Introduction

With increasing global urbanization and consumption, carbon dioxide (CO2) emission from waste have become a significant environmental concern. This study analyses aggregated regional CO2 emissions data from Europe and Central Asia from 1981 to 2023 using data from the World Bank database. We conducted a time series analysis on emissions from 1981 to 2022 and forecasted values for 2023 to 2024.

First, we explored basic statistical properties of the time series, followed by a decomposition to examine its components. To forecast future CO2 emission data, we assessed the suitability of the following models: 1) moving average, 2) simple exponential smoothing, 3) Holt's exponential smoothing, and 4) ARIMA. Model robustness was examined based on model fit and forecast accuracy.

Additionally, our study also examined the relationship between CO2 emission and national income per capita – a widely studied economic-environmental linkage (Grossman & Krueger, 1995). All data preparation, transformation, analysis and evaluation were conducted using R 4.4.3, with the corresponding R code provided in the Appendix.

Data preparation

To prepare each time series, we first use `gsub()` to extract the year, replace missing data – represented using ". ." in the original downloaded data with NA, and used `select()` to keep our columns of interest. We reshaped the data from wide to long format using `pivot_longer()`, convert year column to numeric using `mutate()`, and checked for missing values and outliers using the interquartile range method. Erroneous or missing values are then replaced using linear interpolation. Using `is.na()` and `quantile()`, we detected no missing values or outliers in CO2 emission data set and two missing values in national income data set for year 2022 and 2023. We used `approx()` to obtain values for the two missing data points. In this report, CO2 emissions will serve as the primary time series while national income will serve as the secondary time series for relationship analyses with CO2 emissions.

Basic statistics of time series

Table 1. Overview of basic statistics of CO2 emission (MtCO2e) time series

	MtCO2e
Min	5.114
Max	8.027
Mean	6.597
Median	6.544
Standard Deviation (SD)	0.740
Variance	0.548

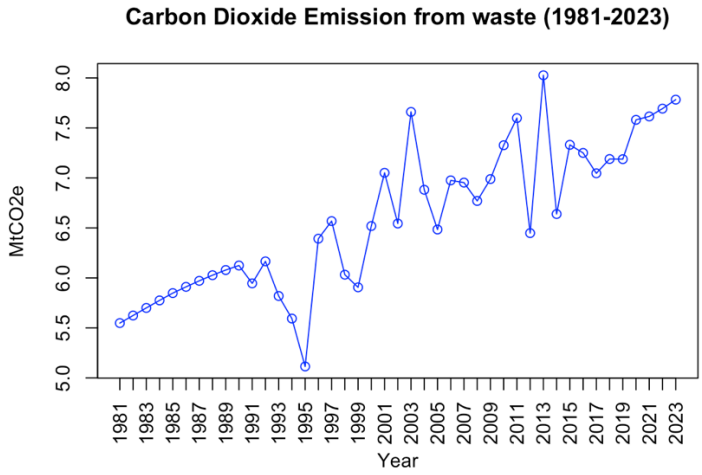

Figure 1. CO2 Emission from waste between 1981 to 2023

Figure 1 shows a general upward trend in CO2 emissions, with fluctuations between 1993 and 2014. These fluctuations could be attributed to environmental policies or legislation passed during the time.

To determine if further transformation of the time series is necessary, we will examine the normality of the distribution of the current data.

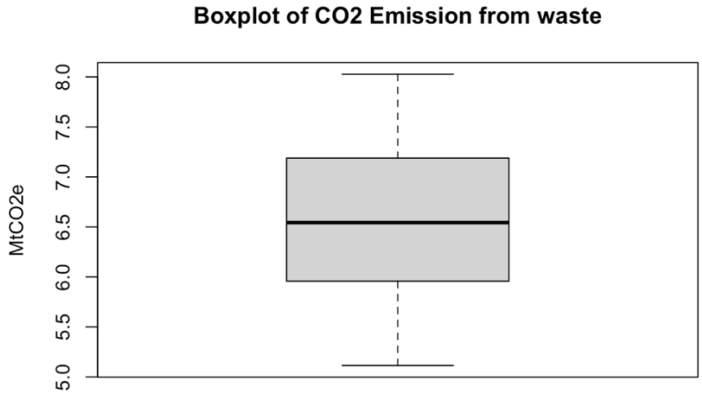

Figure 2. Boxplot of CO2 emission with equal whisker lengths and a central median line, indicating that emission data is normally distributed.

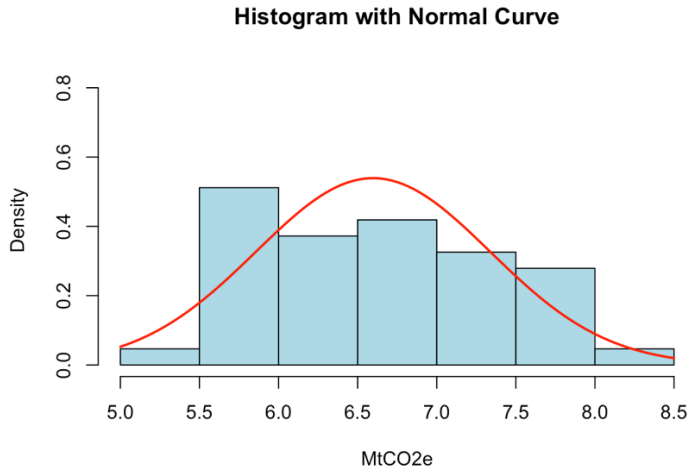

Figure 3. Histogram plot of CO2 emission data indicates that the data is normally distributed.

Both boxplot and histogram suggest normality of CO2 emission data and as such, we will not perform further data transformation.

Time Series Decomposition Analysis

To examine our time series in their individual components, we will use `decompose()` function. However, the result was erroneous as there is a lack of seasonality in the annual CO2 emission data. To confirm the lack of seasonality, we plotted an autocorrelation function (ACF) plot (Figure 4 below).

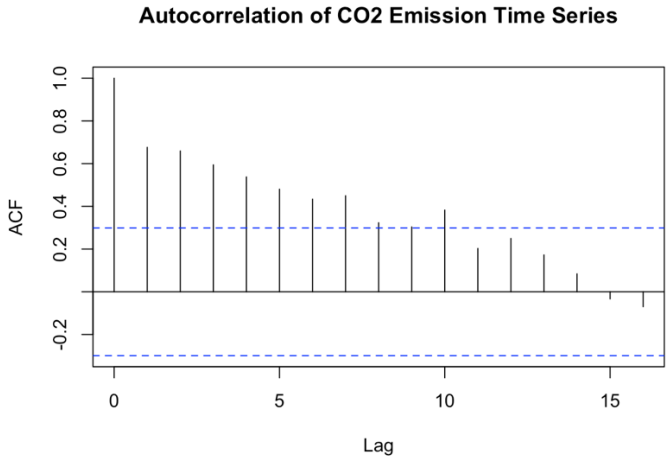

Figure 4. ACF plot showing slow decay to 0. This indicates that there is a trend and no seasonality in CO2 emission.

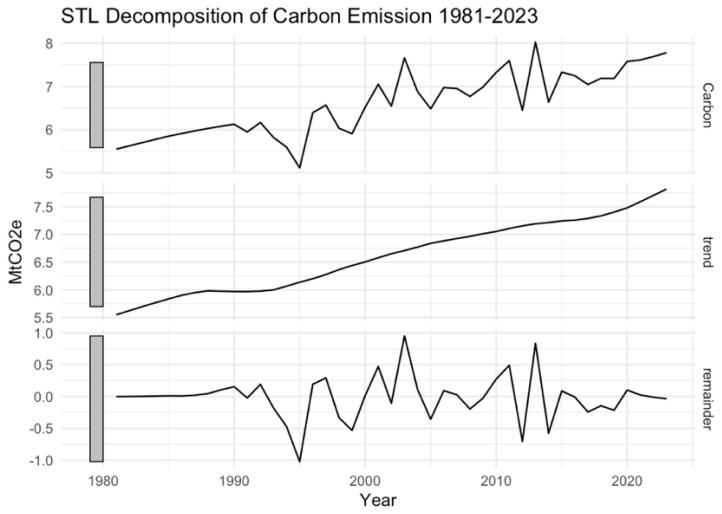

Figure 5. Time series decomposition components

As the data exhibits a trend but no seasonality, we used `STL()` instead to extract the trend and remainder components despite the absence of seasonality. In Figure 5, CO2 emission exhibits a clear upward trend. To further examine the remainder (residuals), we used `acf()` to plot an autocorrelation plot of the remainders (Figure 6 below). The plot shows no significant autocorrelation, indicating that the decomposition has fully captured the structure of the data in terms of trend and residuals.

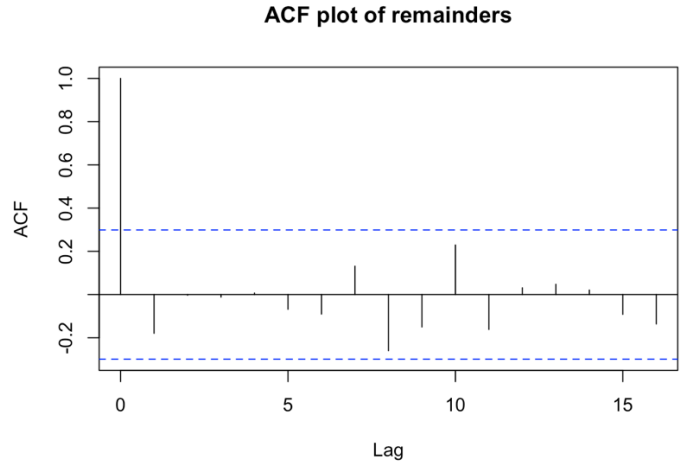

Figure 6. ACF plot of remainders from STL decomposition

To examine the trend component, we fitted a linear regression line (Figure 7). Examining the model parameters, we see an R-squared value of 0.986, indicating over 98% of the variability within the trend component can be explained by time (year), indicating good model fit.

Linear regression equation: Trend (CO2 Mte) = -94.98 + 0.0507 (Year)

This equation suggests that CO2 emissions increased by a unit of 0.0507 each year, indicating an upward trend. As the p-values of both intercept (-94.98) and coefficient of year (0.0507) are <0.05, they are significant in explaining the trend in our data.

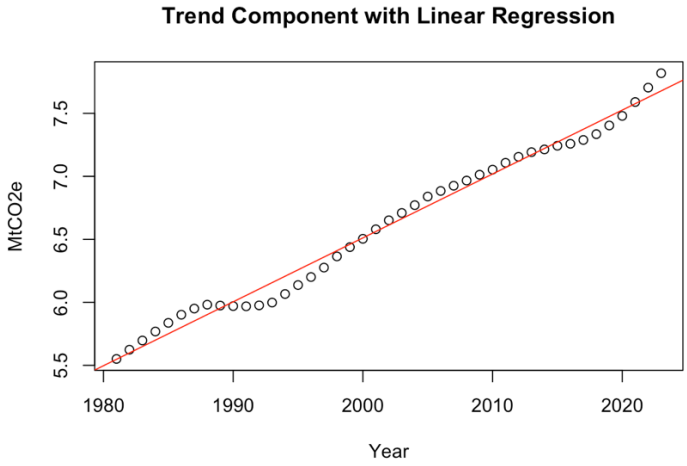

Figure 7. Trend component with fitted linear regression line

Moving Average Model

To forecast past series CO2 emission (1981 to 2022), we will create a timeseries object for our emission data beginning in 1981 and ending in 2022. For constructing moving averages, we will use `rollapply()` with order sizes of 3 and 5. The model's performance will be assessed and compared between the two order sizes.

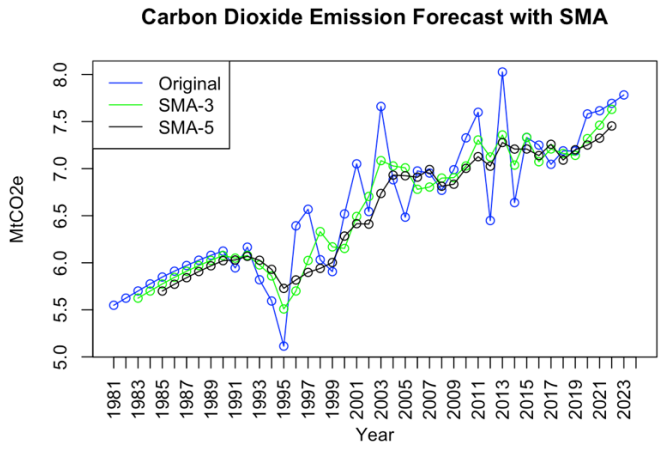

Figure 8. Simple moving average (SMA) plot with original, moving average order 3 and moving average order 5

Figure 8 shows that a larger order (5 instead of 3), resulted in greater smoothing of the data as an order of 5 uses the average of the 5 previous values as compared to 3. A larger order is better at capturing long term trends but performs poorly in responding to short-term fluctuations in comparison to using a smaller order. Generally, moving averages may not be ideal for CO2 emissions data– while it smooths past values, it adapts poorly to data with strong trend or seasonality. However, since our CO2 emission data suggests a trend with no seasonality, using moving averages may still be appropriate.

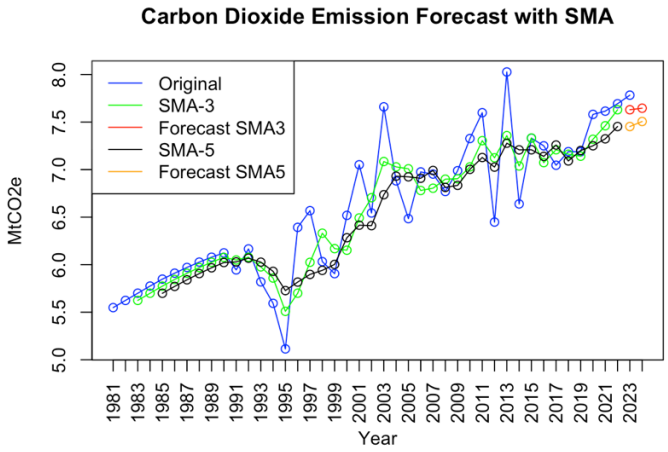

Figure 9. Moving average plot with forecasted values for 2023 and 2024

Based on visual inspection of Figure 9, we see that forecasted values (2023 and 2024) of SMA-5 (in orange) are lower in comparison to SMA-3 (in red).

Table 2. Forecasted values for 2023 and 2024 using SMA3 and SMA5

	SMA-3 Forecast	SMA-5 Forecast
2023	7.63	7.45
2024	7.65	7.51

Evaluation on SMA forecasted values

Since we have the actual CO2 emission value for 2023, we will compare this value with the forecasted values constructed using SMA3 and SMA5. For forecast accuracy, we will use mean squared deviation (MSD) and mean absolute percentage error (MAPE) as evaluation parameters.

Table 3. Evaluation metrics for forecasted values computed using SMA-3 and SMA-5 in comparison to actual

	MSD	**MAPE**
SMA-3	0.0235	1.970
SMA-5	0.1088	4.238

Using SMA-3, the percentage of error is smaller at 1.97%, compared to SMA-5's error percentage of 4.24%. Similarly, SMA-3 has a lower mean squared deviation (0.02) than SMA-5 (0.11). Given the smaller evaluation metrics, we conclude that using a smaller order SMA-3 provides a closer fit to the actual CO2 emission data in 2023.

Simple Exponential Smoothing (SES)

While SES is effective for short term forecasting, it may perform poorly on time series with trend and/or seasonality. As such, using SES on the current CO2 emission data may not be the best method given that earlier decomposition has revealed an upward trend. We applied `ses()` function to emission data and allowed R to optimize the smoothing parameters. To assess model performance, we examined model residuals using `acf()` and `pacf()`. We then calculated MAPE, MSD, MAD and AIC values to evaluate model fit.

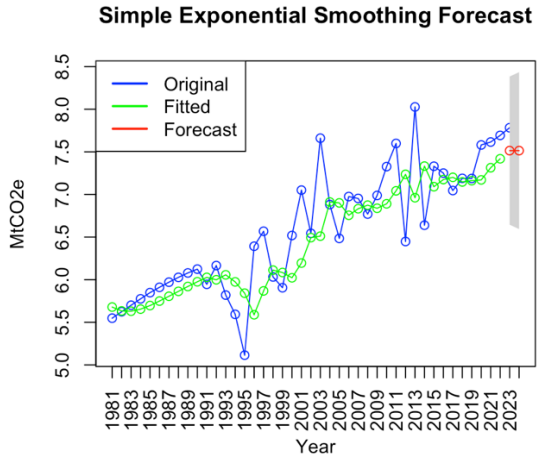

Figure 10. SES plot with forecasted values for 2023 and 2024

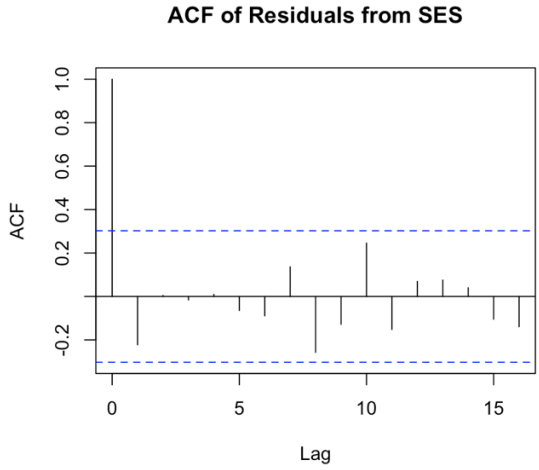

Figure 11. ACF of residuals from SES model exhibiting white noise behavior with no significant autocorrelation

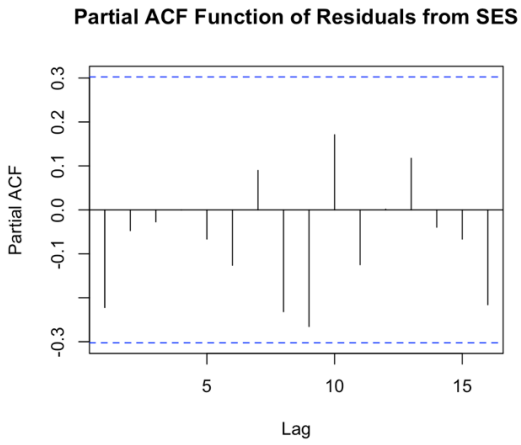

Figure 12. Partial ACF (PACF) of residuals from SES model exhibiting white noise behavior, showing no significant partial autocorrelation

Table 4. Evaluation metrics for model fit using SES

MAPE	4.739
MSD	0.187
MAD	0.317
AIC	92.602

SES model forecasts CO2 emission for 2023 and 2024 to be 7.514 after smoothing, as shown in Figure 10. Figures 11 and 12 demonstrate that the residuals from the SES model exhibit characteristics of white noise – suggesting adequate capture of CO2 emission data structure by the SES model. This is further supported by a p-value of 0.29 (>0.05) from the Ljung-Box test on the residuals, confirming no significant autocorrelation, supporting that the SES model is appropriate for CO2 emission forecasting.

Holt's Double Exponential Smoothing

As Holt's double exponential smoothing method assumes linear trend and non-seasonality of data, this method is most appropriate as it aligns with the structure of CO2 emission data as established in the decomposition earlier. While Holt's method works well for consistent linear trend, it has limited flexibility in terms of drastic directional changes in the trend. To fit our data, we used `holt()` and allowed R to optimize the alpha and beta parameters. We then evaluated model performance using `acf()` and `pacf()` on the model's residuals, and calculated MAPE, MSD, MAD and AIC for model fit.

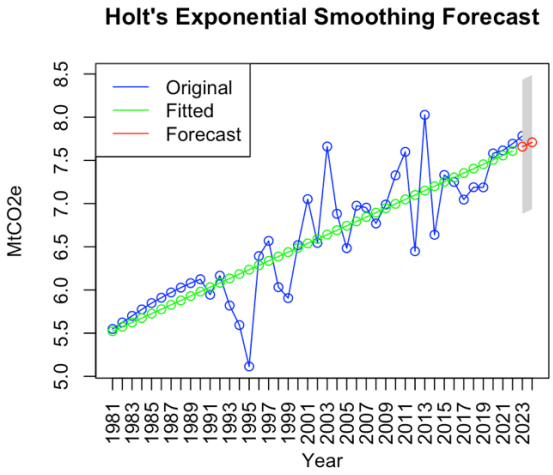

Figure 13. Holt's linear exponential smoothing forecast

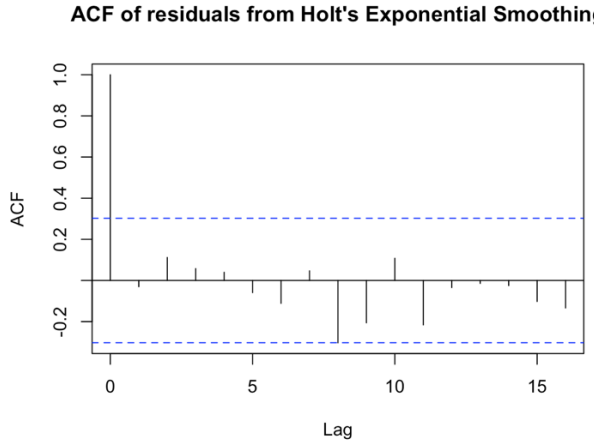

Figure 14. ACF of residuals from Holt's double exponential smoothing demonstrating characteristics of white noise except for Lag 8, where significant autocorrelation is present in the residuals

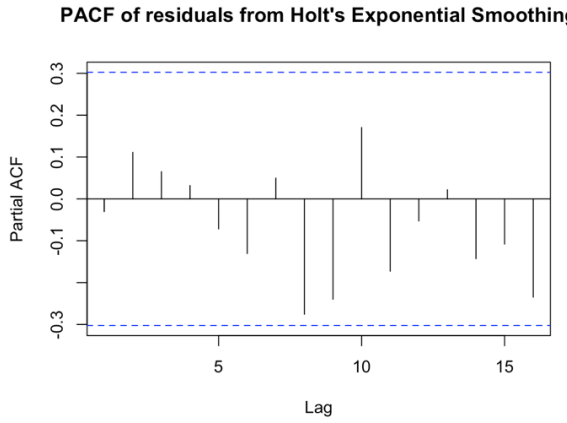

Figure 15. PACF of residuals from Holt's double exponential smoothing indicates no significant autocorrelation unlike ACF

Table 5. Evaluation metrics for model fit using Holt's double exponential smoothing

MAPE	4.078
MSD	0.142
MAD	0.265
AIC	84.954

Holt's double exponential smoothing has fitted past values into a linear line with an upward trend and forecasted CO2 emission for 2023 and 2024 to be 7.659 and 7.709 respectively (Figure 13). While the model has captured the CO2 emission data reasonably well based on ACF (Figure 14) and PACF (Figure 15) of the residuals, significant autocorrelation of residuals might still be present at one lag in the ACF plot. However, the Ljung-Box test on residuals returned a p-value of 0.46 (>0.05), indicating no significant autocorrelation. Additionally, Holt's method also demonstrated a good model fit in comparison to SES, based on the evaluation metrics in Table 5.

ARIMA

To construct the ARIMA model to forecast CO2 emission, we first need to determine the values for p (AR order), d (differencing order) and q (MA order). As ARIMA requires stationary data, we tested the stationarity of CO2 emission data using the Augmented Dickey-Fuller (ADF) test. After conducting `adf.test()`, the initial p-value was 0.386 (>0.05), indicating non-stationarity. After using `diff()` to difference the data and conducting another ADF test, the new p-value is less than 0.01, indicating stationarity after first differencing (d=1). To determine the values for p and q, we conducted ACF and PACF analyses on the differenced data.

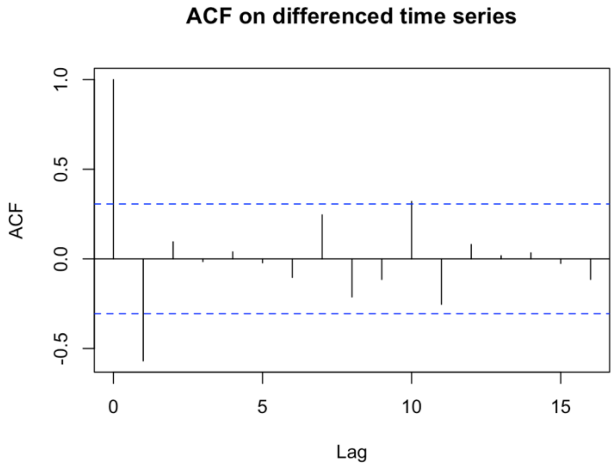

Figure 16. ACF plot showing exponential decay that quickly cuts off to 0 after lag 1 indicating that an MA (q) order of 1 may be appropriate

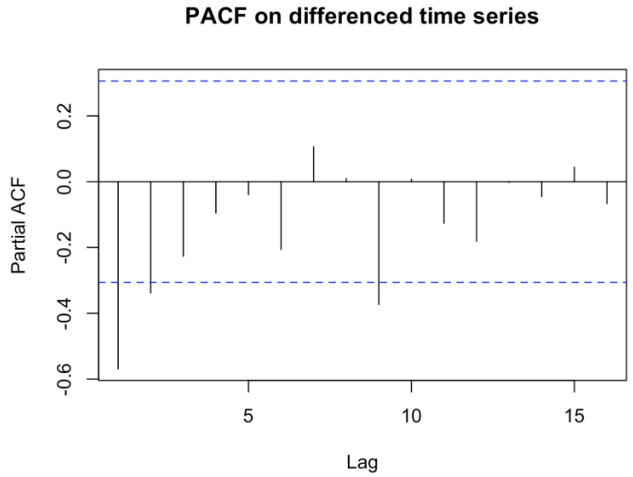

Figure 17. PACF plot showing slow decay, indicating that an AR (p) order of 0 may be appropriate

Taken together, we establish that p,d,q to be 0,1,1 repsectively. With this, we will fit our CO2 emission data using `Arima()` with the respective parameters for AR, difference order and MA.

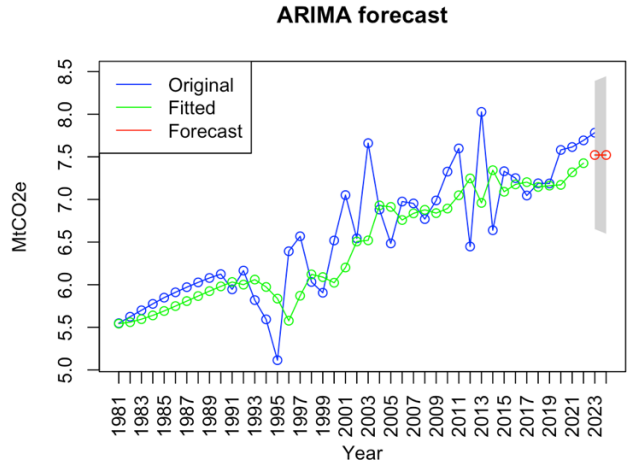

Figure 18. ARIMA forecast

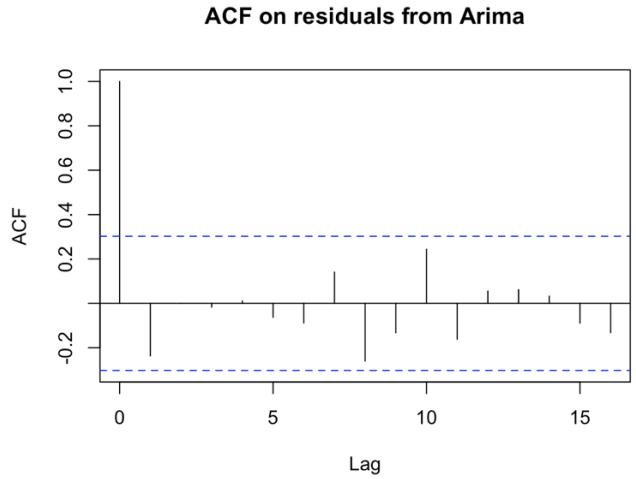

Figure 19. ACF plot of residuals showing no significant autocorrelation

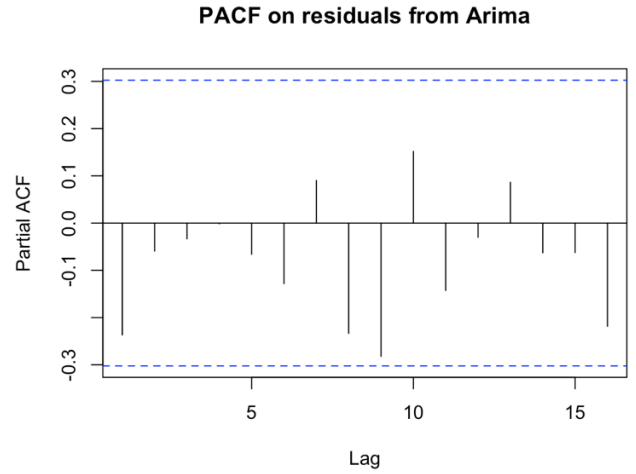

Figure 20. PACF plot of residuals showing no significant autocorrelation

Table 6. Evaluation metrics for model fit using ARIMA

MAPE	4.735
MSD	0.187
MAD	0.316
AIC	53.176

Both ACF and PACF plots (Figure 19 and 20) show no autocorrelation of residuals, indicating adequate capture of data structure by ARIMA model. This is further confirmed by Ljung-Box test on residuals returning p-value of 0.27.

Using `summary()`, the equation for ARIMA(0,1,1) is : $y_t = y_{t-1} - 0.6404\epsilon_{t-1} + \epsilon_t$

With a significant negative MA term (-0.64), the model suggests a partial correction in the opposite direction if the previous step experienced a large increment or decrement.

The ARIMA model, based on the first-order differencing, has removed the upward trend component previously seen during decomposition – indicating a stochastic and random, rather than deterministic trend. While earlier exploratory analyses using moving averages and Holt's double exponential smoothing suggested an upward trend, ARIMA's differencing has removed this trend, suggesting random pattern in the CO2 emission data.

Comparison of forecast accuracy between the different models

To further assess the forecast accuracy of the different models, we compared forecasted 2023 CO2 emission value to the actual 2023 CO2 emission value and calculated the following error metrics:

Table 7. Evaluation metrics measuring forecast accuracy

	SMA-3	**SES**	**Holt's method**	**ARIMA**
MAPE	1.970	3.452	1.598	1.970
MSD	0.0235	0.0722	0.0155	0.0684

In conclusion, Holt's double exponential smoothing provided the most accurate forecast for CO2 emission in 2023, as evidenced with the lowest MAPE and MSD. These metrics indicate that it has the smallest forecast error, making the Holt's double exponential model the most reliable for CO2 emission forecasting.

Relationship between CO2 emission and net national income per capita

We first examined the correlation between CO2 emission and income per capita, and obtained a Pearson correlation coefficient r = 0.827, indicating strong positive linear correlation between the two variables – suggesting that CO2 emissions increase as national income per capita increases. We then constructed a linear regression model to examine how income per capita influence CO2 emission.

```
Call:
lm(formula = Carbon ~ Income, data = merged_df)

Residuals:
     Min       1Q   Median       3Q      Max
-1.19261 -0.17964  0.06141  0.21560  1.11306

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 5.317e+00  1.506e-01  35.302  < 2e-16 ***
Income      9.487e-05  1.009e-05   9.404 8.63e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4216 on 41 degrees of freedom
Multiple R-squared:  0.6832,    Adjusted R-squared:  0.6755
F-statistic: 88.43 on 1 and 41 DF,  p-value: 8.632e-12
```

Figure 21. Screenshot of linear regression model

The linear regression equation obtained is: CO2 emission (y) = 5.317 + (9.487×10^{-5}) x Income

The regression results show a statistically significant relationship between CO2 emission and income per capita. Adjusted R-squared of 0.675 also suggests that income explains

approximately 67.5% of variation in CO2 emissions, demonstrating a substantial relationship between the two variables.

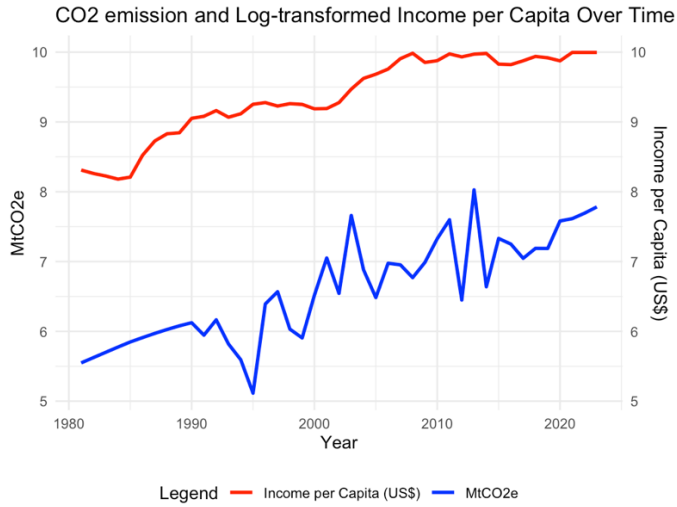

Figure 22. Dual-axis plot showing trends in CO2 emission (blue) and income per capita (red) over time. Income per capita has been log-transformed for scaling consistency. The strong positive correlation between the two variables is also evident with both displaying an upward trend over time.

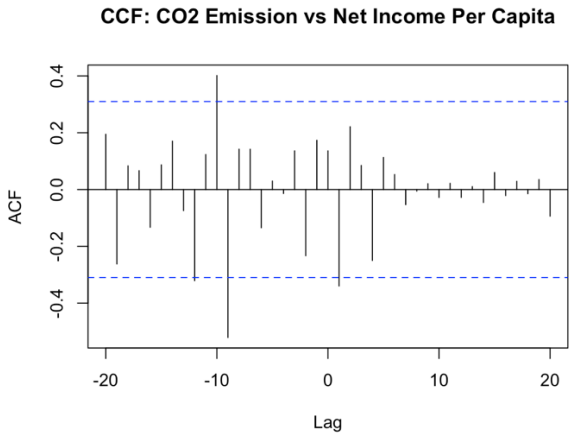

Figure 23. Cross correlation function (CCF) plot between CO2 emission and net national income per capita reveals two key patterns: 1) negative spike at lag -9 – indicating higher emissions 9 years ago correlate with lower income in 2023 and 2) positive spike at lag -10 – indicating higher emissions 10 years ago correlate with higher income in 2023

Although the strong Pearson coefficient shows that CO2 emission and income moves together over time, the CCF plot suggests a more nuanced and delayed relationship between CO2 emissions and national income, where past emissions have both positive and negative impact on national income over time. A granger test conducted also indicates a non-causal relationship between the two (p-value = 0.39).

Taken together, our results suggest that while there is significant correlation between CO2 emissions and national income, they do not share a causal relationship. This implies that the

observations between the two may be influenced by broader regional economic conditions rather than a direct effect on each other.

Conclusion

This analysis primarily focused on CO2 emissions, using time series decomposition to examine underlying trends and residuals. We applied various smoothing and forecasting methods – including moving averages of different orders, SES, Holt's Double Exponential Smoothing and ARIMA – to predict future CO2 emissions. Amongst the models, Holt's model demonstrated most potential in accurately predicting future emission. Additionally, our examination of the relationship between CO2 emissions and national income revealed a more nuanced and indirect connection. While there is correlation, the relationship is not causal, suggesting influence of broader conditions on observed trends and patterns.

References

Grossman, G. M., & Krueger, A. B. (1995). Economic growth and the environment. *The Quarterly Journal of Economics,* 110(2), 353–377. https://doi.org/10.2307/2118443

Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *Journal of Economic and Social Measurement*, 29(1-3), 123-125. https://doi.org/10.3233/JEM-2004-0211

Appendix

```
# Load libraries

library (tidyverse)

library(dplyr)

library (zoo)

library (ggplot2)

library (forecast)

library (tseries)

library(scales)

library(urca)

library(stats)

library(lubridate)

library(TTR)

library(tsibble)

library(fable)

library(feasts)

library(lmtest)


# Data import and cleaning ####

# Import carbon emission csv

carbon_df <- read.csv("~/Desktop/ANL557/ANL557 ECA/carbon.csv")


# Keep only the first row

carbon_df <- carbon_df[1, ]


# Rename year columns to keep only the years

colnames(carbon_df) <- gsub("^X|\\.\\.YR[0-9]+\\.", "",
colnames(carbon_df))

print(carbon_df)


# Remove specific columns using select()
```

```
carbon_df <- carbon_df %>%
  select(-Country.Name, -Country.Code, -Series.Code)

# Convert year columns to numeric
carbon_df[, -1] <- lapply(carbon_df[, -1], as.numeric)

# Reshape from wide to long format
carbon_df <- carbon_df %>%
  pivot_longer(cols = -Series.Name, names_to = "Year", values_to = "Carbon") %>%
  mutate(Year = as.numeric(Year))  # Convert Year column to numeric

# Remove the Series.Name column and keep only Year and Income
carbon_df <- carbon_df %>%
  select(Year, Carbon)

# Check for outliers
carbon_df_iqr <- quantile(carbon_df$Carbon, probs = c(0.25, 0.75), na.rm = TRUE) + c(-1.5, 1.5) * IQR(carbon_df$Carbon, na.rm=TRUE)
carbon_df$Carbon[(carbon_df$Carbon< carbon_df_iqr[1]) | (carbon_df$Carbon > carbon_df_iqr[2])]

# No outliers and missing values after checking
carbon_df <- as.data.frame(carbon_df)  # Converts to a base R data frame for analysis
# Statistics####
# Basic statisitics
summary(carbon_df) # Standard Statistics
var(carbon_df$Carbon) # Variance
sd(carbon_df$Carbon) # Standard Deviation
max(carbon_df$Carbon) # Maximum
min(carbon_df$Carbon) # Minimum
```

```
range(carbon_df$Carbon) # Range


# Time series plot
plot(carbon_df, type = "o", col = "blue",
     xlab = "Year", ylab = "MtCO2e",
     main = "Carbon Dioxide Emission from waste (1981-2023)",
     xaxt = "n")  # Disable x axis so we can customize it)


# Customize x-axis to display all the years and slant the labels
axis(1, at = seq(1981, 2023, by = 1), labels = seq(1981, 2023, by =
1), las = 2)  # Slanted labels


# Convert to time series object (incl 2023)
carbon_ts <- ts(carbon_df$Carbon, start=1981,end=2023, frequency =
1)


# Boxplot for normality test
boxplot(carbon_ts, main="Boxplot of CO2 Emission from waste", ylab=
"MtCO2e")


# Plot the histogram
hist_model <- hist(carbon_ts, breaks = 20, probability = TRUE, plot
= FALSE)


# Set ylim so that peak of curve does not cut off
hist(carbon_ts, probability = TRUE, col = "lightblue", main =
"Histogram with Normal Curve", xlab = "MtCO2e",ylim = c(0,
max(hist_model$density) * 1.2))


# Add the normal distribution curve
curve(dnorm(x, mean = mean(carbon_ts), sd = sd(carbon_ts)), col =
"red", lwd = 2, add = TRUE)


# ACF
```

```
acf(carbon_ts, main = "Autocorrelation of CO2 Emission Time Series")


# ACF will also show that our time series has no seasonality



# STL decomposition method

carbon_df = carbon_df %>% as_tsibble(index = Year) #convert df to
tsibble

df_decom = carbon_df %>% model(stl = STL(Carbon ~ trend())) %>%
components()

autoplot(df_decom) + scale_y_continuous(labels = label_comma()) +
theme_minimal() +

  labs(title = "STL Decomposition of Carbon Emission 1981-2023",
subtitle = NULL, x = "Year", y = "MtCO2e")


# No seasonality, explore trend and remainder

# Examine trend component

trend_df <- data.frame(

  year = 1981:2023,  # or the range of years in your time series

  trend = df_decom$trend

)


# Fit a linear regression model

trend_model <- lm(trend ~ year, data = trend_df)


# Plot the trend and the fitted linear regression line

plot(trend_df$year, trend_df$trend, main = "Trend Component with
Linear Regression", xlab = "Year", ylab = "MtCO2e")

abline(trend_model, col = "red")  # Adds the regression line to the
plot


# View the model summary

summary(trend_model)
```

```
# Fit a linear regression model to the data
lm_model_carbon <- lm(Carbon ~ Year, data = carbon_df)
summary(lm_model_carbon)

# Plot the trend line
ggplot(carbon_df, aes(x = Year, y = Carbon)) +
  geom_line() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  labs(title = "Trend Analysis with Linear Regression")

# Explore remainders from STL
summary(df_decom)

# ACF and PACF of residuals
acf(df_decom$remainder, main="ACF plot of remainders") #from the plot can see that remainder behaves like white noise random none crossed threshold
pacf(df_decom$remainder, main="Partial ACF plot of remainders")

carbon_df <- as.data.frame(carbon_df)  # Converts to a base R data frame

# Create time series for carbon data 1981 to 2022  (exc 2023)
carbon_ts_2022 <- ts(carbon_df$Carbon, start=1981, end=2022)

# Moving average####
carbon_ts_2022_ma3 <- carbon_ts_2022 #time series for ma order3

# SMA window size 3
window_size3 <- 3
```

```
# Calculate the 3year Simple Moving Average (SMA3) for population
sma3 <- zoo::rollapply(carbon_ts_2022_ma3, width = window_size3, FUN = mean, align = 'right', fill = NA)

# Create a vector to store the forecasted values
forecasted_values_ma3 <- numeric(2)  # 2 years forecast (2023-2024)

# Start the rolling forecast for 2023 to 2024
for (i in 1:2) {
  # Get the rolling moving average for the current window
  sma <- zoo::rollapply(carbon_ts_2022_ma3, width = window_size3, FUN = mean, align = 'right', fill = NA)

  # Forecast for next year is the last value of the moving average
  forecasted_values_ma3[i] <- tail(sma, n = 1)

  # Extend the time series to include the forecasted year
  carbon_ts_2022_ma3 <- ts(c(carbon_ts_2022_ma3, forecasted_values_ma3[i]), start = c(1981), frequency = 1)
}

# Combine the forecasted values with the original data
forecast_years_ma3 <- 2023:2024
forecast_data_ma3 <- data.frame(Year = forecast_years_ma3, Carbon = forecasted_values_ma3)
forecast_data_ma3
sma3

# Combine original data with forecasted data
carbon_forecast3_df <- rbind(carbon_df, forecast_data_ma3)
```

```
# SMA for order 5

carbon_ts_2022_ma5 <- carbon_ts_2022 #time series for ma order5


# SMA window size 3

window_size5 <- 5


# Calculate the 5year Simple Moving Average (SMA3) for population

sma5 <- zoo::rollapply(carbon_ts_2022_ma5, width = window_size5, FUN
= mean, align = 'right', fill = NA)


# Create a vector to store the forecasted values

forecasted_values_ma5 <- numeric(2)  # 2 years forecast (2023-2024)


# Start the rolling forecast for 2023 to 2024

for (i in 1:2) {

  # Get the rolling moving average for the current window

  sma <- zoo::rollapply(carbon_ts_2022_ma5, width = window_size5,
FUN = mean, align = 'right', fill = NA)


  # Forecast for next year is the last value of the moving average

  forecasted_values_ma5[i] <- tail(sma, n = 1)


  # Extend the time series to include the forecasted year

  carbon_ts_2022_ma5 <- ts(c(carbon_ts_2022_ma5,
forecasted_values_ma5[i]), start = c(1981), frequency = 1)

}


# Combine the forecasted values with the original data

forecast_years_ma5 <- 2023:2024

forecast_data_ma5 <- data.frame(Year = forecast_years_ma5, Carbon =
forecasted_values_ma5)

forecast_data_ma5
```

```
# Plot the original data
plot(carbon_df$Year, carbon_df$Carbon, type = 'o', col = 'blue',
     main = 'Carbon Dioxide Emission Forecast with SMA', xlab = 'Year', ylab = 'MtCO2e',
     xlim = c(1981, 2024), xaxt = 'n')  # Adjust xlim to show 1981 to 2028 range
axis(1, at = 1981:2024, labels = 1981:2024, las = 2)  # Set custom axis labels

# Add the 3-year moving average (SMA3) to the plot
lines(sma3, col = 'green', type = 'o')

# Add the 5-year moving average (SMA5) to the plot
lines(sma5, col = 'black', type = 'o')

# Add legend for original, sma3 and sma5
legend("topleft", legend = c("Original", "SMA-3", "SMA-5"), col = c("blue", "green","black"), lty = 1)

# Add the forecasted values for sma3
lines(forecast_years_ma3, forecasted_values_ma3, col = 'red', type = 'o')

# Add the forecasted values for sma5
lines(forecast_years_ma5, forecasted_values_ma5, col = 'orange', type = 'o')

# Add legend for all
legend("topleft", legend = c("Original", "SMA-3", "Forecast SMA3", "SMA-5", "Forecast SMA5"), col = c("blue", "green", "red", "black","orange"), lty = 1)
```

```
# Extract actual 2023 emission value

actual_2023 <- carbon_df[carbon_df$Year == 2023,"Carbon" ]

actual_2023 <- as.numeric(actual_2023) #Convert to numeric for
calculation



# Extract 2023 from forecast

forecast_2023_sma3 <- forecasted_values_ma3[1]

forecast_2023_sma3 <- as.numeric(forecast_2023_sma3) #Convert to
numeric for calculation


# Calculate MSD for SMA3

msd_sma3 <- mean((forecast_2023_sma3 - actual_2023)^2)

msd_sma3


# Calculate MAPE for SMA3

mape_sma3 <- mean(abs((forecast_2023_sma3 - actual_2023) /
actual_2023)) * 100

mape_sma3


# Calculate MSD for SMA5

# Extract 2023 from forecast

forecast_2023_sma5 <- forecasted_values_ma5[1]

forecast_2023_sma5 <- as.numeric(forecast_2023_sma5)


# Calculate MSD for SMA5

msd_sma5 <- mean((forecast_2023_sma5 - actual_2023)^2)

msd_sma5


# Calculate MAPE for SMA5

mape_sma5 <- mean(abs((forecast_2023_sma5 - actual_2023) /
actual_2023)) * 100
```

```
mape_sma5


# Simple Exponential Smoothing####

# Create time series for exponential smoothing

carbon_ts_2022_ses <- carbon_ts_2022


# Fit SES model

ses_model <- ses(carbon_ts_2022_ses, h=2)  # Forecast for the 2023
to 2024


# Evaluation

accuracy(ses_model)


# SES model fit

summary(ses_model)

# MAD

mean(abs(ses_model$residuals))

# MSD

mean(ses_model$residuals^2)

# AIC

ses_model$model$aic


# Extract 95CI bounds

lower_ci_ses <- ses_model$lower[, 2]

upper_ci_ses <- ses_model$upper[, 2]


# Extract time indices

time_original_ses <- as.numeric(time(carbon_ts_2022_ses))  # Convert
to numeric

time_forecast_ses <- seq(max(time_original_ses) + 1, by = 1,
length.out = length(ses_model$mean))  # Future years
```

```
# Total range for y-axis

range_total_ses <- range(carbon_ts_2022_ses,
ses_model$mean,lower_ci_ses, upper_ci_ses)

range_total_ses


# Plot original data (incl 2023)

plot(carbon_ts, type = "o", col = "blue",lwd = 1, xlab = "Year",
ylab = "MtCO2e",

     main = "Simple Exponential Smoothing Forecast",
xlim=c(1981,2024), ylim= range_total_ses, xaxt = 'n')

axis(1, at = 1981:2024, labels = 1981:2024, las = 2)  # Set custom
axis labels

# Add fitted values (smoothed historical data)

lines(ses_model$fitted, col = "green", lwd = 1, type="o")


# Add 95% CI

polygon(c(time_forecast_ses, rev(time_forecast_ses)),

        c(upper_ci_ses, rev(lower_ci_ses)),

        col = "lightgrey", border = NA)


# Add forecasted values

lines(ses_model$mean, col = "red", lwd = 1, lty = 1, type="o")


# Add legend

legend("topleft", legend = c("Original", "Fitted", "Forecast"),

       col = c("blue", "green", "red"), lwd = 2, lty = c(1,1,1))


# Forecasted SES values

ses_model$mean



# ACF PACF on SES residuals
```

```
acf(ses_model$residuals, main= "ACF of Residuals from SES")

pacf(ses_model$residuals, main="Partial ACF Function of Residuals
from SES")


# p-value SES residue

Box.test(ses_model$residuals, lag=10, type = "Ljung-Box")


# Extract 2023 SES forecast

forecast_2023_ses <- ses_model$mean [1]

forecast_2023_ses <- as.numeric(forecast_2023_ses)


# Calculate MSD for SES

msd_ses <- mean((forecast_2023_ses - actual_2023)^2)

msd_ses


# Calculate MAPE for SES

mape_ses <- mean(abs((forecast_2023_ses - actual_2023) /
actual_2023)) * 100

mape_ses



# Holt Exponential Smoothing####

# Create time series to use for Holt ES

carbon_ts_2022_h <- carbon_ts_2022

# Holt Exponential smoothing

holt_model <- holt(carbon_ts_2022_h, h=2)


# Extract 95CI

lower_ci_h <- holt_model$lower[, 2]

upper_ci_h <- holt_model$upper[, 2]


# Total range
```

```
range_total_h <- range(carbon_ts_2022_h, holt_model$mean,lower_ci_h,
upper_ci_h)

range_total_h


# Extract time indices

time_original_h <- as.numeric(time(carbon_ts_2022_h))  # Convert to
numeric

time_forecast_h <- seq(max(time_original_h) + 1, by = 1, length.out
= length(holt_model$mean))  # Future years


# Plot original and forecast (incl 2023)

plot(carbon_ts, type = "o", col = "blue", lwd = 1, xlab = "Year",
ylab = "MtCO2e",

     main = "Holt's Exponential Smoothing Forecast",

     xlim = c(1981, 2024), xaxt = 'n',

     ylim= range_total_h)

axis(1, at = 1981:2024, labels = 1981:2024, las = 2) #custom axis


# Add fitted values (green line)

lines(time_original_h, holt_model$fitted, col = "green", lwd = 1,
type="o")


# Add 95% CI(shaded grey area)

polygon(c(time_forecast_h, rev(time_forecast_h)),

        c(upper_ci_h, rev(lower_ci_h)),

        col = "lightgrey", border = NA)

# Add forecasted values (red dashed line) with correct time index

lines(time_forecast_h, holt_model$mean, col = "red", lwd = 1, lty
=1, type="o")


# Add a legend

legend("topleft", legend = c("Original", "Fitted", "Forecast"),

       col = c("blue", "green", "red"), lwd = 1, lty = c(1, 1, 1))
```

```
# Holt forecast values
holt_model$mean

# ACF PACF residues
acf(holt_model$residuals, main="ACF of residuals from Holt's
Exponential Smoothing")
pacf(holt_model$residuals, main="PACF of residuals from Holt's
Exponential Smoothing")

# p-value SES residue
Box.test(holt_model$residuals, lag=10, type = "Ljung-Box")

# qqplot of residuals
qqnorm(holt_model$residuals, main="QQ Plot of Holt Model Residuals")
qqline(holt_model$residuals, col="red", lwd=2)  # Add reference line

# Evaluation of SES model fit
accuracy(holt_model)
# MAD of model fit
mean(abs(holt_model$residuals))
# MSD of model fit
mean(holt_model$residuals^2)
# AIC of model fit
holt_model$model$aic

# Compare actual 2023 to holt 2023 forecast
forecast_2023_h <- holt_model$mean [1]
forecast_2023_h <- as.numeric(forecast_2023_h)

# Calculate MSD for Holt
msd_h <- mean((forecast_2023_h - actual_2023)^2)
```

```
msd_h


# Calculate MAPE for Holt

mape_h <- mean(abs((forecast_2023_h - actual_2023) / actual_2023)) *
100

mape_h



# Holt Winter Exponential smoothing####

# Create time series for HW

carbon_ts_2022_hw <- carbon_ts_2022


# Apply Holt-Winters (automatically selects best alpha, beta, gamma)

hw_model <- hw(carbon_ts_2022_hw, h = 2,seasonal = "additive")  #
Forecast 2 years


#unable to proceed with HW as timeseries freq needs to be >1


# ARIMA####

# Create time series for arima

carbon_ts_2022_am <- carbon_ts_2022


# Stationary test - if p<0.05, reject null hypothesis, series is
stationary

# Perform the Augmented Dickey-Fuller Test

adf.test(carbon_ts_2022_am)

# p=0.386, need to difference the data


# Check how many differences are needed

ndiffs(carbon_ts_2022_am)


#ndiff is 1 indicate first order differencing
```

```
# Perform differencing
carbon_am_diff <- diff(carbon_ts_2022_am, differences = 1)

#adf test on differenced ts
adf.test(carbon_am_diff)
#p<0.01 hence ts now stationary and can be used for arima

# ACF PACF on differenced data to find p,d,q
acf(carbon_am_diff, main= "ACF on differenced time series")
pacf(carbon_am_diff, main= "PACF on differenced time series")

#acf decays quickly to 0
#pacf decays slowly to 0
#suggestive of MA (0,1,q)
#acf cuts off after lag1 indicative of q=1

# For the model with ARIMA(0,1,1) assuming use the original ts and
not differenced
manual_arima_model <- Arima(carbon_ts_2022_am, order = c(0, 1, 1))

# Evaluation of model fit
summary(manual_arima_model)
# MAD of model fit
mean(abs(manual_arima_model$residuals))
# MSD of model fit
mean(manual_arima_model$residuals^2)
# AIC of model fit
AIC(manual_arima_model)

# ARIMA forecast with the manual ari
```

```
forecast_values_am <- forecast::forecast(manual_arima_model, h=2)

forecast_values_am

forecast_values_am$mean #Forecasted values


# Extract 95CI

lower_ci_am <- forecast_values_am$lower[, 2]

upper_ci_am <- forecast_values_am$upper[, 2]


# Total range

range_total_am <- range(carbon_ts_2022_am,
forecast_values_am$mean,lower_ci_am, upper_ci_am)

range_total_am


# Extract time indices

time_original_am <- as.numeric(time(carbon_ts_2022_am))  # Convert
to numeric

time_forecast_am <- seq(max(time_original_am) + 1, by = 1,
length.out = length(forecast_values_am$mean))  # Future years


# Plot original and forecast (incl 2023)

plot(carbon_ts, type = "o", col = "blue", lwd = 1, xlab = "Year",
ylab = "MtCO2e",

     main = "ARIMA forecast",

     xlim = c(1981, 2024), xaxt = 'n',

     ylim= range_total_h)

axis(1, at = 1981:2024, labels = 1981:2024, las = 2) #custom axis


# Add fitted values (green line)

lines(time_original_am, manual_arima_model$fitted, col = "green",
lwd = 1, type="o")


# Add 95% CI(shaded grey area)

polygon(c(time_forecast_am, rev(time_forecast_am)),
```

```
        c(upper_ci_am, rev(lower_ci_am)),
        col = "lightgrey", border = NA)

# Add forecasted values (red dashed line) with correct time index
lines(time_forecast_am, forecast_values_am$mean, col = "red", lwd =
1, lty =1, type="o")

# Add a legend
legend("topleft", legend = c("Original", "Fitted", "Forecast"),
       col = c("blue", "green", "red"), lwd = 1, lty = c(1, 1, 1))

# Evaluation of coeff
summary(manual_arima_model)
pvalue(manual_arima_model$coef)

# ACF PACF on arima residuals
acf(manual_arima_model$residuals, main="ACF on residuals from
Arima")
pacf(manual_arima_model$residuals, main="PACF on residuals from
Arima")
checkresiduals(manual_arima_model)

# Box test for p-value of residuals
Box.test(manual_arima_model$residuals, lag=10, type = "Ljung-Box")

# Compare actual 2023 to arima 2023 forecast
# Extract 2023 forecast from ARIMA
forecast_2023_arima <- forecast_values_am$mean [1]
forecast_2023_arima <- as.numeric(forecast_2023_arima)

# Calculate MSD for arima
msd_arima <- mean((forecast_2023_arima - actual_2023)^2)
```

```
msd_arima


# Calculate MAPE for arima

mape_arima <- mean(abs((forecast_2023_sma3 - actual_2023) /
actual_2023)) * 100

mape_arima

# Data import and cleaning of national income data ####

# Import national income csv

income_df <- read.csv("~/Desktop/ANL557/ANL557 ECA/income.csv")


# Keep only the first row

income_df <- income_df[1, ]


# Rename year columns to keep only the years

colnames(income_df) <- gsub("^X|\\.\\.YR[0-9]+\\.", "",
colnames(income_df))

print(income_df)


# Convert ".." to NA

income_df[income_df == ".."] <- NA  # Replaces ".." with NA


# Remove specific columns using select()

income_df <- income_df %>%

  select(-Country.Name, -Country.Code, -Series.Code)


# Convert year columns to numeric

income_df[, -1] <- lapply(income_df[, -1], as.numeric)


# Reshape from wide to long format

income_df <- income_df %>%

  pivot_longer(cols = -Series.Name, names_to = "Year", values_to =
"Income") %>%
```

```
  mutate(Year = as.numeric(Year))  # Convert Year column to numeric


# Remove the Series.Name column and keep only Year and Income
income_df <- income_df %>%
  select(Year, Income)


# Check for missing values
income_df_missing <- is.na(income_df$Income)
income_df_missing


# Check for outliers
income_df_iqr <- quantile(income_df$Income, probs = c(0.25, 0.75),
na.rm = TRUE) + c(-1.5, 1.5) * IQR(income_df$Income, na.rm=TRUE)
income_df$Income[(income_df$Income< income_df_iqr[1]) |
(income_df$Income > income_df_iqr[2])]


# Ensure Year and Income are numeric
income_df$Year <- as.numeric(income_df$Year)
income_df$Income <- as.numeric(income_df$Income)


# Linear extrapolation
income_df$Income <- approx(income_df$Year, income_df$Income, xout =
income_df$Year, rule = 2)$y


# Merge income and carbon df
merged_df <- full_join(carbon_df, income_df, by = "Year")


# View merged data
head(merged_df)


# Convert to base R df
merged_df <- as.data.frame(merged_df)
```

```
# Correlation between CO2 and income
cor(merged_df$Carbon, merged_df$Income, use = "complete.obs")

# Fit a linear regression model
model_carbon_income <- lm(Carbon ~ Income, data = merged_df)

# View the model summary
summary(model_carbon_income)

# Plot merged_df
ggplot(merged_df) +
  # Plot carbon emissions on the left y-axis
  geom_line(aes(x = Year, y = Carbon, color = "Carbon Emissions"), size = 1) +
  scale_y_continuous(name = "Carbon Emissions", sec.axis = sec_axis(~ ., name = "Income per Capita")) +

  # Plot income on the right y-axis
  geom_line(aes(x = Year, y = Income / 1000, color = "Income per Capita"), size = 1) + # Scale income for better visibility

  # Customize the appearance of the plot
  labs(title = "Carbon Emissions and Income per Capita Over Time",
       x = "Year",
       y = "Carbon Emissions (units)",
       color = "Legend") +

  # Use a manual color scale for the legend
  scale_color_manual(values = c("Carbon Emissions" = "blue", "Income per Capita" = "red")) +
```

```
  # Customize the theme for better visibility
  theme_minimal() +
  theme(legend.position = "bottom")



# Log scale plot
ggplot(merged_df) +
  geom_line(aes(x = Year, y = Carbon, color = "MtCO2e"), size = 1) +
  scale_y_continuous(name = "MtCO2e", sec.axis = sec_axis(~ ., name
= "Income per Capita (US$)")) +
  geom_line(aes(x = Year, y = log(Income), color = "Income per
Capita (US$)"), size = 1) +
  labs(title = "CO2 emission and Log-transformed Income per Capita
Over Time", x = "Year", y = "MtCO2e", color = "Legend") +
  scale_color_manual(values = c("MtCO2e" = "blue", "Income per
Capita (US$)" = "red")) +
  theme_minimal() +
  theme(legend.position = "bottom")


# Convert income df to time series
# Convert to time series object (incl 2023)
income_ts <- ts(income_df$Income, start=1981,end=2023, frequency =
1)


# Check if income ts is stationary
adf.test(income_ts) #p=0.41 need to difference


# Differece
income_diff <- diff(income_ts, differences = 1)


# Check if differenced ts is stationary
adf.test(income_diff) #still non-stationary
```

```
# Difference again

income_diff2 <- diff(income_diff, differences = 1)

adf.test(income_diff2) # differenced data is now stationary


# Use stationary differenced carbon_am_diff

# Cross-Correlation Function (CCF)

ccf(carbon_am_diff, income_diff2, lag.max=20, main = "CCF: CO2
Emission vs Net Income Per Capita")


# Granger test to determine casuality

grangertest(income_diff2 ~ carbon_am_diff, order = 10)
```