1.0 Introduction

In terms of capture fisheries production, Malaysia ranks sixth in the ASEAN region and is in the top 20 countries with fishermen catching around one million tonnes of aquatic animals every year.

However, due to the large amount of fish and other seafood getting fished up, this puts pressure on the population of fish, causing them to be overfished. Overfishing happens when fish are caught at a rate faster than their population growth.

Experts predict that if overfishing continues to happen, all of the species being caught for food would go extinct by 2048. And in 2019, it has been revealed by the fisheries department that 96% of Malaysia's demersal fish stock have been lost in less than 60 years (Selan, 2021).

A way to combat the overexploitation of fish stocks is aquaculture, which is the practice of breeding and cultivating aquatic animals and plants. It acts as an alternative way to produce food for humans and helps ease the pressure put on fish stocks in the wild, allowing their populations to be able to be rebuilt. However, when compared to agriculture, aquaculture is still a recent development, with it only beginning in Malaysia during the 1920s (FOA, 2024).

2.0 Objective

To analyse the dataset and evaluate how effective is aquaculture in Malaysia and whether or not it has been improving over the years.

3.0 Problem Statements

1. Has the increased consumption of fish led to the increase in fish production levels?
2. Has the existence of aquaculture decreased the number of fish caught in the wild?
3. Has the existence of aquaculture helped decrease the number of overexploited fish?

4.0 Data Collection

The data used was collected from the database, Kaggle, under the name "Fish and Overfishing" by Serge Geukjian.

```
#Reading the datasets
df_Aquaculture = pd.read_csv('/Users/foongchoiwan/Desktop/Fish and Overfishing/aquaculture-farmed-fish-production.csv')
df_Aquaculture.head()
```

	Entity	Code	Year	Aquaculture production (metric tons)
0	Afghanistan	AFG	1969	60.0
1	Afghanistan	AFG	1970	60.0
2	Afghanistan	AFG	1971	60.0
3	Afghanistan	AFG	1972	60.0
4	Afghanistan	AFG	1973	60.0

```
df_CaptureFishery = pd.read_csv('/Users/foongchoiwan/Desktop/Fish and Overfishing/capture-fishery-production.csv')
df_CaptureFishery.head()
```

	Entity	Code	Year	Capture fisheries production (metric tons)
0	Afghanistan	AFG	1960	200.0
1	Afghanistan	AFG	1961	300.0
2	Afghanistan	AFG	1962	300.0
3	Afghanistan	AFG	1963	300.0
4	Afghanistan	AFG	1964	300.0

```
df_consumption = pd.read_csv('/Users/foongchoiwan/Desktop/Fish and Overfishing/fish-and-seafood-consumption-per-capita.csv')
df_consumption.head()
```

	Entity	Code	Year	Fish, Seafood- Food supply quantity (kg/capita/yr) (FAO, 2020)
0	Afghanistan	AFG	1961	0.03
1	Afghanistan	AFG	1962	0.03
2	Afghanistan	AFG	1963	0.03
3	Afghanistan	AFG	1964	0.03
4	Afghanistan	AFG	1965	0.03

```
df_types = pd.read_csv('/Users/foongchoiwan/Desktop/Fish and Overfishing/seafood-and-fish-production-thousand-tonnes.csv')
df_types.head()
```

	Entity	Code	Year	Commodity Balances - Livestock and Fish Primary Equivalent - Pelagic Fish - 2763 - Production - 5510 - tonnes	Commodity Balances - Livestock and Fish Primary Equivalent - Crustaceans - 2765 - Production - 5510 - tonnes	Commodity Balances - Livestock and Fish Primary Equivalent - Cephalopods - 2766 - Production - 5510 - tonnes	Commodity Balances - Livestock and Fish Primary Equivalent - Demersal Fish - 2762 - Production - 5510 - tonnes	Commodity Balances - Livestock and Fish Primary Equivalent - Freshwater Fish - 2761 - Production - 5510 - tonnes	Commodity Balances - Livestock and Fish Primary Equivalent - Molluscs, Other - 2767 - Production - 5510 - tonnes	Commodity Balances - Livestock and Fish Primary Equivalent - Marine Fish, Other - 2764 - Production - 5510 - tonnes
0	Afghanistan	AFG	1961	NaN	NaN	NaN	NaN	300.0	NaN	NaN
1	Afghanistan	AFG	1962	NaN	NaN	NaN	NaN	300.0	NaN	NaN
2	Afghanistan	AFG	1963	NaN	NaN	NaN	NaN	300.0	NaN	NaN
3	Afghanistan	AFG	1964	NaN	NaN	NaN	NaN	300.0	NaN	NaN
4	Afghanistan	AFG	1965	NaN	NaN	NaN	NaN	300.0	NaN	NaN

```
df_fishstocks = pd.read_csv('/Users/foongchoiwan/Desktop/Fish and Overfishing/fish-stocks-within-sustainable-levels.csv')
df_fishstocks.head()
```

	Entity	Code	Year	Share of fish stocks within biologically sustainable levels (FAO, 2020)	Share of fish stocks that are overexploited
0	Eastern Central Atlantic	NaN	2015	57.142860	42.857140
1	Eastern Central Atlantic	NaN	2017	57.142857	42.857143
2	Eastern Central Pacific	NaN	2015	86.666670	13.333330
3	Eastern Central Pacific	NaN	2017	86.666667	13.333333
4	Eastern Indian Ocean	NaN	2015	73.076920	26.923080

Figure 1.1

As seen in Figure 1.1 above, the data used from the dataset showcases:

1. The production of fish from capture fisheries and aquaculture per year
2. Seafood consumption levels
3. Types of seafood produced
4. Levels of sustainable and overexploited fish

5.0 Data Preprocessing & Cleaning

Since the data obtained is large and only data regarding Malaysia was analysed, only Malaysia was chosen via selecting the entity. Malaysia is part of the Southeast Pacific, so the "Entity" for "Southeast Pacific" was selected as well.

```
#Due to the large dataset, Malaysia is chosen
df_Aquaculture_MYS = df_Aquaculture[df_Aquaculture["Entity"]=="Malaysia"]
df_CaptureFishery_MYS = df_CaptureFishery[df_CaptureFishery["Entity"]=="Malaysia"]
df_consumption_MYS = df_consumption[df_consumption["Entity"]=="Malaysia"]
df_types_MYS = df_types[df_types["Entity"]=="Malaysia"]
```

```
#Malaysia is part of the Southeast Pacific
df_fishstocks_MYS =df_fishstocks[df_fishstocks["Entity"]=="Southeast Pacific"]
df_fishstocks_MYS.head()
```

The data frames only consist of Malaysia's, so the "Entity" and "Code" columns can be removed.

```
df_Aquaculture_MYS = df_Aquaculture_MYS.select_dtypes(include = 'number')
df_CaptureFishery_MYS = df_CaptureFishery_MYS.select_dtypes(include = 'number')
df_consumption_MYS = df_consumption_MYS.select_dtypes(include = 'number')
df_types_MYS = df_types_MYS.select_dtypes(include = 'number')
df_fishstocks_MYS = df_fishstocks_MYS.select_dtypes(include = 'number')
```

Once the data has been preprocessed, it can now be checked for any errors or missing values.

```
print(df_Aquaculture_MYS.info())
print(df_CaptureFishery_MYS.info())
print(df_consumption_MYS.info())
print(df_types_MYS.info())
print(df_fishstocks_MYS.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 59 entries, 6332 to 6390
Data columns (total 2 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Year                                59 non-null     int64
 1   Aquaculture production (metric tons)  59 non-null     float64
dtypes: float64(1), int64(1)
memory usage: 1.4 KB
None
<class 'pandas.core.frame.DataFrame'>
Index: 59 entries, 8200 to 8258
Data columns (total 2 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Year                                    59 non-null     int64
 1   Capture fisheries production (metric tons)  59 non-null     float64
dtypes: float64(1), int64(1)
memory usage: 1.4 KB
None
<class 'pandas.core.frame.DataFrame'>
Index: 57 entries, 5901 to 5957
Data columns (total 2 columns):
 #   Column                                                  Non-Null Count  Dtype
---  ------                                                  --------------  -----
 0   Year                                                    57 non-null     int64
 1   Fish, Seafood- Food supply quantity (kg/capita/yr) (FAO, 2020)  57 non-null     float64
dtypes: float64(1), int64(1)
memory usage: 1.3 KB
None
<class 'pandas.core.frame.DataFrame'>
Index: 53 entries, 5511 to 5563
Data columns (total 8 columns):
 #   Column                                                                                                  Non-Null Count  Dtype
---  ------                                                                                                  --------------  -----
 0   Year                                                                                                    53 non-null     int64
 1   Commodity Balances - Livestock and Fish Primary Equivalent - Pelagic Fish - 2763 - Production - 5510 - tonnes       53 non-null     float64
 2   Commodity Balances - Livestock and Fish Primary Equivalent - Crustaceans - 2765 - Production - 5510 - tonnes       53 non-null     float64
 3   Commodity Balances - Livestock and Fish Primary Equivalent - Cephalopods - 2766 - Production - 5510 - tonnes       53 non-null     float64
 4   Commodity Balances - Livestock and Fish Primary Equivalent - Demersal Fish - 2762 - Production - 5510 - tonnes       53 non-null     float64
 5   Commodity Balances - Livestock and Fish Primary Equivalent - Freshwater Fish - 2761 - Production - 5510 - tonnes     53 non-null     float64
 6   Commodity Balances - Livestock and Fish Primary Equivalent - Molluscs, Other - 2767 - Production - 5510 - tonnes     53 non-null     float64
 7   Commodity Balances - Livestock and Fish Primary Equivalent - Marine Fish, Other - 2764 - Production - 5510 - tonnes  53 non-null     float64
dtypes: float64(7), int64(1)
memory usage: 3.7 KB
None
<class 'pandas.core.frame.DataFrame'>
Index: 2 entries, 18 to 19
Data columns (total 3 columns):
 #   Column                                                  Non-Null Count  Dtype
---  ------                                                  --------------  -----
 0   Year                                                    2 non-null      int64
 1   Share of fish stocks within biologically sustainable levels (FAO, 2020)  2 non-null      float64
 2   Share of fish stocks that are overexploited             2 non-null      float64
dtypes: float64(2), int64(1)
memory usage: 64.0 bytes
None
```

Figure 2.1

As seen from the image above, there are no missing values. However, the “Year” was in integer format, so it had to be changed to datetime.

```
df_Aquaculture_MYS.reset_index(drop=True, inplace=True)
df_Aquaculture_MYS['Year'] =pd.to_datetime(df_Aquaculture_MYS['Year'], format='%Y')
df_Aquaculture_MYS.head()
```

	Year	Aquaculture production (metric tons)
0	1960-01-01	7841.0
1	1961-01-01	7911.0
2	1962-01-01	9968.0
3	1963-01-01	21941.0
4	1964-01-01	21302.0

```
df_CaptureFishery_MYS.reset_index(drop=True, inplace=True)
df_CaptureFishery_MYS['Year'] =pd.to_datetime(df_CaptureFishery_MYS['Year'], format='%Y')

df_CaptureFishery_MYS.head()
```

	Year	Capture fisheries production (metric tons)
0	1960-01-01	147800.0
1	1961-01-01	160100.0
2	1962-01-01	179877.0
3	1963-01-01	198200.0
4	1964-01-01	192300.0

```
df_consumption_MYS.reset_index(drop=True, inplace=True)
df_consumption_MYS['Year'] =pd.to_datetime(df_consumption_MYS['Year'], format='%Y')

df_consumption_MYS.head()
```

	Year	Fish, Seafood- Food supply quantity (kg/capita/yr) (FAO, 2020)
0	1961-01-01	20.36
1	1962-01-01	21.53
2	1963-01-01	23.33
3	1964-01-01	21.24
4	1965-01-01	20.01

```
df_fishstocks_MYS.reset_index(drop=True, inplace=True)
df_fishstocks_MYS['Year'] =pd.to_datetime(df_fishstocks_MYS['Year'], format='%Y')
```

```
df_fishstocks_MYS.head()
```

	Year	Share of fish stocks within biologically sustainable levels (FAO, 2020)	Share of fish stocks that are overexploited
0	2015-01-01	38.461540	61.538460
1	2017-01-01	45.454545	54.545455

Figure 2.3

The columns in df_types_MYS were renamed to have shorter names for easier readability when plotting the graph.

```
df_types_MYS.reset_index(drop=True, inplace=True)

df_types_MYS.head()
```

	Year	Commodity Balances - Livestock and Fish Primary Equivalent - Pelagic Fish - 2763 - Production - 5510 - tonnes	Commodity Balances - Livestock and Fish Primary Equivalent - Crustaceans - 2765 - Production - 5510 - tonnes	Commodity Balances - Livestock and Fish Primary Equivalent - Cephalopods - 2766 - Production - 5510 - tonnes	Commodity Balances - Livestock and Fish Primary Equivalent - Demersal Fish - 2762 - Production - 5510 - tonnes	Commodity Balances - Livestock and Fish Primary Equivalent - Freshwater Fish - 2761 - Production - 5510 - tonnes	Commodity Balances - Livestock and Fish Primary Equivalent - Molluscs, Other - 2767 - Production - 5510 - tonnes	Commodity Balances - Livestock and Fish Primary Equivalent - Marine Fish, Other - 2764 - Production - 5510 - tonnes
0	1961	66800.0	17500.0	1000.0	21100.0	3831.0	6480.0	51000.0
1	1962	77400.0	22600.0	1000.0	24100.0	3888.0	8580.0	52000.0
2	1963	84200.0	21400.0	1500.0	26900.0	4441.0	20400.0	61200.0
3	1964	90000.0	21800.0	1700.0	32700.0	5402.0	20900.0	41000.0
4	1965	96600.0	24600.0	1200.0	34200.0	5908.0	20800.0	46400.0

```
df_types_MYS.columns = ['Year', 'Pelagic Fish', 'Crustaceans', 'Cephalopods', 'Demersal Fish', 'Freshwater Fish', 'Molluscs, Other', 'Marine Fish, Other']
df_types_MYS = df_types_MYS.set_index('Year')
df_types_MYS.head()
```

Year	Pelagic Fish	Crustaceans	Cephalopods	Demersal Fish	Freshwater Fish	Molluscs, Other	Marine Fish, Other
1961	66800.0	17500.0	1000.0	21100.0	3831.0	6480.0	51000.0
1962	77400.0	22600.0	1000.0	24100.0	3888.0	8580.0	52000.0
1963	84200.0	21400.0	1500.0	26900.0	4441.0	20400.0	61200.0
1964	90000.0	21800.0	1700.0	32700.0	5402.0	20900.0	41000.0
1965	96600.0	24600.0	1200.0	34200.0	5908.0	20800.0	46400.0

Figure 2.4

6.0 Exploratory Data Analysis (EDA)

6.1 Data Visualisation

6.1.1 Production of Fish from Aquaculture

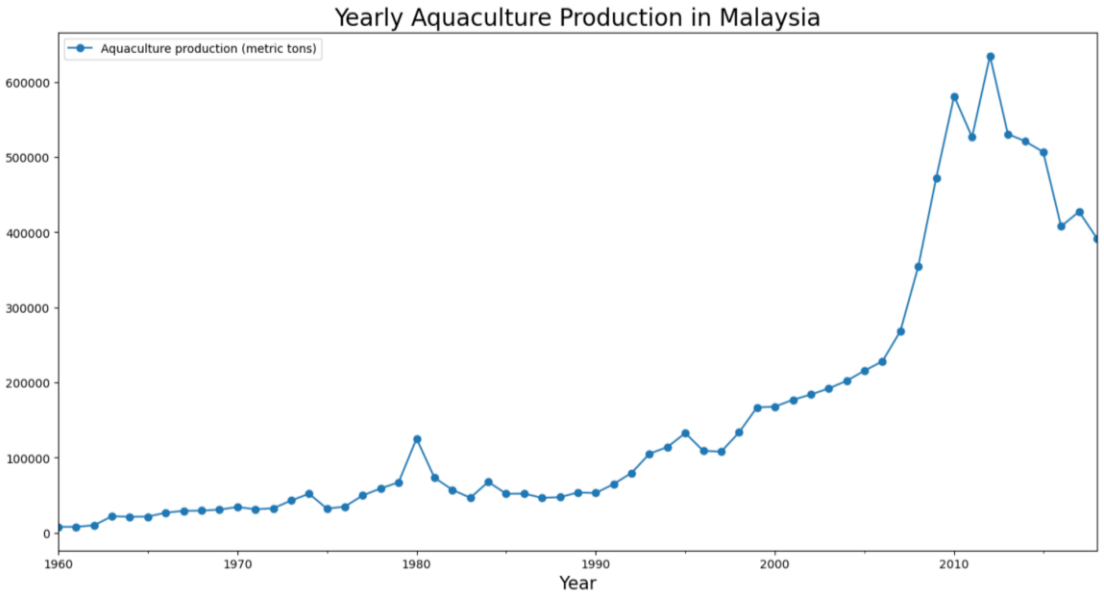


Figure 3.1.1 Yearly Aquaculture Production in Malaysia

As seen in Figure 3.1.1 above, the production of aquaculture in Malaysia per year has been slowly increasing until around 2006 with the exception of a sudden spike in 1980. Afterwards, it started

to see a significant increase in the production before reaching its peak in 2011. It then started to decrease after.

As the data is a time series, it can be decomposed into multiple components to gain a better understanding of its patterns and trends. The process of separating a time series into three main components: trend, seasonality, and noise is called “Time Series Decomposition”.

The trend in the time series represents the long-term movement in the data and the underlying pattern. Seasonality represents any repeating, short-term fluctuations caused by cycles. Residual, also known as noise, represents random variability that remains after the trend and seasonality.

```
#decomposition of the time series
decomposition = sm.tsa.seasonal_decompose(AquacultureMYS)
decomposed_AQ=pd.DataFrame()

decomposed_AQ["trend"] = decomposition.trend
decomposed_AQ["seasonal"]=decomposition.seasonal
decomposed_AQ["random_noise"]=decomposition.resid
fig2,(ax1, ax2, ax3) =plt.subplots(nrows=3, ncols=1, figsize=(12,15))
ax1.set_title("trend", fontsize =15)
decomposed_AQ["trend"].plot(ax=ax1)
ax2.set_title("seasonal", fontsize =15)
decomposed_AQ["seasonal"].plot(ax=ax2)
ax3.set_title("random noise", fontsize =15)
decomposed_AQ["random_noise"].plot(ax=ax3)
```

Additive decomposition of AquacultureMYS was done and the result was stored under decomposition so that it can be visualised using a line graph.

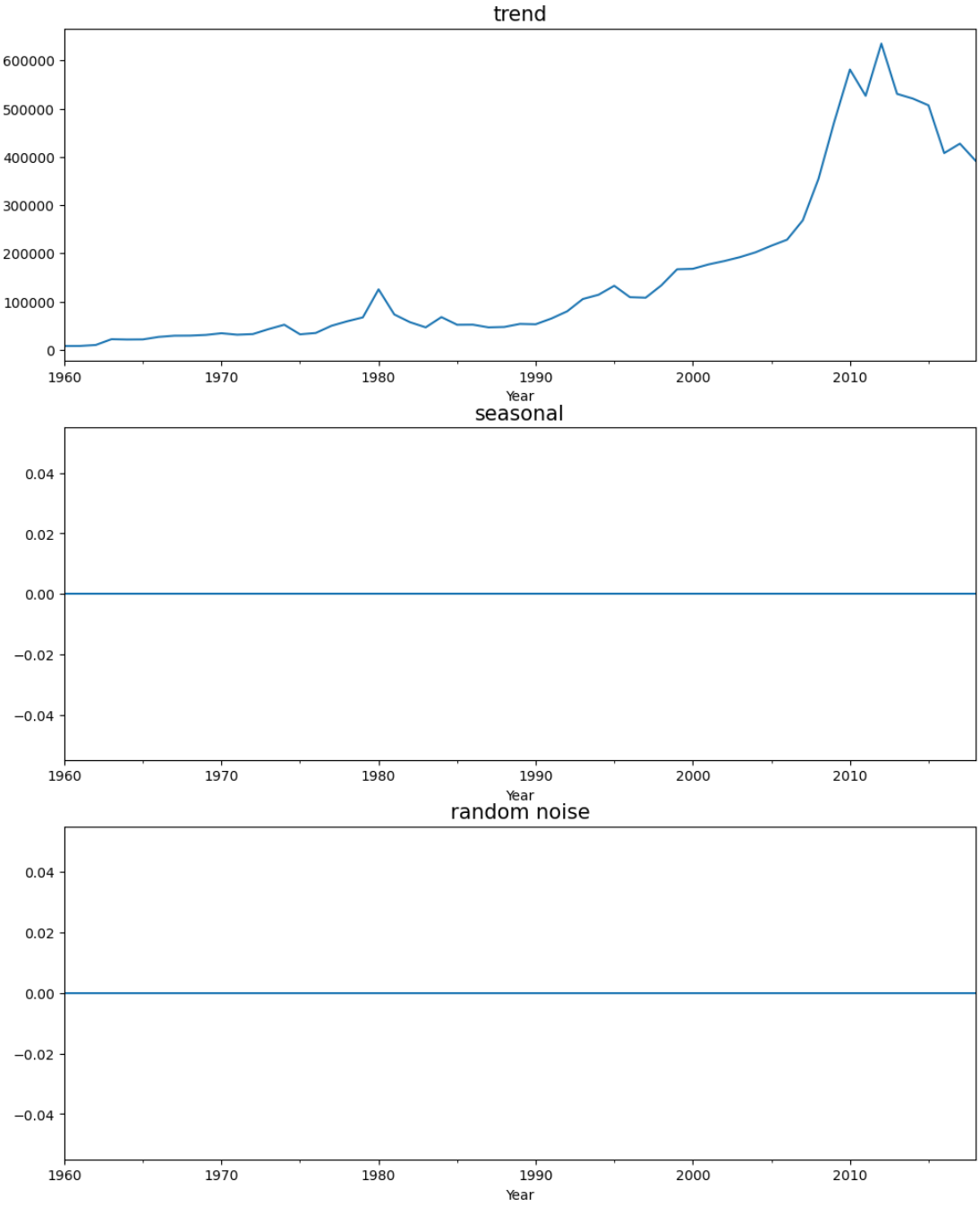


Figure 3.1.2

As seen in Figure 3.2.2, the trend is the graph itself. There is no seasonality in the data that can be observed, which may be due to the data being yearly and there is also no random noise remaining.

6.1.2 Production of Fish from Capture Fisheries

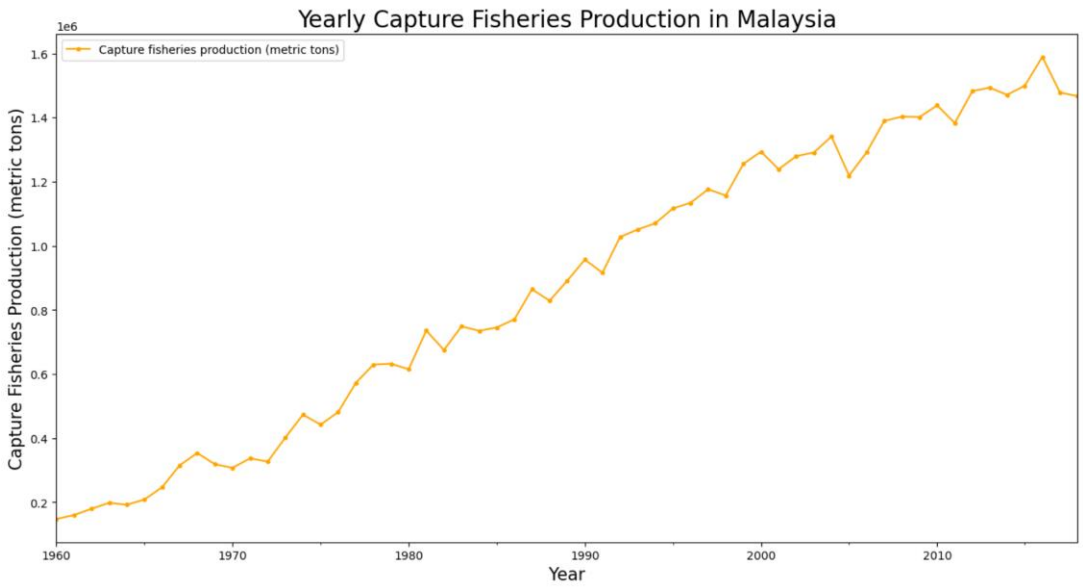


Figure 3.2.1

Overall, the production from capture fisheries is increasing with a few ups and downs in between, with it reaching its peak at around the year 2016, after which it started to decrease.

```
#decomposition of the time series
decomposition = sm.tsa.seasonal_decompose(CaptureFisheryMYS)
decomposed_CF=pd.DataFrame()

decomposed_CF["trend"] = decomposition.trend
decomposed_CF["seasonal"]=decomposition.seasonal
decomposed_CF["random_noise"]=decomposition.resid
fig4,(ax1, ax2, ax3) =plt.subplots(nrows=3, ncols=1, figsize=(12,15))
ax1.set_title("trend", fontsize =15)
decomposed_CF["trend"].plot(color = 'orange', ax=ax1)
ax2.set_title("seasonal", fontsize =15)
decomposed_CF["seasonal"].plot(color = 'orange',ax=ax2)
ax3.set_title("random noise", fontsize =15)
decomposed_CF["random_noise"].plot(color = 'orange',ax=ax3)
```

Additive decomposition of CaptureFisheryMYS was done and the result was stored under decomposition so that it can be visualised using a line graph.

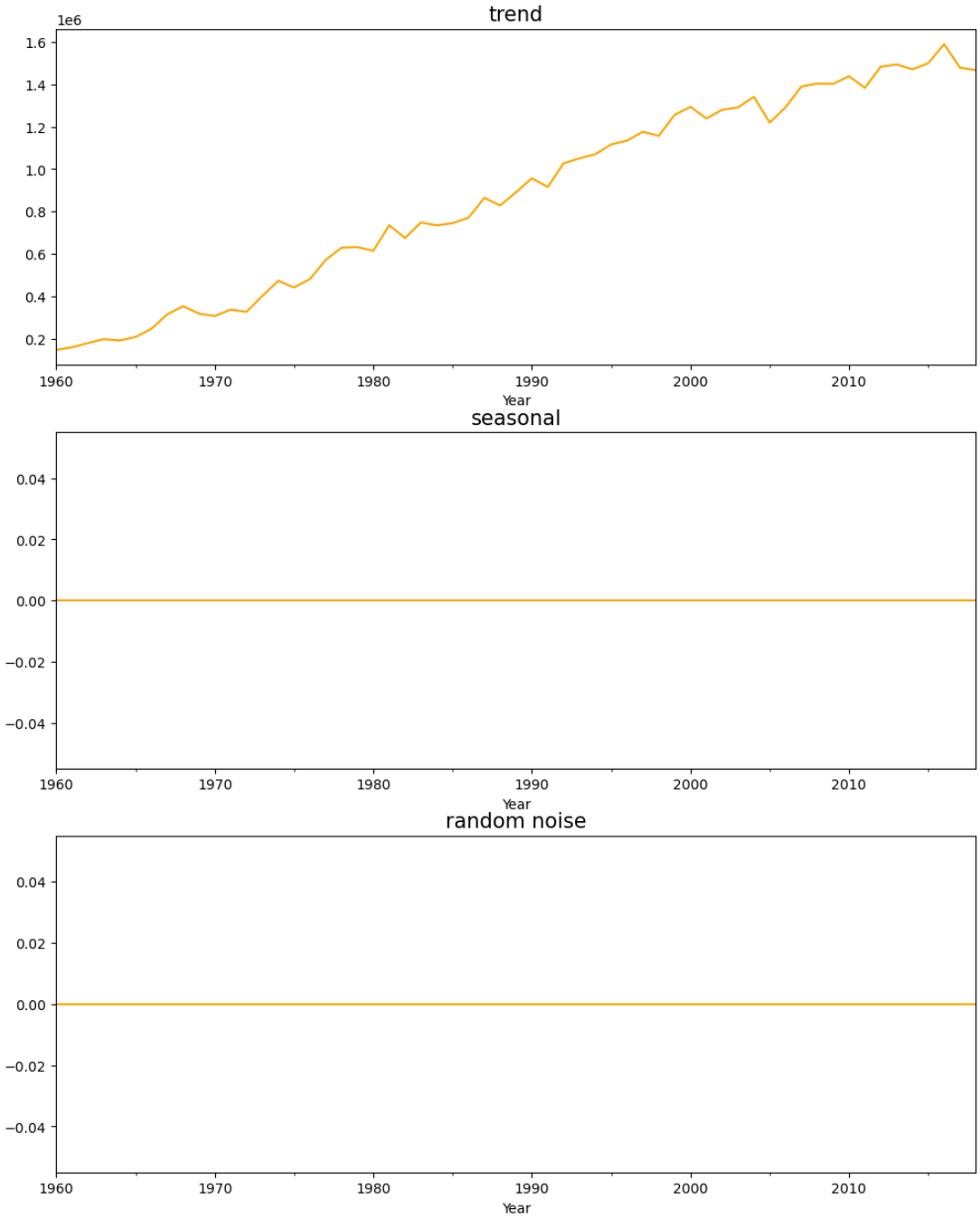


Figure 3.2.2

As seen in Figure 3.2.2, the decomposition of CaptureFisheryMYS is similar to the decomposition of AquacultureMYS as there is only a trend, which is the graph itself.

6.1.3 Consumption of Seafood by Malaysians

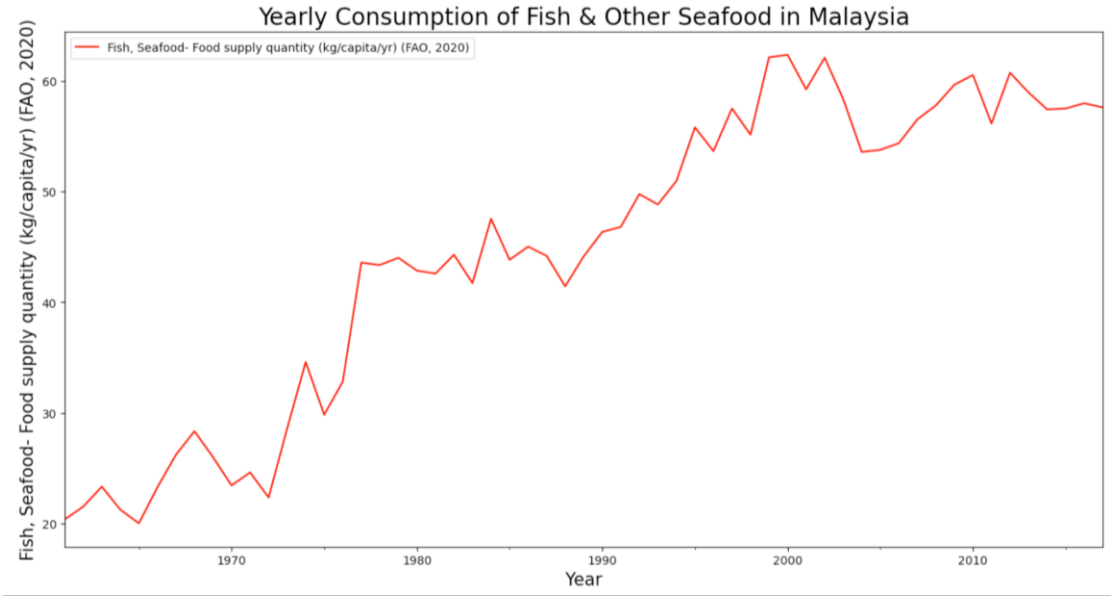


Figure 3.3.1

Overall, the graph above shows that the consumption of seafood by Malaysians increases each year until the sudden drop at around 2004.

```
decomposition = sm.tsa.seasonal_decompose(consumptionMYS)
decomposed_consum=pd.DataFrame()

decomposed_consum["trend"] = decomposition.trend
decomposed_consum["seasonal"]=decomposition.seasonal
decomposed_consum["random_noise"]=decomposition.resid
fig6,(ax1, ax2, ax3) =plt.subplots(nrows=3, ncols=1, figsize=(12,15))
ax1.set_title("trend", fontsize =15)
decomposed_consum["trend"].plot(color = 'red',ax=ax1)
ax2.set_title("seasonal", fontsize =15)
decomposed_consum["seasonal"].plot(color = 'red',ax=ax2)
ax3.set_title("random noise", fontsize =15)
decomposed_consum["random_noise"].plot(color = 'red',ax=ax3)
```

Additive decomposition is then done on consumptionMYS and graphed to be visualized.

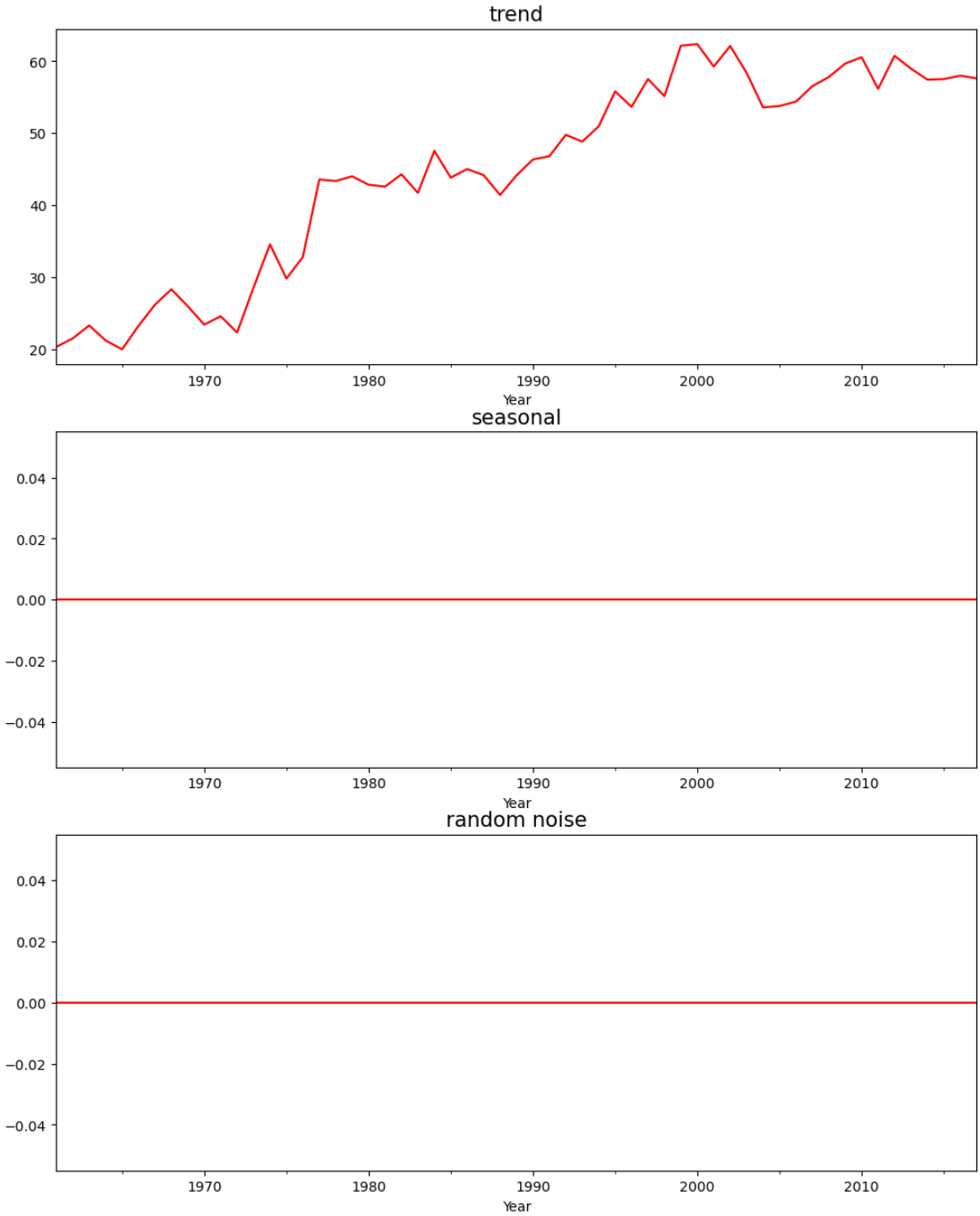


Figure 3.3.2

As seen in Figure 3.3.2, the trend is the time series itself and due to it being yearly data, no seasonality is shown. There is also no random noise.

6.1.4 Overexploited and Sustainable Fish Stocks in the Southeast Pacific

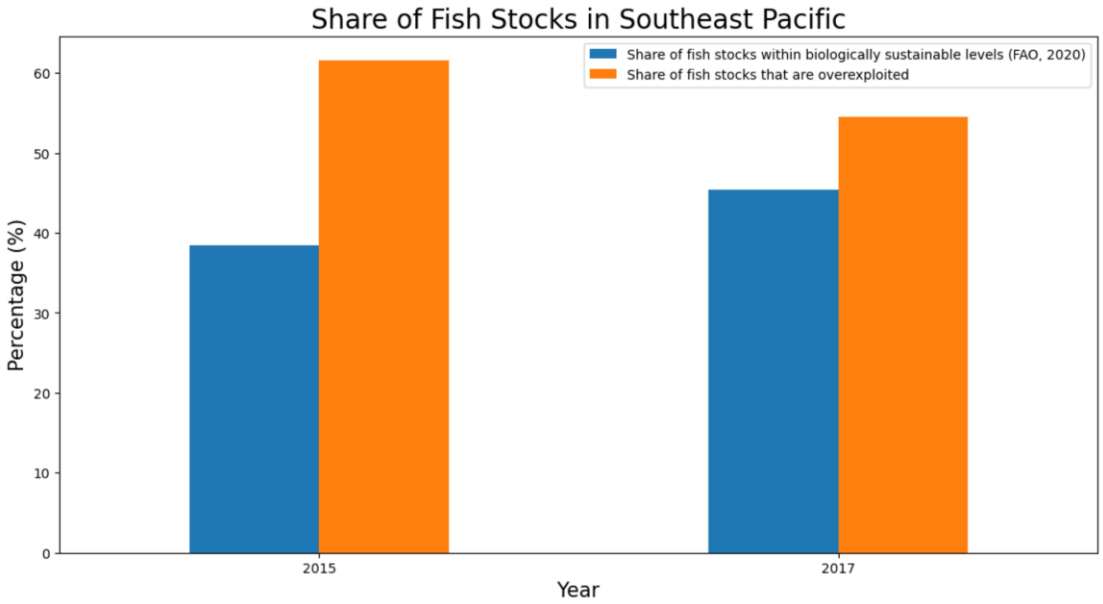


Figure 3.4

From 2015 to 2017, the percentage of overexploited fish stocks have decreased and the share of sustainable stocks have managed to increase.

6.1.5 Types of Seafood Produced

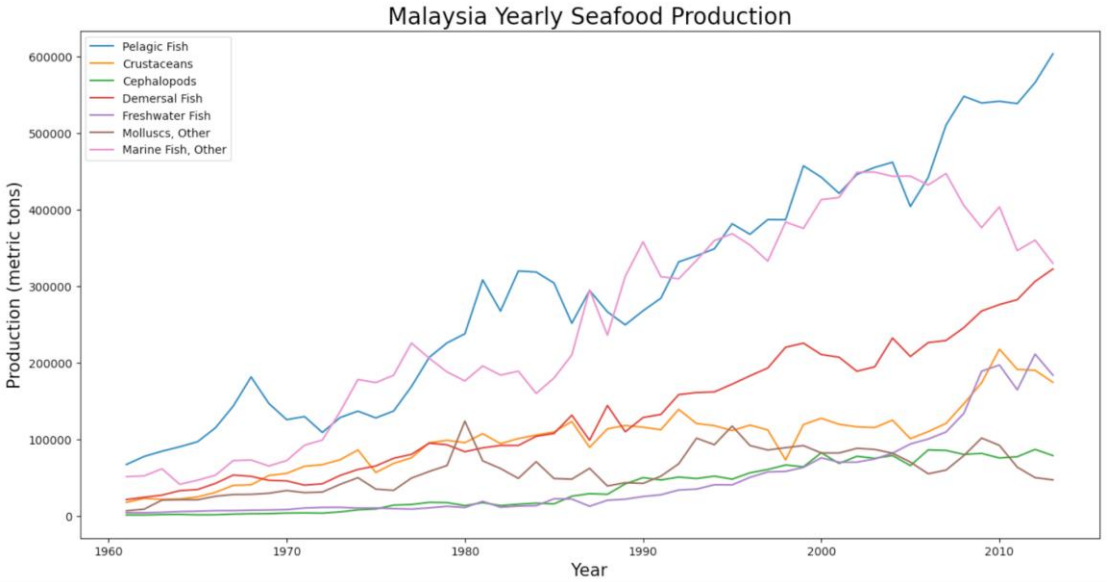


Figure 3.5.1

Figure 3.5.1 depicts the types of seafood produced in the form of a line graph. By visualizing the data in a line graph, the type of seafood with the largest production can be observed. It can also visualize the increase and decrease of the production levels of each type of marine animal.

For example, by using Figure 3.5.1, we are able to observe that the production levels of marine fish and other equivalent species started to decrease whereas the production levels of pelagic fish increased.

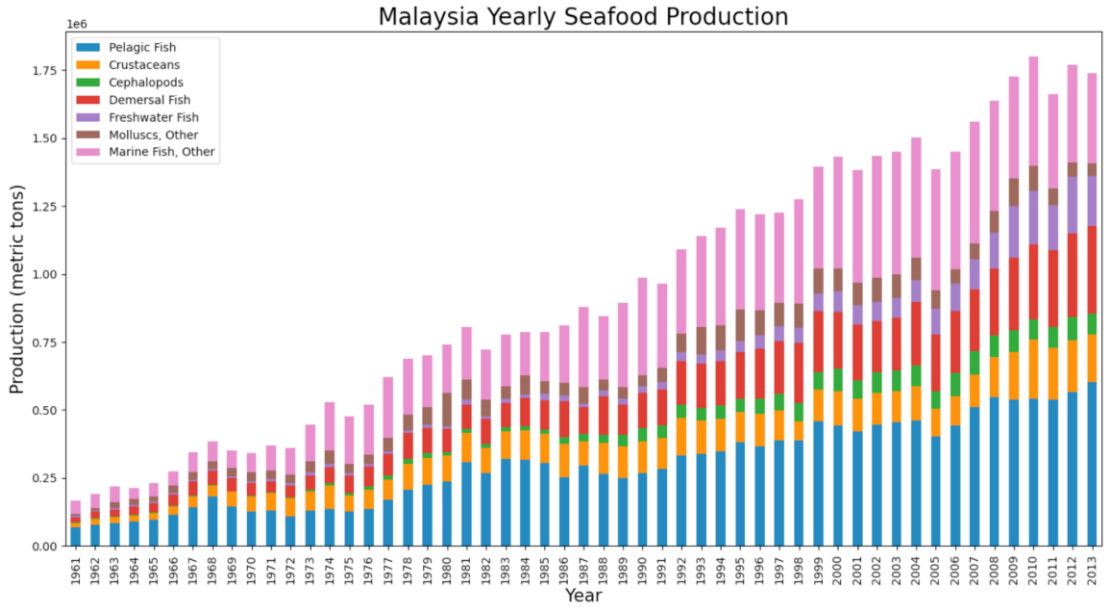


Figure 3.5.2

Figure 3.5.2 depicts the same data as Figure 3.5.1 in a bar chart format. This way the total production of fish and other seafood can be visualized. From the figure above, the highest total amount observed was during 2010.

6.1.6 Comparison between Capture Fisheries Production and Seafood Consumption

```
#The data is in different units, therefore they are normalised
Normalized_CaptureFishery = df_CaptureFishery_MYS.copy()
Normalized_Consumption = df_consumption_MYS.copy()

#Show percentages of total
Normalized_CaptureFishery['Capture fisheries production (metric tons)'] = df_CaptureFishery_MYS['Capture fisheries production (metric tons)'].div(np.sum(df_CaptureFishe
Normalized_Consumption['Fish, Seafood- Food supply quantity (kg/capita/yr) (FAO, 2020)'] = df_consumption_MYS['Fish, Seafood- Food supply quantity (kg/capita/yr) (FAO,
```

Consumption levels are in a different unit from capture fisheries, so the data had to be normalised by converting df_consumption_MYS and df_CaptureFishery_MYS into the percentages of their total.

Once converted, we can then plot the graph:

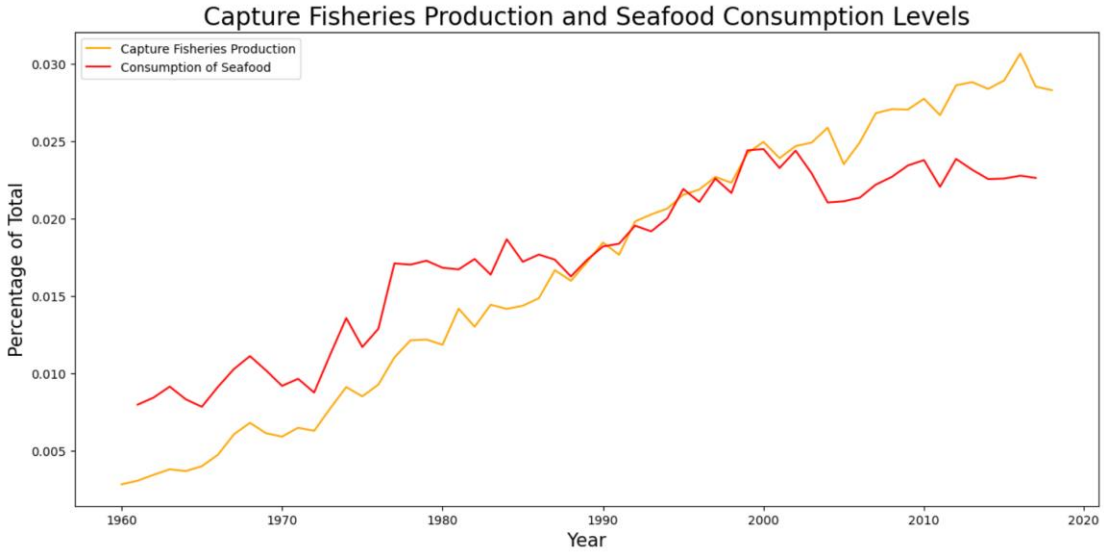


Figure 3.6

From the graph above, consumption levels appear to increase at a slower rate compared to capture fisheries production. Based on the overall shape, consumption levels and capture fisheries production may have an effect on each other as they have the same sudden spikes and drops. For example, around the year 2012, there was a sudden decrease in both of the graphs.

6.1.7 Comparison between Aquaculture Production & Seafood Consumption

```
Normalized_Aquaculture['Aquaculture production (metric tons)'] = df_Aquaculture_MYS['Aquaculture production (metric tons)'].div(np.sum(df_Aquaculture_MYS['Aquaculture production (metric tons)'].values))
```

Since df_consumption_MYS and df_Aquaculture_MYS were also in different units, df_Aquaculture_MYS had to be normalized into its percentage of total as well. Once normalized, the graph can then be plotted.

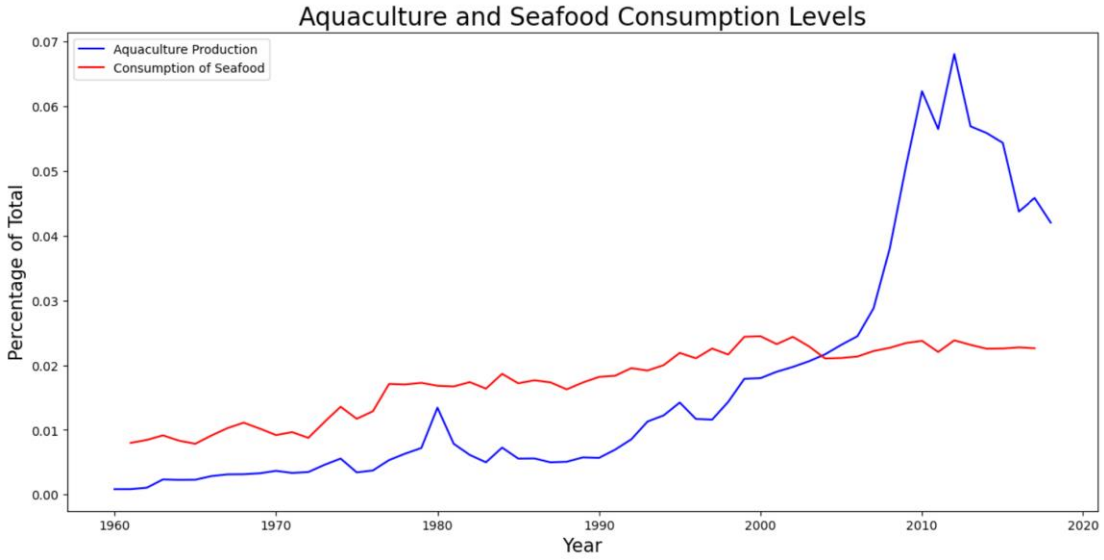


Figure 3.7

From the graph above, it can be observed that the yearly seafood consumption levels in Malaysia appear more stationary than its yearly aquaculture production.

6.1.8 Comparison between Aquaculture Production and Capture Fisheries Production

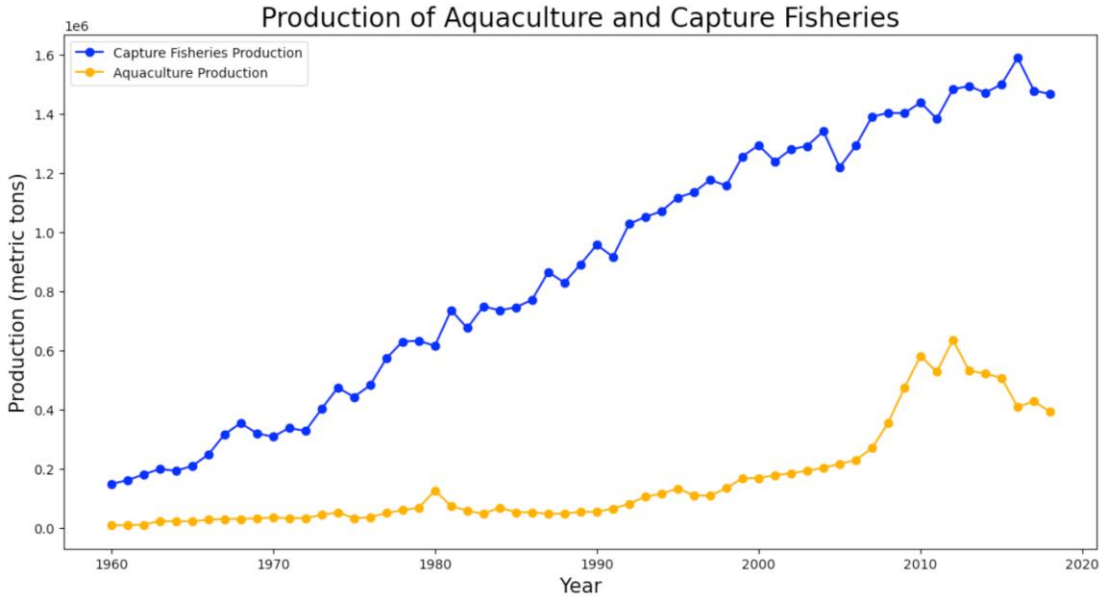


Figure 3.8

Aquaculture production is still rather low compared to capture fisheries, and it has been decreasing ever since the year 2016 while capture fisheries production has increased.

6.2 Cross Correlation

For a better understanding of the relationship between the data, cross correlation is performed. Cross-correlation analysis is able to calculate the similarity between two series at different time lags to reveal how one series correlates to another at a different time point.

To perform cross correlation on df_Consumption_MYS with df_Aquculture_MYS and df_CaptureFishery_MYS, the normalized version of the data is used.

```
print(Normalized_Consumption.info())
print(Normalized_Aquaculture.info())
print(Normalized_CaptureFishery.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 57 entries, 0 to 56
Data columns (total 2 columns):
 #   Column                                                      Non-Null Count  Dtype         
---  ------                                                      --------------  -----         
 0   Year                                                        57 non-null     datetime64[ns]
 1   Fish, Seafood- Food supply quantity (kg/capita/yr) (FAO, 2020)  57 non-null     float64       
dtypes: datetime64[ns](1), float64(1)
memory usage: 1.0 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59 entries, 0 to 58
Data columns (total 2 columns):
 #   Column                              Non-Null Count  Dtype         
---  ------                              --------------  -----         
 0   Year                                59 non-null     datetime64[ns]
 1   Aquaculture production (metric tons)  59 non-null     float64       
dtypes: datetime64[ns](1), float64(1)
memory usage: 1.1 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59 entries, 0 to 58
Data columns (total 2 columns):
 #   Column                                   Non-Null Count  Dtype         
---  ------                                   --------------  -----         
 0   Year                                     59 non-null     datetime64[ns]
 1   Capture fisheries production (metric tons)  59 non-null     float64       
dtypes: datetime64[ns](1), float64(1)
memory usage: 1.1 KB
None
```

However, Normalized_Consumption only has 57 entries compared to Normalized_Aquaculture and Normalized_CaptureFishery, which have 59.

```
print(Normalized_Consumption.head())
print(Normalized_Aquaculture.head())
print(Normalized_CaptureFishery.head())
```

```
        Year  Fish, Seafood- Food supply quantity (kg/capita/yr) (FAO, 2020)
0 1961-01-01                                           0.007993
1 1962-01-01                                           0.008453
2 1963-01-01                                           0.009160
3 1964-01-01                                           0.008339
4 1965-01-01                                           0.007856
        Year  Aquaculture production (metric tons)
0 1960-01-01                              0.000841
1 1961-01-01                              0.000849
2 1962-01-01                              0.001069
3 1963-01-01                              0.002353
4 1964-01-01                              0.002285
        Year  Capture fisheries production (metric tons)
0 1960-01-01                                    0.002850
1 1961-01-01                                    0.003087
2 1962-01-01                                    0.003469
3 1963-01-01                                    0.003822
4 1964-01-01                                    0.003708
```

```
print(Normalized_Consumption.tail())
print(Normalized_Aquaculture.tail())
print(Normalized_CaptureFishery.tail())
```

```
         Year  Fish, Seafood- Food supply quantity (kg/capita/yr) (FAO, 2020)
52 2013-01-01                                           0.023152
53 2014-01-01                                           0.022551
54 2015-01-01                                           0.022583
55 2016-01-01                                           0.022767
56 2017-01-01                                           0.022622
         Year  Aquaculture production (metric tons)
54 2014-01-01                              0.055884
55 2015-01-01                              0.054377
56 2016-01-01                              0.043750
57 2017-01-01                              0.045855
58 2018-01-01                              0.042043
         Year  Capture fisheries production (metric tons)
54 2014-01-01                                    0.028370
55 2015-01-01                                    0.028918
56 2016-01-01                                    0.030653
57 2017-01-01                                    0.028516
58 2018-01-01                                    0.028289
```

To find any differences, the first and last 5 rows are printed out. We can then see that the years ‘1960’ and ‘2018’ are missing from Normalized_Consumption.

```
Normalized_Aquaculture.drop(Normalized_Aquaculture.index[0], inplace=True)
Normalized_Aquaculture.reset_index(drop=True, inplace=True)
Normalized_CaptureFishery.drop(Normalized_CaptureFishery.index[0], inplace=True)
Normalized_CaptureFishery.reset_index(drop=True, inplace=True)
```

```
print(Normalized_Aquaculture.head())
print(Normalized_Aquaculture.tail())
print(Normalized_CaptureFishery.head())
print(Normalized_CaptureFishery.tail())
```

```
       Year  Aquaculture production (metric tons)
0 1961-01-01                              0.000849
1 1962-01-01                              0.001069
2 1963-01-01                              0.002353
3 1964-01-01                              0.002285
4 1965-01-01                              0.002307
         Year  Aquaculture production (metric tons)
53 2014-01-01                              0.055884
54 2015-01-01                              0.054377
55 2016-01-01                              0.043750
56 2017-01-01                              0.045855
57 2018-01-01                              0.042043
       Year  Capture fisheries production (metric tons)
0 1961-01-01                                    0.003087
1 1962-01-01                                    0.003469
2 1963-01-01                                    0.003822
3 1964-01-01                                    0.003708
4 1965-01-01                                    0.004021
         Year  Capture fisheries production (metric tons)
53 2014-01-01                                    0.028370
54 2015-01-01                                    0.028918
55 2016-01-01                                    0.030653
56 2017-01-01                                    0.028516
57 2018-01-01                                    0.028289
```

```
Normalized_Aquaculture.drop(Normalized_Aquaculture.index[57], inplace=True)
Normalized_Aquaculture.reset_index(drop=True, inplace=True)
Normalized_CaptureFishery.drop(Normalized_CaptureFishery.index[57], inplace=True)
Normalized_CaptureFishery.reset_index(drop=True, inplace=True)
```

```
print(Normalized_Aquaculture.head())
print(Normalized_Aquaculture.tail())
print(Normalized_CaptureFishery.head())
print(Normalized_CaptureFishery.tail())
```

```
       Year  Aquaculture production (metric tons)
0 1961-01-01                              0.000849
1 1962-01-01                              0.001069
2 1963-01-01                              0.002353
3 1964-01-01                              0.002285
4 1965-01-01                              0.002307
         Year  Aquaculture production (metric tons)
52 2013-01-01                              0.056923
53 2014-01-01                              0.055884
54 2015-01-01                              0.054377
55 2016-01-01                              0.043750
56 2017-01-01                              0.045855
       Year  Capture fisheries production (metric tons)
0 1961-01-01                                    0.003087
1 1962-01-01                                    0.003469
2 1963-01-01                                    0.003822
3 1964-01-01                                    0.003708
4 1965-01-01                                    0.004021
         Year  Capture fisheries production (metric tons)
52 2013-01-01                                    0.028806
53 2014-01-01                                    0.028370
54 2015-01-01                                    0.028918
55 2016-01-01                                    0.030653
56 2017-01-01                                    0.028516
```

The rows containing year ‘1960’ and ‘2018’ were then dropped from Normalized_Aquaculture and Normalized_CaptureFishery so that the time range of the three data are the same.

6.2.1 Aquaculture Production and Capture Fisheries Production

To perform cross correlation, the data should be stationary, which means the mean and variance of the two series are approximately constant and aren't affected by time. If the data isn't stationary, the time series must be detrended.

To check whether or not the time series is stationary, the Augmented Dickey Fuller (ADF) Test was conducted. The null hypothesis for the ADF test is that the series in question is not stationary. If the p-value is less than 0.05, we say that the null hypothesis is rejected.

```
#Time series must be stationary to perform cross correlation

result = adfuller(df_Aquaculture_MYS['Aquaculture production (metric tons)'])
print("statistics:",result[0])
print("p-value:",result[1])
```

```
statistics: -0.5279542104763244
p-value: 0.8864450186935084
```

```
result = adfuller(df_CaptureFishery_MYS['Capture fisheries production (metric tons)'])
print("statistics:",result[0])
print("p-value:",result[1])
```

```
statistics: -1.1537947128011408
p-value: 0.6931498053186675
```

The p-value is more than 0.05 for df_Aquaculture_MYS and df_CaptureFishery_MYS, therefore the null hypothesis is not rejected and the data is not stationary.

To stationarise the data, we can perform differencing.

```
aq_detrend=df_Aquaculture_MYS['Aquaculture production (metric tons)'].diff().dropna()
result = adfuller(aq_detrend)
print("statistics:",result[0])
print("p-value:",result[1])
```

```
statistics: -3.5581219492903156
p-value: 0.0066140054386771745
```

```
cap_detrend=df_CaptureFishery_MYS['Capture fisheries production (metric tons)'].diff().dropna()
result = adfuller(cap_detrend)
print("statistics:",result[0])
print("p-value:",result[1])
```

```
statistics: -8.173011487348974
p-value: 8.522275489474179e-13
```

The p-value is now less than 0.05, so the null hypothesis is rejected and the series is now stationary.

```
def ccf_values(series1, series2):
    x = series1
    y = series2
    x = (x - np.mean(x)) / (np.std(x) * len(x))
    y = (y - np.mean(y)) / (np.std(y))
    ccf = np.correlate(x, y, 'full')
    return ccf


ccf_production = ccf_values(aq_detrend, cap_detrend)
ccf_production
```

```
array([ 2.01182462e-03,  9.18787742e-03, -1.36937507e-04, -7.18801707e-03,
        1.62116041e-02,  6.53457372e-03, -7.40965283e-03,  9.44800989e-03,
        1.61175126e-02, -3.44132559e-03,  7.82867988e-03,  1.14326864e-02,
       -5.39178032e-03,  3.27103995e-03,  1.43975728e-02,  2.43442515e-02,
       -1.17080085e-02, -5.00629219e-03,  1.56363361e-02, -2.10043382e-02,
       -5.54749582e-02,  1.26424509e-01, -9.20212406e-03, -8.31624819e-03,
        9.74576942e-03,  6.40172834e-02, -9.48231830e-02,  7.87214639e-02,
        2.33767239e-03,  1.82511014e-03,  2.43788560e-02,  1.03730636e-02,
       -1.41522248e-01,  3.49733561e-02,  2.67583803e-02,  6.83244787e-03,
       -3.90151093e-02,  4.95077438e-02, -8.06740854e-03, -4.79451057e-02,
        1.25785962e-02, -4.06638330e-02,  4.21333233e-02, -1.96772843e-02,
        2.53451183e-02, -3.97936422e-02, -8.75529091e-02, -7.59613194e-02,
       -7.69963664e-02, -1.35816654e-01, -1.93374492e-02, -9.93749165e-03,
       -1.48195585e-01,  1.65056899e-01,  2.48319549e-02, -1.01404283e-01,
        1.32137369e-01,  3.13527288e-02, -5.52086080e-02,  1.64001374e-01,
        6.95068411e-02, -2.30311743e-01, -1.10633873e-03,  9.76051095e-02,
       -1.94993783e-01,  6.91604349e-02, -1.36513808e-01,  1.12202041e-01,
        4.56368025e-02,  5.20904632e-02,  1.79210000e-01, -1.19948670e-01,
        1.38896069e-01, -2.44865522e-02,  7.51294490e-02,  1.02340085e-01,
       -1.32918863e-01,  1.88408565e-01, -1.30775037e-01,  1.01494866e-01,
        1.71903038e-02, -2.29044875e-01,  1.70773970e-01, -1.37341474e-01,
        3.65766780e-02, -2.51000682e-02,  5.36817659e-02, -1.74484694e-01,
        2.57327190e-01,  2.92082273e-02,  1.47921247e-02,  1.01737636e-01,
       -7.24078059e-02,  1.27036035e-01, -1.24160819e-01, -2.86748983e-02,
       -9.10001545e-02, -1.67012218e-01,  6.58537974e-02, -2.79426240e-03,
       -2.59047780e-03,  5.43531726e-02, -3.72935799e-02, -9.31764796e-03,
        1.81836457e-02, -4.95967541e-02, -4.87791773e-03, -2.61899821e-03,
       -9.73469985e-03,  3.18866204e-02,  5.36094923e-03,  1.49455763e-02,
        1.11380911e-02, -9.52113487e-05,  3.91801283e-03])
```

After detrending the series, cross correlation can now be performed. First, the correlation coefficients are calculated. Since cross correlation coefficients were calculated multiple times for different series, a function is created to be used.

```
#Plotting the graph
lags = signal.correlation_lags(len(aq_detrend), len(cap_detrend))

fig, ax =plt.subplots(figsize=(12, 6))
ax.plot(lags, ccf_production)
ax.axhline(-2/np.sqrt(23), color='red', label='5% confidence interval')
ax.axhline(2/np.sqrt(23), color='red')
ax.axvline(x = 0, color = 'black', lw = 1)
ax.axhline(y = 0, color = 'black', lw = 1)
ax.axhline(y = np.max(ccf_production), color = 'blue', lw = 1, linestyle='--', label = 'highest +/- correlation')
ax.axhline(y = np.min(ccf_production), color = 'blue', lw = 1, linestyle='--')
ax.set(ylim = [-1, 1])
ax.set_title('Cross Correlation between Aquaculture Production and Capture Fisheries Production', weight='bold', fontsize = 15)
ax.set_ylabel('Correlation Coefficients', weight='bold', fontsize = 12)
ax.set_xlabel('Time Lags', weight='bold', fontsize = 12)
plt.legend()
```

A list of lag values was then created to visualise it against the correlation coefficients. Confidence intervals are also set to indicate the value at which the correlation coefficients become important.

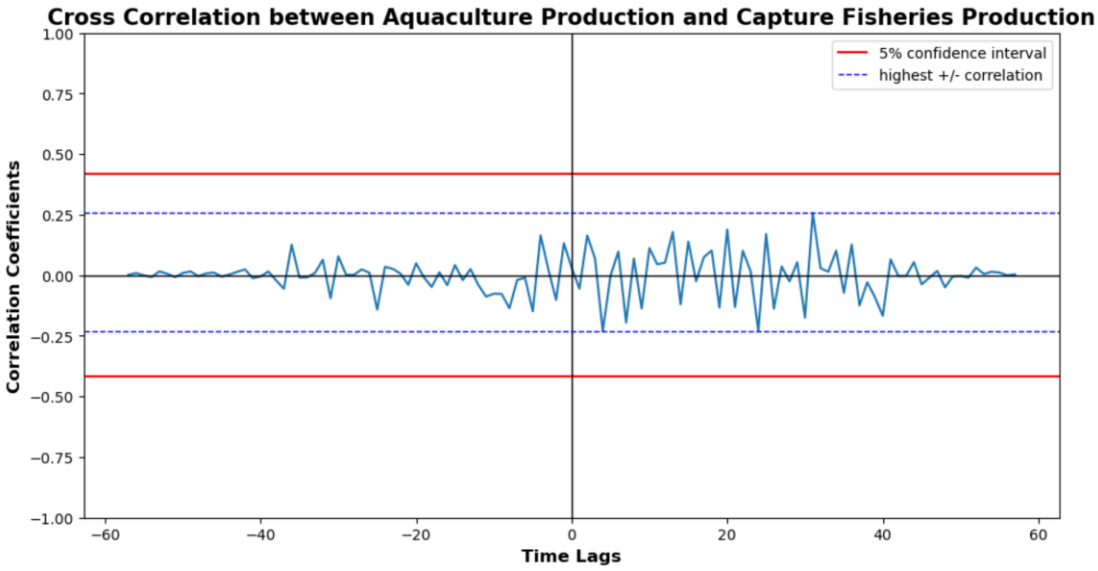


Figure 4.1

The time lags on the right indicate that series 1 lags while series 2 leads, which means the changes in series 1 happens after the changes in series 2. While the time lags on the left indicate that series 2 lags while series 1 leads, which means the changes in series 1 causes the changes in series 2 to happen.

From Figure 4.1, we can notice that there are higher correlation coefficients on the right side, which show that it is more likely that the changes in capture fisheries production affect aquaculture production instead of the other way around.

```
print(lags)
```

```
[-57 -56 -55 -54 -53 -52 -51 -50 -49 -48 -47 -46 -45 -44 -43 -42 -41 -40
 -39 -38 -37 -36 -35 -34 -33 -32 -31 -30 -29 -28 -27 -26 -25 -24 -23 -22
 -21 -20 -19 -18 -17 -16 -15 -14 -13 -12 -11 -10  -9  -8  -7  -6  -5  -4
  -3  -2  -1   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32
  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50
  51  52  53  54  55  56  57]
```

```
for i in ccf_production:
    if i == max(ccf_production) :
        print(i)
```

```
0.25732719009542754
```

```
#To find the exact time lag
j=-57
for i in ccf_production:
    if i == max(ccf_production):
        print(j)
    j=j+1
```

```
31
```

```
for i in ccf_production:
    if i == min(ccf_production) :
        print(i)
```

```
-0.23031174322004233
```

```
j=-57
for i in ccf_production:
    if i == min(ccf_production):
        print(j)
    j=j+1
```

```
4
```

To get the precise lag value, the code above was used so the time lag for which the positive and negative correlation coefficient was the highest can be found.

The highest positive correlation coefficient is around +0.26, which happens around the +31 lag. This indicates there is a weak positive correlation between aquaculture production today and capture fisheries production 31 years ago.

The highest negative correlation coefficient is around -0.23, which happens around the +4 lag. It indicates a weak negative correlation between aquaculture production today and capture fisheries 4 years ago.

6.2.2 Aquaculture Production and Consumption levels

```
result = adfuller(Normalized_Aquaculture['Aquaculture production (metric tons)'])
print("statistics:",result[0])
print("p-value:",result[1])
```

```
statistics: -0.6295353085723003
p-value: 0.864167636618753
```

```
aqnorm_detrend=Normalized_Aquaculture['Aquaculture production (metric tons)'].diff().dropna()
result = adfuller(aqnorm_detrend)
print("statistics:",result[0])
print("p-value:",result[1])
```

```
statistics: -3.388106543595348
p-value: 0.011371062681056067
```

```
result = adfuller(Normalized_Consumption['Fish, Seafood- Food supply quantity (kg/capita/yr) (FAO, 2020)'])
print("statistics:",result[0])
print("p-value:",result[1])
```

```
statistics: -1.4559040326270212
p-value: 0.555201401158352
```

```
consumnorm_detrend=Normalized_Consumption['Fish, Seafood- Food supply quantity (kg/capita/yr) (FAO, 2020)'].diff().dropna()
result = adfuller(consumnorm_detrend)
print("statistics:",result[0])
print("p-value:",result[1])
```

```
statistics: -8.937990099516348
p-value: 9.419769947501313e-15
```

```
ccf_AqConsump = ccf_values(aqnorm_detrend, consumnorm_detrend)
ccf_AqConsump
```

```
array([ 0.0008421 , -0.00053513,  0.00160489,  0.00275973,  0.00179758,
       -0.00068173,  0.01358825, -0.0033243 ,  0.0049508 ,  0.01072108,
        0.00259167,  0.00262944,  0.00436954,  0.01333104, -0.00230267,
        0.00359668,  0.0274835 ,  0.00801293, -0.02594221,  0.04031227,
        0.00425   , -0.00713054,  0.00382418,  0.07072744, -0.06243563,
        0.031054  ,  0.04448808,  0.03411064, -0.03584205,  0.00846312,
       -0.01019341, -0.07314792,  0.02762463,  0.04498843, -0.00069549,
       -0.00034899,  0.05949047, -0.03821564,  0.04890162, -0.04982535,
        0.07180399, -0.02412679, -0.01002502, -0.05824609, -0.0174223 ,
       -0.00555456, -0.07817678, -0.05191041, -0.01734468, -0.04424786,
       -0.13178386,  0.05683029, -0.09372073, -0.02942275, -0.10475838,
        0.29244628, -0.11296629,  0.12246219,  0.13502875, -0.22315608,
       -0.13052644, -0.12823872, -0.02303492, -0.16486749, -0.03176251,
        0.20778643,  0.04271575,  0.07864715,  0.25288515, -0.19400784,
        0.24326507, -0.11134882,  0.06114672,  0.10161057, -0.14677567,
        0.07586603, -0.11339896, -0.06718237, -0.04780097, -0.14241346,
        0.05558097,  0.10940373, -0.17682671,  0.22061518, -0.19455708,
        0.05950066,  0.18236247,  0.09941565,  0.25569976, -0.10989565,
        0.21556608,  0.09086272, -0.16943893,  0.01012793, -0.20475439,
       -0.2278266 ,  0.16003895, -0.05406366, -0.03349974,  0.09625967,
       -0.06796244,  0.05976078,  0.01087433, -0.05556265,  0.02134399,
       -0.03540545,  0.03246644,  0.03553954, -0.02472201, -0.00598654,
        0.0009163 ])
```

The process is repeated for Normalized_Aquaculture and Normalized_Consumption, since the function to calculate cross correlation coefficients has been defined earlier, it can be used.

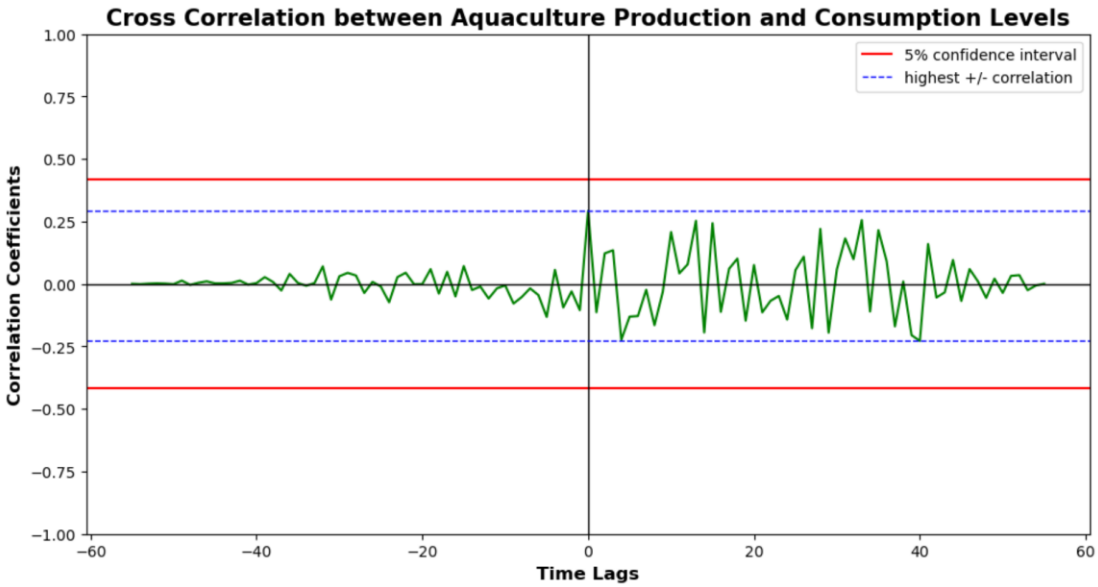


Figure 4.2

From Figure 4.2, there are higher correlation coefficients on the right side, which show that it is more likely that the changes in seafood consumption levels affect aquaculture production.

```
print(lags)
```

```
[-55 -54 -53 -52 -51 -50 -49 -48 -47 -46 -45 -44 -43 -42 -41 -40 -39 -38
 -37 -36 -35 -34 -33 -32 -31 -30 -29 -28 -27 -26 -25 -24 -23 -22 -21 -20
 -19 -18 -17 -16 -15 -14 -13 -12 -11 -10  -9  -8  -7  -6  -5  -4  -3  -2
  -1   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52
  53  54  55]
```

```
for i in ccf_AqConsump:
    if i == max(ccf_AqConsump) :
        print(i)
```

```
0.2924462821736922
```

```
j=-55
for i in ccf_AqConsump:
    if i == max(ccf_AqConsump):
        print(j)
    j=j+1
```

```
0
```

```
for i in ccf_AqConsump:
    if i == min(ccf_AqConsump) :
        print(i)
```

```
-0.2278265984533069
```

```
j=-55
for i in ccf_AqConsump:
    if i == min(ccf_AqConsump):
        print(j)
    j=j+1
```

```
40
```

From the code, the highest positive correlation coefficient is +0.29, which happens around the 0 lag, this suggests there is weak positive correlation between aquaculture production and seafood consumption levels at the same timeframe.

The highest negative correlation is -0.22 coefficient, which is around the +40 time lag. This suggests that there is a weak negative correlation between aquaculture production today and seafood consumption levels 40 years ago.

6.2.3 Capture Fisheries Production and Consumption Levels

```
result = adfuller(Normalized_CaptureFishery['Capture fisheries production (metric tons)'])
print("statistics:",result[0])
print("p-value:",result[1])
```

```
statistics: -1.1580378682230068
p-value: 0.6913814438288586
```

```
CFnorm_detrend=Normalized_CaptureFishery['Capture fisheries production (metric tons)'].diff().dropna()
result = adfuller(CFnorm_detrend)
print("statistics:",result[0])
print("p-value:",result[1])
```

```
statistics: -8.452628456469153
p-value: 1.646662666801625e-13
```

Data in Normalized_CaptureFishery is detrended and the detrended value for Normalized_Consumption is taken from earlier.

```
ccf_CFConsump = ccf_values(CFnorm_detrend, consumnorm_detrend)
ccf_CFConsump
```

```
array([ 4.24838974e-04,  6.68810828e-04,  3.66892537e-03,  2.68745388e-03,
        2.62146784e-03,  1.92719768e-03,  5.83613248e-03, -7.93521031e-03,
        2.03971617e-03, -3.03609128e-03,  2.18757010e-02, -3.08077375e-03,
       -3.19083480e-02,  3.43404237e-02,  2.73342233e-02, -2.84159801e-02,
        4.65675002e-02, -2.37710520e-02, -1.01614402e-01, -1.27368208e-02,
        4.35231149e-02,  2.42656690e-02, -2.92054537e-02,  4.77912248e-02,
        7.95188790e-02, -1.93230350e-01,  1.17217147e-01, -3.11541069e-02,
       -1.09906585e-01,  2.28605042e-02, -7.34561637e-04, -7.16261113e-02,
        1.31495847e-02,  1.88398015e-02,  2.56469970e-02, -5.95014874e-02,
        3.55349048e-03,  8.74156560e-02, -7.94660105e-02,  1.51853508e-01,
       -1.06044343e-01, -8.23234584e-02,  8.08216813e-02, -2.03373614e-02,
       -1.26854324e-01,  1.70527439e-01, -7.19938943e-02, -8.09296682e-03,
       -1.11237975e-01,  1.14112674e-01,  4.49922678e-02, -2.66730321e-01,
        2.86748064e-01, -1.49397947e-01, -9.39579389e-02,  4.71403743e-01,
        2.26660261e-02, -1.58041465e-01, -1.18468471e-02,  1.17701152e-01,
        9.18716444e-03, -1.32731209e-01,  8.38502290e-02, -6.34538414e-04,
       -1.55137579e-01,  2.11675431e-01, -1.18032814e-01, -1.44833911e-01,
        3.12081557e-01, -1.39516567e-01,  8.51036890e-02,  1.00876195e-01,
       -4.91209283e-02,  2.59925013e-02, -1.36722711e-02,  9.06566479e-03,
       -3.05236949e-02,  1.82893751e-02,  6.22307063e-02, -9.89205593e-02,
       -6.47602215e-02,  1.98399750e-01, -1.31044417e-01, -1.84028322e-01,
        4.96641875e-02,  1.63402959e-01, -7.37986685e-02, -3.38753300e-02,
        6.56057367e-02, -1.52999426e-02, -1.41089567e-02,  1.11278965e-01,
       -1.39810318e-01, -6.66287748e-02,  1.69704659e-01, -1.40865976e-01,
       -1.62129107e-02,  8.02190694e-02, -1.00759544e-01, -6.56155005e-02,
        6.21506100e-02, -1.24486208e-02,  1.89325748e-02,  1.36313037e-02,
        1.48966535e-02,  9.26265222e-04, -5.94927741e-02,  5.85817813e-03,
        4.87496750e-02, -1.29400195e-02, -7.37755853e-03])
```

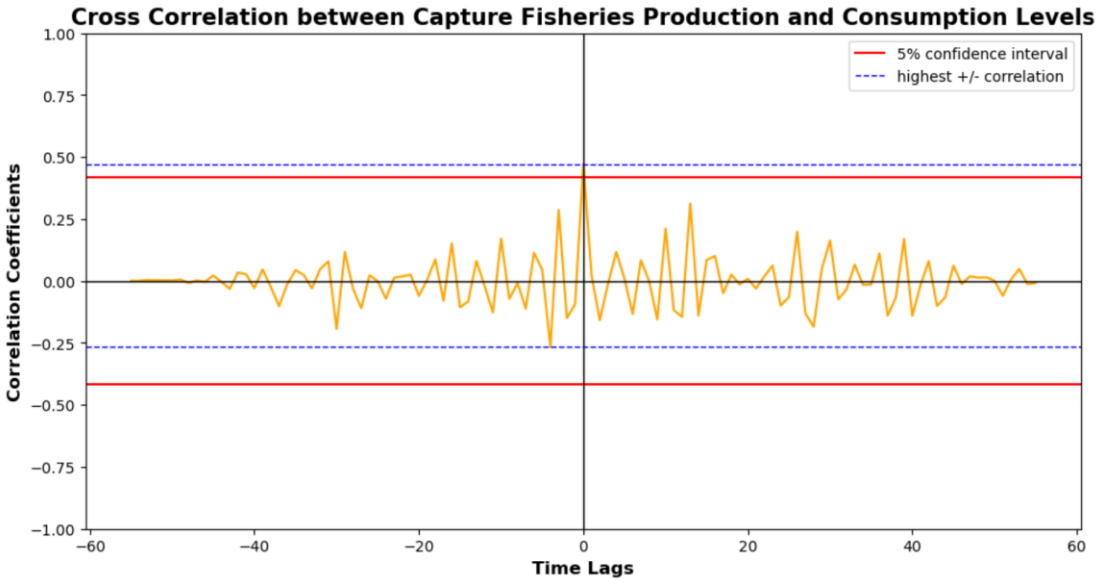


Figure 4.3

From Figure 4.3, the correlation coefficients with higher values appear to be spread out between the right side and the left side. This could mean that the production of capture fisheries and consumption levels affect each other.

```
print(lags)
```

```
[-55 -54 -53 -52 -51 -50 -49 -48 -47 -46 -45 -44 -43 -42 -41 -40 -39 -38
 -37 -36 -35 -34 -33 -32 -31 -30 -29 -28 -27 -26 -25 -24 -23 -22 -21 -20
 -19 -18 -17 -16 -15 -14 -13 -12 -11 -10  -9  -8  -7  -6  -5  -4  -3  -2
  -1   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52
  53  54  55]
```

```
for i in ccf_CFConsump:
    if i == max(ccf_CFConsump) :
        print(i)
```

```
0.47140374302157273
```

```
j=-55
for i in ccf_CFConsump:
    if i == max(ccf_CFConsump):
        print(j)
    j=j+1
```

```
0
```

```
for i in ccf_CFConsump:
    if i == min(ccf_CFConsump) :
        print(i)
```

```
-0.26673032109044664
```

```
j=-55
for i in ccf_CFConsump:
    if i == min(ccf_CFConsump):
        print(j)
    j=j+1
```

```
-4
```

The highest positive correlation coefficient is +0.47 when the time lag is 0, this indicates there is a moderate positive correlation between capture fisheries production and consumption levels at the same timeframe.

The highest negative correlation coefficient is approximately -0.27, which is at the -4 lag. This shows a weak negative correlation between capture fisheries production today and consumption levels four years later.

6.3 Granger Causality Test between Aquaculture Production and Capture Fisheries Production

To further analyse the relationship between the production levels of aquaculture and capture fisheries, the granger causality test was performed.

Granger causality test helps assess whether the values from one time series can be used to forecast another. It also gauges the degree to which past data on one variable may be used to predict how the other variable will behave in the future.

The Granger Causality test can only be performed on the condition that the data needs to be stationary i.e it should have a constant mean, constant variance, and no seasonal component. One limitation of the Granger causality test is that it doesn't prove direct causation since it doesn't provide any insight on the relationship between the variables unlike 'cause and effect' analysis (Pallavi, 2024).

```
from statsmodels.tsa.stattools import grangercausalitytests
data = pd.DataFrame({'Aquaculture production (metric tons)':aq_detrend,'Capture fisheries production (metric tons)':cap_detrend})
```

```
data.head()
```

	Aquaculture production (metric tons)	Capture fisheries production (metric tons)
1	70.0	12300.0
2	2057.0	19777.0
3	11973.0	18323.0
4	-639.0	-5900.0
5	206.0	16200.0

Since the data needed to be stationary, the detrended values of df_Aquaculture_MYS and df_CaptureFishery_MYS that had been calculated earlier: aq_detrend and cap_detrend, were used.

```
#Does the production of capture fisheries affect aquaculture production?
grangercausalitytests(data, maxlag=4)
```

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=0.1679  , p=0.6836  , df_denom=54, df_num=1
ssr based chi2 test:   chi2=0.1773  , p=0.6737  , df=1
likelihood ratio test: chi2=0.1770  , p=0.6740  , df=1
parameter F test:         F=0.1679  , p=0.6836  , df_denom=54, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=0.8633  , p=0.4279  , df_denom=51, df_num=2
ssr based chi2 test:   chi2=1.8958  , p=0.3876  , df=2
likelihood ratio test: chi2=1.8644  , p=0.3937  , df=2
parameter F test:         F=0.8633  , p=0.4279  , df_denom=51, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=1.2546  , p=0.3005  , df_denom=48, df_num=3
ssr based chi2 test:   chi2=4.3126  , p=0.2296  , df=3
likelihood ratio test: chi2=4.1519  , p=0.2455  , df=3
parameter F test:         F=1.2546  , p=0.3005  , df_denom=48, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=1.7282  , p=0.1604  , df_denom=45, df_num=4
ssr based chi2 test:   chi2=8.2952  , p=0.0813  , df=4
likelihood ratio test: chi2=7.7166  , p=0.1025  , df=4
parameter F test:         F=1.7282  , p=0.1604  , df_denom=45, df_num=4
```

“Y”, or the predictor variable, is “capture fisheries production” and “X”, the response variable is “aquaculture production”. So the null hypothesis for the Granger causality test is “Capture fisheries production does not ‘granger cause’ aquaculture production”.

Based on the results, the overall p-value is larger than 0.05, so the null hypothesis is not rejected.

```
#Does the production of aquaculture affect capture fisheries production?
grangercausalitytests(data[["Capture fisheries production (metric tons)", "Aquaculture production (metric tons)"]], maxlag=4)
```

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=1.2363  , p=0.2711  , df_denom=54, df_num=1
ssr based chi2 test:   chi2=1.3050  , p=0.2533  , df=1
likelihood ratio test: chi2=1.2903  , p=0.2560  , df=1
parameter F test:         F=1.2363  , p=0.2711  , df_denom=54, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=0.3680  , p=0.6939  , df_denom=51, df_num=2
ssr based chi2 test:   chi2=0.8081  , p=0.6676  , df=2
likelihood ratio test: chi2=0.8023  , p=0.6695  , df=2
parameter F test:         F=0.3680  , p=0.6939  , df_denom=51, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=0.4569  , p=0.7136  , df_denom=48, df_num=3
ssr based chi2 test:   chi2=1.5707  , p=0.6661  , df=3
likelihood ratio test: chi2=1.5487  , p=0.6711  , df=3
parameter F test:         F=0.4569  , p=0.7136  , df_denom=48, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=0.5431  , p=0.7049  , df_denom=45, df_num=4
ssr based chi2 test:   chi2=2.6070  , p=0.6256  , df=4
likelihood ratio test: chi2=2.5460  , p=0.6364  , df=4
parameter F test:         F=0.5431  , p=0.7049  , df_denom=45, df_num=4
```

The Granger causality test is then done the other way around, with the predictor variable being "aquaculture production" with "capture fisheries production" acting as the predictor variable. Which makes the null hypothesis "Aquaculture production does not 'granger cause' capture fisheries production".

Based on the results, the overall p-value is also larger than 0.05, so the null hypothesis is not rejected.

7.0 Time Series Forecasting

To determine whether or not aquaculture in Malaysia is improving, the data used to build the prediction model would be from df_Aquaculture_MYS. As it is a time series, one of the suitable models that could be used is the Autoregressive Integrated Moving Average (ARIMA) model, which uses past values to predict future values.

To build the model, the ARIMA(p, d, q) model's parameters must first be found.

- *p*: the number of lag observations in the model, also known as the lag order.
- *d*: the number of times the raw observations are differenced; also known as the degree of differencing.
- *q*: the size of the moving average window, also known as the order of the moving average.

```
n_splits = 10
tscv = TimeSeriesSplit(n_splits=n_splits)
```

```
dataAQ = df_Aquaculture_MYS['Aquaculture production (metric tons)']
```

```
#Showcasing each split
i=1
for train_index, test_index in tscv.split(dataAQ):
    print('Split', i)
    print('TRAIN:', train_index)
    print('TEST:', test_index)
    i=i+1
```

To assess the performance of our model throughout the entire time series data, time series cross validation is performed. It is able to assess the performance of a model by training and testing it on different subsets of the data to ensure that the model generalizes well to unseen data. For the forecasting model, ten splits were used.

```
Split 1
TRAIN: [0 1 2 3 4 5 6 7 8]
TEST: [ 9 10 11 12 13]
Split 2
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13]
TEST: [14 15 16 17 18]
Split 3
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18]
TEST: [19 20 21 22 23]
Split 4
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]
TEST: [24 25 26 27 28]
Split 5
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28]
TEST: [29 30 31 32 33]
Split 6
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33]
TEST: [34 35 36 37 38]
Split 7
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38]
TEST: [39 40 41 42 43]
Split 8
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43]
TEST: [44 45 46 47 48]
Split 9
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48]
TEST: [49 50 51 52 53]
Split 10
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53]
TEST: [54 55 56 57 58]
```

Figure 5.1

The image above shows the individual splits, indexes of the previous data being used to train the model and indexes of the test data used for comparisons.

```
[678]: result = adfuller(train_data)
       print("statistics",result[0])
       print("p-value",result[1])

       statistics 1.7146127685926573
       p-value 0.9981637384269566

[680]: #The original data is not stationary
       #To sationarise the data, apply differencing

       # check the stationality of 1 order difference data
       #1st order differencing
       train_data_stationary=train_data.diff().dropna()
       result = adfuller(train_data_stationary)
       print("statistics",result[0])
       print("p-value",result[1])
       train_data_stationary.plot()

       #p-value is less than 0.05, it is stationary

       statistics -3.1115393850828763
       p-value 0.02571092475130434
[680]: <Axes: >
```

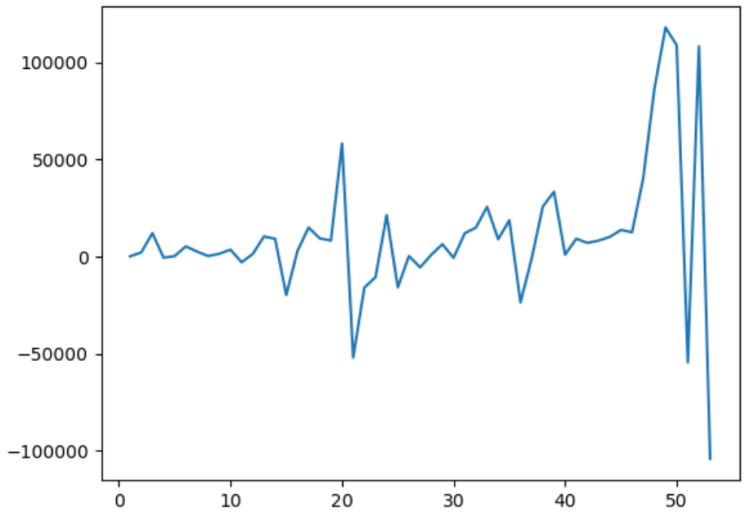


Figure 5.2

To find the parameters of the ARIMA model, the test data from the most recent split was used. For the data to be stationary, 1st order differencing was applied. Thus, the value of d would be 1.

The value of p & q can be found by plotting the autocorrelation function (ACF) and the partial autocorrelation function (PACF)

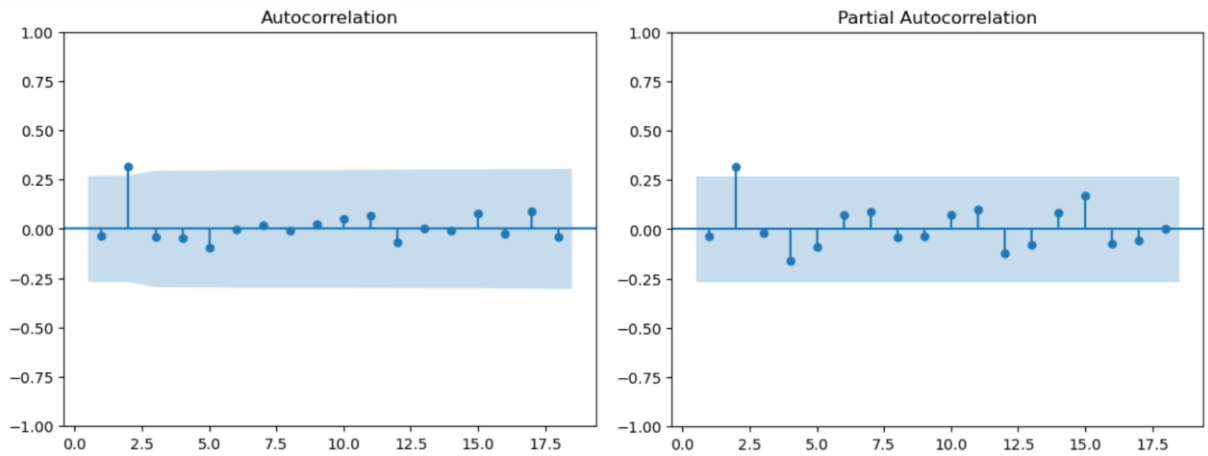


Figure 5.3

From the autocorrelation and partial autocorrelation function, p and q both have the value of 2. Thus, the model could have the parameters (2,1,0) or (0,1,2).

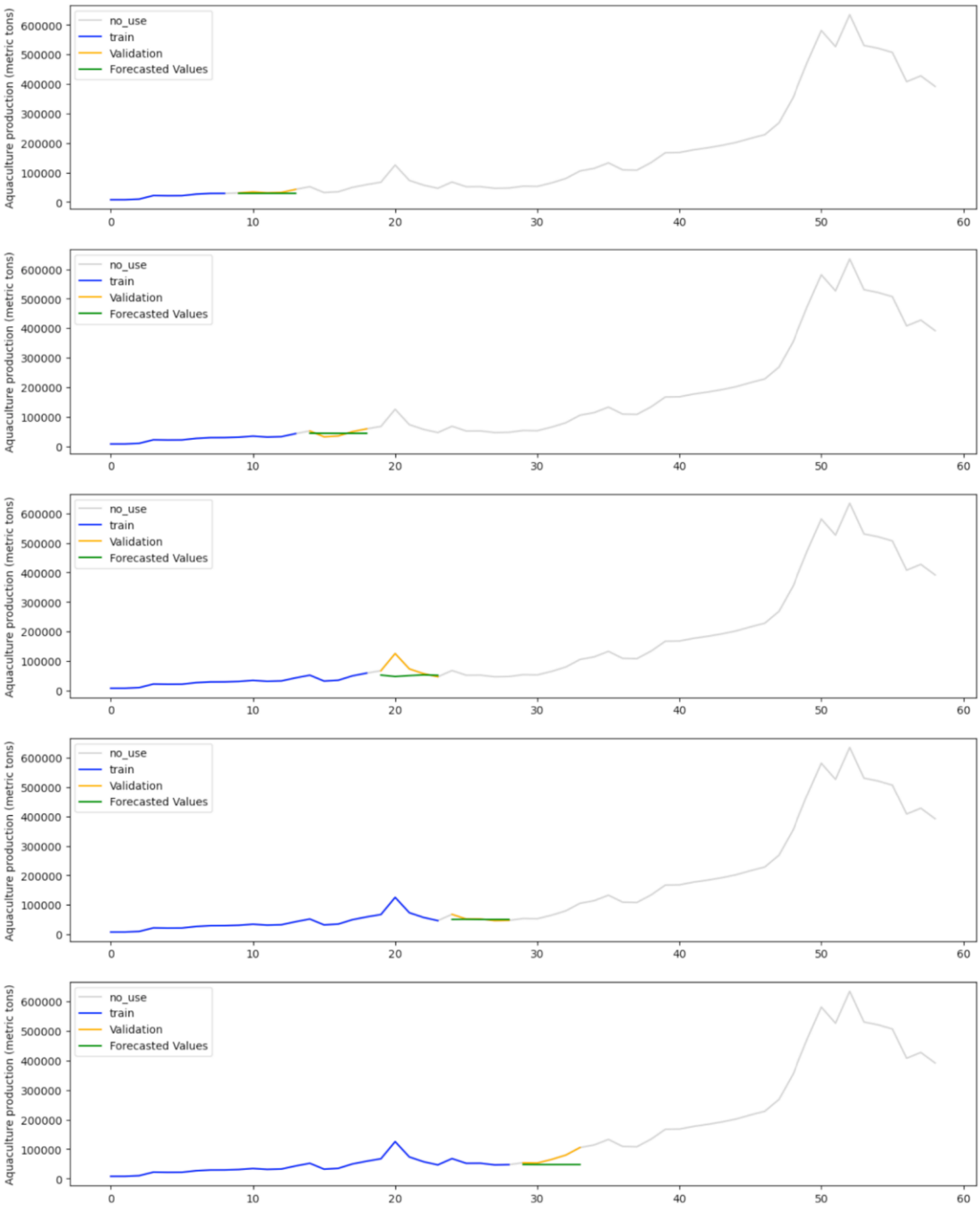
```
mse_scores1 = []
rmse_scores1=[]
i=1

for train_index, test_index in tscv.split(dataAQ):
    train_data = dataAQ.iloc[train_index]
    test_data = dataAQ.iloc[test_index]

    model1 = ARIMA(train_data, order=(2, 1, 0))
    fitted_model1 = model1.fit()

    predictions1 = fitted_model1.forecast(steps=len(test_data))

    #Calculate Mean Squared Error for each split
    mse = mean_squared_error(test_data, predictions1)
    mse_scores1.append(mse)

    #Root Mean Squared Error for each split
    rmse= np.sqrt(mean_squared_error(test_data, predictions1))
    rmse_scores1.append(rmse)

    print('Split', i)
    print(f'Mean Squared Error for current split: {mse}')
    print(f'Root Mean Squared Error for current split: {rmse} ')
    i=i+1

#Calculate average
average_mse1 = np.mean(mse_scores1)
average_rmse1=np.mean(rmse_scores1)
print(" ")
print(f'Average Mean Squared Error across all splits: {average_mse1}')
print(f'Average Root Mean Squared Error across all splits: {average_rmse1}')
```

By using the ARIMA model with parameters (2,1,0), the predictions for each individual split are calculated. The performance of the model can be evaluated by calculating the mean squared error (MSE) and root mean squared error for each split and the average across all splits.

Split	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)
1	42552402.87955734	6523.220284457466
2	109449709.11275014	10461.821500711534
3	1367850825.4351304	36984.46735367606
4	64618789.36226402	8038.581302833481
5	940318564.8741324	30664.614213685003
6	542563731.5258989	23292.997478338828
7	2446948091.2728624	49466.63614268573
8	6900966682.861753	83072.05717244369
9	23633677110.71527	153732.4855413301
10	15139209351.532227	123041.49442985577
Average	5118815525.957185	52527.83754200176

Figure 5.4

From the table above, the MSE and RMSE values are all significantly large, indicating that the model is not very accurate.


no_use
train
Validation
Forecasted Values
Aquaculture production (metric tons)
0
100000
200000
300000
400000
500000
600000
0
10
20
30
40
50
60

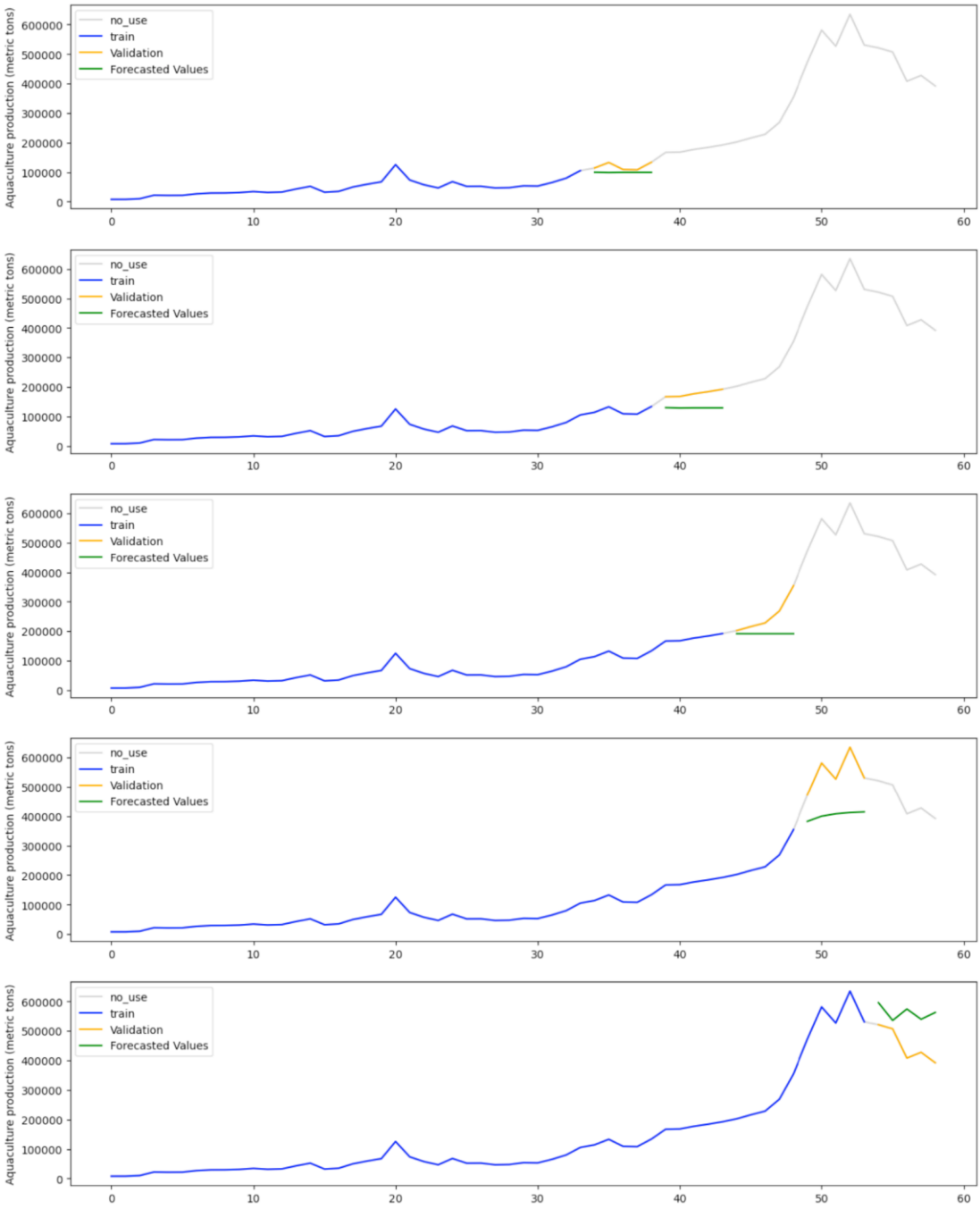


Figure 5.5

To get a better visualisation of the model's accuracy, each individual prediction for every split is plotted alongside its train data and test data. From Figure 5.5, it can be seen that the inaccuracies are due to the sudden spikes or changes in the data. This shows that the model isn't capable of handling sudden fluctuations in the data.

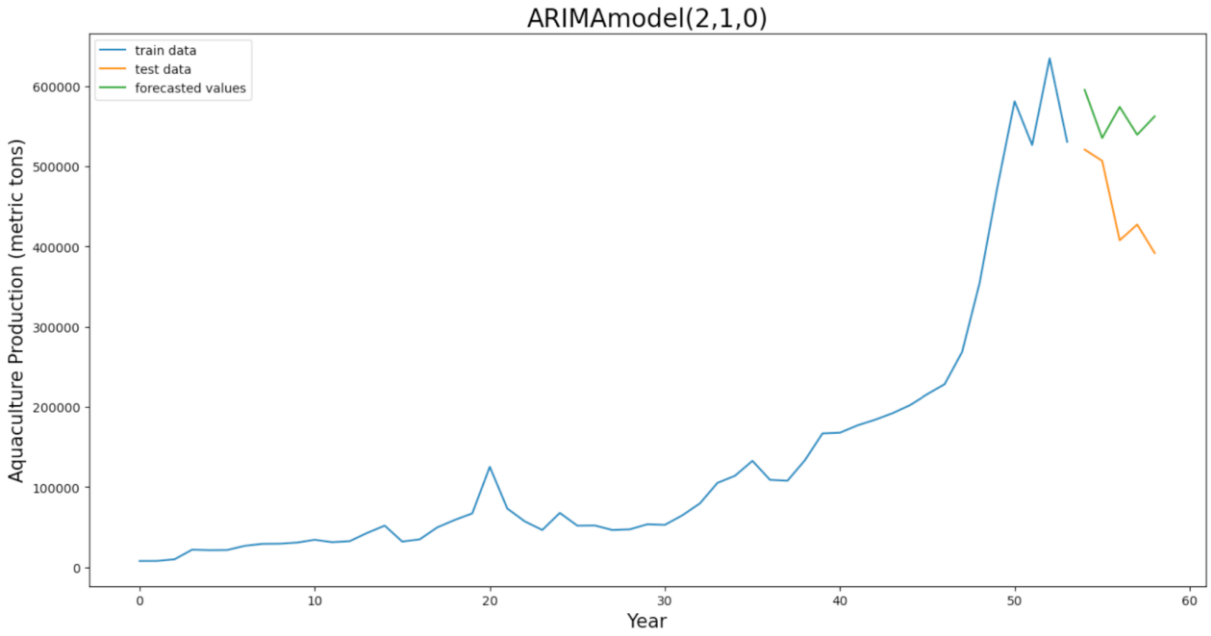


Figure 5.6

For example, in Figure 5.6, the model uses the data it was trained on to forecast that aquaculture production would fluctuate between increasing and decreasing. However, the test data shows a sudden decrease in the production of aquaculture.

The model can be deployed by companies specialising in aquaculture to be used to forecast future aquaculture production and use the forecasted results as a target.

8.0 Discussion

8.1 Production vs Consumption

Based on the visualisation and cross correlation done between the production of aquaculture and production of capture fisheries with seafood consumption levels, there is a higher correlation between capture fisheries production and consumption levels compared to aquaculture production and consumption levels.

This indicates that when the amount of seafood getting consumed increases, it encourages the production of capture fisheries so that it is able to supply the demand.

Moreover, it also suggests that the production from aquaculture is not used for consumption and is instead contributing to other factors, such as exports, rebuilding the populations of endangered species or breeding marine animals for zoos and aquariums.

8.2 Aquaculture vs Capture Fisheries

From the data analysis done, it appears that aquaculture production does not affect capture fisheries production. Meaning that fish are still being caught in the wild despite being able to cultivate them. It could also indicate that aquaculture in Malaysia isn't effective enough to be able to completely replace catching seafood in the wild.

8.3 Aquaculture & The overexploited fish stocks

The percentage of overexploited fish stocks have decreased from 2015 to 2017, which could indicate that the existence of aquaculture has been able to decrease the overexploitation of fish. However, due to insufficient data, it's difficult to make assumptions that it was because of the existence of aquaculture. Furthermore, from the range of countries who are part of the Southeast Pacific, it is not guaranteed that the decrease was due to Malaysia's aquaculture production alone.

9.0 Conclusion

Malaysia's fishing industry plays an important role in the country by acting as one of the major suppliers for animal protein as well as contributing to Malaysia's exports. However, due to the fish being caught faster than they can reproduce, it has caused their populations to dwindle.

To remedy this, aquaculture is introduced to help repopulate threatened species and rebuild their habitats. Even though Malaysia's aquaculture has been developing over the years, the data shows that in recent years, the output from aquaculture has been continuously decreasing.

Therefore, Malaysia's aquaculture still has a long way to go for it to have a steady and significant production to ensure that it is effective enough to elevate the pressure put on fish stocks by capture fisheries.

10. References

9.5 Non-seasonal ARIMA models | Forecasting: Principles and Practice (3rd ed). (n.d.). https://otexts.com/fpp3/non-seasonal-arima.html

amikang. (2023, January 21). *S.Korea_Aquaculture_Time_Series_Prediction*. Kaggle.com; Kaggle. https://www.kaggle.com/code/amikang/s-korea-aquaculture-time-series-prediction

Bobbitt, Z. (2023, July 20). *How to Detrend Data (With Examples)*. Statology. https://www.statology.org/detrend-data/

FAO. (2024). Malaysia. Text by Mazuki Hashim. In: *Fisheries and Aquaculture*. https://www.fao.org/fishery/en/countrysector/my/en

GeeksforGeeks. (2023, October 20). *Time Series Decomposition techniques*. GeeksforGeeks. https://www.geeksforgeeks.org/time-series-decomposition-techniques/

GeeksforGeeks. (2024, February 15). *Time Series CrossValidation*. GeeksforGeeks. https://www.geeksforgeeks.org/time-series-cross-validation/

Hayes, A. (2024, July 31). *Autoregressive Integrated Moving Average (ARIMA) Prediction Model*. Investopedia. https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp

Pallavi. (2024, November 12). *Granger Causality in Time Series – Explained using Chicken and Egg problem*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/08/granger-causality-in-time-series-explained-using-chicken-and-egg-problem/

Salami, A. (2024, November 24). Cross Correlation with Two Time Series in Python. *Data Insight*. https://www.datainsightonline.com/post/cross-correlation-with-two-time-series-in-python

Selan, S. (2021, January 12). There aren't plenty fish in the sea anymore, Malaysians warned. *MalaysiaNow*. https://www.malaysianow.com/news/2021/01/11/there-arent-plenty-fish-in-the-sea-anymore-malaysians-warned

Serge Geukjian. (2024). *Fish and Overfishing*. Kaggle.com. https://www.kaggle.com/datasets/sergegeukjian/fish-and-overfishing?select=aquaculture-farmed-fish-production.csv

What is aquaculture. (n.d.). ASC International. https://asc-aqua.org/learn-about-seafood-farming/what-is-aquaculture/