.4 1538.4
[56] 1266.8 1955.7 2045.6 1708.2 1367.5 1886.6 2809.6 2207.2
```

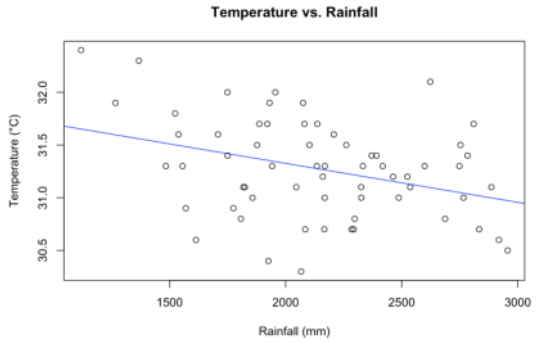


Fig. 1. Overall plot of temperature and rainfall time series.

Pre-Whitening Time Series

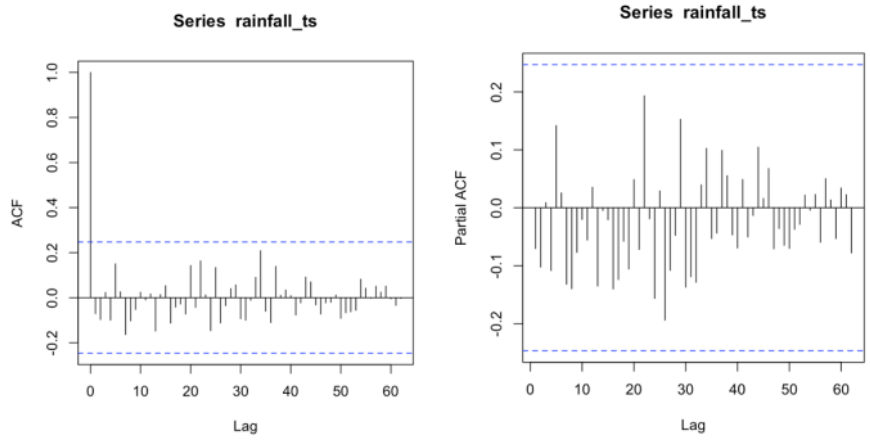


Fig. 2. ACF and PACF of rainfall time series.

From Fig. 2, the ACF does not show clear cut off after lag 1 as there are still a few spikes even if it is still below the threshold, and the PACF does not show a clear cut off either. An ARIMA model with a mix of AR and MA components might be considered for pre whitening the rainfall time series. Comparing all the p-value below, ARIMA(1,0,1) is the most suitable model.

p-value of ARIMA(1,0,1)

```
> pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 # Compute p-values
        ar1         ma1    intercept
3.483937e-14 1.142050e-32 1.215082e-82
```

p-value of ARIMA(2,0,1)

```
> pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 # Compute
p-values
       ar1          ar2          ma1    intercept
4.566464e-02 2.319672e-01 5.970737e-02 2.318210e-47
```

p-value of ARIMA(1,0,2)

```
> pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 # Compute
p-values
       ar1          ma1          ma2    intercept
2.348840e-01 3.122404e-01 2.166175e-01 2.279915e-48
```

p-value of ARIMA(1,1,1)

```
> pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 # Compute
p-values
       ar1          ma1
6.626442e-01 1.076388e-28
```

Calculating Cross-Correlation Function (CCF)

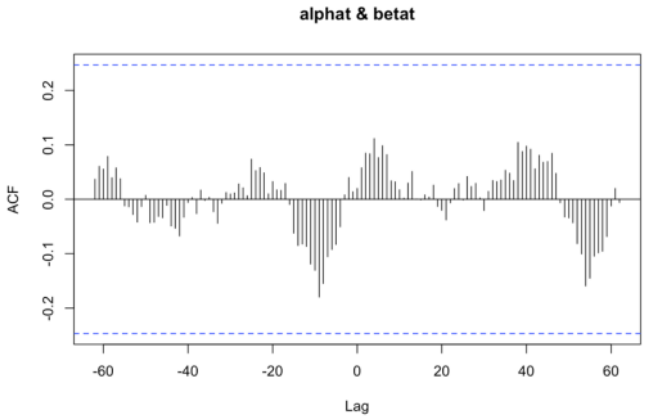


Fig. 3. CCF

Next, the residuals of the ARIMA(1,0,1) will be extracted as “alphat” to be the pre-whitened series of rainfall time series. Then, we fit the same ARIMA(1,0,1) model on the temperature time series without changing the coefficients. The residual of the pre-whitened series of temperature time series is extracted as “betat”. Given the correlations in Fig. 3, none of the values cross the threshold, which implies that there are no statistically significant linear relationships between rainfall and temperature at any of the tested lags within the confidence level used to set the threshold. There isn't a strong linear relationship where the amount of rainfall significantly predicts temperature changes, or vice versa, at any specific lag in the period analyzed.

Fitting Dynamic Regression

The ACF of the residuals of Linear Regression in Fig. 4 gradually decreases but stay above the significance threshold up to lag 2 and sporadically afterwards. The PACF does show a cut off after lag 1 but there are spikes that do not have clear pattern, observed under the threshold. This suggests dynamic regression using ARIMA of AR & MA mixture.

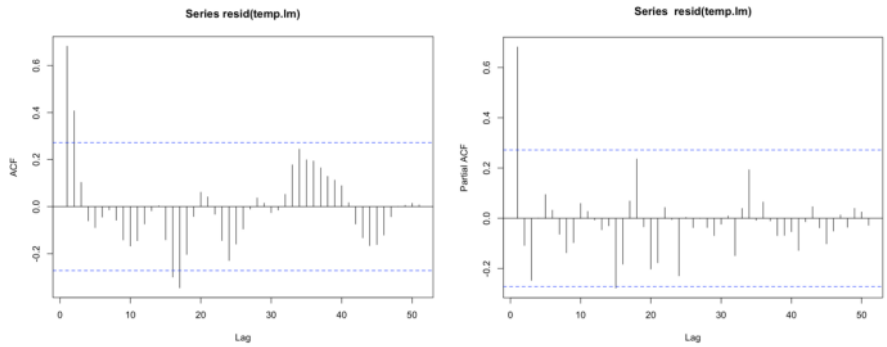


Fig. 4. Residuals ACF & PACF of Linear Regression

Below are the p-values of ARIMA models tested, suggesting that ARIMA(0,1,1) or ARIMA(0,0,1) is the best model as both show significant statistic for both MA and Xreg. Next, we need to compare the ACF and PACF of the residuals for both models.

p-value of ARIMAX(1,0,1)

```
> pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 # Compute p-values
        ar1         ma1   intercept        xreg
3.796802e-33 1.031936e-06         NaN         NaN
```

p-value of ARIMAX(2,0,1)

```
> pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 # Compute p-values
        ar1         ar2         ma1   intercept        xreg
6.361394e-01 1.132867e-02 2.544811e-06         NaN         NaN
```

p-value of ARIMAX(1,0,2)

```
> pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 # Compute p-values
        ar1         ma1         ma2   intercept        xreg
3.978195e-36 1.197154e-02 1.374976e-01         NaN         NaN
```

p-value of ARIMAX(0,0,1)

```
> pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 # Compute p-values
        ma1   intercept        xreg
7.466616e-11 2.626349e-75 1.201168e-02
```

p-value of ARIMAX(0,0,2)

```
> pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 # Compute p-values
        ma1         ma2   intercept        xreg
6.680884e-06 2.767055e-01 7.916529e-73 4.208061e-02
```

p-value of ARIMAX(1,0,0)

```
> pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 # Compute p-values
        ar1   intercept        xreg
8.685465e-12         NaN         NaN
```

p-value of ARIMAX(0,1,1)

```
> pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 # Compute p-values
        ma1        xreg
5.687232e-09 1.834789e-02
```

Evaluation of Dynamic Regression

Fig. 5 shows that there is still presence of significant autocorrelation in ACF and PACF of the residuals, suggesting ARIMAX(0,0,1) may not be adequate. In contrast, Fig. 6 shows the absence of clear, significant patterns in the ACF and PACF of the residuals of the dynamic regression ARIMAX(0,1,1), suggesting that there is no strong autocorrelated behavior left unmodeled.

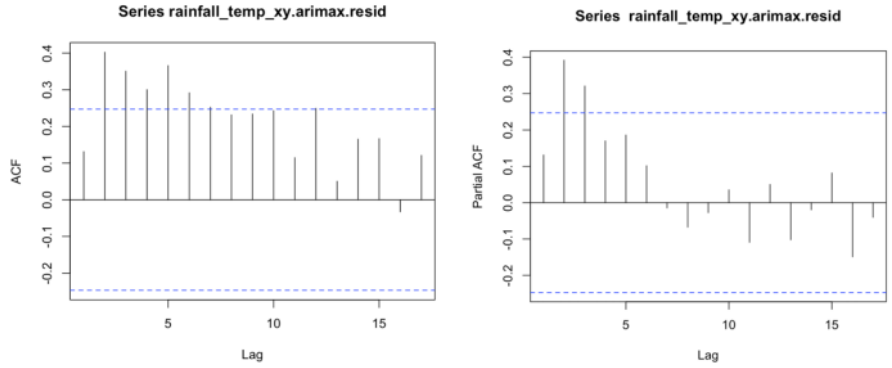


Fig. 5. ACF & PACF of residuals of ARIMAX(0,0,1).

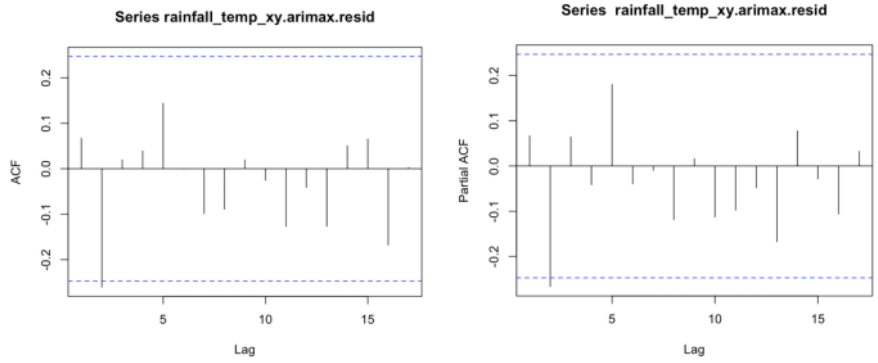
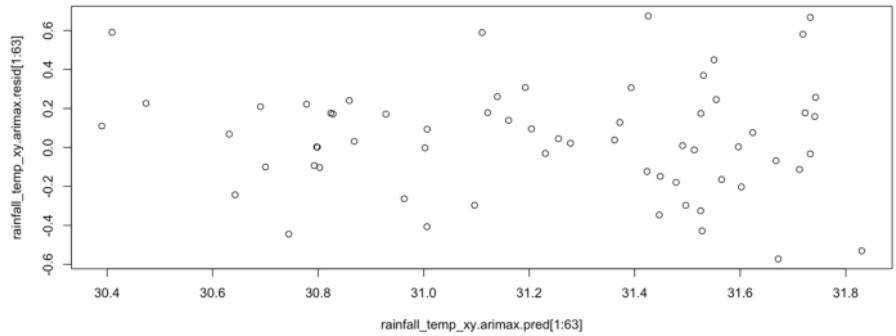


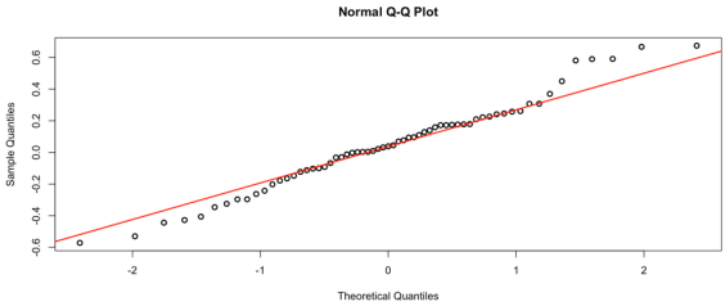
Fig. 6. ACF & PACF of residuals of ARIMAX(0,1,1).




```
> mean_resid
[1] 0.0436089
```

Fig. 7. Forecast against residuals of ARIMAX(0,1,1)

Fig. 7 shows that the mean of the residuals is 0.0436 which is very close to zero, suggesting there is very slight positive bias in the model's prediction where the forecasts are slightly underestimating the actual values. However, the small magnitude of this mean still suggests that the bias is minimal.




```
> shapiro.test(rainfall_temp_xy.arimax.resid[1:63])

        Shapiro-Wilk normality test

data:  rainfall_temp_xy.arimax.resid[1:63]
W = 0.9811, p-value = 0.4433
```

Fig. 8 QQ Plot and Shapiro Test of residuals of Dynamic Regression ARIMAX(0,1,1)

Finally, Fig. 8 illustrates the QQ plot and Shapiro Test shows p-value of 0.4433 > 0.05, suggesting the null hypothesis is not rejected, concluding that the residuals of the dynamic regression model are normally distributed. In conclusion, ARIMAX(0,1,1) is a well-fitted model with a very minimal bias of underestimating the actual values.

Temperature Forecast

For forecasting the temperature using the ARIMAX(0,1,1), we need to provide the forecasted value of the rainfall. Since the rainfall time series is already stationary, we can directly use the ARIMA(1,0,1) tested in the pre-whitening part of the report to forecast the rainfall for year 2023 to 2032.

```
> print(rainfall_fc)
Time Series:
Start = 2023
End = 2032
Frequency = 1
 [1] 2283.630 2259.484 2239.894 2224.003 2211.110 2200.652 2192.167 2185.284 2179.700
2175.170
```

After getting the rainfall forecast above, we can substitute it into the "newxreg" parameter into the ARIMAX(0,1,1) to forecast future temperature. The forecasted value below is very similar to the mean of the historical data which is 31.27302.

```
> # Forecast the future temperatures
> arimax.fc <- predict(rainfall_temp_xy.arimax, n.ahead = 10, newxreg = rainfall_fc)
> future_temp_fc <- c(arimax.fc$pred)
> print(future_temp_fc)
 [1] 31.64679 31.65538 31.66235 31.66801 31.67260 31.67632 31.67934 31.68179 31.68378
31.68539
```

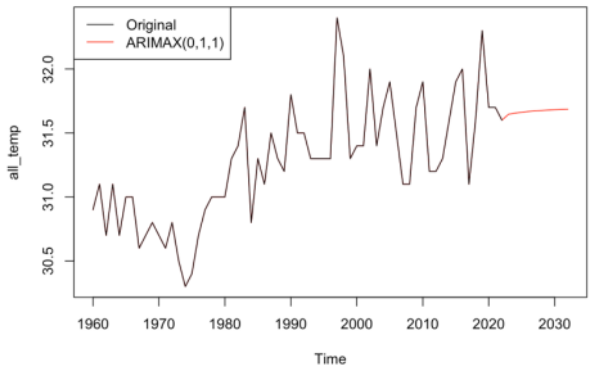


Fig. 9. ARIMAX forecast plot.

Fig 9 above shows the forecast plot with milder increase in temperature.

Conclusion

```
> summary(rainfall_temp_xy.arimax)

Call:
arimax(x = temp_max_ts, order = c(0, 1, 1), xreg = rainfall_ts)

Coefficients:
          ma1    xreg
      -0.6582  -4e-04
s.e.   0.0969   1e-04

sigma^2 estimated as 0.0809:  log likelihood = -10.31,  aic = 24.62

Training set error measures:
                    ME RMSE MAE MPE MAPE
Training set NaN  NaN NaN NaN  NaN
Warning message:
In trainingaccuracy(object, test, d, D) :
  test elements must be within sample
```

In conclusion, the dynamic regression model ARIMAX(0,1,1) is well-fitted. The AIC of the dynamic regression model is 24.62 which is much lower compared to ARIMA MA(1) model with AIC value of 44.15 in Question 1. The additional predictor used here seems to increase the forecasting quality as it lowers the AIC value. Table 1 shows that the accuracy metrics of ARIMAX(0,1,1) are all lower than that of ARIMA MA(1), suggesting that the dynamic regression ARIMAX(0,1,1) is more preferred.

Model	MAPE	MAD	MSD	AIC
ARIMA MA(1)	0.8426017	0.264614	0.1088727	44.15
ARIMAX(0,1,1)	0.8119677	0.254445	0.1076616	24.62

Table 1. Models' accuracy metrics

Code Snippet for calculating accuracy metrics:

```
# Calculating predicted values for the actual period (1960-2022)
temp_max_df$predicted_temp <- rainfall_temp_xy.arimax.pred

# Calculating MAPE
MAPE_arimax <- mean(abs((temp_max_df$temp -
temp_max_df$predicted_temp) / temp_max_df$temp)) * 100

# Calculating MAD
MAD_arimax <- mean(abs(temp_max_df$temp -
temp_max_df$predicted_temp))

# Calculating MSD
MSD_arimax <- mean((temp_max_df$temp -
temp_max_df$predicted_temp)^2)
```

Word count: 639

Reference

Chidiac, S., Yao, L., & Liu, P. (2022). Climate change effects on heating and cooling demands of buildings in Canada. CivilEng, 3(2), 277–295. https://doi.org/10.3390/civileng3020017

Lin, M., Tsai, C. W., & Chen, C. (2021). Daily maximum temperature forecasting in changing climate using a hybrid of Multi-dimensional Complementary Ensemble Empirical Mode Decomposition and Radial Basis Function Neural Network. Journal of Hydrology. Regional Studies, 38, 100923. https://doi.org/10.1016/j.ejrh.2021.100923

Appendix

Question 1 Source Code

```
# Set your working directory
setwd("/Users/sallyyeo/Desktop/ANL557-ECA")
raw <- read.csv("SG_Enviroment_TemperatureRainfall.csv")

##### Shaping Data #####
library(tidyverse) # to use pivot_longer()

# Reshape the data by pivot_longer() & pivot_wider()
long_data <- raw %>%
  pivot_longer(cols = starts_with("X"), names_to = "Year",
values_to = "Value")

wide_data <- long_data %>%
  pivot_wider(names_from = Data.Series, values_from = Value)

# Remove 'X' from the Year column
wide_data$Year <- sub("X", "", wide_data$Year)

# Convert all columns to numeric
wide_data <- wide_data %>%
  mutate_all(function(x) {
    x <- gsub(",", "", x)   # to remove comma
    parse_number(x, na = "na")}) # set empty values to NA

# Convert year to date type
wide_data$Year <- as.Date(paste(wide_data$Year, "-01-01", sep=""),
format="%Y-%m-%d")

# Ensure the data is in ascending chronological order
wide_data <- wide_data[order(wide_data$Year), ]

print(wide_data, n=63)

##### Prepare time-series object #####
library(forecast)
library(tsibble)
library(ggplot2)
```

```
# Convert 'Air Temperature Means Daily Maximum (Degree Celsius)' to
a ts object
temp_max_ts <- ts(wide_data$`Air Temperature Means Daily Maximum
(Degree Celsius)`,
                  start = min(year(wide_data$Year)),
                  end = max(year(wide_data$Year)),
                  frequency = 1) # Since the data is yearly,
frequency is 1

print(temp_max_ts)

# Create a data frame with years and temperature values
years <- seq(1960, 2022)  # Sequence from start year to end year
temp_max_df <- data.frame(year = years, temp = temp_max_ts)

########## Explore time-series object ##########
# View the time series object
plot(temp_max_ts, xlab = "Year", ylab = "Temperature (°C)",
     main = "Annual Average Maximum Temperature in Singapore")

# create time index
time_index <- time(temp_max_ts)

# Generate the linear regression
temp_max_trend_model <- lm(temp_max_ts ~ time_index)
summary(temp_max_trend_model)

# Plot the original time series and the trend
plot(temp_max_ts, xlab = "Year", ylab = "Temperature (°C)",
     main = "Annual Average Maximum Temperature and Trend in
Singapore")
abline(a = temp_max_trend_model$coefficients[1], b =
temp_max_trend_model$coefficients[2], col = "red")
legend("topleft", legend = c("Original", "Trend"), col = c("black",
"red"), lty = 1)

# Check for stationarity using ACF
acf(temp_max_ts, lag.max = 63)
pacf(temp_max_ts, lag.max = 63)

#install.packages("tseries")
library(tseries)

######################### HOLT DESM ########################
```

```
# Set forecast length to 10 yrs
fc_length = 10

# Apply Holt's linear trend method
temp_max_ts.desm <- holt(temp_max_ts, h = fc_length)
print(temp_max_ts.desm)

# Create time series combining the fitted value of a model with the
forecasted values
temp_max_ts.desm_fc        <-        ts(c(temp_max_ts.desm$fitted,
temp_max_ts.desm$mean), start = c(1960, 1), frequency = 1)
print(temp_max_ts.desm_fc)

# Get the extended time periods
x_val <- time(temp_max_ts.desm_fc)

# Append the exact number of NA as the forecast length
y_val <- c(wide_data$`Air Temperature Means Daily Maximum (Degree
Celsius)`, rep(NA, fc_length))

# Plot using the extended series
plot(x = x_val, y = y_val, type = "l", xlab = "Year", ylab =
"Temperature (°C)", lwd = 2,
     main = "Annual Average Maximum Temperature and Trend in
Singapore")

# Overlay the double exponential smoothing to the plot
lines(x = x_val, y = temp_max_ts.desm_fc, col = "red", lwd = 3,
type = "l")
legend("topleft", legend = c("Original", "Holt's"), col = c("black",
"red"), lty = 1)

################## HOLT Model Evaluation ######################
# Holt's model summary
temp_max_ts.desm$model

# Compute the Mean Absolute Percentage Error (MAPE)
MAPE_holt <- mean(abs(temp_max_ts.desm$residuals) / wide_data$`Air
Temperature Means Daily Maximum (Degree Celsius)`) * 100

# Compute the Mean Absolute Deviance (or Error) (MAD)
MAD_holt <- mean(abs(temp_max_ts.desm$residuals))

# Compute the Mean Square Deviance (or Error) (MSD)
```

```
MSD_holt <- mean(temp_max_ts.desm$residuals ^ 2)

# Extract the Akaike Information Criterion (AIC)
AIC_holt <- temp_max_ts.desm$model$aic

# Print the metrics
print(paste("MAPE_holt:", MAPE_holt))
print(paste("MAD_holt:", MAD_holt))
print(paste("MSD_holt:", MSD_holt))
print(paste("AIC_holt:", AIC_holt))

#########################             LINEAR             REGRESSION
#########################
# Fitting a linear model
linear_model <- lm(temp ~ year, data = temp_max_df)
summary(linear_model)  # View model summary for details

# Create a new data frame that includes both historical and future
years
all_years <- data.frame(year = c(1960:2022, 2023:2032))

# Combining fitted with forecasted values
temp_max_df.linear_fc <- predict(linear_model, newdata = all_years)
all_years$temp <- temp_max_df.linear_fc

# Convert to time series object
temp_max_ts.linear_fc <- ts(temp_max_df.linear_fc, start = 1960,
end = 2032, frequency = 1)
# Print the forecasted values
print(temp_max_ts.linear_fc)

# Get the extended time periods & append the exact number of NA as
the forecast length
x_val <- time(temp_max_ts.linear_fc)
y_val <- c(temp_max_df$temp, rep(NA, fc_length))

# Plot using the extended series
plot(x = x_val, y = y_val, type = "l", xlab = "Year", ylab =
"Temperature (°C)", lwd = 2,
     main = "Annual Average Maximum Temperature and Trend in
Singapore")

abline(linear_model, col = "green") # Plot linear model fit
```

```
# Plot forecasted values
lines(x = x_val, y = temp_max_ts.linear_fc, col = "red", lty = 1,
type ="l")
legend("topleft", legend = c("Original", "Linear Regression",
"Forecasted Value"), col = c("black", "green", "red"), lty = 1)

################# Linear Model Evaluation ######################
anova(linear_model)

# Extract the residuals of the linear regression
linear_model.res <- residuals(linear_model)

# Create a qq-plot for the model residuals
qqnorm(y = linear_model.res)

# Add a 45-degree line to it. If the residuals are lying closely on
the line, the residuals
# of the linear regression are most likely normally distributed.
qqline(linear_model.res, col = "red")

# Generate a histogram to have another check on the distribution of
the residuals
hist(linear_model.res, breaks = 8)

# The result should be supported by a numerical test.
# The Shapiro-Wilk test is most common for it.
shapiro.test(linear_model.res)

# Check on the independence of the residuals by looking at the ACF
acf(linear_model.res, lag.max = 63)

#install.packages("car")
library(car)
durbinWatsonTest(linear_model)

# Calculating predicted values for the actual period (1960-2022)
temp_max_df$predicted_temp <- predict(linear_model, newdata =
temp_max_df)

# Calculating MAPE
MAPE_lm <- mean(abs((temp_max_df$temp - temp_max_df$predicted_temp)
/ temp_max_df$temp)) * 100

# Calculating MAD
```

```
MAD_lm <- mean(abs(temp_max_df$temp - temp_max_df$predicted_temp))

# Calculating MSD
MSD_lm <- mean((temp_max_df$temp - temp_max_df$predicted_temp)^2)

# Extracting AIC
AIC_lm <- AIC(linear_model)

# Print the metrics
print(paste("MAPE:", MAPE_lm))
print(paste("MAD:", MAD_lm))
print(paste("MSD:", MSD_lm))
print(paste("AIC:", AIC_lm))

########################## ARIMA ##########################
# First differencing
temp_max_ts_diff1 <- diff(temp_max_df$temp, lag = 1)

# Check variance
plot(temp_max_ts_diff1)
plot(x = 1:62, y = temp_max_ts_diff1)

# Check stationarity with ACF
acf(temp_max_ts_diff1, lag.max=62)
pacf(temp_max_ts_diff1, lag.max=62)

# Log transformation
temp_max_ts_log <- log(temp_max_df$temp, base = exp(1))

# Check variance
plot(x = wide_data$Year, y = temp_max_ts_log)

# Check stationarity with ACF
acf(temp_max_ts_log)
pacf(temp_max_ts_log)

# Differencing the log transformed
temp_max_ts_log_diff1 <- diff(temp_max_ts_log, lag = 1)

# Check variance
plot(temp_max_ts_log_diff1)
plot(x = 1:62, y = temp_max_ts_log_diff1)

# Check stationarity with ACF
```

```
acf(temp_max_ts_log_diff1, lag.max = 63)
pacf(temp_max_ts_log_diff1, lag.max = 63)

# MA(1) after diff the log transformed
ma_process.ma1 <- arima(temp_max_ts_log_diff1, order = c(0, 0, 1))
print(ma_process.ma1)

# Find p-value of the MA(1)
param_ma1 <- ma_process.ma1$coef
covmat_ma1 <- ma_process.ma1$var.coef
se_ma1 <- sqrt(diag(covmat_ma1))
ttest_ma1 <- abs(param_ma1) / se_ma1
pt(ttest_ma1, df = length(ma_process.ma1) - length(param_ma1),
lower.tail = FALSE) * 2
summary(ma_process.ma1)

# MA(2) after diff the log transformed
ma_process.ma2 <- arima(temp_max_ts_log_diff1, order = c(0, 0, 2))
print(ma_process.ma2)

# Find p-value of the MA(2)
param_ma2 <- ma_process.ma2$coef
covmat_ma2 <- ma_process.ma2$var.coef
se_ma2 <- sqrt(diag(covmat_ma2))
ttest_ma2 <- abs(param_ma2) / se_ma2
pt(ttest_ma2, df = length(ma_process.ma2) - length(param_ma2),
lower.tail = FALSE) * 2
summary(ma_process.ma2)

########## Model Diagnostic ##########
# Extract the residuals of the model MA(1)
ma_process.ma1.res <- residuals(ma_process.ma1)
acf(ma_process.ma1.res)
pacf(ma_process.ma1.res)

# Check model's normality
hist(ma_process.ma1.res)
qqnorm(ma_process.ma1.res)
qqline(ma_process.ma1.res, col = "red")
shapiro.test(ma_process.ma1.res)

# Accuracy metrics
arima011 <- arima(temp_max_df$temp, order = c(0, 1, 1))
summary(arima011)
```

```
# Forecast using MA(1)
fc_length = 10
ma1_fc = predict(ma_process.ma1, n.ahead = fc_length)

# Reverse the differencing
fcval <- c(temp_max_ts_log_diff1, ma1_fc$pred) # Combine historical
& forecasted
fcval <- ts(fcval, start = c(1960), frequency = 1)

# Reverse the log transformation
newval <- c(temp_max_ts_log[1], temp_max_ts_log[1] + cumsum(fcval))
newval <- ts(newval, start = c(1960), frequency = 1)
ma1_original_scale_fc = exp(newval)

# Print forecast in original scale
print(ma1_original_scale_fc)

# Compare forecast with mean of historical data
mean(temp_max_df$temp)

# Plot the forecast values
plot(ma1_original_scale_fc, type = "l", col = "red")
lines(temp_max_df$temp, col = "black")
legend("topleft", legend = c("Original", "ARIMA(0,0,1)"), col =
c("black", "red"), lty = 1)
```

Question 2 Source Code

```
# Convert 'Air Temperature Means Daily Maximum (Degree Celsius)' to
a ts object

temp_max_ts <- ts(wide_data$`Air Temperature Means Daily Maximum
(Degree Celsius)`,

                  start = min(year(wide_data$Year)),

                  end = max(year(wide_data$Year)),

                  frequency = 1) # Since the data is yearly,
frequency is 1


# Convert 'Air Temperature Means Daily Maximum (Degree Celsius)' to
a ts object

rainfall_ts <- ts(wide_data$`Total Rainfall (Millimetre)`,

                  start = min(year(wide_data$Year)),

                  end = max(year(wide_data$Year)),

                  frequency = 1) # Since the data is yearly,
frequency is 1


# Adding new column total_rainfall to the temp_max_df created in Q1

temp_max_df %>% mutate(total_rainfall = rainfall_ts) ->
temp_rainfall_df


# Check stationarity for the maximum temperature time series

adf_test_temp = adf.test(temp_max_ts, alternative = "stationary")

print(adf_test_temp) # non-stationary


# Check stationarity for the rainfall time series

adf_test_rainfall = adf.test(rainfall_ts, alternative =
"stationary")

print(adf_test_rainfall) # stationary
```

```
# Check ACF and PACF
rainfall_acf <- acf(rainfall_ts, lag.max = 63)
rainfall_pacf <- pacf(rainfall_ts, lag.max= 63)
temp_acf <- acf(temp_max_ts, lag.max= 63)
temp_pacf <- pacf(temp_max_ts, lag.max= 63)

# Create a combined data frame
years <- seq(1960, 2022)
data_df <- data.frame(year = years,
                      temperature = as.numeric(temp_max_ts),
                      rainfall = as.numeric(rainfall_ts))

# Scatter plot of temperature vs. rainfall
plot(data_df$rainfall, data_df$temperature, xlab = "Rainfall (mm)",
ylab = "Temperature (°C)", main = "Temperature vs. Rainfall")
abline(lm(temperature ~ rainfall, data = data_df), col = "blue")  #
Adds a regression line


###### TESTING ARIMA MODEL for PRE-WHITENING ######
# For dynamic regression, we must first prewhiten both the series
using the suitable ARIMA model for rainfall_ts
rainfall.arima101 <- arima(rainfall_ts, order = c(1, 0, 1))
print(rainfall.arima101)
param <- rainfall.arima101$coef # Extract coefficient estimates
se <- sqrt(diag(rainfall.arima101$var.coef)) # Extract standard
errors of the coefficient estimates
```

```
degree_freedom <- rainfall.arima101$nobs - length(param) # Degrees
of freedom

pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 #
Compute p-values


# For dynamic regression, we must first prewhiten both the series
using the suitable ARIMA model for rainfall_ts

rainfall.arima201 <- arima(rainfall_ts, order = c(2, 0, 1))

print(rainfall.arima201)

param <- rainfall.arima201$coef # Extract coefficient estimates

se <- sqrt(diag(rainfall.arima201$var.coef)) # Extract standard
errors of the coefficient estimates

degree_freedom <- rainfall.arima201$nobs - length(param) # Degrees
of freedom

pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 #
Compute p-values


# For dynamic regression, we must first prewhiten both the series
using the suitable ARIMA model for rainfall_ts

rainfall.arima102 <- arima(rainfall_ts, order = c(1, 0, 2))

print(rainfall.arima102)

param <- rainfall.arima102$coef # Extract coefficient estimates

se <- sqrt(diag(rainfall.arima102$var.coef)) # Extract standard
errors of the coefficient estimates

degree_freedom <- rainfall.arima102$nobs - length(param) # Degrees
of freedom

pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 #
Compute p-values


# For dynamic regression, we must first prewhiten both the series
using the suitable ARIMA model for xt
```

```
rainfall.arima111 <- arima(rainfall_ts, order = c(1, 1, 1))

print(rainfall.arima111)

param <- rainfall.arima111$coef # Extract coefficient estimates

se <- sqrt(diag(rainfall.arima111$var.coef)) # Extract standard
errors of the coefficient estimates

degree_freedom <- rainfall.arima111$nobs - length(param) # Degrees
of freedom

pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 #
Compute p-values


# Take the residuals of the model for rainfall_ts as the prewhitened
series of rainfall_ts

alphat <- resid(rainfall.arima101)

print(alphat)


# Chosen ARIMA(1,0,1)

# Fit the same ARIMA model on temperature time series without
changing the coefficients at all.

temperature.arima101 <- arima(temp_max_ts, order = c(1, 0, 1),
                              include.mean = TRUE,
                              fixed = rainfall.arima101$coef,
                              optim.control = list(maxit = 1))

print(temperature.arima101)


# Take the residuals of the model for temp_max_ts as the pre-
whitened series of temp_max_ts

betat <- resid(temperature.arima101)

print(betat)
```

```
# Calculate the cross-correlation function of alphat (prewhitened
rainfall_ts) and betat (temp_max_ts).

ab_ccf <- ccf(x = alphat, y = betat, lag.max = 63)

print(ab_ccf)

plot(ab_ccf)


# Load the libraries TSA and forecast to run the final dynamic
regression model

library(TSA)

library(forecast)


########### TESTING ARIMA MODELS for DYNAMIC REGRESSION
###############

# Run the dynamic regression model and specify the ARIMA order for
temperature time series.

rainfall_temp_xy.arimax <- arimax(x = temp_max_ts, order = c(1, 0,
1), xreg = rainfall_ts)

summary(rainfall_temp_xy.arimax)


# Compute the p-values

param <- rainfall_temp_xy.arimax$coef # Extract coefficient
estimates

se <- sqrt(diag(rainfall_temp_xy.arimax$var.coef)) # Extract
standard errors of the coefficient estimates

degree_freedom <- rainfall_temp_xy.arimax$nobs - length(param) #
Degrees of freedom

pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 #
Compute p-values


# Run the dynamic regression model and specify the ARIMA order for
temperature time series.
```

```
rainfall_temp_xy.arimax <- arimax(x = temp_max_ts, order = c(2, 0,
1), xreg = rainfall_ts)

summary(rainfall_temp_xy.arimax)


# Compute the p-values

param <- rainfall_temp_xy.arimax$coef # Extract coefficient
estimates

se <- sqrt(diag(rainfall_temp_xy.arimax$var.coef)) # Extract
standard errors of the coefficient estimates

degree_freedom <- rainfall_temp_xy.arimax$nobs - length(param) #
Degrees of freedom

pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 #
Compute p-values


# Run the dynamic regression model and specify the ARIMA order for
temperature time series.

rainfall_temp_xy.arimax <- arimax(x = temp_max_ts, order = c(1, 0,
2), xreg = rainfall_ts)

summary(rainfall_temp_xy.arimax)


# Compute the p-values

param <- rainfall_temp_xy.arimax$coef # Extract coefficient
estimates

se <- sqrt(diag(rainfall_temp_xy.arimax$var.coef)) # Extract
standard errors of the coefficient estimates

degree_freedom <- rainfall_temp_xy.arimax$nobs - length(param) #
Degrees of freedom

pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 #
Compute p-values


# Run the dynamic regression model and specify the ARIMA order for
temperature time series.
```

```
rainfall_temp_xy.arimax <- arimax(x = temp_max_ts, order = c(0, 0,
1), xreg = rainfall_ts)

summary(rainfall_temp_xy.arimax)


# Compute the p-values

param <- rainfall_temp_xy.arimax$coef # Extract coefficient
estimates

se <- sqrt(diag(rainfall_temp_xy.arimax$var.coef)) # Extract
standard errors of the coefficient estimates

degree_freedom <- rainfall_temp_xy.arimax$nobs - length(param) #
Degrees of freedom

pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 #
Compute p-values


# Run the dynamic regression model and specify the ARIMA order for
temperature time series.

rainfall_temp_xy.arimax <- arimax(x = temp_max_ts, order = c(0, 0,
2), xreg = rainfall_ts)

summary(rainfall_temp_xy.arimax)


# Compute the p-values

param <- rainfall_temp_xy.arimax$coef # Extract coefficient
estimates

se <- sqrt(diag(rainfall_temp_xy.arimax$var.coef)) # Extract
standard errors of the coefficient estimates

degree_freedom <- rainfall_temp_xy.arimax$nobs - length(param) #
Degrees of freedom

pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 #
Compute p-values


# Run the dynamic regression model and specify the ARIMA order for
temperature time series.
```

```
rainfall_temp_xy.arimax <- arimax(x = temp_max_ts, order = c(1, 0,
0), xreg = rainfall_ts)

summary(rainfall_temp_xy.arimax)


# Compute the p-values

param <- rainfall_temp_xy.arimax$coef # Extract coefficient
estimates

se <- sqrt(diag(rainfall_temp_xy.arimax$var.coef)) # Extract
standard errors of the coefficient estimates

degree_freedom <- rainfall_temp_xy.arimax$nobs - length(param) #
Degrees of freedom

pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 #
Compute p-values



# Run the dynamic regression model and specify the ARIMA order for
temperature time series.

rainfall_temp_xy.arimax <- arimax(x = temp_max_ts, order = c(0, 1,
1), xreg = rainfall_ts)

summary(rainfall_temp_xy.arimax)


# Compute the p-values

param <- rainfall_temp_xy.arimax$coef # Extract coefficient
estimates

se <- sqrt(diag(rainfall_temp_xy.arimax$var.coef)) # Extract
standard errors of the coefficient estimates

degree_freedom <- rainfall_temp_xy.arimax$nobs - length(param) #
Degrees of freedom

pt(abs(param) / se, df = degree_freedom, lower.tail = FALSE) * 2 #
Compute p-values
```

```
########## CHOSEN ARIMAX(0,1,1) ###############

# Run the dynamic regression model and specify the ARIMA order for
temperature ts.

rainfall_temp_xy.arimax <- arimax(x = temp_max_ts, order = c(0, 1,
1), xreg = rainfall_ts)

summary(rainfall_temp_xy.arimax)


# Extract the residuals of the dynamic regression model for model
diagnostics

rainfall_temp_xy.arimax.resid <- resid(rainfall_temp_xy.arimax)


# Examine the ACF and PACF of the residuals, filter out NA.

acf_resid_dynreg = acf(rainfall_temp_xy.arimax.resid, na.action =
na.omit)

pacf_resid_dynreg = pacf(rainfall_temp_xy.arimax.resid, na.action
= na.omit)


# Plot forecasted values against residuals to check on the zero-
mean and constant variance assumptions.

rainfall_temp_xy.arimax.pred       <-       temp_max_ts       -
rainfall_temp_xy.arimax.resid

plot(x     =     rainfall_temp_xy.arimax.pred[1:63],     y     =
rainfall_temp_xy.arimax.resid[1:63], type = "p")

mean_resid <- mean(rainfall_temp_xy.arimax.resid)

print(mean_resid)


# Check normality of the residuals

qqnorm(rainfall_temp_xy.arimax.resid[1:63], pch = 1, lwd = 2)

qqline(rainfall_temp_xy.arimax.resid[1:63], col = "red", lwd = 2)
```

```
shapiro.test(rainfall_temp_xy.arimax.resid[1:63])


############## another dynamic regression ###################

# Forecast rainfall using ARIMA(101)

fc_length = 10

rainfall.arima101.fc   =   predict(rainfall.arima101,   n.ahead   =
fc_length)


# Convert forecast to time series

rainfall_fc <- c(rainfall.arima101.fc$pred)

rainfall_fc <- ts(rainfall_fc, start = c(2023), frequency = 1)

print(rainfall_fc)


# Forecast the future temperatures

arimax.fc <- predict(rainfall_temp_xy.arimax, n.ahead = 10, newxreg
= rainfall_fc)

future_temp_fc <- c(arimax.fc$pred)

print(future_temp_fc)


# Calculating predicted values for the actual period (1960-2022)

temp_max_df$predicted_temp <- rainfall_temp_xy.arimax.pred


# Calculating MAPE

MAPE_arimax         <-         mean(abs((temp_max_df$temp         -
temp_max_df$predicted_temp) / temp_max_df$temp)) * 100


# Calculating MAD

MAD_arimax         <-          mean(abs(temp_max_df$temp          -
temp_max_df$predicted_temp))
```

```
# Calculating MSD
MSD_arimax <- mean((temp_max_df$temp - temp_max_df$predicted_temp)^2)

# Print the metrics
print(paste("MAPE:", MAPE_arimax))
print(paste("MAD:", MAD_arimax))
print(paste("MSD:", MSD_arimax))

# Plot the forecast values
all_temp = c(temp_max_df$temp, future_temp_fc)
all_temp = ts(all_temp, start = c(1960), frequency = 1)

plot(all_temp, type = "l", col = "red")
lines(temp_max_df$temp, col = "black")
12
legend("topleft", legend = c("Original", "ARIMAX(0,1,1)"), col = c("black", "red"), lty = 1)
```

End of Report

ANL557_ECA_Sallyyeo001_SallyMarcellinaYeo.docx

ORIGINALITY REPORT

7%	5%	4%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	rstudio-pubs-static.s3.amazonaws.com Internet Source	1%
2	link.springer.com Internet Source	1%
3	brainmass.com Internet Source	<1%
4	stackoverflow.com Internet Source	<1%
5	Submitted to Queen Mary and Westfield College Student Paper	<1%
6	Submitted to RMIT University Student Paper	<1%
7	Submitted to Georgia Institute of Technology Main Campus Student Paper	<1%
8	na-cordex.org Internet Source	<1%

9	Bożena Gajdzik. "Post-Pandemic Steel Production Scenarios for Poland Based on Forecasts of Annual Steel Production Volume", Management Systems in Production Engineering, 2023 Publication	<1 %
10	Q. L. Jing, Q. Cheng, J. M. Marshall, W. B. Hu, Z. C. Yang, J. H. Lu. "Imported cases and minimum temperature drive dengue transmission in Guangzhou, China: evidence from ARIMAX model", Epidemiology and Infection, 2018 Publication	<1 %
11	Submitted to University of Nicosia Student Paper	<1 %
12	scholar.ufs.ac.za Internet Source	<1 %
13	Şalk, Selma. "Financial Data Analysis by Exponential Smoothing and ATA Method", Dokuz Eylul Universitesi (Turkey), 2024 Publication	<1 %
14	Submitted to University of New England Student Paper	<1 %
15	alexanderdemos.org Internet Source	<1 %
16	Submitted to University of Exeter Student Paper	

		<1%
17	github.com Internet Source	<1%
18	Submitted to Liverpool John Moores University Student Paper	<1%
19	Submitted to University of Illinois at Urbana-Champaign Student Paper	<1%
20	books.openedition.org Internet Source	<1%
21	Submitted to Sim University Student Paper	<1%
22	repositorio.ufmg.br Internet Source	<1%
23	www.freestatistics.org Internet Source	<1%
24	Guo, Z.. "A corrected hybrid approach for wind speed prediction in Hexi Corridor of China", Energy, 201103 Publication	<1%
25	Submitted to University College London Student Paper	<1%

26	"Handbook of Financial Econometrics and Statistics", Springer Science and Business Media LLC, 2015 Publication	<1 %
27	Hesham Fouli, Rabie Fouli, Bashar Bashir, Oumar A. Loni. "Seasonal forecasting of rainfall and runoff volumes in Riyadh region, KSA", KSCE Journal of Civil Engineering, 2017 Publication	<1 %
28	Submitted to University of Florida Student Paper	<1 %
29	contoharimadatawei1.blogspot.com Internet Source	<1 %
30	www.researchgate.net Internet Source	<1 %
31	Helen da Silva Costa Benaduce, Guilherme Pumi. "SYMARFIMA: A dynamical model for conditionally symmetric time series with long range dependence mean structure", Journal of Statistical Planning and Inference, 2023 Publication	<1 %
32	research-api.cbs.dk Internet Source	<1 %
33	Layton, Blythe Anderson. "Illness and Indicators of Beach Water Quality on the California Coast: An Interdisciplinary Study in	<1 %

Oceans and Human Health.", Stanford University, 2020
Publication

34 ÇEtin, Beyza. "Empirical Investigation of the Properties of Ata Forecasting Method", Dokuz Eylul Universitesi (Turkey), 2024 <1%
Publication

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		