Project Report

Source

Introduction to Data Science
DATA7001

GROUP-3

CROP PRODUCTION CHALLENGES IN A RAPIDLY GROWING WORLD

Members:

Anmol Arora	47945209
Yan Liu	46075707
Leonardo Vaz	47388824
Huijun Yu	47749500
Vivek Kumar Gupta	47793446

We give consent for this report to be used as a teaching resource.

ABSTRACT

This analysis focuses on the challenges posed by population growth in global crop production. The objective is to understand the potential implications of population growth on food security and identify countries that may face food insecurity, security, or a balanced situation in the future based on their population size. However, it is important to note that population growth is just one aspect of the complex dynamics influencing crop production. In the whole process of the survey, the main stakeholders are the one who consume produced crops in any form like human beings.

In section 1, we describe the motivation for this research, the main questions raised, and the people we hope to help. Section 2 describes the datasets we used and their sources. In Section 3 we specifically tuned our data sets to make them fit for use. Section 4 analyzes the dataset we used and creates models for it. Section 5 includes answers to our questions about the crop production challenges in the rapid growing world and recommendations that could be made to stakeholders accordingly.

The project was mostly successful as we have tried our best to follow design thinking and data science processes to answer all our research questions. What we found was that for the next 10-12 years, the world does not need to worry much about food security. In addition, we have tried to provide useful suggestions for stakeholders through the analysis of the data set.

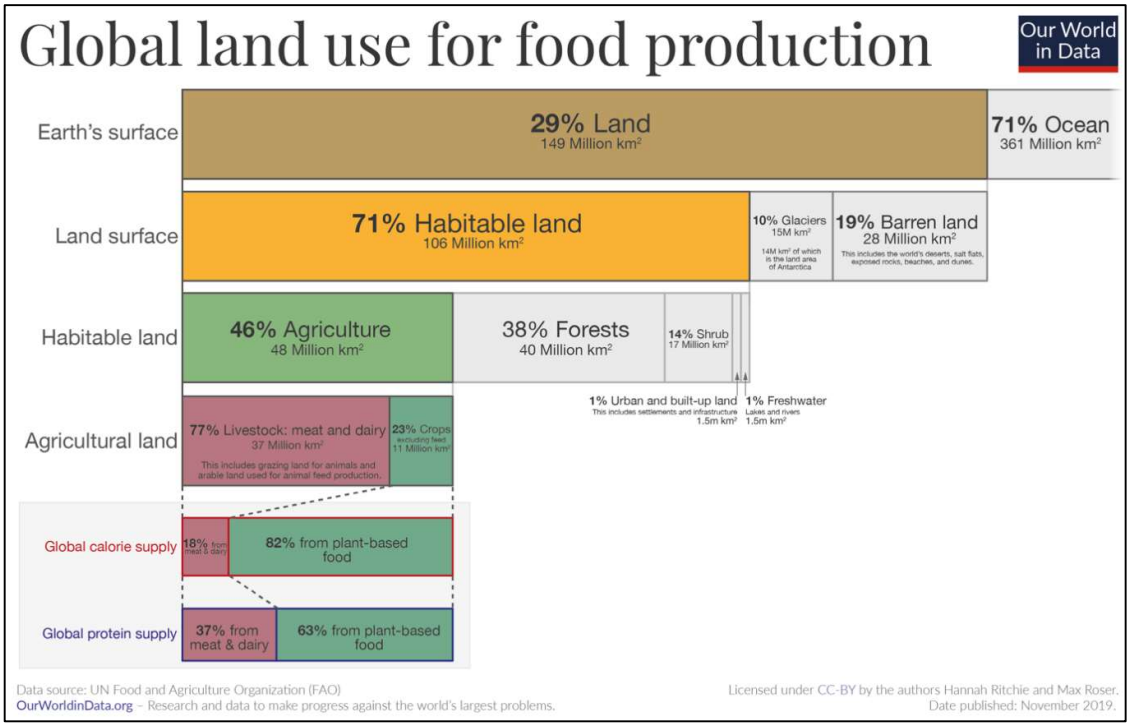
TABLE OF CONTENTS

INTRODUCTION

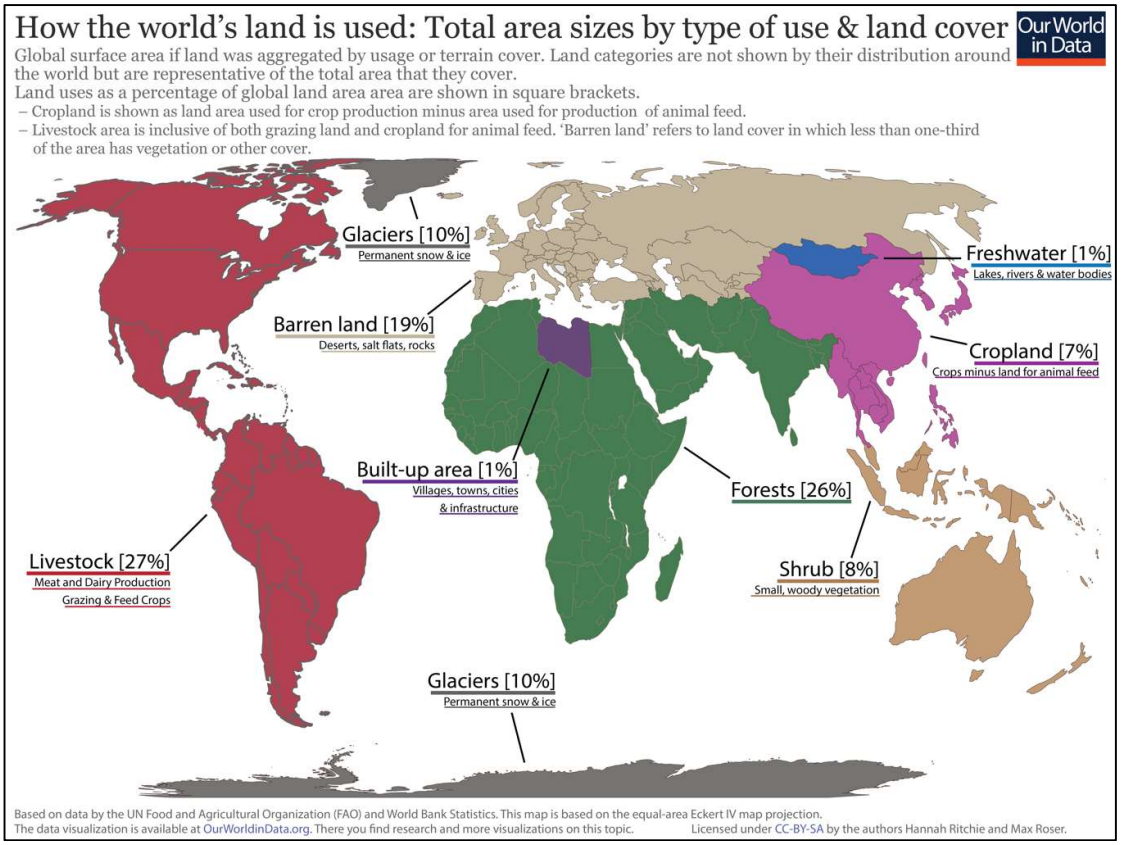
1.1 Why the Analysis?

The global population has been steadily increasing, placing greater demands on agricultural production to meet the growing food needs. At the same time, crop production plays a vital role in ensuring an adequate food supply for the population. Analysing the dynamics between global crop production and population growth provides insights into potential challenges and opportunities for achieving food security.

Another reason for this analysis is, if we have a look at the images shown below, we will get to know, that just 7% of total world land area is available for the crop production (for human population) right now. That means, cropland must be used judiciously for producing crops to effectively sustain the growing population demand and it is not an easy task to do.



Source



Source

1.2 Main challenges in crop production

Global crop production is influenced by various factors such as climate conditions, technological advancements, land availability, and agricultural practices. Increases in crop production are necessary to keep pace with population growth and ensure sufficient food availability. However, population growth rates vary across countries, and this demographic diversity has implications for food security.

1.3 Insights

Countries with high population growth rates and limited agricultural resources may face food insecurity in the future. These regions may struggle to produce enough food to meet the needs of their growing populations. Factors such as limited arable land, water scarcity, inadequate infrastructure, and socio-economic challenges can further exacerbate food insecurity. Countries with high population densities and low agricultural productivity may be particularly vulnerable to food shortages. On the other hand, countries with sustainable agricultural practices, efficient resource management, and investments in technological innovations can maintain or improve their food security status. These countries may have achieved a balance between crop production and population growth through effective agricultural policies, infrastructure development, and support systems. They have the capacity to meet the food demands of their population and even contribute to international food markets.

1.4 Future trends

To determine which countries will be food insecure, secure, or balanced in the future based on their population size, a comprehensive analysis is required. This analysis should consider factors beyond population size, such as agricultural productivity, import and export capabilities, investment in agricultural research and development, and government policies supporting food security initiatives. Additionally, assessments of regional variations, climate change impacts, and socio-economic factors are necessary to make accurate projections.

1.5 Major limitation/s of the analysis

Analysis is focused on population growth only which oversimplifies the complex dynamics of crop production. It neglects the interplay of various factors such as climate change, land availability, use and degradation, water scarcity, pest and disease outbreaks, technological advancements, market dynamics, and policy interventions. Ignoring these factors can lead to an incomplete understanding of the challenges and potential solutions in crop production. The impact of population growth on food security and agricultural systems varies based on factors such as regional resources, agricultural practices, infrastructure, trade relationships, and socio-economic conditions like Income disparities, poverty levels, access to land and resources, educational opportunities, social inequalities, and governance structures.

1.6 Key Stakeholders

- Farmers and Agricultural Producers
- Government and Policy Makers
- Research Institutions and Scientists
- Non-Governmental Organizations.
- Private Sector including agricultural input suppliers, agribusinesses, food processors, and retailers.
- Consumers and Civil Society.
- Financial Institutions.
- Academia and Education Institutions.

GETTING THE DATA I NEED

2.1 Crop Production Dataset (1961-2019)

Data from the Food and Agriculture Organization of the United Nations (FAO)

Area Code	Area	Item Code	Item	Element Code	Element	Unit	Y1961	Y1961F	Y1962	Y1962F	Y1963	Y1963F	Y1964	Y1964F
4	Algeria	221	Almonds, with shell	5312	Area harvested	ha	13300	F	13300	F	13300	F	14200	F
4	Algeria	221	Almonds, with shell	5419	Yield	hg/ha	4511	Fc	4511	Fc	4511	Fc	4507	Fc
4	Algeria	221	Almonds, with shell	5510	Production	tonnes	6000		6000		6000		6400	
4	Algeria	515	Apples	5312	Area harvested	ha	3400	F	3100	F	2800	F	2700	F
4	Algeria	515	Apples	5419	Yield	hg/ha	45294	Fc	45161	Fc	46429	Fc	46078	Fc
4	Algeria	515	Apples	5510	Production	tonnes	15400		14000		13000		12441	
4	Algeria	526	Apricots	5312	Area harvested	ha	4200	F	4600	F	4000	F	4200	F
4	Algeria	526	Apricots	5419	Yield	hg/ha	30286	Fc	30000	Fc	30000	Fc	35476	Fc
4	Algeria	526	Apricots	5510	Production	tonnes	12720		13800		12000		14900	
4	Algeria	366	Artichokes	5312	Area harvested	ha	5000	F	5000	F	5000	F	4520	

Figure 2.1 Crop Production Dataset (1961-2019)

Dataset Overview

The dataset provides comprehensive coverage of production of all primary crops in all countries and regions of the world, including crop statistics for 173 primary and fibre crops on five continents: Africa, the Americas, Asia, Europe and Oceania. The dataset is 34.74 MB in size and contains statistics for the above crops.

Area Code: A code that represents the geographic area.
Area: The name of the geographic area.
Item Code: A code that represents the specific crop item.
Item: The name of the crop item.
Element Code: A code that represents the type of element being measured.
Element: A description of the element being measured.
Unit: The unit of measurement for the crop statistics.
Y1960 to Y2019: Year-wise columns containing crop statistics for each year.
Y1961F, Y1962F to Y2019F: Flag columns indicating the nature of the data (estimated, calculated, imputed, or aggregated) for each year.

2.2 Flags Dataset

In addition to the main dataset, a separate flags dataset was used to provide additional information about the nature of the data. The flags dataset mapped flag codes to flag descriptions, helping to identify whether a particular value in the crop statistics was estimated, calculated, imputed, or aggregated. This information is crucial for accurate interpretation and analysis of the crop statistics.

2.3 Global Population Dataset (1961-2019)

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	1966	1967	1968	1969	1970
0	Aruba	ABW	Population, total	SP.POP.TOTL	54608.0	55811.0	56682.0	57475.0	58178.0	58782.0	59291.0	59522.0	59471.0	59330.0	59106.0
1	Africa Eastern and Southern	AFE	Population, total	SP.POP.TOTL	130692579.0	134169237.0	137835590.0	141630546.0	145605995.0	149742351.0	153955516.0	158313235.0	162875171.0	167596160.0	172475766.0
2	Afghanistan	AFG	Population, total	SP.POP.TOTL	8622466.0	8790140.0	8969047.0	9157465.0	9355514.0	9565147.0	9783147.0	10010030.0	10247780.0	10494489.0	10752971.0
3	Africa Western and Central	AFW	Population, total	SP.POP.TOTL	97256290.0	99314028.0	101445032.0	103667517.0	105959979.0	108336203.0	110798486.0	113319950.0	115921723.0	118615741.0	121424797.0

Figure 2.2 Data from the World Bank

Dataset Overview

The population dataset includes global population data from 1960 to 2021, providing comprehensive information about the population of different countries around the world. The dataset is 181 KB in size and consists of 266 rows representing different countries.

MAKING THE DATA FIT FOR USE

3.1 Data Transformation

- **Crop Production Dataset**

Prior to the analysis, the dataset was transformed from a wide format to a long format. This transformation involved converting the year columns into rows, allowing for easier analysis and visualization of the crop statistics over time.

	Area Code	Area	Item Code	Item	Element Code	Element	Unit	Y1961	Y1961F	Y1962	Y1962F	Y1963	Y1963F
0	4	Algeria	221	Almonds, with shell	5312	Area harvested	ha	13300.0	F	13300.0	F	13300.0	F
1	4	Algeria	221	Almonds, with shell	5419	Yield	hg/ha	4511.0	Fc	4511.0	Fc	4511.0	Fc

Figure 3.1 Crop Production Dataset in Wide Format

	Area Code	Area	Item Code	Item	Element Code	Element	Unit	Year	Values	Flags
0	4	Algeria	221	Almonds, with shell	5312	Area harvested	ha	1961	13300.0	F
1	4	Algeria	221	Almonds, with shell	5419	Yield	hg/ha	1961	4511.0	Fc
2	4	Algeria	221	Almonds, with shell	5510	Production	tonnes	1961	6000.0	NaN
3	4	Algeria	515	Apples	5312	Area harvested	ha	1961	3400.0	F
4	4	Algeria	515	Apples	5419	Yield	hg/ha	1961	45294.0	Fc

Figure 3.2 Crop Production Dataset in Long Format

- **Global Population Dataset**

Despite the large size of the population dataset, the dataset has simpler attributes and cleaner data compared to the crop production dataset. However, some country names are not exactly the same in both, and to ensure consistency and facilitate seamless integration with the crop production dataset, country names were corrected and standardized.

Examples of country name corrections:

- Venezuela (Bolivarian Republic of) -> Venezuela
- Republic of Korea -> South Korea
- Democratic Republic of Korea -> North Korea
- Micronesia (Federated States of Micronesia) -> Micronesia
- Iran, Islamic Rep. -> Iran
- Congo, Dem. Rep. -> Congo
- Bahamas, The -> Bahamas

3.2 Data Quality Issues

Both datasets are derived from publicly available data from authoritative institutions, so the data is generally reliable and clean for further exploration. However, to ensure that the data was suitable for use in this project, we took various steps to correct key issues in order to achieve our goals.

See Appendix: Data Cleaning for excerpts of the code used to clean the data.

3.3 Dealing with missing values using Linear Regression and Randomization

As you can see in the below graph, some of the data in the crop production dataset was missing. Do deal with those missing values, data imputation was used. To fill those missing values, interpolation between existing data points was performed and the data points out of range were removed.

Before Imputation:

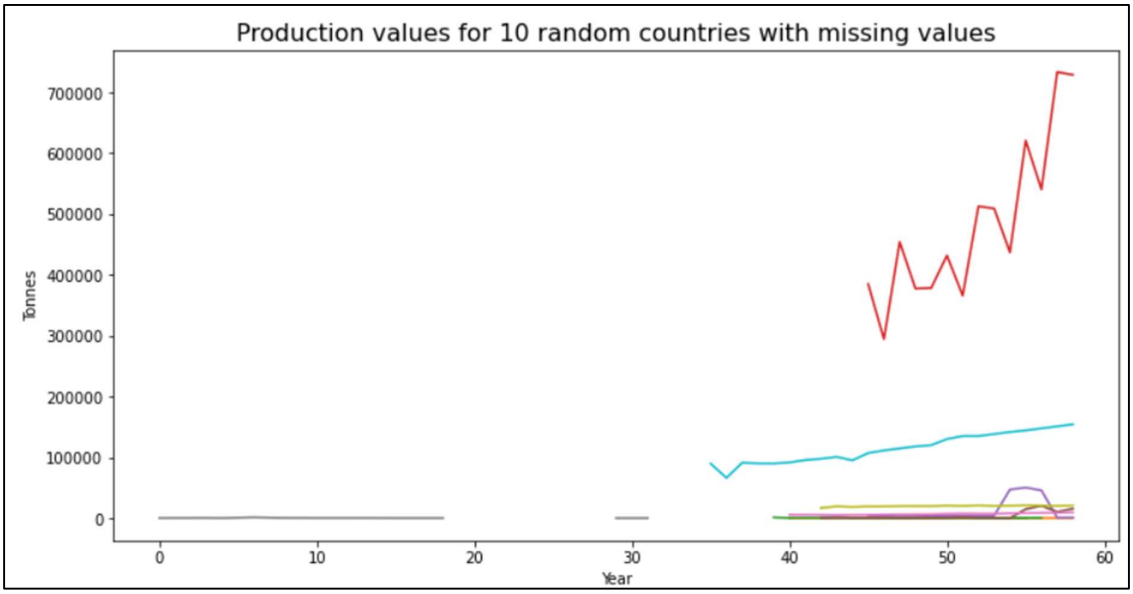


Figure 3.3 Production Values of 10 random countries before imputation

After Imputation:

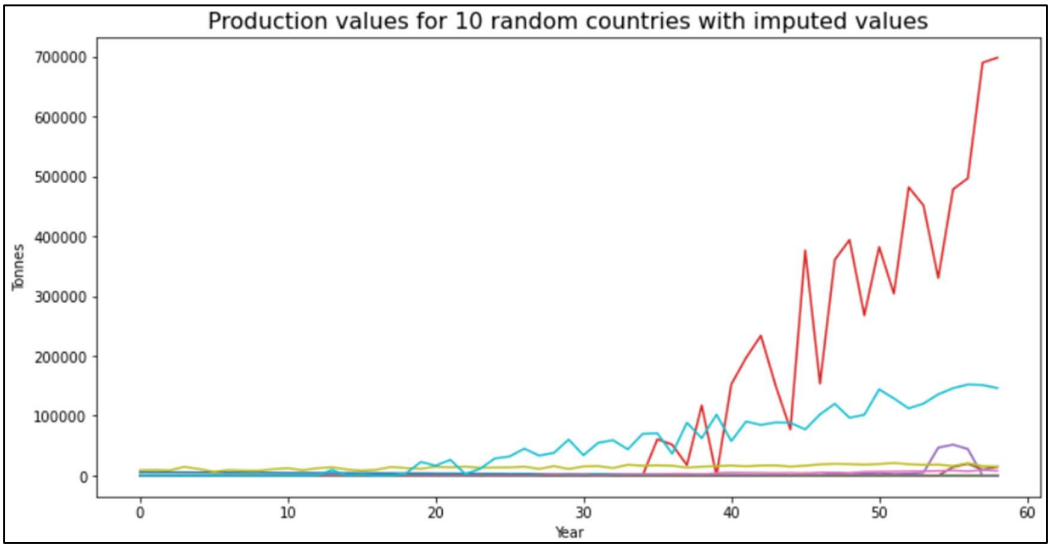


Figure 3.4 Production Values of 10 random countries after imputation

MAKING THE DATA CONFESS

4.1 Exploratory Data Analysis

- **Regional Crop Production Clustering: Unveiling Geographic Similarities in Agricultural Patterns**

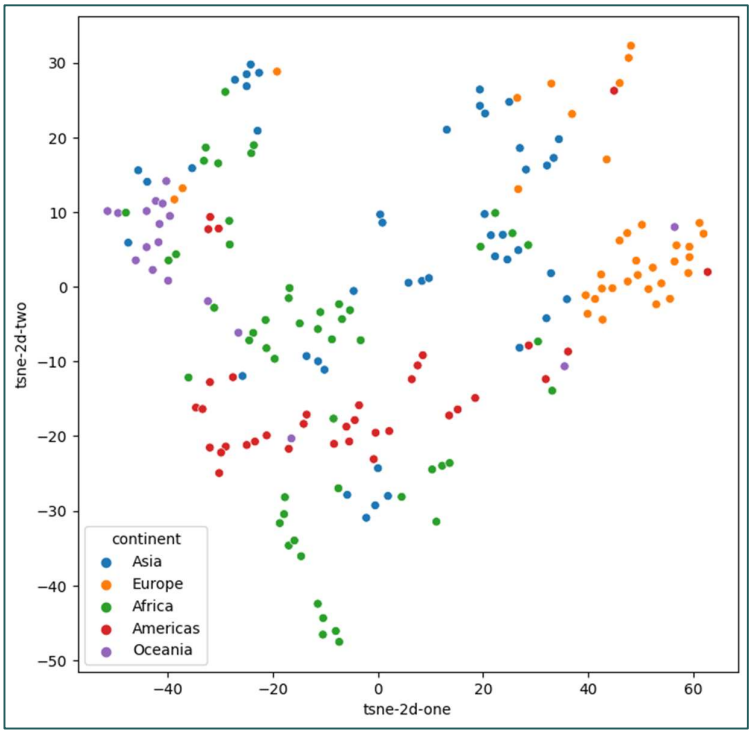


Figure 4.1

The clustering of countries was explored based on the similarity of their crop production patterns. The main objective is to identify clusters that occur as a result of growing similar crops. The color scheme used represents the continents to which these countries belong, suggesting a potential association between geographical proximity and similarity in crop production choices.

By applying clustering techniques, insight into the agricultural landscape was gained, revealing crop production patterns and trends in different regions. The clustering process was able to identify groups of countries that exhibit similar crop cultivation tendencies, highlighting potential regional agricultural similarities and common challenges.

Color-coded representations of continents enhance our understanding of how geographic proximity influences crop production choices. Neighboring countries often share common environmental conditions, climatic characteristics, and agricultural practices. As a result, they often exhibit similarities in the types of crops grown, reflecting the influence of regional factors on agricultural decisions.

Overall, it shows the interconnectedness of crop production, regional factors, and geographic proximity, emphasizing the importance of considering local and global contexts in understanding agricultural systems, enabling stakeholders to make informed decisions, promoting regional cooperation, and advancing sustainable agricultural development.

- **Sankey Diagram: Top produced Crops and Top Crop Producing Countries**

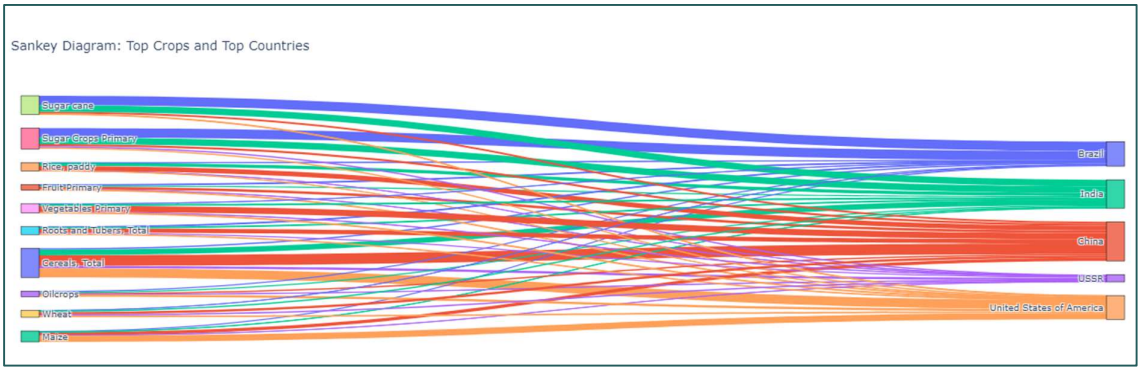


Figure 4.2

This chart visually illustrates the top ten crops and their corresponding top five producing countries. It provides a comprehensive overview of the state of global crop production, revealing the relationship between countries and their unique crops.

By linking the top crops to their major producers, the chart highlights the interconnectedness and interdependence of agricultural systems worldwide. It provides valuable insights into the geographic distribution of crop production and reveals the countries that play an important role in the cultivation of specific crops.
It demonstrates the diversity and specialization of crop production on a global scale. It allows us to identify the major players in each crop category and recognize the unique strengths and competitive advantages of different countries in meeting global food demand.

In addition, the chart allows us to identify patterns and trends in crop production, such as regional clusters of countries that specialize in specific crops. It helps us gain a deeper understanding of the factors that influence crop selection, including climate, soil conditions, market demand, and agricultural practices specific to each region.

It's a powerful tool for policy makers, researchers and stakeholders in the agricultural sector to gain insight into the dynamics of global crop production. It can inform decision-making processes related to food security, trade agreements, resource allocation, and sustainable agricultural development.

- **Hot map of global crop production**

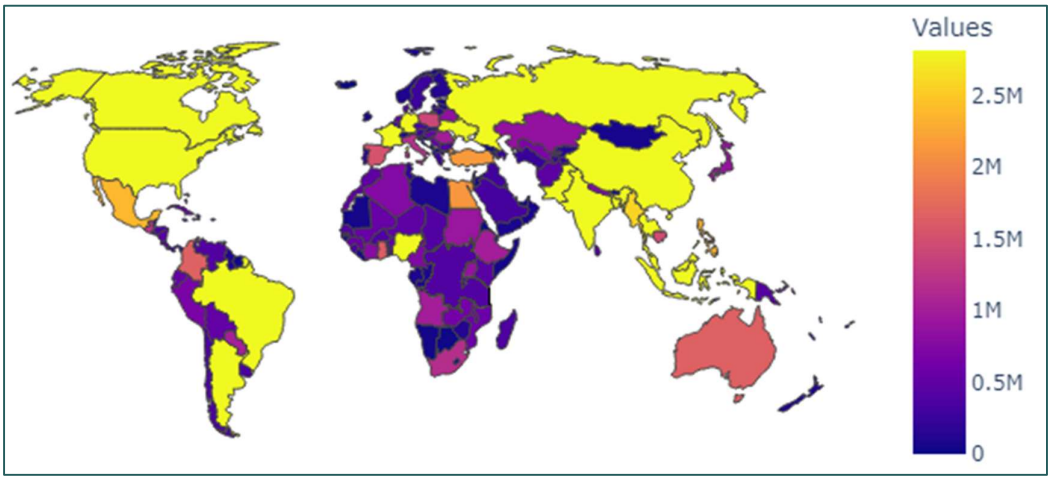


Figure 4.3

The aim was to gain insight into the distribution and trends of crop production on a global scale. Based on the total crop production values for each country, a heat map is generated to visualize crop production patterns and changes in different regions, providing a comprehensive view of global crop production patterns. The heat map provides a graphical representation of the data, with each country represented by a cell or tile. The color intensity of each cell reflects the magnitude of the crop yield, with darker colors indicating higher production levels and lighter colors indicating lower production levels.

The analysis of the spatial distribution of crop production allows the observation of patterns and trends in agricultural productivity.

4.2 Research Questions

- **Analysing Crop Production Data using Time Series**

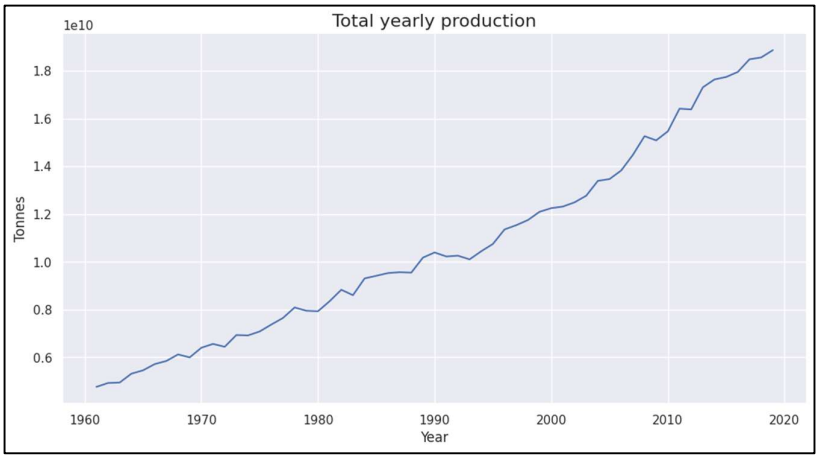


Figure 4.4

Using time series analysis of crop production data, the analysis found that total annual crop production showed a steady upward trend from year to year from 1961 to 2019.

- **Analysing Population Data using Time Series**

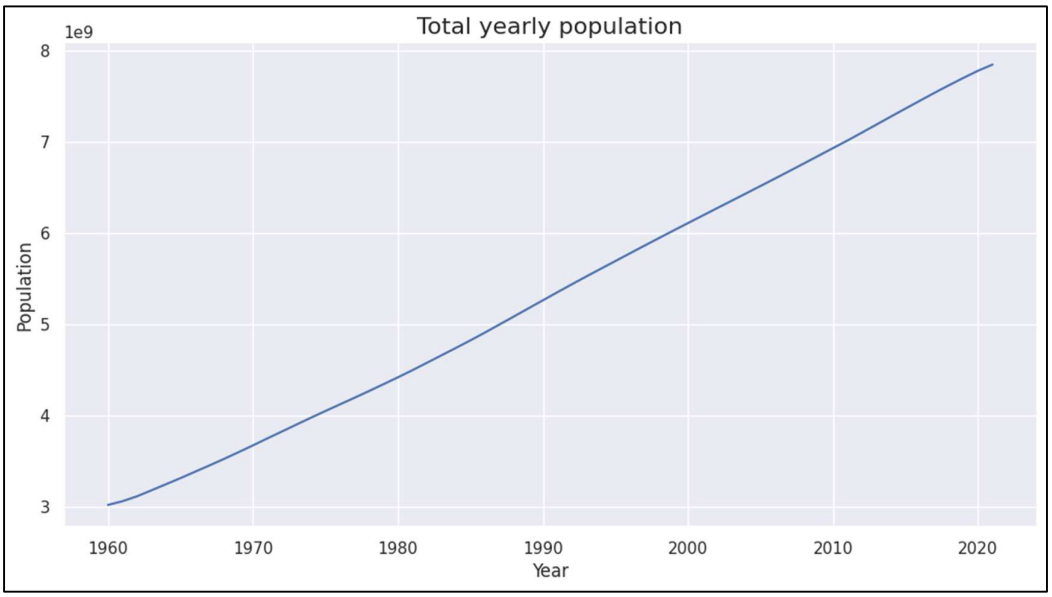


Figure 4.5

Using time series analysis of crop production data, the analysis found a linear increase in the total global population from 1961 to 2019.

- **Q1: To what extent does the rate of crop production align with population growth, and does these present potential challenges in meeting global food demand?**

- Fitting Linear Regression Model on Crop Production Dataset

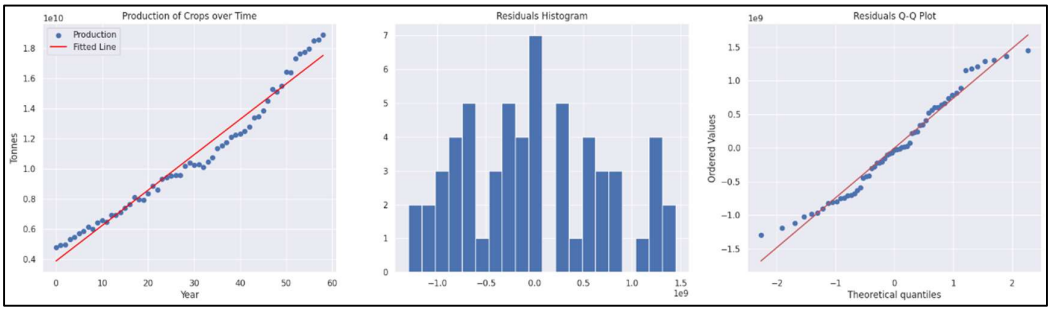


Figure 4.6

p-values for residuals: 0.099
See Appendix: Data….

- Fitting Exponential Regression Model on Crop Production Dataset

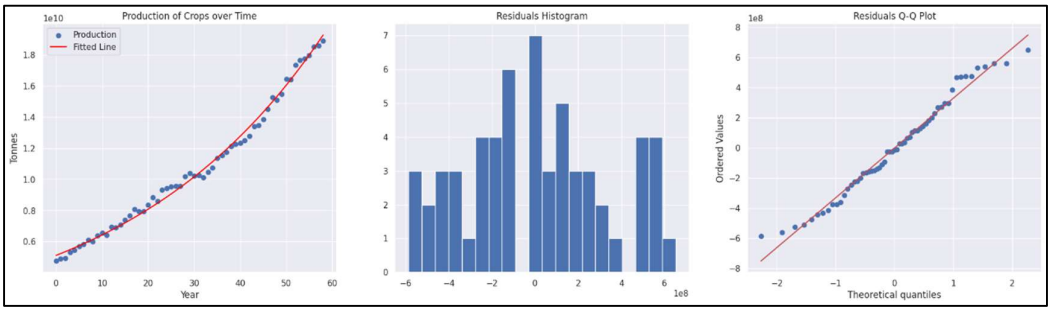


Figure 4.7

p-values for residuals: 0.172

1. A linear regression model and an exponential regression model were fitted on the Crop Production Dataset to analyze the relationship between the independent variable (years) and the dependent variable (crop production). The p-values for the residuals were calculated as 0.099 for the linear regression model and 0.172 for the exponential regression model.

2. The p-value for residuals in statistical analysis measures the probability of observing the residuals or errors as extreme or more extreme than what was observed, assuming the null hypothesis is true. In this case, a lower p-value indicates stronger evidence against the null hypothesis, suggesting that the model's residuals deviate significantly from the expected values.

3. Comparing the p-values for residuals between the linear regression model (0.099) and the exponential regression model (0.172), we can draw some conclusions.

4. For the linear regression model, the p-value of 0.099 suggests that there is a moderate level of evidence against the null hypothesis. It indicates that the residuals may deviate to some extent from the expected values, potentially indicating a lack of perfect fit between the independent variable (years) and the dependent variable (crop production).

5. On the other hand, the exponential regression model has a slightly higher p-value for residuals (0.172), indicating a weaker level of evidence against the null hypothesis compared to the linear regression model. This suggests that the exponential model's residuals may deviate less from the expected values, implying a relatively better fit between the independent variable and the dependent variable compared to the linear model.

6. The analysis showed that both linear and exponential regression models provided reasonable fits to the crop production data set. However, based on the p-values of the residuals, the exponential regression model exhibited a relatively better fit compared to the linear regression model. This implies that the exponential regression model may capture the underlying relationship between year and crop yield with greater precision and accuracy.

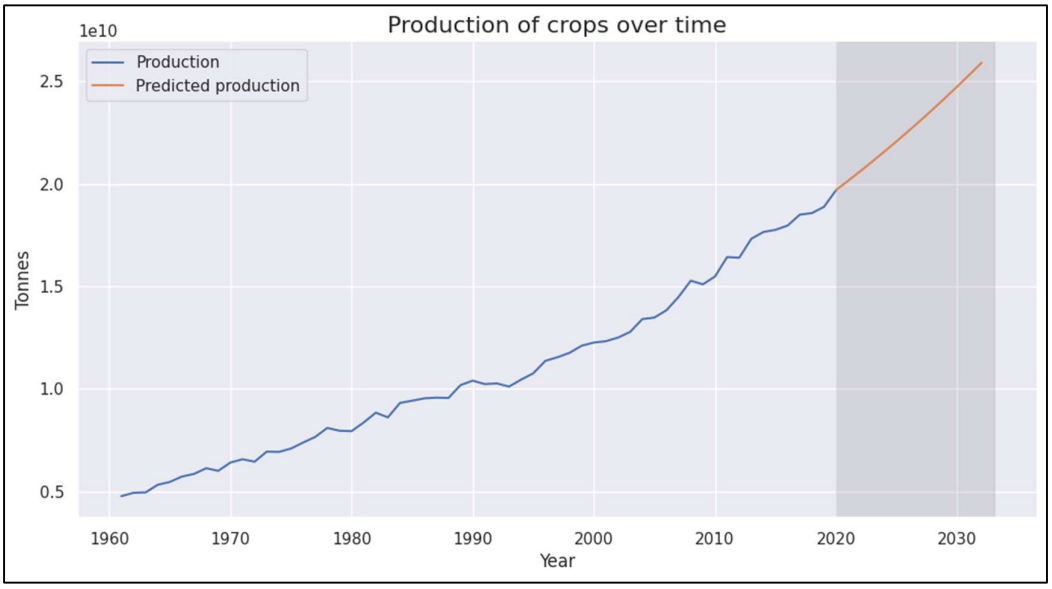


Figure 4.8

7. Therefore, an exponential fit regression was chosen to predict future crop production data (up to 2033). The projections are shown in the figure above.

- Fitting Linear Regression Model on Population Dataset

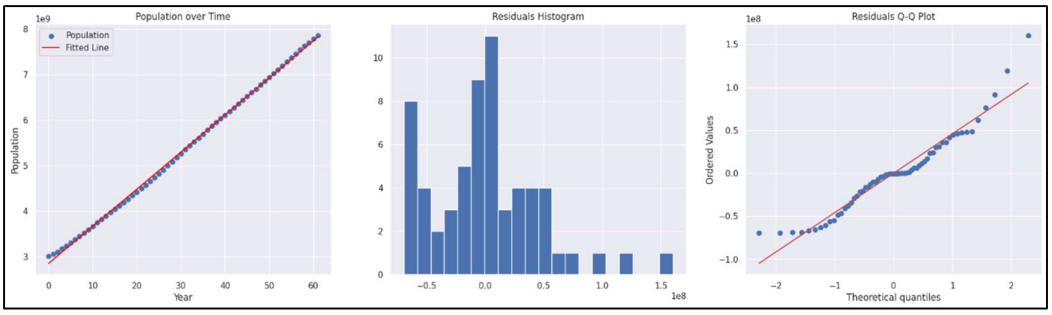


Figure 4.9

p-values for residuals: 0.004

- Fitting Exponential Regression Model on Population Dataset

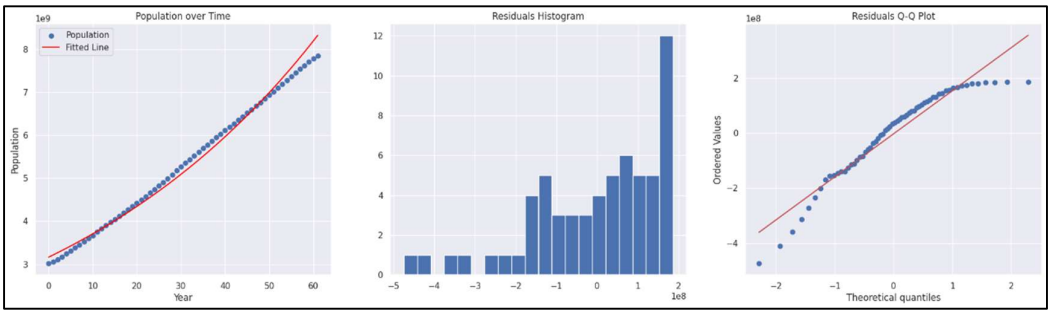


Figure 4.10

p-values for residuals: 0.010

1. A linear regression model and an exponential regression model were fitted on the Population Dataset to analyse the relationship between the independent variable (years) and the dependent variable (population). The p-values for the residuals were calculated as 0.004 for the linear regression model and 0.010 for the exponential regression model.
2. The p-value for residuals in statistical analysis measures the probability of observing the residuals or errors as extreme or more extreme than what was observed, assuming the null hypothesis is true. A lower p-value indicates stronger evidence against the null hypothesis, suggesting that the model's residuals deviate significantly from the expected values.
3. Comparing the p-values for residuals between the linear regression model (0.004) and the exponential regression model (0.010), we can draw some conclusions.
4. For the linear regression model, the p-value of 0.004 indicates strong evidence against the null hypothesis. It suggests that the residuals significantly deviate from the expected values, indicating a potential lack of perfect fit between the independent variable (years) and the dependent variable (population). The low p-value implies that the linear regression model may not adequately capture the underlying relationship between years and population.
5. On the other hand, the exponential regression model has a slightly higher p-value for residuals (0.010), indicating a weaker level of evidence against the null hypothesis compared to the linear regression model. This suggests that the exponential model's residuals may deviate less from the expected values, implying a relatively better fit between the independent variable and the dependent variable compared to the linear model.
6. The analysis shows that both linear and exponential regression models can be used to analyze the relationship between year and population. However, based on the p-values of the residuals, the linear regression model exhibited stronger evidence of fit compared to the exponential regression model. This suggests that linear models may capture the underlying relationship between year and population with relatively better accuracy and precision.

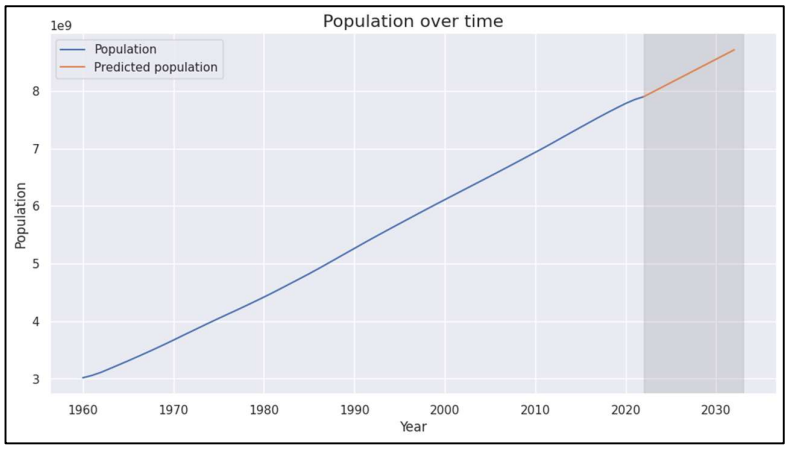


Figure 4.11

7. Therefore, a linear regression was chosen to predict future population data (up to 2033). The projections are shown in the figure above.

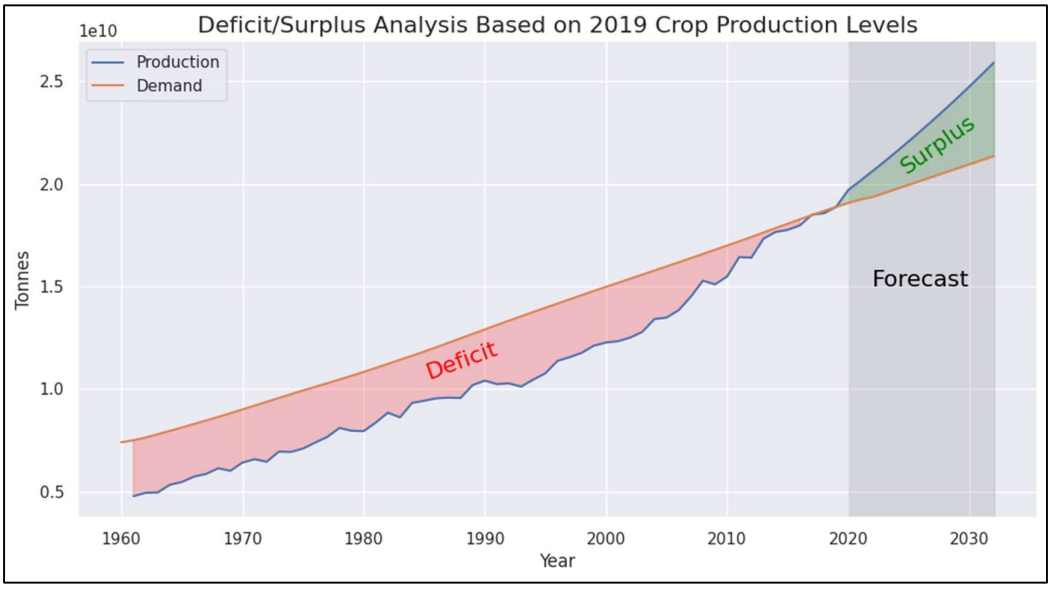


Figure 4.12 Deficit/Surplus Forecasting for the next 10 Years

According to this forecast, the global crop production is anticipated to exceed its demand in the next decade.

To assess the extent to which the crop production rate coincides with FOOD DEMAND, a projection analysis of the FOOD DEMAND situation after 2019 and up to 2033 was conducted using the required global crop production per capita in 2019 as the BASELINE.

The analysis shows that after 2019, the year represented on the right side of the baseline (global crop production in 2019), a crop production surplus is expected by 2033, which implies that there will be enough crop production to meet the food demand of its population and potentially contribute to global food supply.

After analysing the previous graph, we came to the following conclusion:

The data suggests that the rate of growth in crop production surpasses that of population growth, implying a promising outlook for the future. Nevertheless, it is crucial to take into account additional factors, such as alternative applications of crops like animal feed and bio-fuels. It is plausible that the surplus in crop production is allocated for these alternative purposes rather than solely for nutritional needs.

- **Q2: To what extent does the rate of crop production align with population growth, and does these present potential challenges in meeting global food demand?**

Regression Models for each Country

We defined the regression models for each country separately.

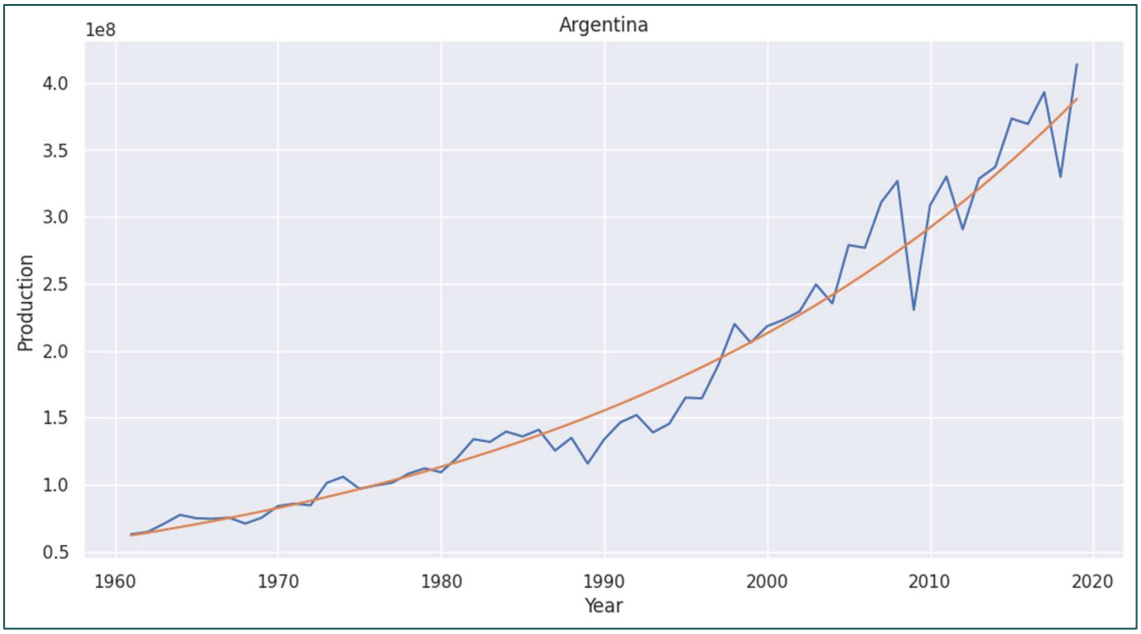


Figure 4.12

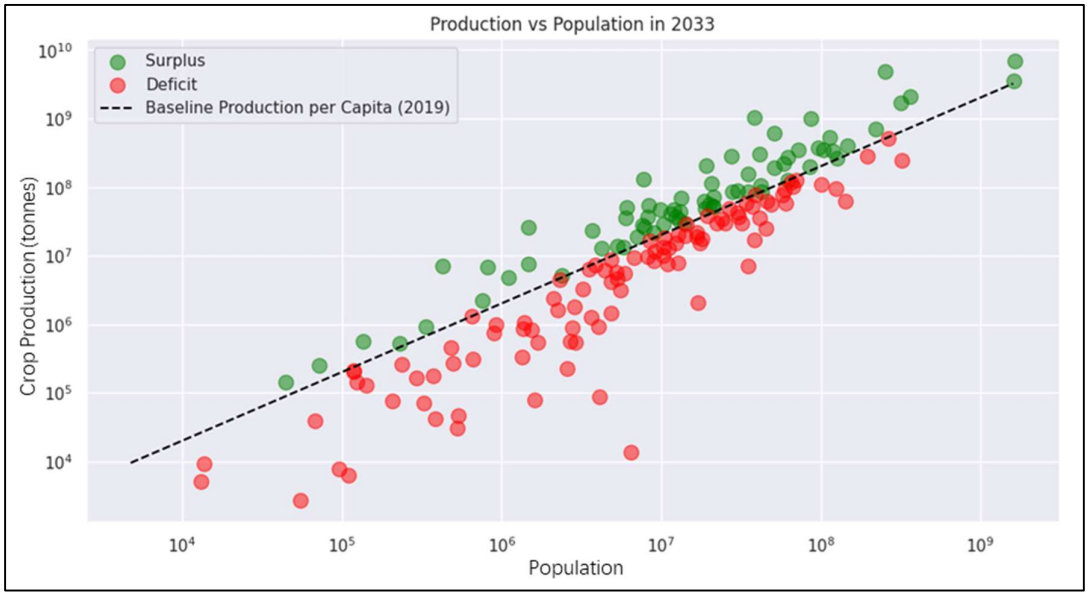


Figure 4.13 Scatter Plot of Production per Capita vs Population in 2033

To assess the extent to which the rate of crop production aligns with population growth and the potential challenges it presents in meeting global food demand, an analysis was conducted based on the projected food security in the near future. Countries were categorized into two groups: those projected to be in surplus and those projected to be in deficit, using the crop production per capita of 2019 as the requirement for the future.

The analysis revealed that a portion of countries, represented by the green dots above the black line (baseline production per capita of 2019), are expected to have a surplus of crop production by 2033. These countries are projected to have sufficient crop production to meet their population's food requirements and potentially contribute to global food supply.

On the other hand, the red dots represent countries that are projected to face a deficit in crop production by 2033, falling below the baseline production per capita of 2019. These countries are expected to experience challenges in meeting their population's food demand, indicating potential food security risks and the need for external support or strategies to bridge the food deficit gap.

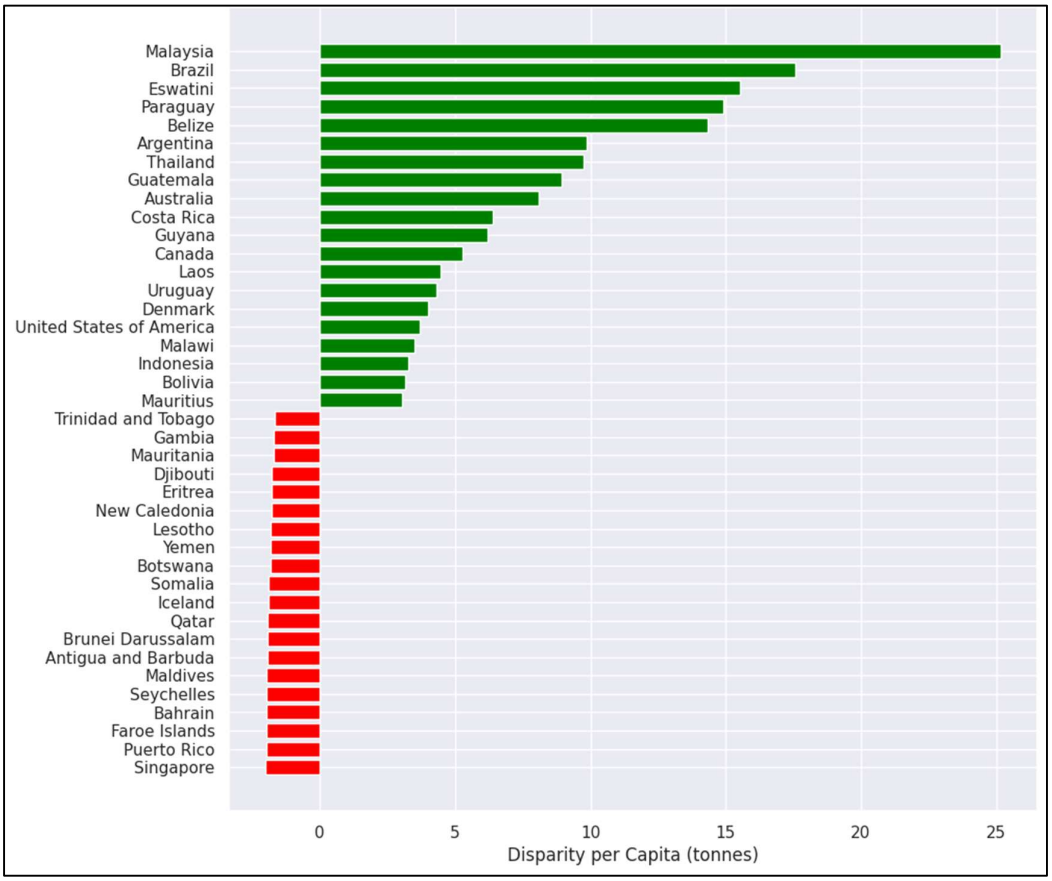


Figure 4.14 Top and Bottom Countries by Disparity for Crop Production per Capita in 2033

The analysis of the top and bottom countries by disparity for crop production per capita in 2033 reveals notable findings. Singapore emerges as the country projected to face the most significant deficit in crop production per capita by 2033, alongside Puerto

Rico, Faroe Islands, and Bahrain. These countries are expected to encounter challenges in meeting their population's food requirements due to a considerable gap between their crop production and population size.

Conversely, Malaysia, Brazil, Eswatini, Paraguay, and Belize are projected to have the highest crop production per capita a decade later. These countries are expected to exhibit a favourable balance between their crop production and population size, indicating their ability to meet or even exceed their population's food requirements.

- **Q3: Considering the anticipated increase in global food demand, what extent of land area will be required for harvesting in order to sufficiently address future needs?**

Forecasting Crop Production and Yield for the Next Decade

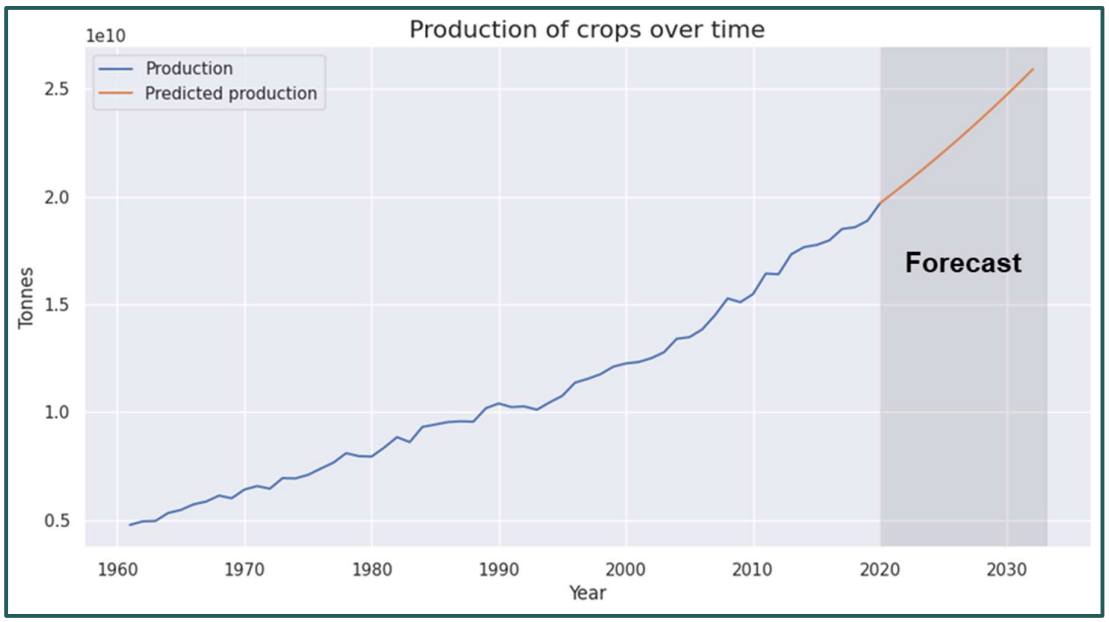


Figure 4.15

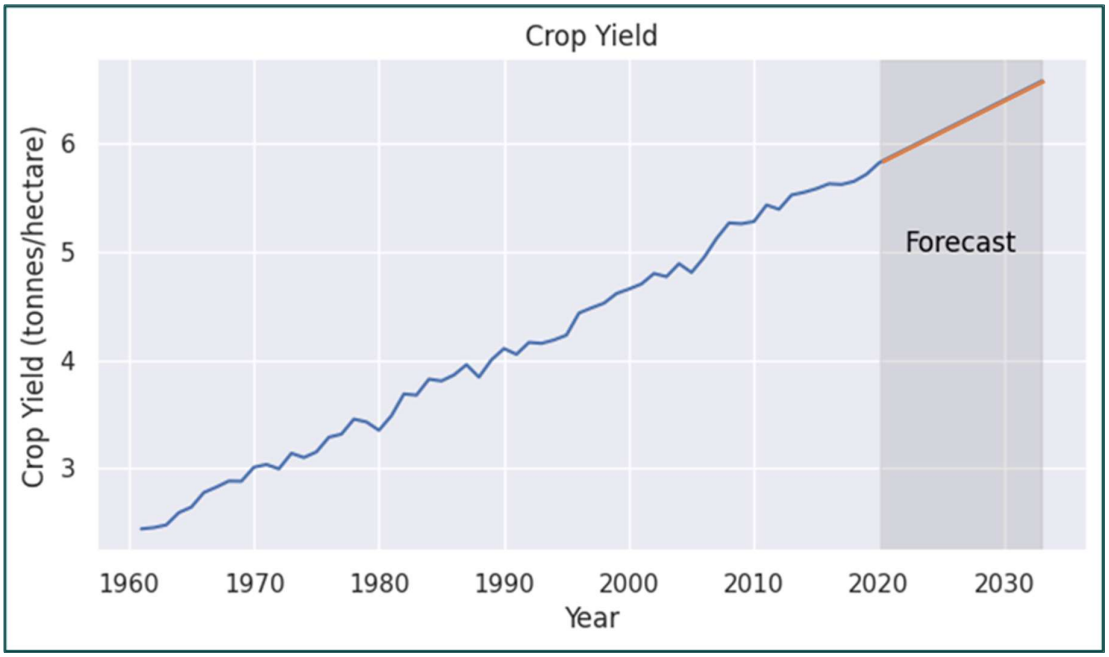


Figure 4.16

- **Forecasted Crop Production for 2033: 25.91 billion tonnes**
- **Forecasted Crop Yield for 2033: 6.58 tonnes/hectare**

After projecting crop production and yield 10 years into the future, we calculated the area that will be required for harvesting in 2033 by dividing the crops produced in tonnes by the crop yield in tonnes/hectare:
Area required for harvesting in 2033 = Crops Produced (tonnes) / Crop Yield (tonnes/hectare)

= (25.91 billion / 6.58) hectares
= 3.94 billion hectares

This means that a staggering 3.94 billion hectares of land would need to be dedicated to agriculture in order to sufficiently address the future food needs. It's important to note that this estimate is based on the projected production and yield, and there are various factors that could influence the actual land requirements, such as advancements in farming practices and technology, changes in dietary habits, and efforts to reduce food waste.

Storytelling with Data

5.1 Conclusion

The analysis highlights the relationship between crop production and population growth and identifies potential challenges in meeting global food demand.

The findings indicate that, based on current projections:

1) Global crop production is expected to exceed demand in the next decade. However, it is important to note that this analysis focuses solely on the rate of crop production and population growth, and other factors such as climate change, technological advancements, and policy interventions are not taken into account. Therefore, the findings should be interpreted with caution and considered in conjunction with a more comprehensive analysis.

2) Additionally, the analysis categorizes countries based on their ability to meet food requirements. It identifies countries, such as Singapore, Puerto Rico, Faroe Islands, and Bahrain, that are projected to face deficits in crop production per capita in the future. Conversely, countries like Malaysia, Brazil, Eswatini, Paraguay, and Belize are expected to have higher crop production per capita. This categorization provides insights into the varying levels of food security across different regions.

3) Furthermore, the analysis estimates the land area required for harvesting to address future food needs. The calculation suggests that approximately 3.94 billion hectares of land would be necessary to meet anticipated global food demand in 2033. However, it is important to acknowledge that this estimate is based on projected production and yield, and other factors, such as advancements in farming practices and technology, changes in dietary habits, and efforts to reduce food waste, can influence actual land requirements.

5.2 Future Steps

To further enhance the analysis and provide a more comprehensive understanding of the challenges in global food production, several future steps and areas of work can be considered:

1) **Incorporate additional factors**: Expand the analysis to include other critical factors influencing crop production and food security, such as climate change, water availability, technological advancements, market dynamics, policy interventions, and socio-economic factors. This will provide a more comprehensive and accurate assessment of the challenges and potential solutions.

2) **Assess regional and local variations**: Conduct a more granular analysis that considers regional and local variations in crop production and food security. Different regions have distinct resources, agricultural practices, infrastructure, and socio-economic conditions, which significantly impact their ability to meet food requirements.

3) **Evaluate sustainability**: Incorporate sustainability considerations into the analysis, including the environmental impact of crop production, resource management, and the adoption of sustainable farming practices. This will ensure that future food production strategies align with long-term environmental goals.

4) **Explore technological advancements**: Investigate the potential of technological advancements, such as precision agriculture, digital farming, and genetic engineering, to enhance crop productivity and address food production challenges. Assess the feasibility and impact of implementing these technologies on a global scale.

5) **Consider policy interventions**: Analyse the effectiveness of different policy interventions, such as agricultural subsidies, research and development funding, trade policies, and food security programs, in promoting sustainable crop production and addressing food security concerns. Identify best practices and policy recommendations for governments and international organizations.

6) **Monitor and update projections**: Continuously monitor and update projections of crop production, population growth, and other relevant factors to ensure the analysis remains up-to-date and reflective of changing global dynamics. Regularly reassess the findings and refine the analysis as new data becomes available.

By undertaking these future steps and expanding the scope of the analysis, policymakers, researchers, and stakeholders can gain deeper insights into the challenges of crop production and develop informed strategies to ensure global food security in a rapidly growing world.

5.3 Answering the feedbacks/peer reviews

In response to feedback, we have improved our report to be more professional and descriptive. Taking into consideration the suggestions provided by our professors, we have incorporated a more detailed explanation of our data imputation methodology on page 10. By doing so, we aim to provide a clearer understanding of how we handled missing data in our analysis.

Peer's feedback suggested us to explain more about choosing models. We have provided more comprehensive justifications for utilizing exponential regression in production analysis and linear regression in population analysis. This will help readers gain insights into our decision-making process and understand the rationale behind our chosen models.

Furthermore, we have implemented the suggestion to include maps in our report. Specifically, we have created a hot map visualization of crop production, which offers a comprehensive overview of the distribution and trends of crop production on a global scale. By incorporating these maps, we aim to provide readers with a visual representation that supports the analysis and findings presented in our report.

REFERENCES

- **Crop Production Dataset**

data.world. (n.d.). data.world. https://data.world/agriculture/crop-production/workspace/project-summary?agentid=agriculture&datasetid=crop-production

- **Global Population Dataset**

World Bank Open Data. (n.d.). World Bank Open Data. https://data.worldbank.org/indicator/SP.POP.TOTL?end=2021&start=1960&view=chart

APPENDIX

Code:

Loading Libraries

In [ ]:
```
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import scipy.stats as stats
from sklearn.linear_model import LinearRegression
import seaborn as sns
import matplotlib.pyplot as plt

pd.options.display.max_columns = None
```

Loading & Transforming Data

Load Crop Production Data

In [ ]:
```
# Get filenames of crop production datasets
prod_crops_ds_filename = [filename for filename in os.listdir('data') if filename.startswith('Production')]

# Read all crop production datasets into a single dataframe
df = pd.DataFrame()
for filename in prod_crops_ds_filename:
    temp_df = pd.read_csv(os.path.join('data', filename), encoding='ISO-8859-1')
    df = pd.concat([df, temp_df])
```

In [ ]:
```
df.dropna().head()
```

Out[ ]:

	Area Code	Area	Item Code	Item	Element Code	Element	Unit	Y1961	Y1961F	Y1962	Y1962F	Y1963	Y1963F	Y1964	Y1964F	Y1965
7	10	Australia	515	Apples	5419	Yield	hg/ha	110682.0	Fc	122396.0	Fc	127242.0	Fc	132186.0	Fc	125340.0
10	10	Australia	526	Apricots	5419	Yield	hg/ha	107251.0	Fc	107251.0	Fc	104022.0	Fc	86337.0	Fc	113069.0
13	10	Australia	367	Asparagus	5419	Yield	hg/ha	39833.0	Fc	39208.0	Fc	38960.0	Fc	33275.0	Fc	37620.0
19	10	Australia	486	Bananas	5419	Yield	hg/ha	119640.0	Fc	119640.0	Fc	111881.0	Fc	124690.0	Fc	130432.0
22	10	Australia	44	Barley	5419	Yield	hg/ha	9759.0	Fc	10940.0	Fc	12080.0	Fc	13390.0	Fc	10204.0

Transform Years from columns to rows

In [ ]:
```
# Get column names
cols = df.columns[:7].to_list()
value_columns = [col for col in df.columns if col.startswith('Y') and not col.endswith('F')]
flag_columns = [col for col in df.columns if col.startswith('Y') and col.endswith('F')]

# Transform dataset from wide to long format (year columns to rows)
df_values = df.melt(id_vars=cols, value_vars=value_columns, var_name='Year', value_name='Values')
df_values['Year'] = df_values['Year'].str[1:].astype(int)
df_flags = df.melt(id_vars=cols, value_vars=flag_columns, var_name='Year', value_name='Flags')
df_flags['Year'] = df_flags['Year'].str[1:-1].astype(int)
df = pd.merge(df_values, df_flags, on=cols + ['Year'])
```

In [ ]:
```
df.head()
```

Out[ ]:

	Area Code	Area	Item Code	Item	Element Code	Element	Unit	Year	Values	Flags
0	10	Australia	221	Almonds, with shell	5312	Area harvested	ha	1961	NaN	M
1	10	Australia	221	Almonds, with shell	5419	Yield	hg/ha	1961	NaN	NaN

In [ ]:
```
countries_conversion_table = pd.read_csv('country_names_conversion.csv')
countries_conversion_table = countries_conversion_table[~(countries_conversion_table['Final'].isna())]
```

In [ ]:
```
crop_country_name_map = countries_conversion_table[['Area', 'Final']].dropna().set_index('Area')['Final'].to_dict()
```

In [ ]:
```
countries_conversion_table[~(countries_conversion_table['Country Name'].isna())].sample(10)
```

Out[ ]:

	Area	Country Name	Final
84	NaN	Iran, Islamic Rep.	Iran
86	NaN	St. Kitts and Nevis	Saint Kitts and Nevis
155	Bhutan	Bhutan	Bhutan
226	Luxembourg	Luxembourg	Belgium-Luxembourg
170	Colombia	Colombia	Colombia
251	North Macedonia	North Macedonia	North Macedonia
193	Gabon	Gabon	Gabon
195	Germany	Germany	Germany
130	NaN	Tanzania	Tanzania
151	Belarus	Belarus	Russia (USSR)

In [ ]:
```
df['Area'] = df['Area'].map(crop_country_name_map)
```

In [ ]:
```
demographics_country_name_map = countries_conversion_table[['Country Name', 'Final']].dropna().set_index('Country Name')['Final'].to_dict()
```

In [ ]:
```
# Read flags dataset (flags are used to indicate whether a value is an estimate, calculated, imputed, or aggregated)
flags = pd.read_csv('data/flags.csv', encoding='ISO-8859-1')
flags.columns = ['Flag', 'Flag Description']
flags.set_index('Flag', inplace=True)
flag_to_description = flags.to_dict(orient='dict')['Flag Description']

# Map flag codes to flag descriptions
df['Flag Description'] = df['Flags'].map(flag_to_description)
```

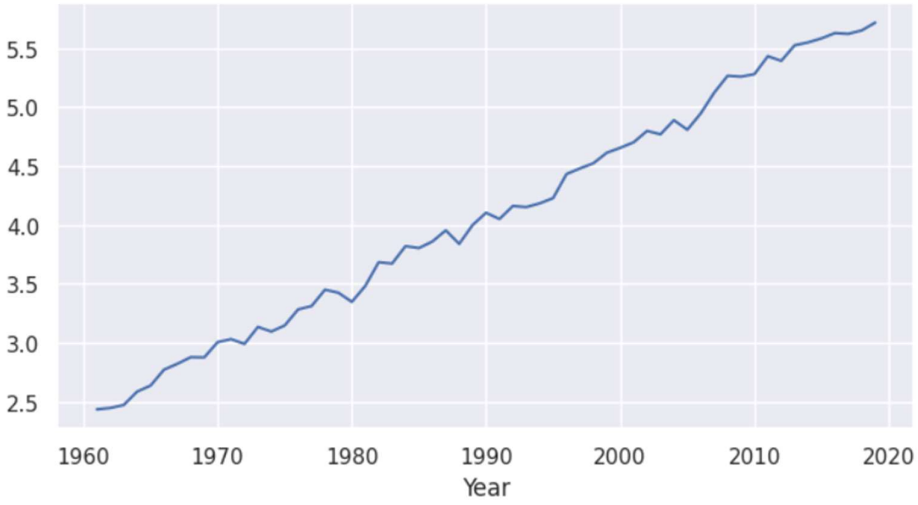
In [ ]:
```
df.sample(10)
```

Out[ ]:

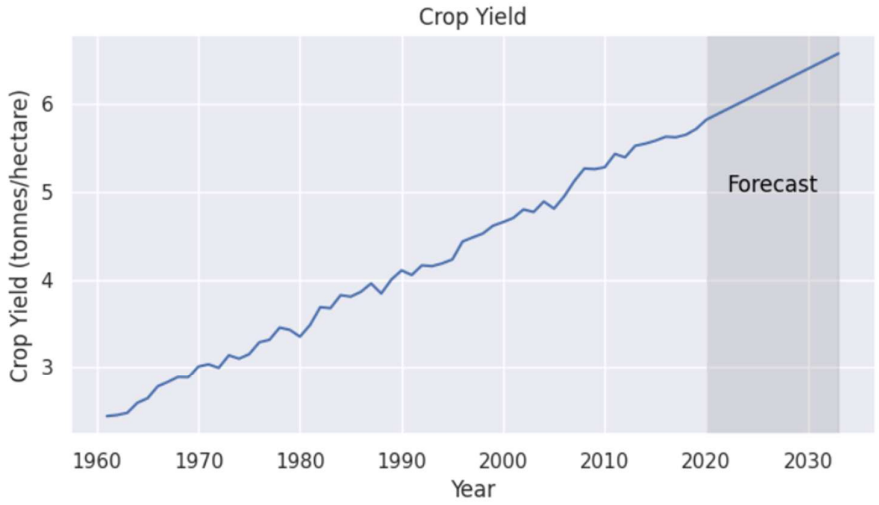
	Area Code	Area	Item Code	Item	Element Code	Element	Unit	Year	Values	Flags	Flag Description
911890	89	Guatemala	242	Groundnuts, with shell	5312	Area harvested	ha	1984	910.0	NaN	Not applicable
2074743	209	Eswatini	463	Vegetables, fresh nes	5312	Area harvested	ha	2015	1205.0	Im	FAO data based on imputation methodology
1652048	228	Russia (USSR)	554	Cranberries	5510	Production	tonnes	2004	NaN	NaN	Not applicable
1273635	209	Eswatini	490	Oranges	5419	Yield	hg/ha	1994	74092.0	Fc	Calculated data
2207214	44	Colombia	1735	Vegetables Primary	5419	Yield	hg/ha	2018	142426.0	Fc	Calculated data
319499	72	Djibouti	176	Beans, dry	5312	Area harvested	ha	1969	NaN	M	Data not available
1353237	158	Niger	195	Cow peas, dry	5312	Area harvested	ha	1996	3043968.0	NaN	Not applicable
1138547	33	Canada	236	Soybeans	5312	Area harvested	ha	1990	483600.0	NaN	Not applicable
804746	167	Czechoslovakia	490	Oranges	5510	Production	tonnes	1982	NaN	NaN	Not applicable
434620	238	Ethiopia	702	Nutmeg, mace and cardamoms	5419	Yield	hg/ha	1972	NaN	NaN	Not applicable

In [ ]:
```
crop_yield = (df[df['Element'] == 'Production'].groupby('Year')['Values'].sum()/df[df['Element'] == 'Area harvested'].groupby('Year')['Values'].sum())
crop_yield.plot()
```

Out[ ]:
```
<Axes: xlabel='Year'>
```



In [ ]:
```
yield_linear_model = LinearRegression()
yield_linear_model.fit(crop_yield.index.values.reshape(-1, 1), crop_yield.values.reshape(-1, 1))
yield_linear_model.score(crop_yield.index.values.reshape(-1, 1), crop_yield.values.reshape(-1, 1))
yield_forecast = yield_linear_model.predict(np.arange(2020, 2034).reshape(-1, 1))
yield_forecast_2033 = yield_forecast[-1]
yield_all = np.concatenate([crop_yield.values, yield_forecast.reshape(-1)])
plt.plot(np.arange(1961, 2034), yield_all)
plt.title('Crop Yield')
plt.xlabel('Year')
plt.ylabel('Crop Yield (tonnes/hectare)')
plt.text(2022, 5, 'Forecast', fontsize=12, color='black')
plt.axvspan(2020, 2033, alpha=0.2, color='grey')
plt.show()
print('Crop yield forecast for 2033: {:.2f} tonnes/hectare'.format(yield_forecast_2033[0]))

# Continues after the graph for production forecast
```




```
Crop yield forecast for 2033: 6.58 tonnes/hectare
```

n [ ]:
```
df[df['Element'] == 'Area harvested'].groupby('Year')['Values'].sum()
```

ut[ ]:
```
Year
1961    2.174839e+09
1962    2.210998e+09
1963    2.240979e+09
1964    2.283463e+09
```

Load Demographic Data

In [ ]:
```
# Load demographics dataset
demographics = pd.read_csv('data/demographics.csv')
demographics.head()
```

Out[ ]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	1966	
0	Aruba	ABW	Population, total	SP.POP.TOTL	54608.0	55811.0	56682.0	57475.0	58178.0	58782.0	59291.0	59
1	Africa Eastern and Southern	AFE	Population, total	SP.POP.TOTL	130692579.0	134169237.0	137835590.0	141630546.0	145605995.0	149742351.0	153955516.0	158313
2	Afghanistan	AFG	Population, total	SP.POP.TOTL	8622466.0	8790140.0	8969047.0	9157465.0	9355514.0	9565147.0	9783147.0	1001(
3	Africa Western and Central	AFW	Population, total	SP.POP.TOTL	97256290.0	99314028.0	101445032.0	103667517.0	105959979.0	108336203.0	110798486.0	113319
4	Angola	AGO	Population, total	SP.POP.TOTL	5357195.0	5441333.0	5521400.0	5599827.0	5673199.0	5736582.0	5787044.0	5827

In [ ]:
```
demographics['Country Name'] = demographics['Country Name'].map(demographics_country_name_map)
demographics = demographics[~(demographics['Country Name'].isna())]
```

In [ ]:
```
# Transform dataset from wide to long format (year columns to rows)
year_columns = [x for x in demographics.columns if x.isnumeric()]
demographics = demographics.melt(id_vars=['Country Name', 'Country Code'], value_vars=year_columns, var_name='Year', value_name='Populat
demographics['Year'] = demographics['Year'].astype(int)
demographics.head()
```

Out[ ]:

	Country Name	Country Code	Year	Population
0	Afghanistan	AFG	1960	8622466.0
1	Angola	AGO	1960	5357195.0
2	Albania	ALB	1960	1608800.0
3	United Arab Emirates	ARE	1960	133426.0
4	Argentina	ARG	1960	20349744.0

In [ ]:
```
# Filter out countries that are not in crop production dataset
demographics = demographics[demographics['Country Name'].isin(df['Area'].unique())]
```

In [ ]:
```
demographics[demographics['Year']==2021]['Population'].sum()
```

Out[ ]:
```
7849333635.0
```

Dealing with Missing Values

In [ ]:
```
df[df['Element']=='Production'].isna().sum()
```

Out[ ]:
```
Area Code              0
Area                22125
Item Code              0
Item                   0
Element Code           0
Element                0
Unit                   0
Year                   0
Values            206468
Flags             437663
Flag Description       0
dtype: int64
```

In [ ]:
```
demographics.isna().sum()
```

Out[ ]:
```
Country Name    0
Country Code    0
Year            0
Population      0
dtype: int64
```

In [ ]:
```
pivot = df[df['Element']=='Production'].pivot_table(index=['Area', 'Item'], columns='Year', values='Values', aggfunc='max')
```

In [ ]:
```
pivot.isna()
```

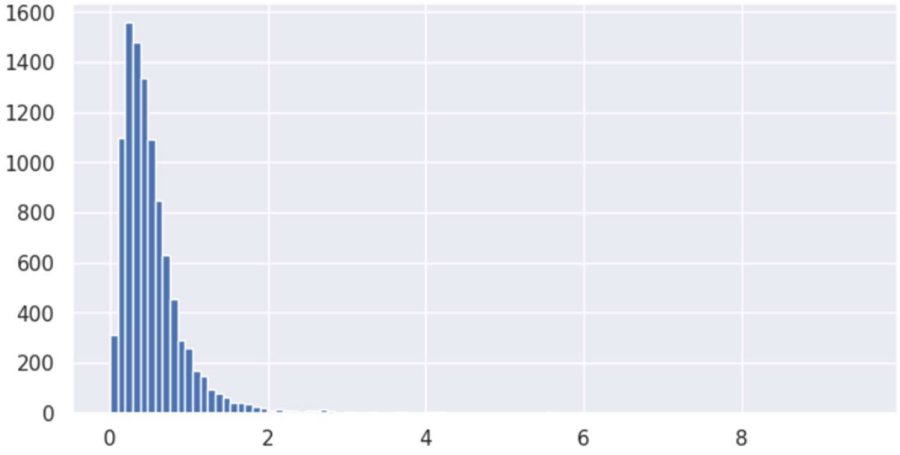
Out[ ]:

	Year	1961	1962	1963	1964	1965	1966	1967	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979
Area	**Item**																			
Afghanistan	**Almonds, with shell**	True	True	True	True	True	True	True	True	True	True	True	True	True	True	False	False	False	False	False
	Anise, badian, fennel, coriander	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
	Apples	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
	Apricots	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
	Barley	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	**...**	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Zimbabwe	**Vanilla**	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
	Vegetables Primary	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
	Vegetables, fresh nes	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False

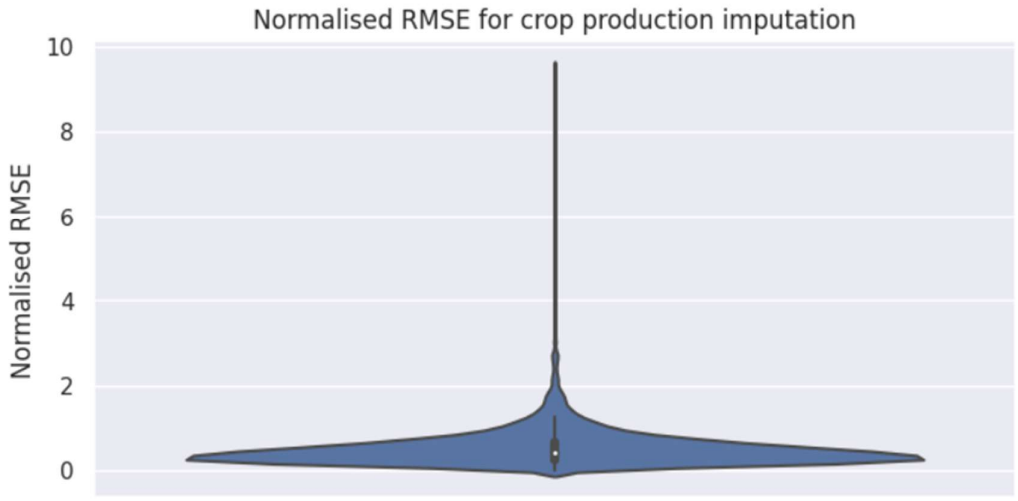
In [ ]:
```
from sklearn.model_selection import TimeSeriesSplit
errors = {}
def linear_regression(row):
    # x_axis
    x = np.arange(row.shape[0])
    isna = row.isna()
    x = x[~isna]
    y = row[~isna]
    n_folds = 5
    tscv = TimeSeriesSplit(n_splits=n_folds)
    error = 0
    mean = y.mean()
    if len(x) > n_folds:
        for fold_index, (train_index, test_index) in enumerate(tscv.split(x)):
            x_train, x_test = x[train_index], x[test_index]
            y_train, y_test = y.values[train_index], y.values[test_index]
            model = LinearRegression()
            model.fit(x_train.reshape(-1, 1), y_train)
            pred = model.predict(x_test.reshape(-1, 1))
            error += np.sqrt(np.mean((pred - y_test)**2))
            # errors[row.name] = {f'fold_{fold_index}': np.sqrt(np.mean((pred - y_test)**2))}
        error /= n_folds
        # error /= mean
        errors[row.name] = {'error': error, 'mean': mean}
        noise_baseline = mean * 0.05
        noise = np.random.normal(-noise_baseline, noise_baseline, size=row.shape[0])
        model = LinearRegression()
        model.fit(x.reshape(-1, 1), y.values)
        pred = model.predict(np.arange(row.shape[0]).reshape(-1, 1))
        row = row.fillna(pd.Series(pred, index=row.index)) + noise
        row = np.maximum(row, 0)
    return row

pivot = df[df['Element']=='Production'].pivot_table(index=['Area', 'Item'], columns='Year', values='Values', aggfunc='max')
imputed_df = pivot.apply(lambda row: linear_regression(row), axis=1)
```

In [ ]:
```
error_df_crops = pd.DataFrame(errors).T
error_df_crops.reset_index(inplace=True)
error_df_crops.columns = ['Area', 'Crop', 'Error', 'Mean']
error_df_crops['Normalised Error'] = error_df_crops['Error'] / error_df_crops['Mean']
error_df_crops['Normalised Error'].hist(bins=100)
```

Out[ ]: <Axes: >

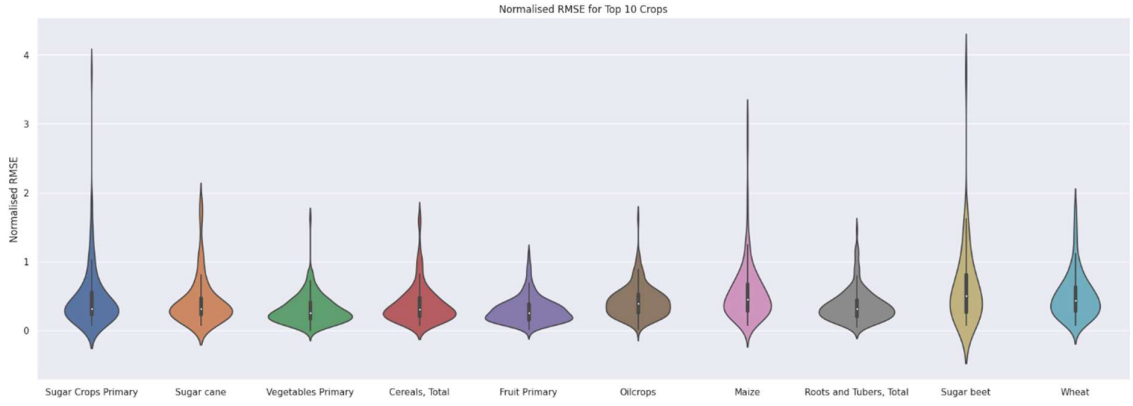


In [ ]:
```
# Plot the violin plot
sns.set(rc={'figure.figsize':(8, 4)})
sns.violinplot(error_df_crops, y="Normalised Error")

# Set labels and title
plt.xlabel("")
plt.ylabel("Normalised RMSE")
plt.title("Normalised RMSE for crop production imputation")

# Show the plot
plt.show()
```

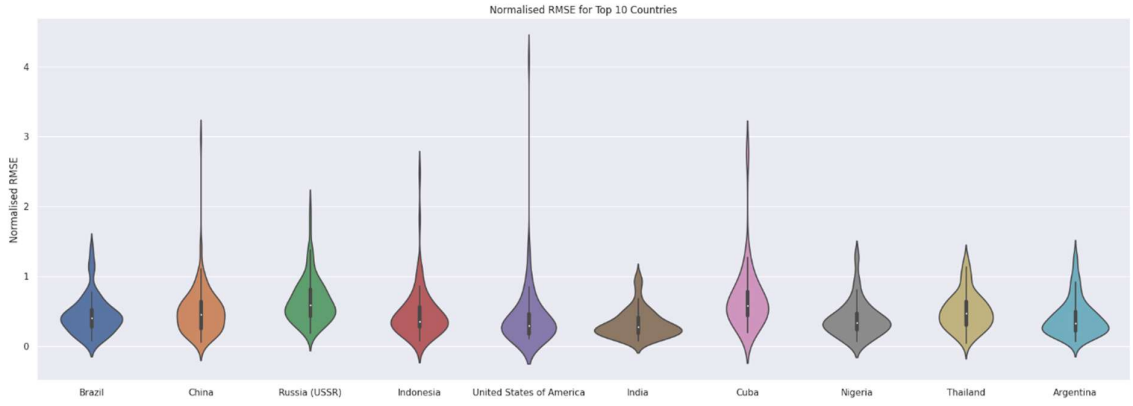


In [ ]:
```
err = [x['error'] for x in errors.values()]
means = [x['mean'] for x in errors.values()]
crops = [x[1] for x in errors.keys()]
crop_country = [x[0] for x in errors.keys()]
error_df = pd.DataFrame({'error': err, 'mean': means, 'crop': crops, 'country': crop_country}).sort_values('error', ascending=False)
error_df['error_scaled'] = error_df['error'] / error_df['mean']
top_10_crops = error_df.groupby('crop').sum().sort_values('error', ascending=False).reset_index()[:10]['crop'].to_list()
top_10_countries = error_df.groupby('country').sum().sort_values('error', ascending=False).reset_index()[:10]['country'].to_list()
error_df_crops = error_df[error_df['crop'].isin(top_10_crops)]
error_df_countries = error_df[error_df['country'].isin(top_10_countries)]
```

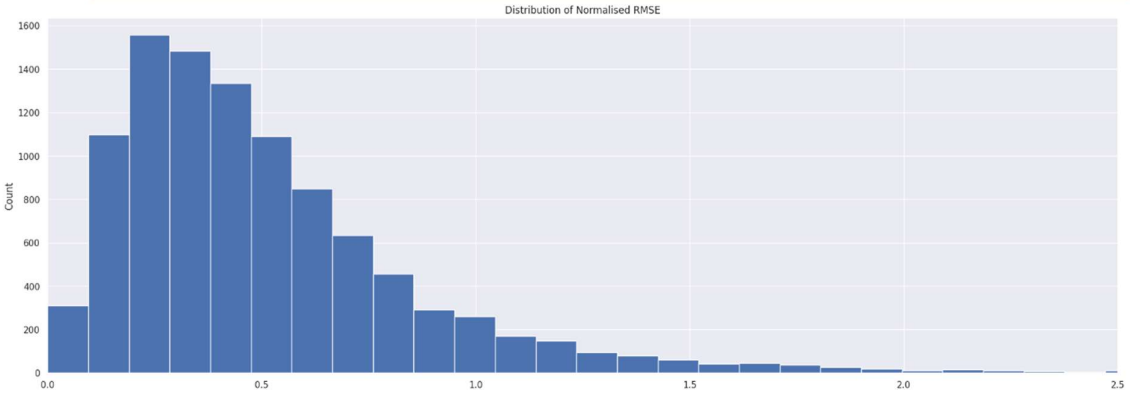
In [ ]:
```
# Plot the violin plot
sns.violinplot(error_df_crops, x="crop", y="error_scaled")
sns.set(rc={'figure.figsize':(24, 8)})
# Set labels and title
plt.xlabel("")
plt.ylabel("Normalised RMSE")
plt.title("Normalised RMSE for Top 10 Crops")

# Show the plot
plt.show()
```



In [ ]:
```
sns.violinplot(error_df_countries, x="country", y="error_scaled")
sns.set(rc={'figure.figsize':(24, 8)})
# Set labels and title
plt.xlabel("")
plt.ylabel("Normalised RMSE")
plt.title("Normalised RMSE for Top 10 Countries")

# Show the plot
plt.show()
```



In [ ]:
```
plt.hist(np.array(err)/np.array(means), bins=100);
plt.xlim(0, 2.5)
plt.ylabel('Count')
plt.title('Distribution of Normalised RMSE');
```



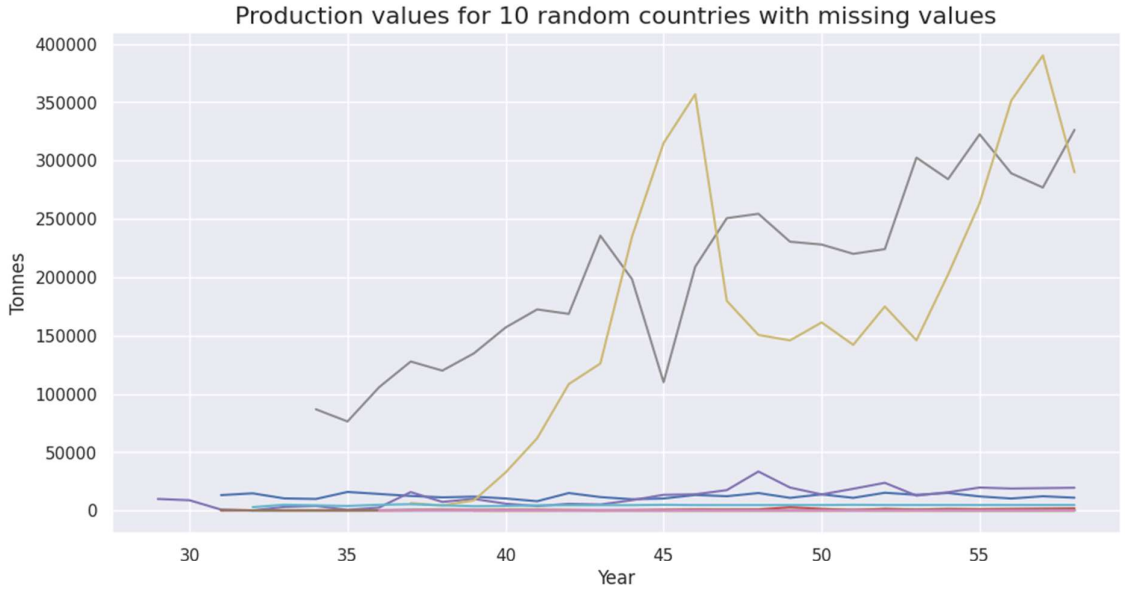
In [ ]:
```
from sklearn.model_selection import TimeSeriesSplit
def linear_regression(row):
    # x_axis
    x = np.arange(row.shape[0])
    isna = row.isna()
    x = x[~isna]
    y = row[~isna]
    mean = y.mean()
    noise_baseline = mean * 0.1
    noise = np.random.normal(-noise_baseline, noise_baseline, size=row.shape[0])
    if len(x) > 1:
        model = LinearRegression()
        model.fit(x.reshape(-1, 1), y.values)
        pred = model.predict(np.arange(row.shape[0]).reshape(-1, 1))
        row = row.fillna(pd.Series(pred, index=row.index)) + noise
        row = np.maximum(row, 0)
    return row

pivot = df[df['Element']=='Production'].pivot_table(index=['Area', 'Item'], columns='Year', values='Values', aggfunc='max')
imputed_df = pivot.apply(lambda row: linear_regression(row), axis=1)
```

[ ]:
```
pivot.loc[sample]
```

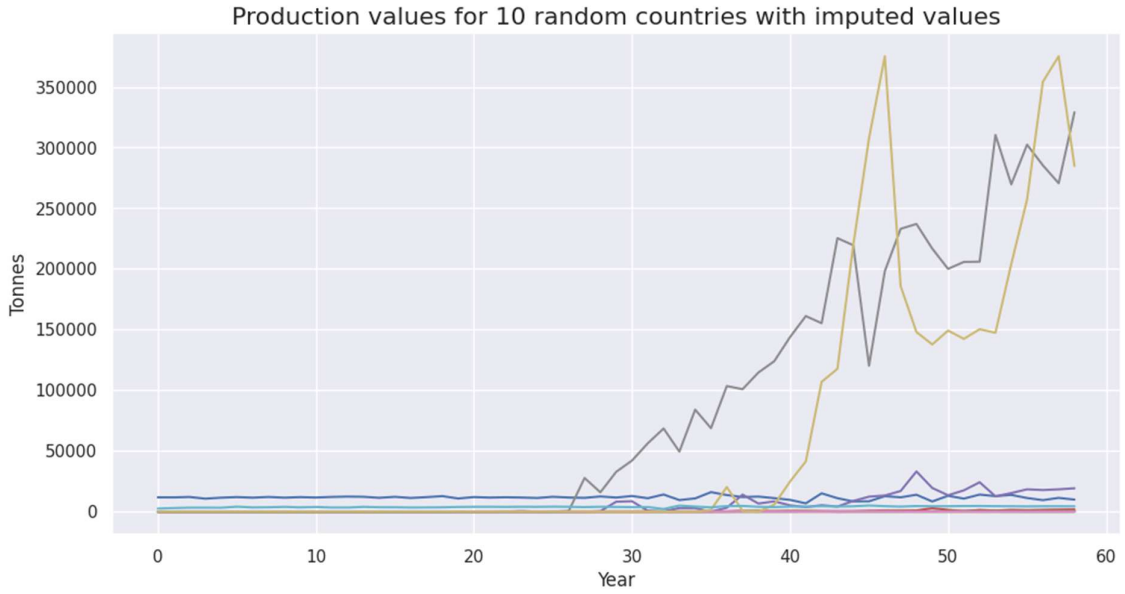
t[ ]:

Year		1961	1962	1963	1964	1965	1966	1967	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977
Area	Item																	
Venezuela	Sugar beet	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Serbia and Montenegro	Hops	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
South Africa	Rapeseed	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Guatemala	Beans, green	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Norway	Maize,	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [ ]:
```
# Visualising missing values
sample = pivot.isna().sum(axis=1).sort_values(ascending=False)[1000:2000].sample(10).index
plt.figure(figsize=(12, 6))
plt.plot(pivot.loc[sample].reset_index().values[:, 2:].T)
plt.title('Production values for 10 random countries with missing values', fontsize=16)
plt.xlabel('Year')
plt.ylabel('Tonnes')
plt.show()
```



In [ ]:
```
# Visualising imputed values
plt.figure(figsize=(12, 6))
plt.plot(imputed_df.loc[sample].reset_index().values[:, 2:].T)
plt.title('Production values for 10 random countries with imputed values', fontsize=16)
plt.xlabel('Year')
plt.ylabel('Tonnes')
plt.show()
```



In [ ]:
```
# Reshape dataframe back to rows
df = imputed_df.reset_index().melt(id_vars=['Area', 'Item'], var_name='Year', value_name='Productio
```

In [ ]:
```
df.head()
```

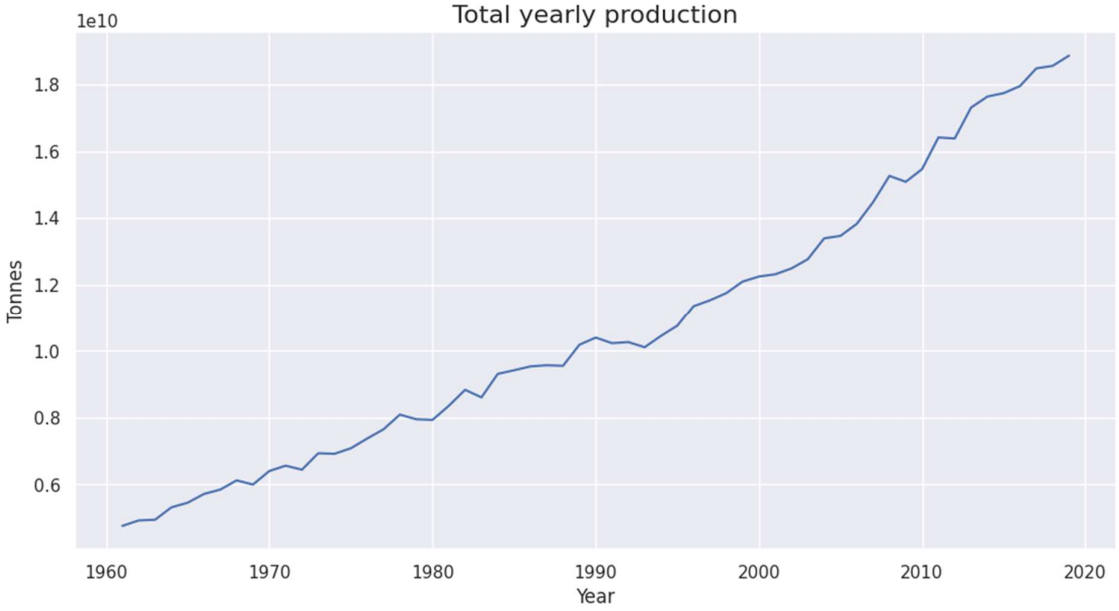
Out[ ]:

	Area	Item	Year	Production
0	Afghanistan	Almonds, with shell	1961	0.000000
1	Afghanistan	Anise, badian, fennel, coriander	1961	0.000000
2	Afghanistan	Apples	1961	12438.264703
3	Afghanistan	Apricots	1961	30644.333305
4	Afghanistan	Barley	1961	368893.164286

Analysing Crop Production Data

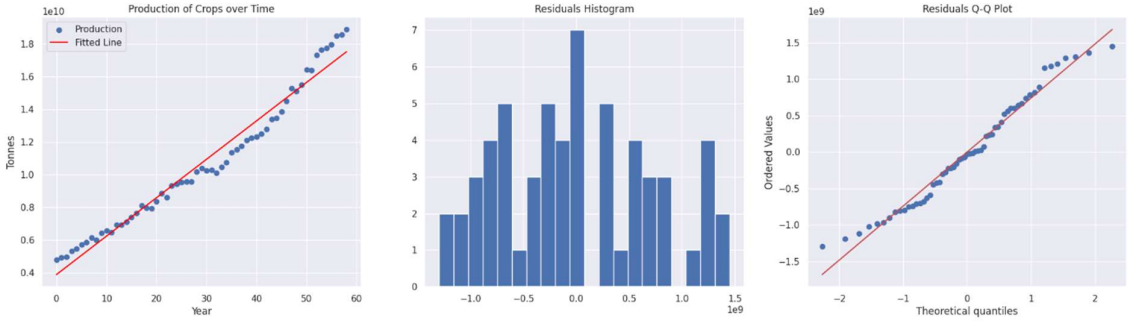
Global Crop Production by Year

n [ ]:
```
yearly_production = df.groupby('Year')['Production'].sum()
plt.figure(figsize=(12, 6))
plt.plot(yearly_production)
plt.title('Total yearly production', fontsize=16)
plt.xlabel('Year')
plt.ylabel('Tonnes')
plt.show()
```



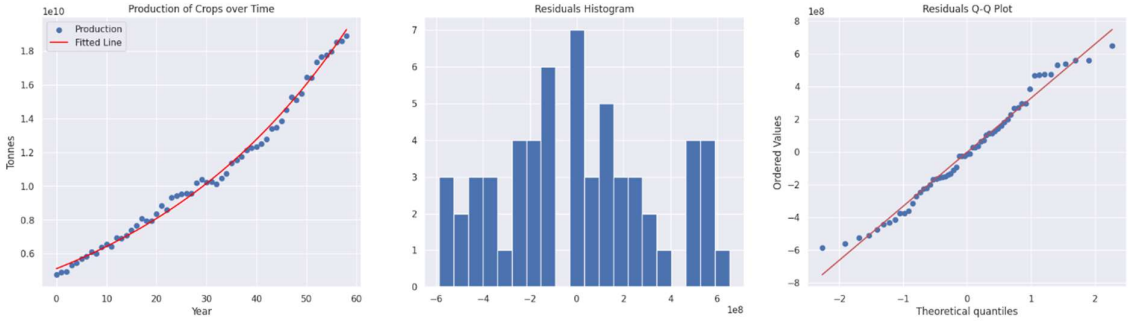
Fit a Linear Regression Model and Analyse the Results

n [ ]:
```
# Fit linear regression model to yearly production
X = np.arange(len(yearly_production)).reshape(-1, 1)
y = yearly_production.values.reshape(-1, 1)
linear_prod_model = LinearRegression()
linear_prod_model.fit(X, y)

# Calculate residuals
residuals = y - linear_prod_model.predict(X)

# Plot fitted line, residuals histogram, and residuals Q-Q plot
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(24, 6))
ax1.scatter(X, y, label='Production')
ax1.plot(X, linear_prod_model.predict(X), label='Fitted Line', color='red')
ax1.set_xlabel('Year')
ax1.set_ylabel('Tonnes')
ax1.set_title('Production of Crops over Time')
ax1.legend()
ax2.hist(residuals, bins=20)
ax2.set_title('Residuals Histogram')
stats.probplot(residuals.flatten(), plot=ax3)
ax3.set_title('Residuals Q-Q Plot')
plt.show()

# Calculate p-values for residuals
_, p_values = stats.normaltest(residuals.flatten())
print('p-values for residuals:', f'{p_values:.3f}')
```



Fit an Exponential Regression Model and Analyse the Results

In [ ]:
```
# Fit an exponential model to yearly production
X = np.arange(len(yearly_production)).reshape(-1, 1)
y = yearly_production.values.reshape(-1, 1)
exp_prod_model = LinearRegression()
exp_prod_model.fit(X, np.log(y))

# Calculate residuals
residuals = y - np.exp(exp_prod_model.predict(X))

# Plot fitted line, residuals histogram, and residuals Q-Q plot
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(24, 6))
ax1.scatter(X, y, label='Production')
ax1.plot(X, np.exp(exp_prod_model.predict(X)), label='Fitted Line', color='red')
ax1.set_xlabel('Year')
ax1.set_ylabel('Tonnes')
ax1.set_title('Production of Crops over Time')
ax1.legend()
ax2.hist(residuals, bins=20)
ax2.set_title('Residuals Histogram')
stats.probplot(residuals.flatten(), plot=ax3)
ax3.set_title('Residuals Q-Q Plot')
plt.show()

# Calculate p-values for residuals (assumes residuals are normally distributed)
_, p_values = stats.normaltest(residuals.flatten())
print('p-values for residuals:', f'{p_values:.3f}')
```

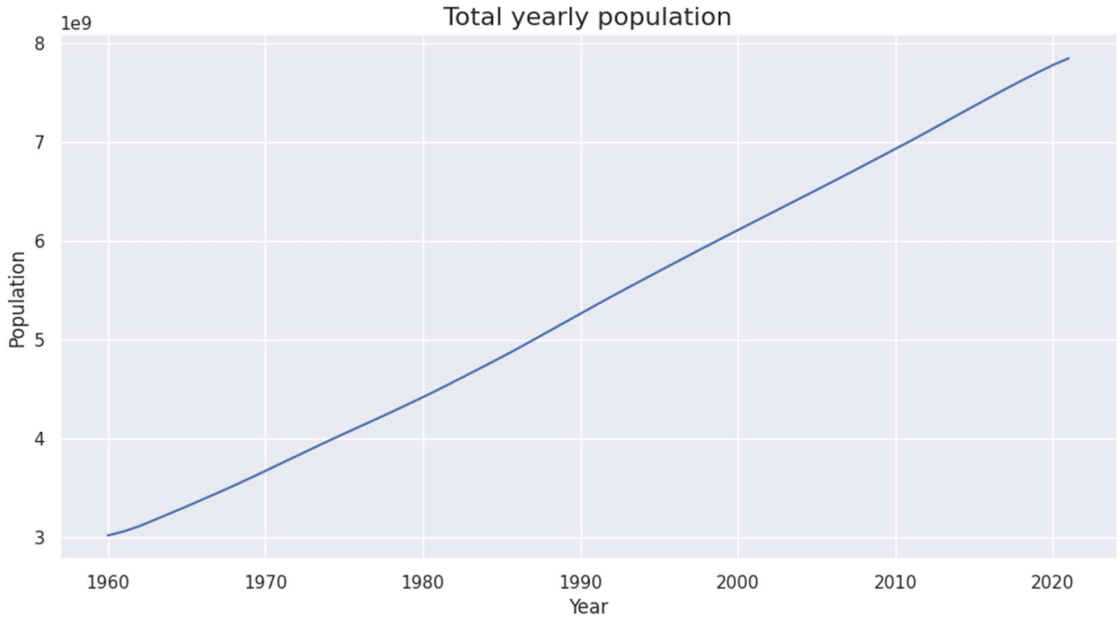



```
p-values for residuals: 0.172
```

Analysing Demographic Data

Population by Year

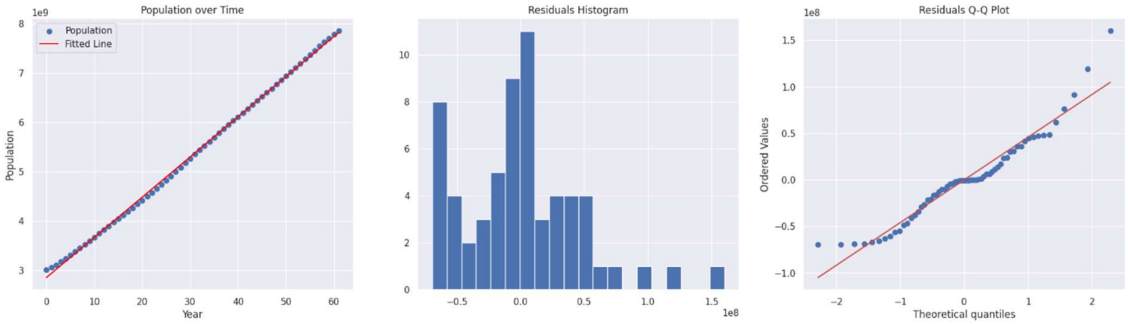
]:
```
population = demographics.groupby('Year')['Population'].sum()
plt.figure(figsize=(12, 6))
plt.plot(population)
plt.title('Total yearly population', fontsize=16)
plt.xlabel('Year')
plt.ylabel('Population')
plt.show()
```



Fit a Linear Regression Model and Analyse the Results

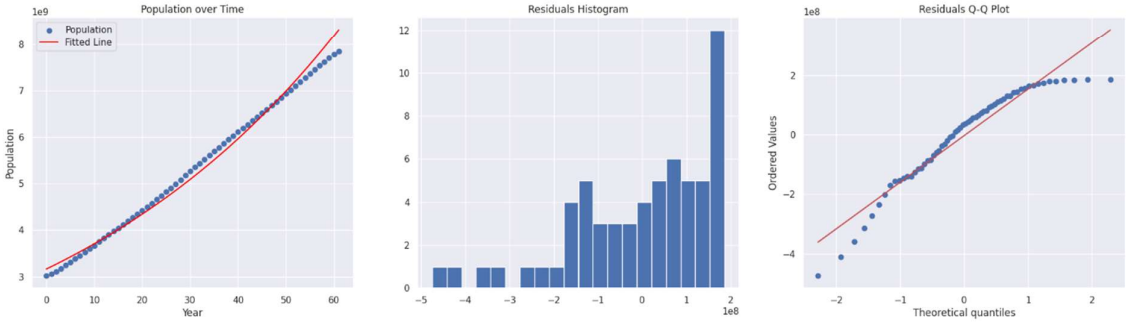
In [ ]:

```
# Fit linear regression model to yearly population
X = np.arange(len(population)).reshape(-1, 1)
y = population.values.reshape(-1, 1)
linear_pop_model = LinearRegression()
linear_pop_model.fit(X, y)

# Calculate residuals
residuals = y - linear_pop_model.predict(X)

# Plot fitted line, residuals histogram, and residuals Q-Q plot
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(24, 6))
ax1.scatter(X, y, label='Population')
ax1.plot(X, linear_pop_model.predict(X), label='Fitted Line', color='red')
ax1.set_xlabel('Year')
ax1.set_ylabel('Population')
ax1.set_title('Population over Time')
ax1.legend()
ax2.hist(residuals, bins=20)
ax2.set_title('Residuals Histogram')
stats.probplot(residuals.flatten(), plot=ax3)
ax3.set_title('Residuals Q-Q Plot')
plt.show()

# Calculate p-values for residuals
_, p_values = stats.normaltest(residuals.flatten())
print('p-values for residuals:', f'{p_values:.3f}')
```



Fit an Exponential Regression Model and Analyse the Results

n [ ]:
```
# Fit an exponential model to yearly population
X = np.arange(len(population)).reshape(-1, 1)
y = population.values.reshape(-1, 1)
exp_pop_model = LinearRegression()
exp_pop_model.fit(X, np.log(y))

# Calculate residuals
residuals = y - np.exp(exp_pop_model.predict(X))

# Plot fitted line, residuals histogram, and residuals Q-Q plot
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(24, 6))
ax1.scatter(X, y, label='Population')
ax1.plot(X, np.exp(exp_pop_model.predict(X)), label='Fitted Line', color='red')
ax1.set_xlabel('Year')
ax1.set_ylabel('Population')
ax1.set_title('Population over Time')
ax1.legend()
ax2.hist(residuals, bins=20)
ax2.set_title('Residuals Histogram')
stats.probplot(residuals.flatten(), plot=ax3)
ax3.set_title('Residuals Q-Q Plot')
plt.show()

# Calculate p-values for residuals
_, p_values = stats.normaltest(residuals.flatten())
print('p-values for residuals:', f'{p_values:.3f}')
```



Analysing food demand (tonnes per capita per year)

Assumptions:

1. Food supply is equal to food demand (tonnes per capita per year)
2. Food supply is equal to crop production (tonnes per capita per year)

In [ ]:
```
# Merge production and population dataframes
country_yearly_production = df.groupby(['Area', 'Year'])['Production'].sum().reset_index()
merge = pd.merge(country_yearly_production, demographics, left_on=['Area', 'Year'], right_on=['Country Name', 'Year'], how='left')
merge.head()
```

Out[ ]:

	Area	Year	Production	Country Name	Country Code	Population
0	Afghanistan	1961	1.119817e+07	Afghanistan	AFG	8790140.0
1	Afghanistan	1962	1.124859e+07	Afghanistan	AFG	8969047.0
2	Afghanistan	1963	1.034671e+07	Afghanistan	AFG	9157465.0
3	Afghanistan	1964	1.105875e+07	Afghanistan	AFG	9355514.0
4	Afghanistan	1965	1.105486e+07	Afghanistan	AFG	9565147.0

In [ ]:
```
merge['Production per capita (tonnes/per capita)'] = merge['Production'] / merge['Population']
merge.head()
```

Out[ ]:

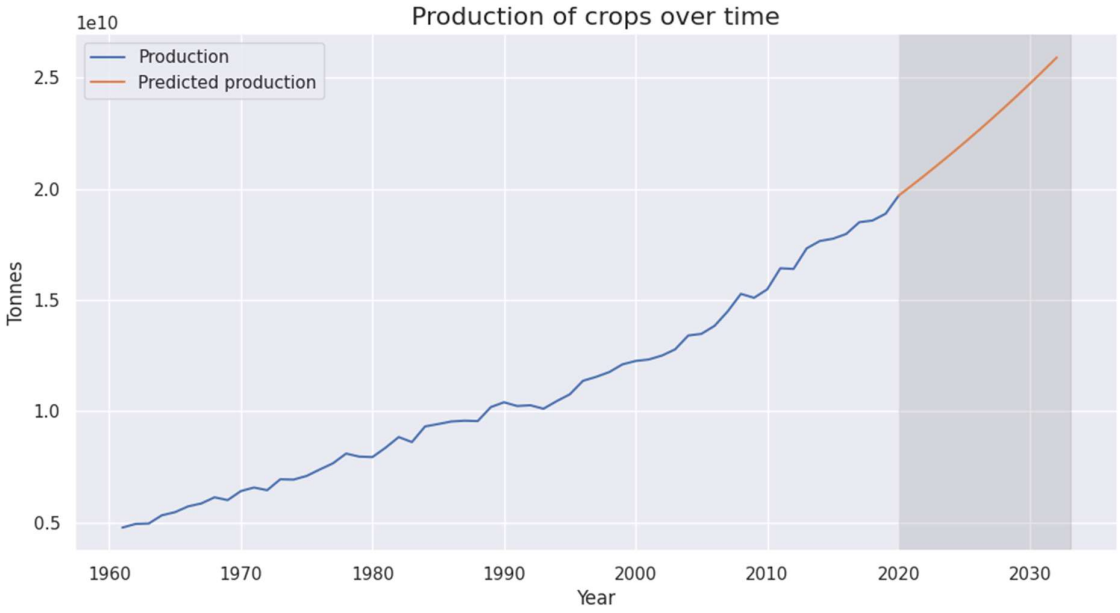
	Area	Year	Production	Country Name	Country Code	Population	Production per capita (tonnes/per capita)
0	Afghanistan	1961	1.119817e+07	Afghanistan	AFG	8790140.0	1.273946
1	Afghanistan	1962	1.124859e+07	Afghanistan	AFG	8969047.0	1.254157

In [ ]:
```
total_production_2019 = df[df['Year'] == 2019]['Production'].sum()
total_population_2019 = demographics[demographics['Year'] == 2019]['Population'].sum()
total_production_per_capita_2019 = total_production_2019 / total_population_2019
print('Total production in 2019:', f'{total_production_2019:.0f}')
print('Total population in 2019:', f'{total_population_2019:.0f}')
print('Production per capita in 2019:', f'{total_production_per_capita_2019:.3f} tonnes/per capita')
```

```
Total production in 2019: 18878841240
Total population in 2019: 7703647794
Production per capita in 2019: 2.451 tonnes/per capita
```

Projecting Crop Production in 2033

In [ ]:
```
# Use our regression model to predict production per capita in 2033
range_ = np.arange(2020, 2033).reshape(-1, 1)
pred = np.exp(exp_prod_model.predict(range_ - 2020 + len(yearly_production))).flatten()
plt.figure(figsize=(12, 6))
plt.plot(np.append(yearly_production.index.values, 2020), np.append(yearly_production.values, pred[0]), label='Production')
plt.plot(range_, pred, label='Predicted production')
plt.title('Production of crops over time', fontsize=16)
plt.xlabel('Year')
plt.ylabel('Tonnes')
plt.legend()
plt.axvspan(2020, 2033, alpha=0.2, color='grey')
plt.show()
```



In [ ]:
```
pred[-1]
```

Out[ ]:
```
25906257501.48927
```

In [ ]:
```
yield_forecast_2033
```

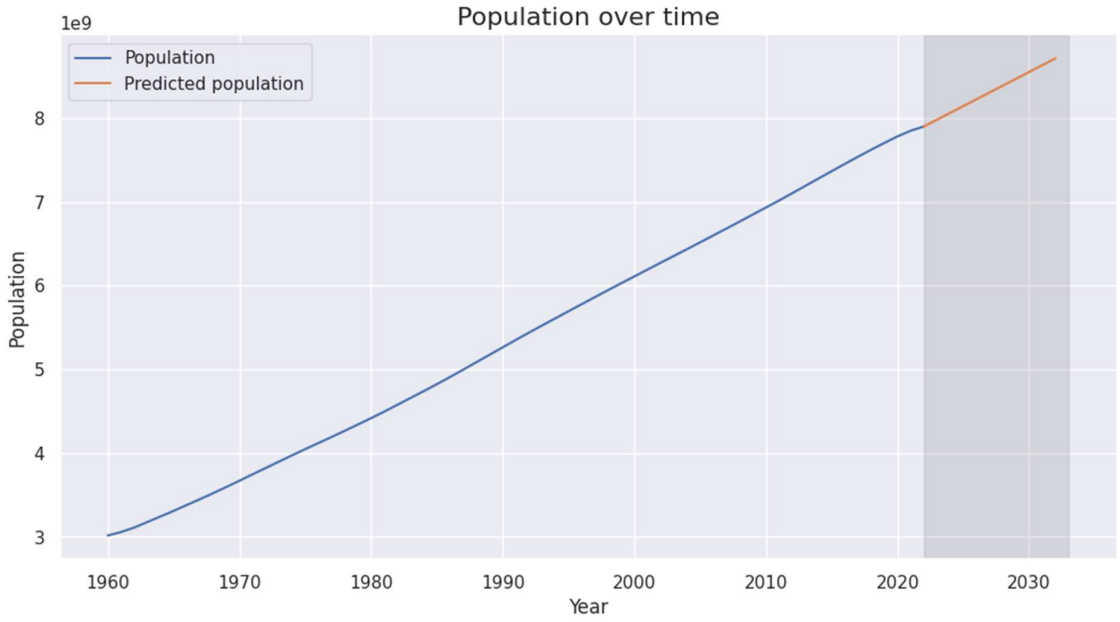
Out[ ]:
```
array([6.57745223])
```

In [ ]:
```
area_hasvasted_2033 = pred[-1] / yield_forecast_2033
print('Area that has to be harvested in 2033:', f'{int(area_hasvasted_2033):.0f} hectares')
```

```
Area that has to be harvested in 2033: 3938646238 hectares
```

Project Population in 2033

In [ ]:
```
# Use our regression model to predict population in 2033
range_ = np.arange(2022, 2033).reshape(-1, 1)
# pred = np.exp(exp_pop_model.predict(range_ - 2020 + len(population))).flatten()
pred = linear_pop_model.predict(range_ - 2022 + len(population)).flatten()
plt.figure(figsize=(12, 6))
plt.plot(np.append(population.index.values, 2022), np.append(population.values, pred[0]), label='Population')
plt.plot(range_, pred, label='Predicted population')
plt.title('Population over time', fontsize=16)
plt.xlabel('Year')
plt.ylabel('Population')
plt.legend()
plt.axvspan(2022, 2033, alpha=0.2, color='grey')
plt.show()
```

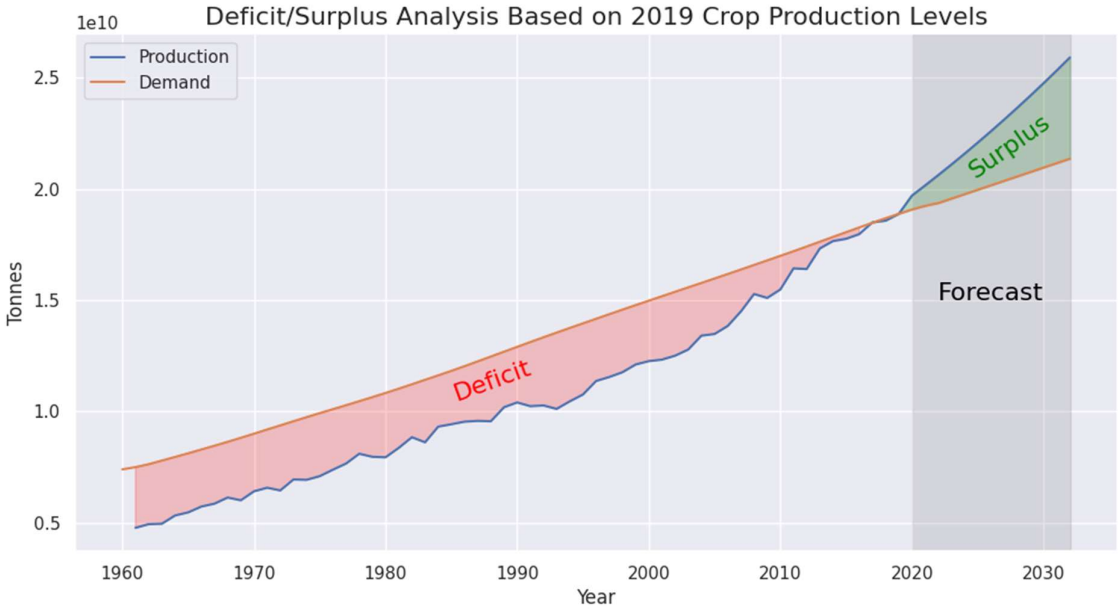


Project Food Demand in 2033

In [ ]:
```
print(f'Assuming per capita crop demand of 2019: {total_production_per_capita_2019:.2f} tonnes per year')
```

```
Assuming per capita crop demand of 2019: 2.45 tonnes per year
```

In [ ]:
```
# Demand vs production in 2033
range_prod = np.arange(yearly_production.index.values[0], 2033).reshape(-1, 1)
range_pop = np.arange(population.index.values[0], 2033).reshape(-1, 1)
demand = population.values * total_production_per_capita_2019
pred_prod = np.exp(exp_prod_model.predict(range_prod[len(yearly_production):] - yearly_production.index.values[0])).flatten()
pred_pop = linear_pop_model.predict(range_pop[len(population):] - population.index.values[0]).flatten()
pred_demand = pred_pop * total_production_per_capita_2019
demand = np.append(demand, pred_demand)
prod = np.append(yearly_production.values, pred_prod)
assert len(demand.flatten()) == len(range_pop.flatten())
assert len(prod.flatten()) == len(range_prod.flatten())
print(len(demand.flatten()), len(range_pop.flatten()))
plt.figure(figsize=(12, 6))
plt.plot(range_prod, prod, label='Production')
plt.plot(range_pop, demand, label='Demand')
plt.fill_between(range_pop.flatten()[1:], demand[1:], prod, where=prod > demand[1:]*.99, alpha=0.2, color='green')
plt.fill_between(range_pop.flatten()[1:], demand[1:], prod, where=prod < demand[1:], alpha=0.2, color='red')
plt.text(2022, 1.5e10, 'Forecast', fontsize=16, color='black')
plt.text(1985, 1.05e10, 'Deficit', fontsize=16, color='red', rotation=20)
plt.text(2024, 2.05e10, 'Surplus', fontsize=16, color='green', rotation=35)
plt.title('Deficit/Surplus Analysis Based on 2019 Crop Production Levels', fontsize=16)
plt.xlabel('Year')
plt.ylabel('Tonnes')
plt.legend()
plt.axvspan(2020, 2032, alpha=0.2, color='grey')
plt.show()
```



Regression Model for each country

In [ ]:
```
df
```

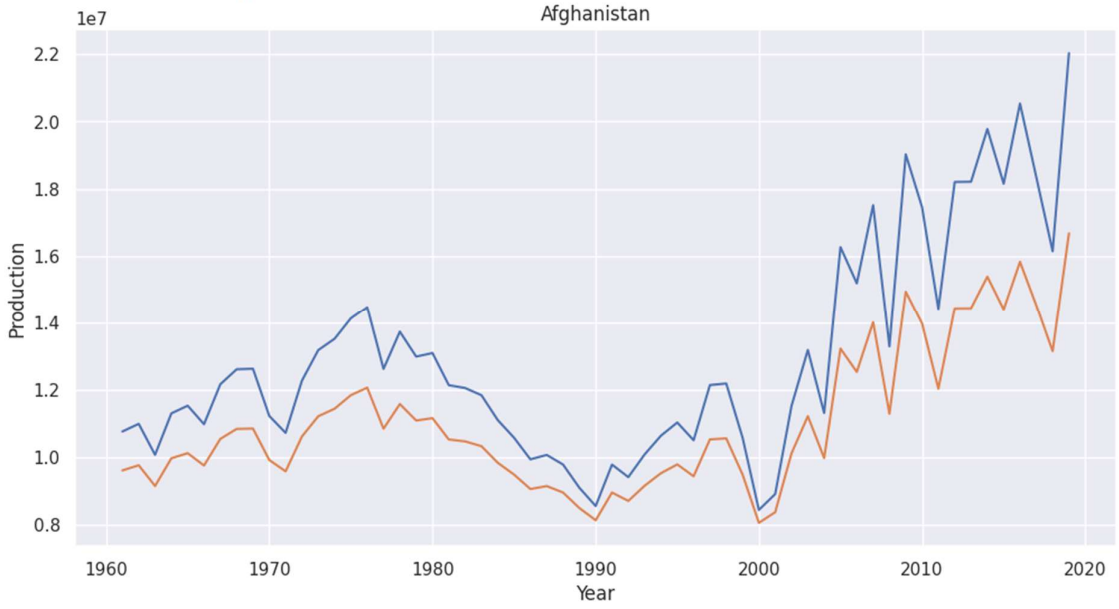
Out[ ]:

	Area Code	Area	Item Code	Item	Element Code	Element	Unit	Year	Values	Flags	Area2	Flag Description
0	10	Australia	221	Almonds, with shell	5312	Area harvested	ha	1961	NaN	M	Australia	Data not available
1	10	Australia	221	Almonds, with shell	5419	Yield	hg/ha	1961	NaN	NaN	Australia	Not applicable
2	10	Australia	221	Almonds, with shell	5510	Production	tonnes	1961	NaN	M	Australia	Data not available
3	10	Australia	711	Anise, badian, fennel, coriander	5312	Area harvested	ha	1961	NaN	M	Australia	Data not available
4	10	Australia	711	Anise, badian, fennel, coriander	5419	Yield	hg/ha	1961	NaN	NaN	Australia	Not applicable
...	...	...	...	...	...	...	...	...	...	...	...	...

In [ ]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

seq_length = 12
n_features = 1

def split_sequences(sequences, n_steps):
    X, y = list(), list()
    for i in range(len(sequences)):
        # Find the end of this pattern
        end_ix = i + n_steps
        # Check if we are beyond the dataset
        if end_ix > len(sequences):
            break
        # Gather input and output parts of the pattern
        seq_x, seq_y = sequences[i:end_ix], sequences[end_ix-1]
        X.append(seq_x)
        y.append(seq_y)
    return np.array(X).reshape(-1, n_steps, n_features), np.array(y)

def lstm_model():
    # Define lstm model
    model = keras.Sequential()
    # model.add(layers.LSTM(512, activation='sigmoid', input_shape=(seq_length, n_features), return_sequences=True))
    # model.add(layers.LSTM(256, activation='sigmoid', input_shape=(seq_length, n_features), return_sequences=True))
    model.add(layers.LSTM(8, activation='sigmoid', input_shape=(seq_length, n_features)))
    model.add(layers.Dense(1, activation='linear'))
    optimizer = tf.keras.optimizers.Adam(learning_rate=0.01)
    model.compile(optimizer, loss='mse')
    return model

# # Split data into train and test sets
# train_size = int(len(df) * 0.8)
# test_size = len(df) - train_size
# train, test = df.iloc[0:train_size], df.iloc[train_size:len(df)]
```

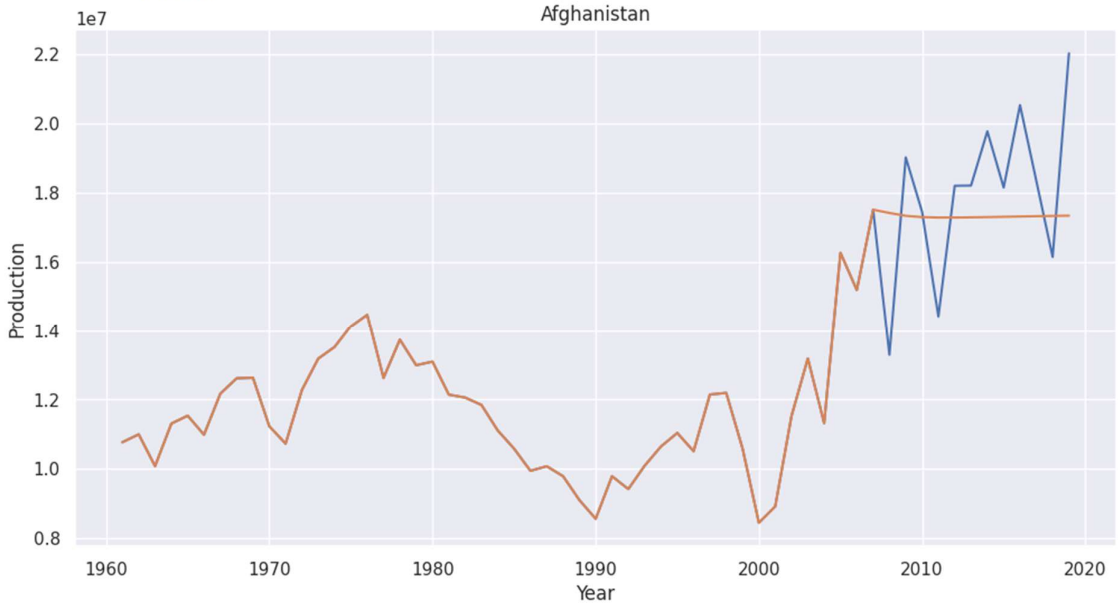
In [ ]:
```
groups = df.groupby(['Area', 'Year'])['Production'].sum()
for country in groups.index.levels[0]:
    model = lstm_model()
    X = groups[country].index.values.reshape(-1, 1)
    values = groups[country].values
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled = scaler.fit_transform(values.reshape(-1, 1))
    x, y = split_sequences(scaled, seq_length)
    model.fit(x, y, epochs=200)
    pred = model.predict(scaled)
    rev_pred = scaler.inverse_transform(pred)
    plt.figure(figsize=(12, 6))
    plt.plot(groups[country].index.values, groups[country].values)
    plt.plot(X, rev_pred.squeeze())
    plt.title(country)
    plt.xlabel('Year')
    plt.ylabel('Production')
    plt.show()
    break
```

```
Epoch 1/200
2/2 [==============================] - 1s 9ms/step - loss: 0.3777
Epoch 2/200
2/2 [==============================] - 0s 9ms/step - loss: 0.2164
Epoch 3/200
2/2 [==============================] - 0s 9ms/step - loss: 0.1094
Epoch 4/200
2/2 [==============================] - 0s 9ms/step - loss: 0.0563
Epoch 5/200
```



In [ ]:
```
groups = df.groupby(['Area', 'Year'])['Production'].sum()
for country in groups.index.levels[0]:
    model = lstm_model()
    test_size = 12
    X = groups[country].index.values.reshape(-1, 1)[:-test_size]
    values = groups[country].values[:-test_size]
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled = scaler.fit_transform(values.reshape(-1, 1))
    x, y = split_sequences(scaled, seq_length)
    model.fit(x, y, epochs=500)
    for i in range(test_size):
        pred = model.predict(scaled[-seq_length:].reshape(1, seq_length, n_features))
        scaled = np.append(scaled, pred)
    # Expand X index (years)
    X_pred = np.append(X, np.arange(X[-1]+1, X[-1]+test_size+1))
    rev_pred = scaler.inverse_transform(scaled.reshape(-1, 1))
    plt.figure(figsize=(12, 6))
    plt.plot(groups[country].index.values, groups[country].values)
    plt.plot(X_pred, rev_pred)
    plt.title(country)
    plt.xlabel('Year')
    plt.ylabel('Production')
    plt.show()
    break
```

```
Epoch 1/500
2/2 [==============================] - 1s 14ms/step - loss: 0.0559
Epoch 2/500
2/2 [==============================] - 0s 10ms/step - loss: 0.0509
Epoch 3/500
2/2 [==============================] - 0s 11ms/step - loss: 0.0613
```

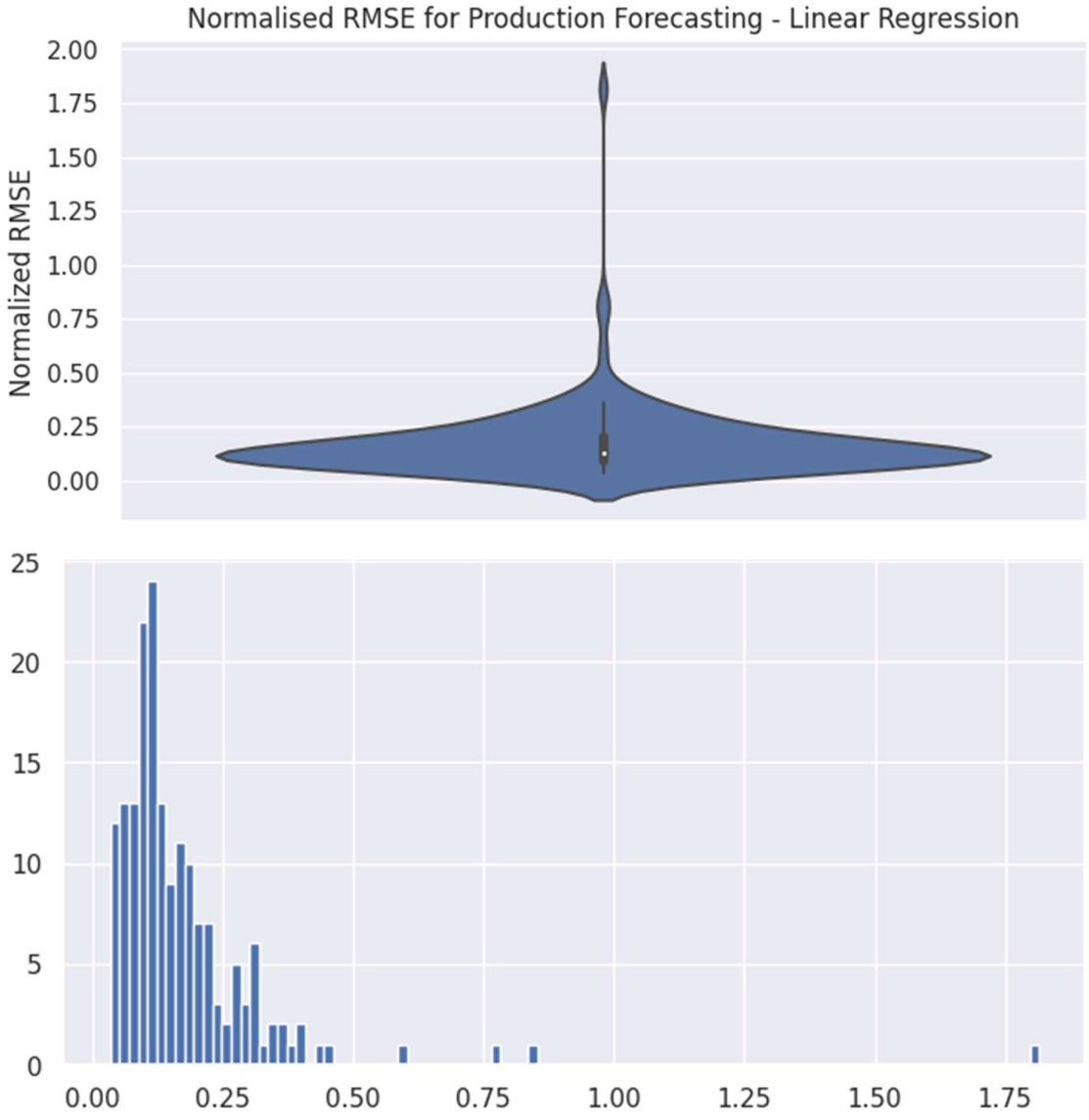


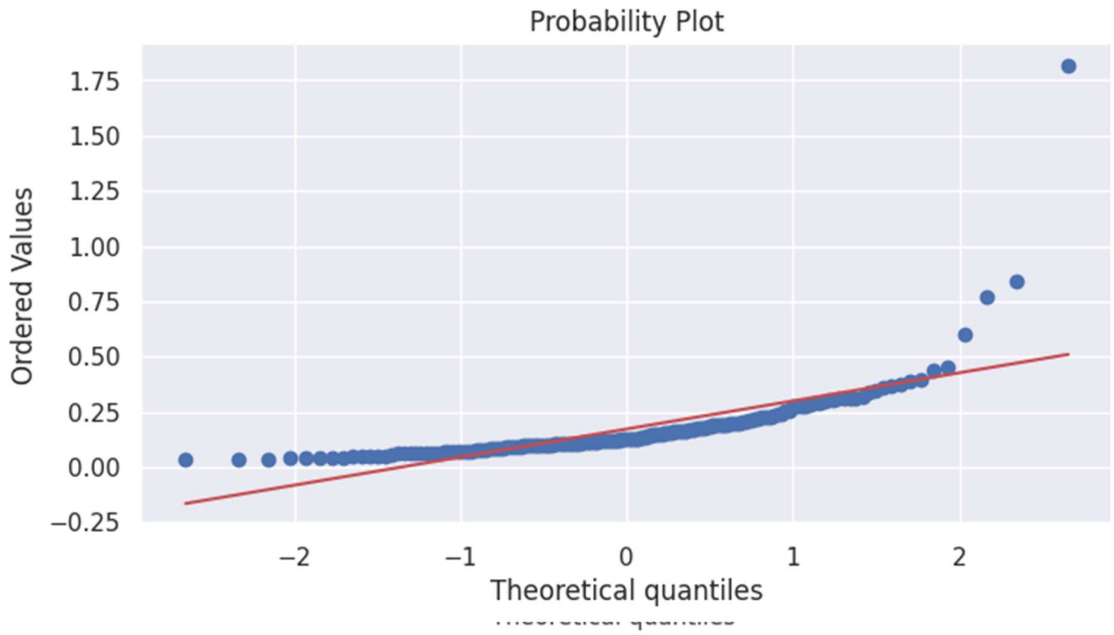
In [ ]:
```
groups = df.groupby(['Area', 'Year'])['Production'].sum()
for country in groups.index.levels[0]:
    model = LinearRegression()
    X = groups[country].index.values.reshape(-1, 1)
    y = np.log(groups[country].values)
    # train_index = np.random.choice(len(X), int(len(X) * 0.8), replace=False)
    # X_train, y_train = X[train_index], y[train_index]
    model.fit(X, y)
    plt.figure(figsize=(12, 6))
    plt.plot(groups[country].index.values, groups[country].values)
    plt.plot(X, np.exp(model.predict(X)))
    plt.title(country)
    plt.xlabel('Year')
    plt.ylabel('Production')
    plt.show()
```

In [ ]:
```
def fit_models(groups, model_type='linear', n_folds=5):
    errors = {}
    models = {}
    for country in groups.index.levels[0]:
        model = LinearRegression()
        X = groups[country].index.values.reshape(-1, 1)
        mean = groups[country].values.mean()
        values = groups[country].values
        if model_type == 'exponential':
            values = np.log(values)
        tscv = TimeSeriesSplit(n_splits=n_folds)
        error = 0
        if len(x) > n_folds:
            for fold_index, (train_index, test_index) in enumerate(tscv.split(x)):
                x_train, x_test = X[train_index], X[test_index]
                y_train, y_test = values[train_index], values[test_index]
                model = LinearRegression()
                model.fit(x_train.reshape(-1, 1), y_train)
                pred = model.predict(x_test.reshape(-1, 1))
                if model_type == 'exponential':
                    pred = np.exp(pred)
                    error += np.sqrt(np.mean((pred - np.exp(y_test)) ** 2))
                else:
                    error += np.sqrt(np.mean((pred - y_test) ** 2))
            error /= n_folds
            model = LinearRegression()
            model.fit(X, values)
            errors[country] = {'error': error, 'mean': mean}
            models[country] = model
    error_df = pd.DataFrame.from_dict(errors, orient='index').reset_index()
    error_df['normalized_error'] = error_df['error'] / error_df['mean']
    return error_df, models
```

In [ ]:
```
def plot_errors(error_df, title):
    # Plot the violin plot
    sns.violinplot(error_df, y="normalized_error")

    # Set labels and title
    plt.xlabel("")
    plt.ylabel("Normalized RMSE")
    plt.title(title)

    # Show the plot
    plt.show()
    error_df['normalized_error'].hist(bins=100)
    plt.show()
    stats.probplot(error_df['normalized_error'], dist="norm", plot=pylab)
    pylab.show()
    # Shapiro-Wilk Test
    print(stats.shapiro(error_df['normalized_error']))
```

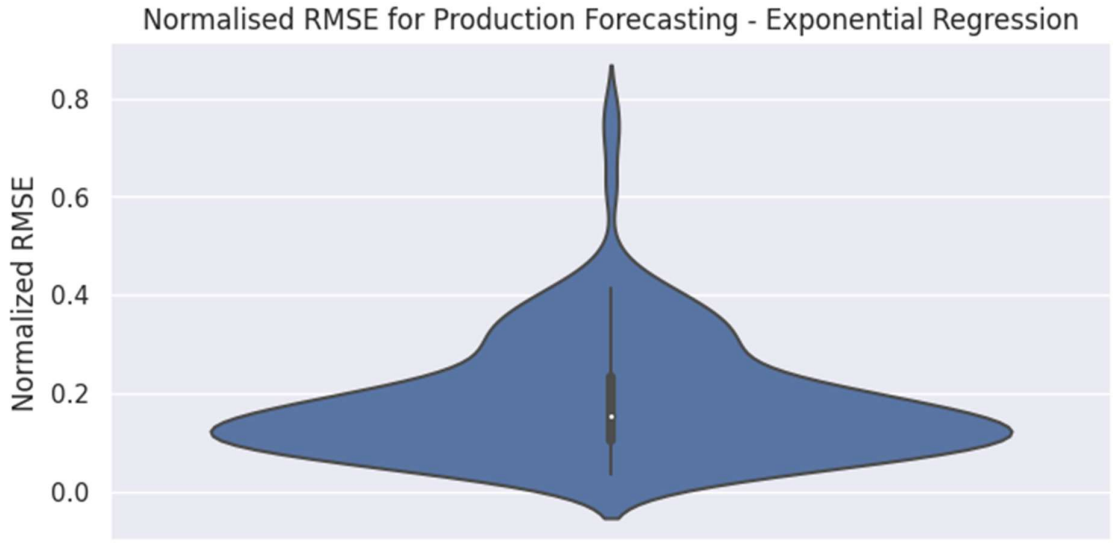
In [ ]:
```
# Linear Regression for Production Forecasting
groups = df.groupby(['Area', 'Year'])['Production'].sum()
error_df_prod_linear, production_linear_models = fit_models(groups, model_type='linear')
plot_errors(error_df_prod_linear, 'Normalised RMSE for Production Forecasting - Linear Regression')
```

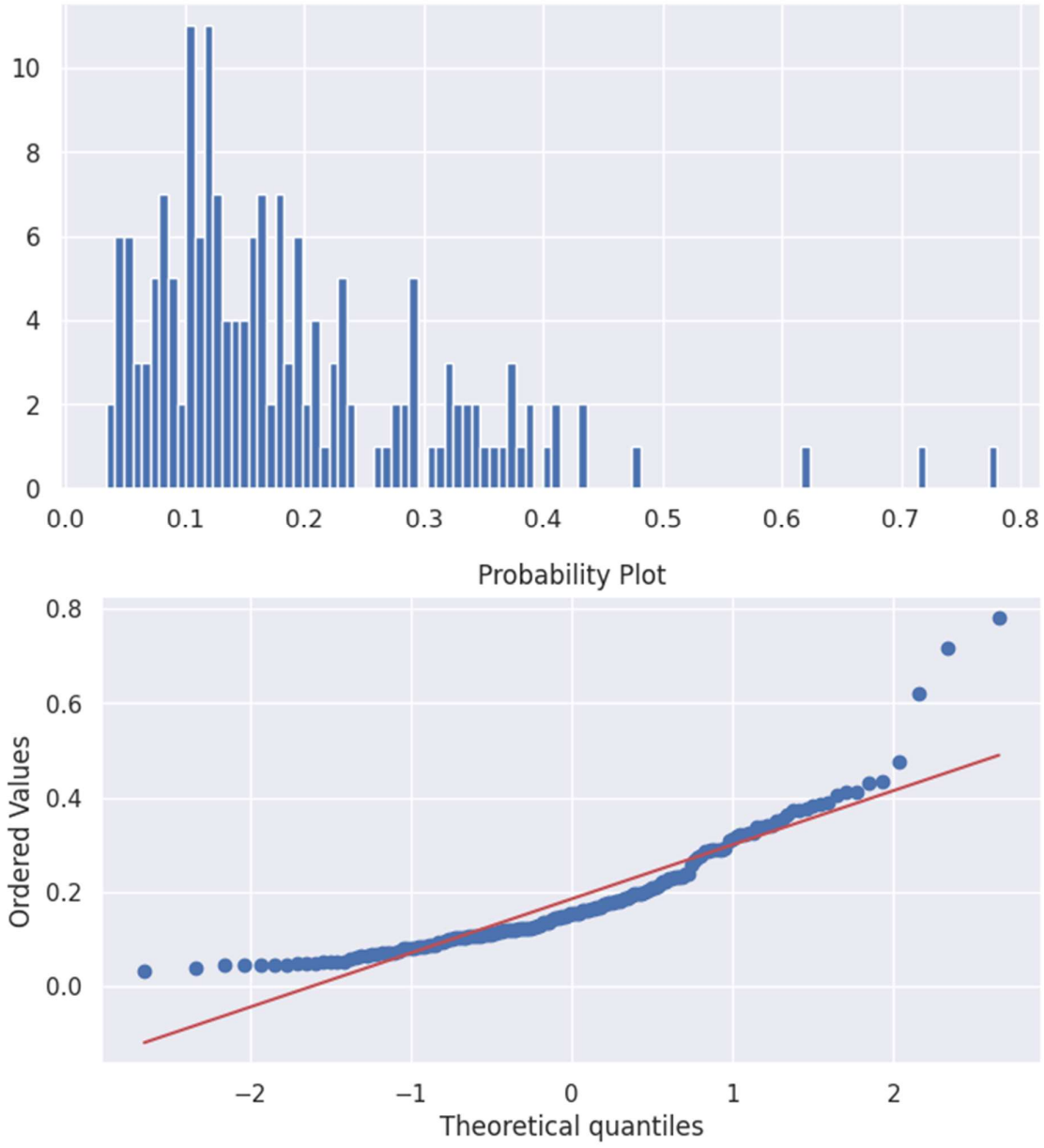





```
ShapiroResult(statistic=0.5486217737197876, pvalue=4.749914223143136e-21)
```

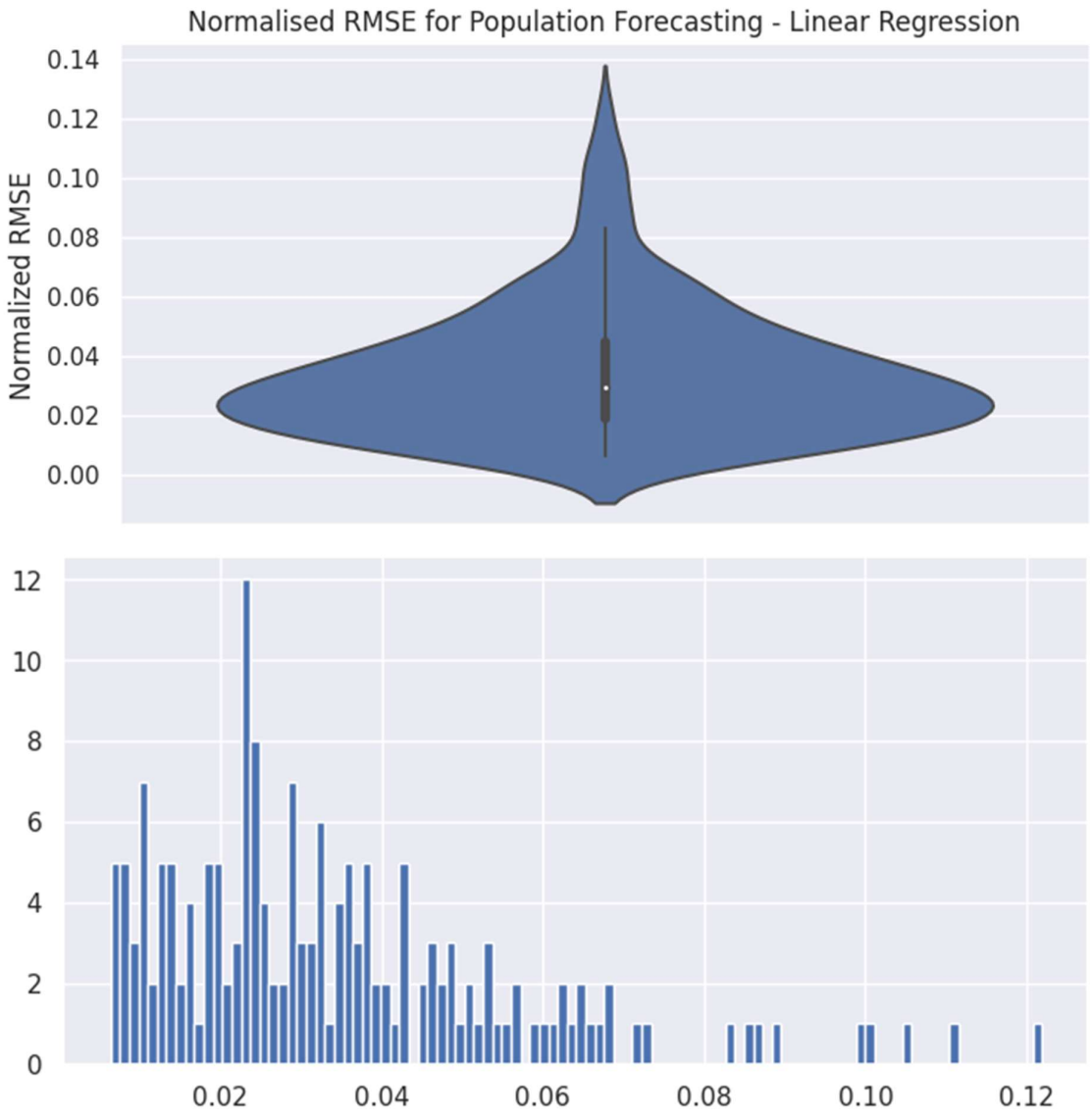
n [ ]:
```
# Exponential Regression for Production Forecasting
groups = df.groupby(['Area', 'Year'])['Production'].sum()
error_df_prod_exp, production_exp_models = fit_models(groups, model_type='exponential')
plot_errors(error_df_prod_exp, 'Normalised RMSE for Production Forecasting - Exponential Regression')
```

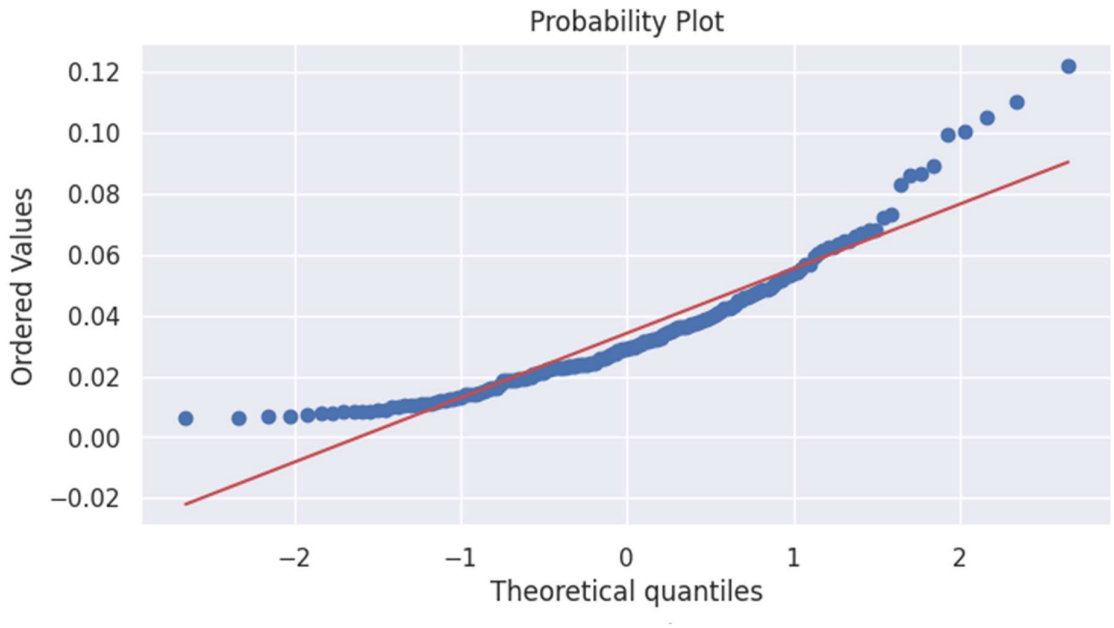





```
ShapiroResult(statistic=0.8574124574661255, pvalue=9.808023317126047e-12)
```

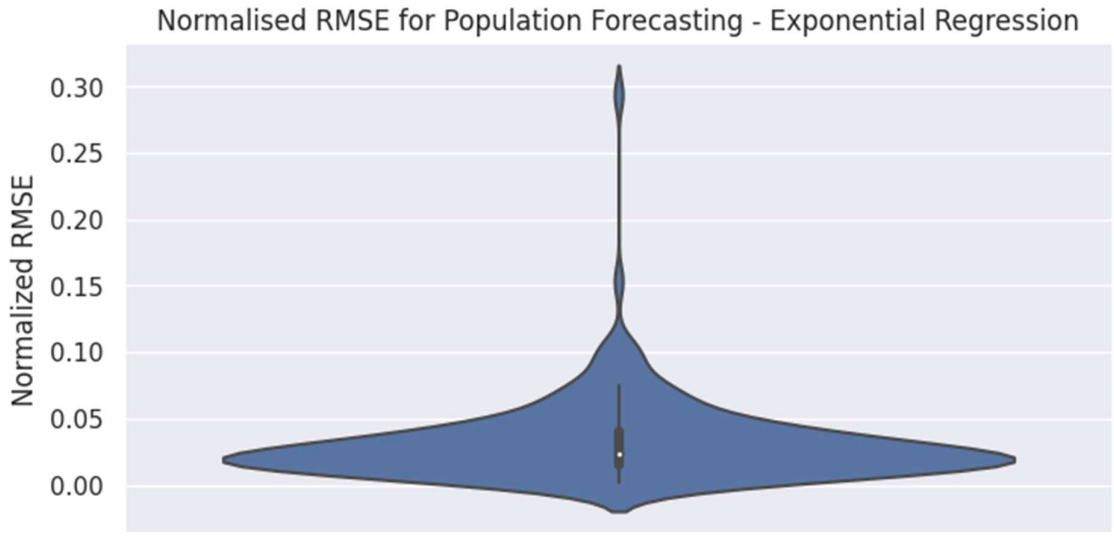
In [ ]:
```
# Linear Model Population
groups = demographics.groupby(['Country Name', 'Year'])['Population'].sum()
error_df_pop_linear, population_linear_models = fit_models(groups, model_type='linear')
plot_errors(error_df_pop_linear, 'Normalised RMSE for Population Forecasting - Linear Regression')
```

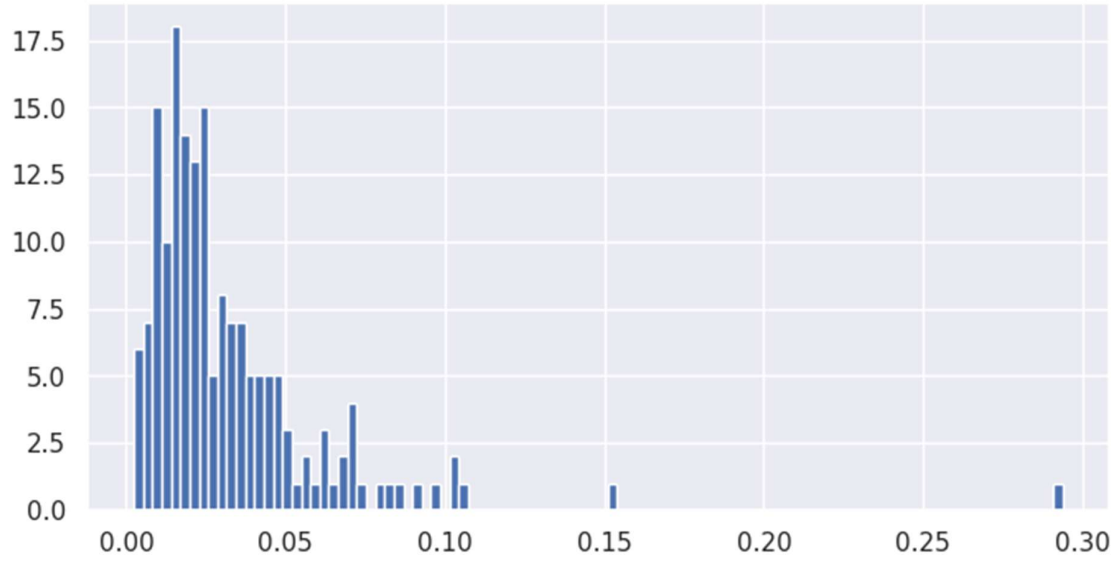

Normalised RMSE for Population Forecasting - Linear Regression
Normalized RMSE
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
12
10
8
6
4
2
0
0.02
0.04
0.06
0.08
0.10
0.12

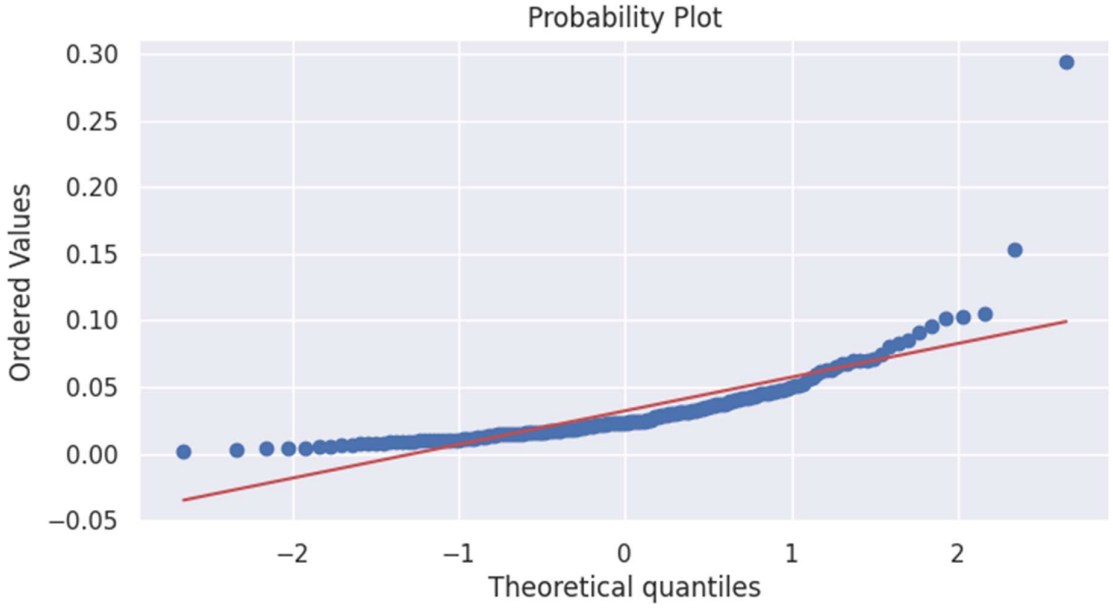


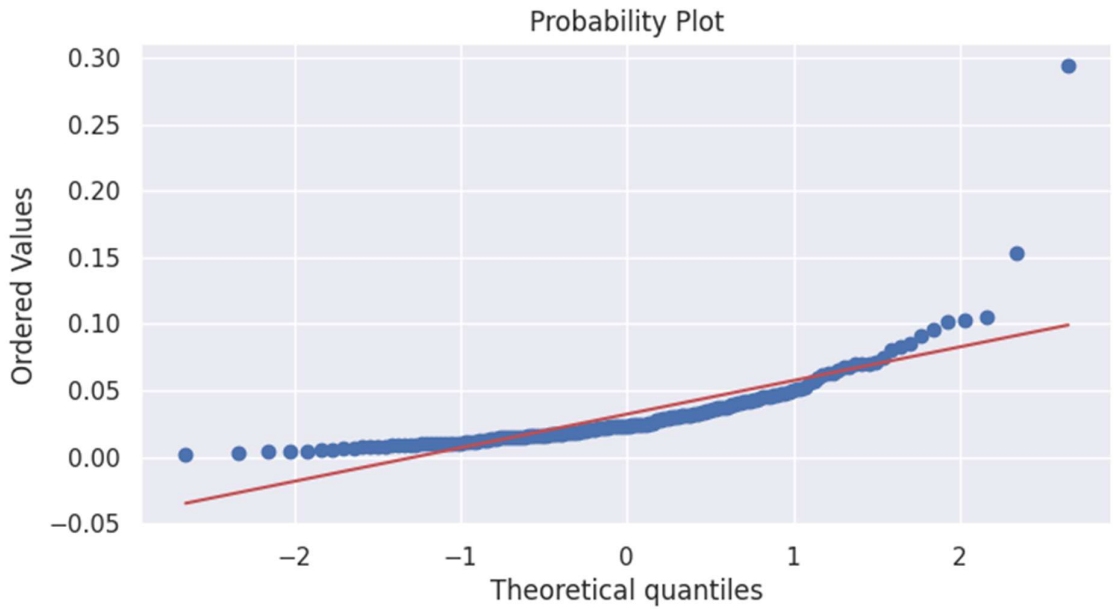

```
ShapiroResult(statistic=0.8902348875999451, pvalue=5.37045796722424e-10)
```

In [ ]:
```
# Exponential Model Population
groups = demographics.groupby(['Country Name', 'Year'])['Population'].sum()
error_df_pop_exp, population_exp_models = fit_models(groups, model_type='exponential')
plot_errors(error_df_pop_exp, 'Normalised RMSE for Population Forecasting - Exponential Regression')
```




17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.00
0.05
0.10
0.15
0.20
0.25
0.30


Probability Plot
Ordered Values
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−2
−1
0
1
2
Theoretical quantiles



ShapiroResult(statistic=0.6638813614845276, pvalue=2.487397862805652e-18)

In [ ]:
```
# Production Forecasting with exponential model
production_forecast = df.pivot_table(index='Area', columns='Year', values='Production', aggfunc='sum')
production_forecast = production_forecast.reset_index()
production_forecast[list(range(2020, 2034))] = production_forecast.apply(lambda x: np.exp(production_exp_models[x['Area']].predict(np.arange(2020, 2034).reshape(-1, 1))), axis=1, re
production_forecast
```

Out[ ]:

Year	Area	1961	1962	1963	1964	1965	1966	1967	1968	1969	1970	1971	1972	
0	Afghanistan	1.119817e+07	1.124859e+07	1.034671e+07	1.105875e+07	1.105486e+07	1.103697e+07	1.239107e+07	1.226512e+07	1.257005e+07	1.156206e+07	1.067025e+07	1.213766e+07	1.301
1	Albania	1.334800e+06	1.353821e+06	1.368386e+06	1.721743e+06	1.553674e+06	1.976284e+06	2.059793e+06	2.177740e+06	2.173712e+06	2.292794e+06	2.256939e+06	2.418420e+06	2.638
2	Algeria	8.062526e+06	1.041639e+07	1.109267e+07	8.660816e+06	9.728327e+06	5.776305e+06	7.641533e+06	1.009690e+07	9.117113e+06	1.029858e+07	9.580494e+06	9.617651e+06	8.399
3	Angola	6.190278e+06	6.009244e+06	6.537206e+06	6.918807e+06	7.315995e+06	6.939868e+06	7.599824e+06	7.211057e+06	7.266417e+06	7.971801e+06	8.424005e+06	8.207479e+06	8.753
4	Antigua and Barbuda	3.773236e+05	4.036513e+05	5.183406e+05	5.362184e+05	2.750808e+05	1.678233e+05	1.059281e+05	3.173634e+04	8.091994e+04	9.584265e+04	2.777607e+05	1.979080e+04	1.548

In [ ]:
```
# Population Forecasting with linear model
population_forecast = demographics.pivot_table(index='Country Name', columns='Year', values='Population', aggfunc='sum')
population_forecast = population_forecast.reset_index()
population_forecast[list(range(2022, 2034))] = population_forecast.apply(lambda x: population_linear_models[x['Country Name']].predict(np.aran
population_forecast
```

Out[ ]:

Year	Country Name	1960	1961	1962	1963	1964	1965	1966	1967	1968	1969	1970	
0	Afghanistan	8622466.0	8790140.0	8969047.0	9157465.0	9355514.0	9565147.0	9783147.0	10010030.0	10247780.0	10494489.0	10752971.0	1101
1	Albania	1608800.0	1659800.0	1711319.0	1762621.0	1814135.0	1864791.0	1914573.0	1965598.0	2022272.0	2081695.0	2135479.0	218
2	Algeria	11394307.0	11598608.0	11778260.0	11969451.0	12179099.0	12381256.0	12613389.0	12897115.0	13190975.0	13491016.0	13795915.0	1411
3	Angola	5357195.0	5441333.0	5521400.0	5599827.0	5673199.0	5736582.0	5787044.0	5827503.0	5868203.0	5928386.0	6029700.0	617
4	Antigua and Barbuda	55342.0	56245.0	57008.0	57778.0	58664.0	59644.0	60615.0	61617.0	62658.0	63742.0	64517.0	6

In [ ]:
```
years = production_forecast.columns[1:]
production_forecast = production_forecast.melt(id_vars=['Area'], value_vars=years, var_name='Year', value_name='Production')
population_forecast = population_forecast.melt(id_vars=['Country Name'], value_vars=years, var_name='Year', value_name='Population')
```

In [ ]:
```
production_per_capita_2019 = (df[df['Year'] == 2019].groupby('Area')['Production'].sum() / demographics[demographics['Year'] == 2019].groupby('Country Name')['Population'].sum()).mean()
production_per_capita_2019
```

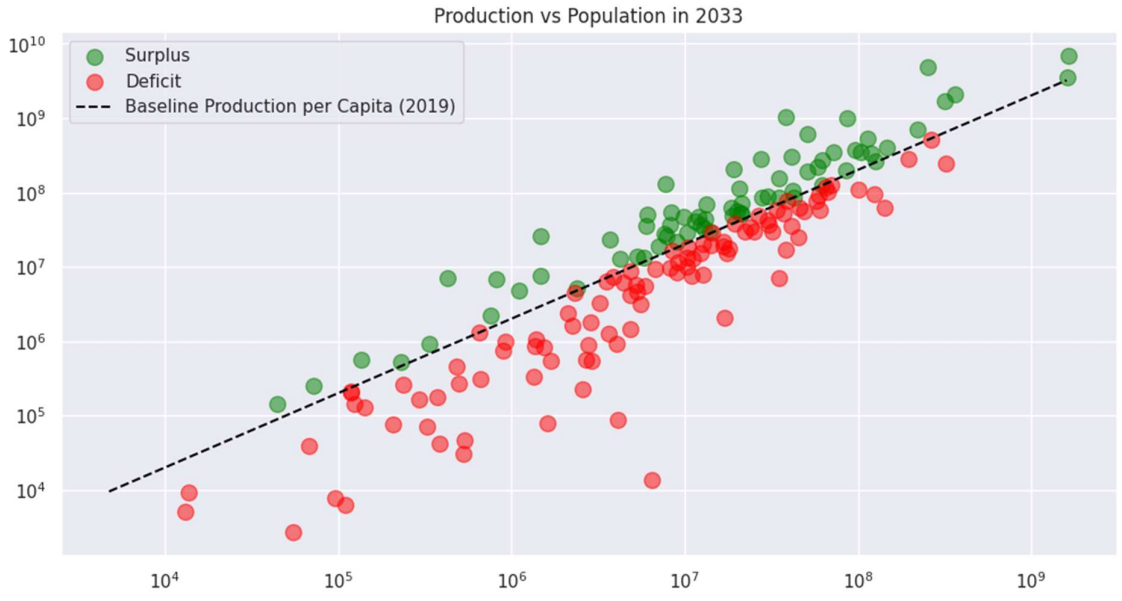
Out[ ]:
```
2.016163493808608
```

In [ ]:
```
population = np.arange(forecast['Population'].min(), forecast['Population'].max(), 100000000)
production_threshold = population * production_per_capita_2019
```

In [ ]:
```
population.shape
```

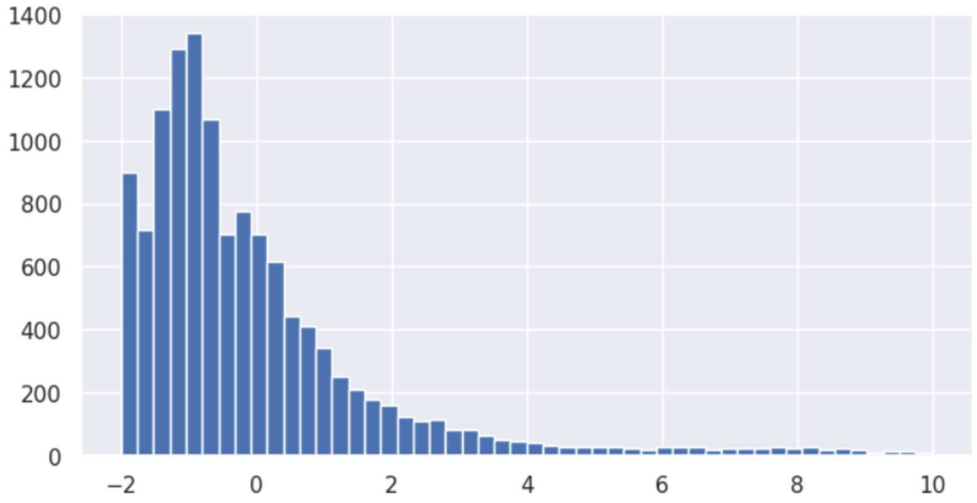
Out[ ]:
```
(17,)
```

n [ ]:
```
annotate = False
fig, ax = plt.subplots(figsize=(12, 6))

forecast = population_forecast.merge(production_forecast, left_on=['Country Name', 'Year'], right_on=['Area', 'Year'], how='inner').drop('Area', ax
forecast['Production per capita'] = forecast['Production'] / forecast['Population']
forecast['Category'] = forecast.apply(lambda x: 'Surplus' if x['Production'] > x['Population'] * production_per_capita_2019 else 'Deficit', axis=1)
forecast = forecast.rename(columns={'Country Name': 'Country'})

# Split data into two categories
surplus = forecast[(forecast['Year'] == 2033) & (forecast['Category'] == 'Surplus')]
deficit = forecast[(forecast['Year'] == 2033) & (forecast['Category'] == 'Deficit')]

# Plot each category with a different color and label
ax.scatter(surplus['Population'], surplus['Production'], color='green', alpha=0.5, s=100, label='Surplus')
ax.scatter(deficit['Population'], deficit['Production'], color='red', alpha=0.5, s=100, label='Deficit')

# Set x and y axis to logarithmic scale
ax.set_xscale('log')
ax.set_yscale('log')

plt.plot(population, production_threshold, color='black', label='Baseline Production per Capita (2019)', linestyle='--')

forecast['Disparity'] = forecast['Production'] - forecast['Population'] * production_per_capita_2019
forecast['Disparity per capita'] = forecast['Disparity'] / forecast['Population']
bottom_disparity = forecast[forecast['Year']==2033].sort_values('Disparity per capita', ascending=True).head(20)
top_disparity = forecast[forecast['Year']==2033].sort_values('Disparity per capita', ascending=False).head(20)

# Annotate the top and bottom 20 countries
if annotate:
    for i, row in top_disparity.iterrows():
        ax.annotate(row['Country'], (row['Population'], row['Production']), xytext=(row['Population'] - 0.5, row['Production'] + 0.5), fontsize=6)

    for i, row in bottom_disparity.iterrows():
        ax.annotate(row['Country'], (row['Population'], row['Production']), xytext=(row['Population'] - 0.5, row['Production'] + 0.5), fontsize=6)


# Now, when you call plt.legend(), it will create a legend with the labels you specified in the scatter plot calls.
plt.legend()

plt.title('Production vs Population in 2033')
```

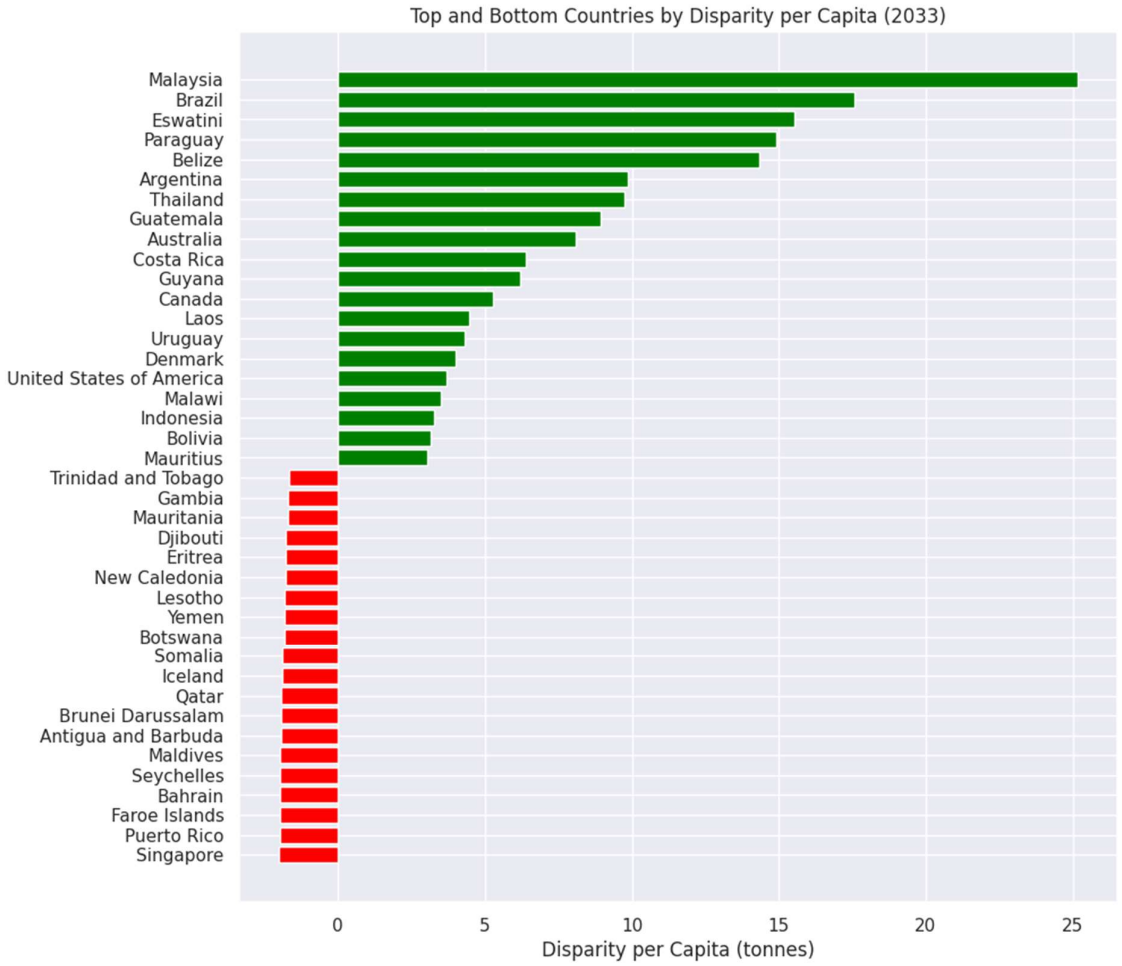


In [ ]:
```
forecast['Disparity per capita'].hist(bins=50, range=(-2, 10))
```

Out[ ]: <Axes: >

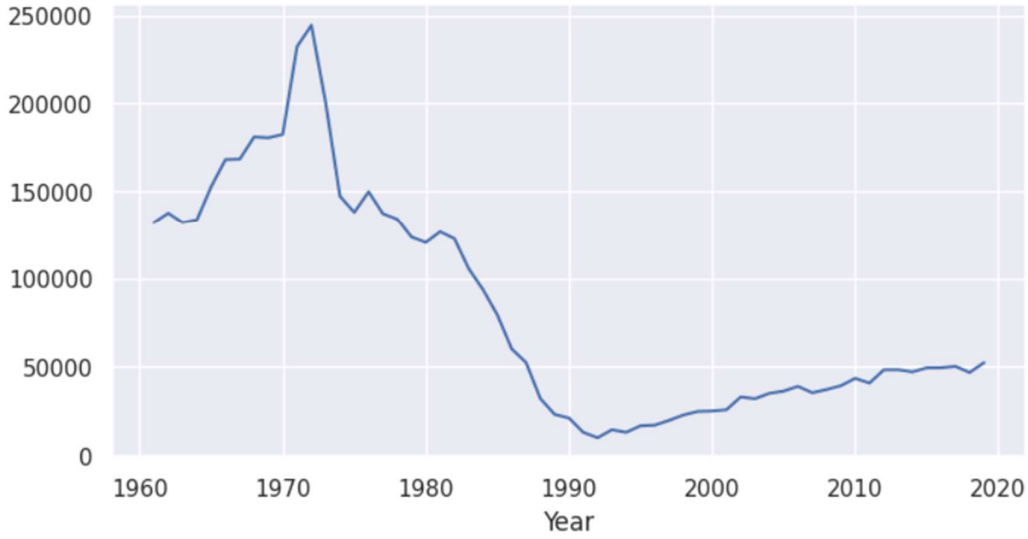


]:
```
fig, ax = plt.subplots(figsize=(10, 10))

# Combine the top and bottom disparities
remainder = forecast[~forecast['Country'].isin(top_disparity['Country']) & ~forecast['Country'].isin(bottom_disparity['Country'])]
remainder = remainder[remainder['Year'] == 2033]
combined = pd.concat([top_disparity, bottom_disparity])#, remainder])

# Sort the values to ensure the bars diverge
combined = combined.sort_values('Disparity per capita', ascending=True)

# Create a diverging bar plot
ax.barh(combined['Country'], combined['Disparity per capita'], color=['red' if x < 0 else 'green' for x in combined['Disparity per capita']])
ax.set_title('Top and Bottom Countries by Disparity per Capita (2033)')
ax.set_xlabel('Disparity per Capita (tonnes)')

plt.show()
```



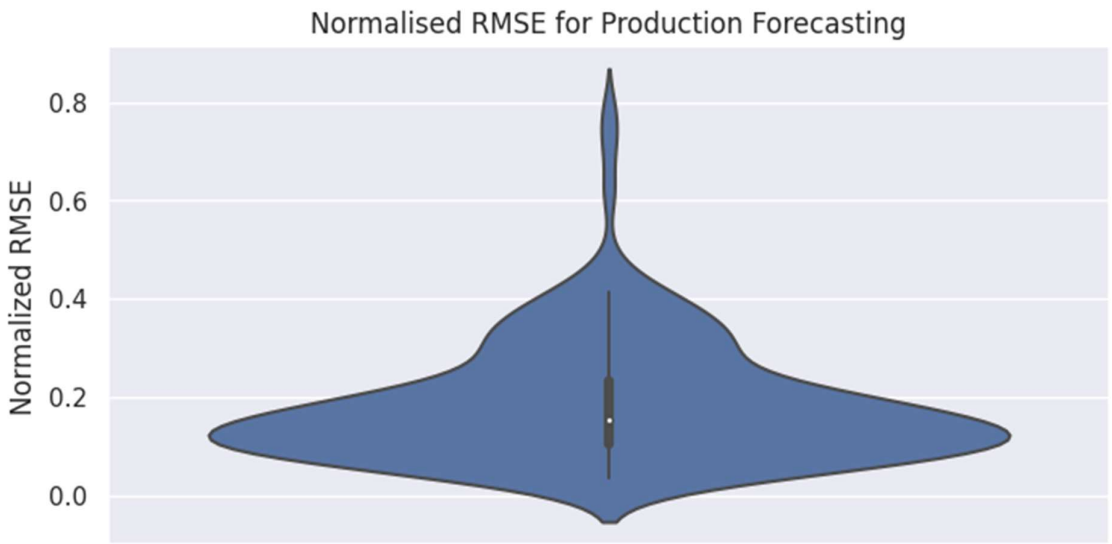

```
[ ]: df[df['Area'] == 'Singapore'].groupby('Year')['Production'].sum().plot()
```

```
[ ]: <Axes: xlabel='Year'>
```



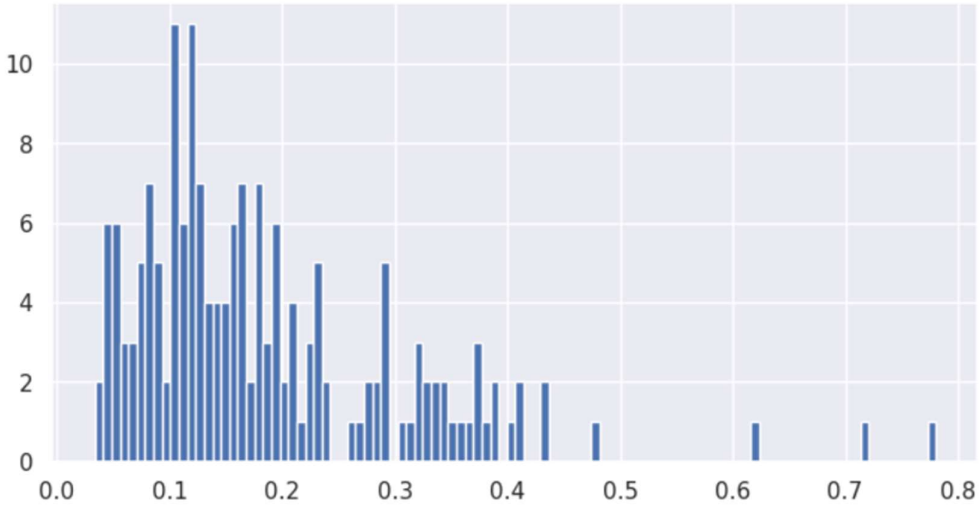
In [ ]:
```
# Exponential Model Crop Production
groups = df.groupby(['Area', 'Year'])['Production'].sum()
n_folds = 5
exponential_models_errors = {}
exponential_models = {}
for country in groups.index.levels[0]:
    model = LinearRegression()
    X = groups[country].index.values.reshape(-1, 1)
    mean = groups[country].values.mean()
    values = np.log(groups[country].values)
    tscv = TimeSeriesSplit(n_splits=n_folds)
    error = 0

    if len(x) > n_folds:
        for fold_index, (train_index, test_index) in enumerate(tscv.split(x)):
            x_train, x_test = X[train_index], X[test_index]
            y_train, y_test = values[train_index], values[test_index]
            model = LinearRegression()
            model.fit(x_train.reshape(-1, 1), y_train)
            pred = np.exp(model.predict(x_test.reshape(-1, 1)))
            error += np.sqrt(np.mean((pred - np.exp(y_test))**2))
        error /= n_folds
        model = LinearRegression()
        model.fit(X, values)
        exponential_models_errors[country] = {'error': error, 'mean': mean}
        exponential_models[country] = model
exponential_models_error_df = pd.DataFrame.from_dict(exponential_models_errors, orient='index').reset_index()
exponential_models_error_df['normalized_error'] = exponential_models_error_df['error'] / exponential_models_error_df['mean']
```

[ ]:
```
sns.set(rc={'figure.figsize':(8, 4)})

# Plot the violin plot
sns.violinplot(exponential_models_error_df, y="normalized_error")

# Set labels and title
plt.xlabel("")
plt.ylabel("Normalized RMSE")
plt.title("Normalised RMSE for Production Forecasting")


# Show the plot
plt.show()
```



In [ ]:
```
error_df['normalized_error'].hist(bins=100)
```

Out[ ]: <Axes: >



In [ ]:
```
# Linear Model Crop Production
groups = df.groupby(['Area', 'Year'])['Production'].sum()
n_folds = 5
errors = {}
models = {}
for country in groups.index.levels[0]:
    model = LinearRegression()
    X = groups[country].index.values.reshape(-1, 1)
    mean = groups[country].values.mean()
    values = groups[country].values
    tscv = TimeSeriesSplit(n_splits=n_folds)
    error = 0

    if len(x) > n_folds:
        for fold_index, (train_index, test_index) in enumerate(tscv.split(x)):
            x_train, x_test = X[train_index], X[test_index]
            y_train, y_test = values[train_index], values[test_index]
            model = LinearRegression()
            model.fit(x_train.reshape(-1, 1), y_train)
            pred = model.predict(x_test.reshape(-1, 1))
            error += np.sqrt(np.mean((pred - y_test)**2))
        error /= n_folds
        model = LinearRegression()
        model.fit(X, values)
        errors[country] = {'error': error, 'mean': mean}
        models[country] = model
error_df = pd.DataFrame.from_dict(errors, orient='index').reset_index()
error_df['normalized_error'] = error_df['error'] / error_df['mean']
```

In [ ]:
```
error_df
```

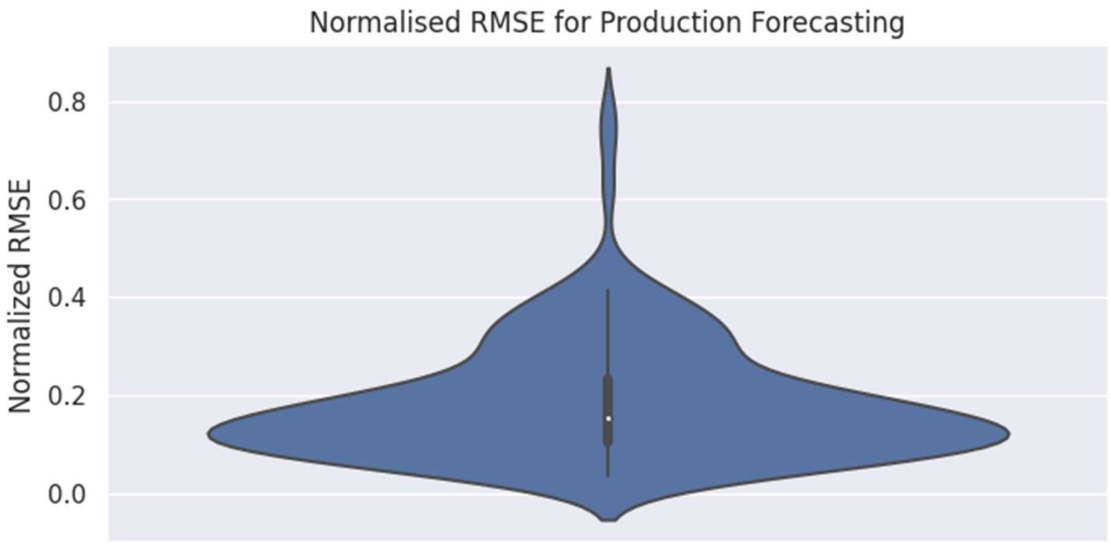
[ ]:
```
error_df
```

t[ ]:

	index	error	mean	normalized_error
0	Afghanistan	1.233741e+07	1.298200e+07	0.950347
1	Albania	4.301018e+06	3.388261e+06	1.269388
2	Algeria	8.624674e+06	1.806249e+07	0.477491
3	Angola	7.217649e+06	1.584729e+07	0.455450
4	Antigua and Barbuda	4.903220e+04	7.739002e+04	0.633573
...	...	...	...	...
169	Venezuela	2.464230e+07	2.493391e+07	0.988305
170	Vietnam	5.884686e+07	1.133515e+08	0.519154
171	Yemen	3.165803e+06	3.922548e+06	0.807078
172	Zambia	6.243769e+06	8.824551e+06	0.707545
173	Zimbabwe	1.573872e+07	1.098410e+07	1.432864

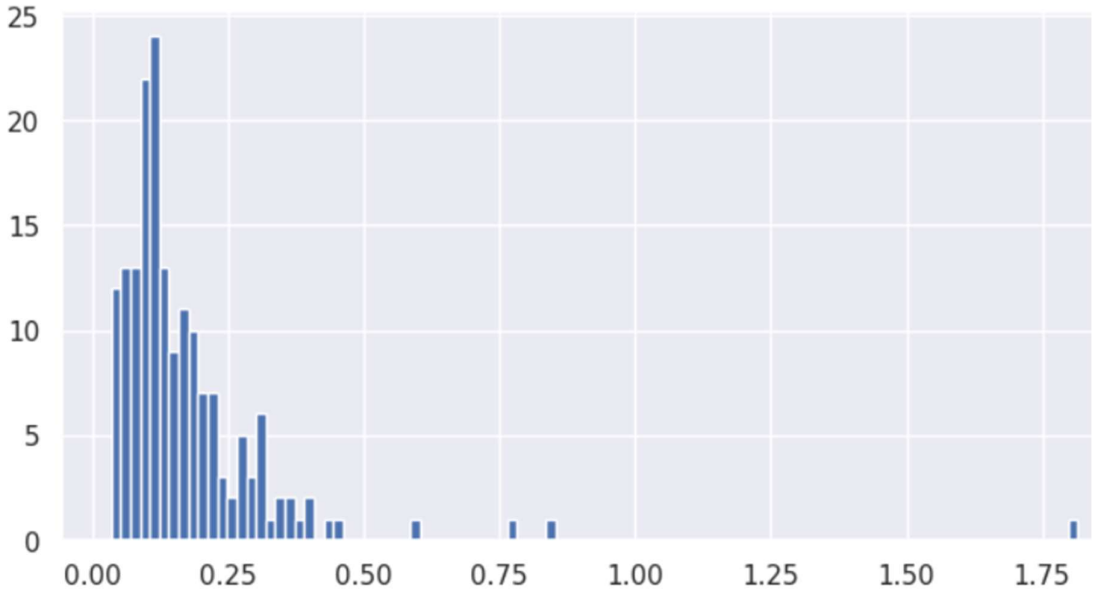
174 rows × 4 columns

[ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.set(rc={'figure.figsize':(8, 4)})

# Plot the violin plot
sns.violinplot(error_df, y="normalized_error")

# Set labels and title
plt.xlabel("")
plt.ylabel("Normalized RMSE")
plt.title("Normalised RMSE for Production Forecasting")
```

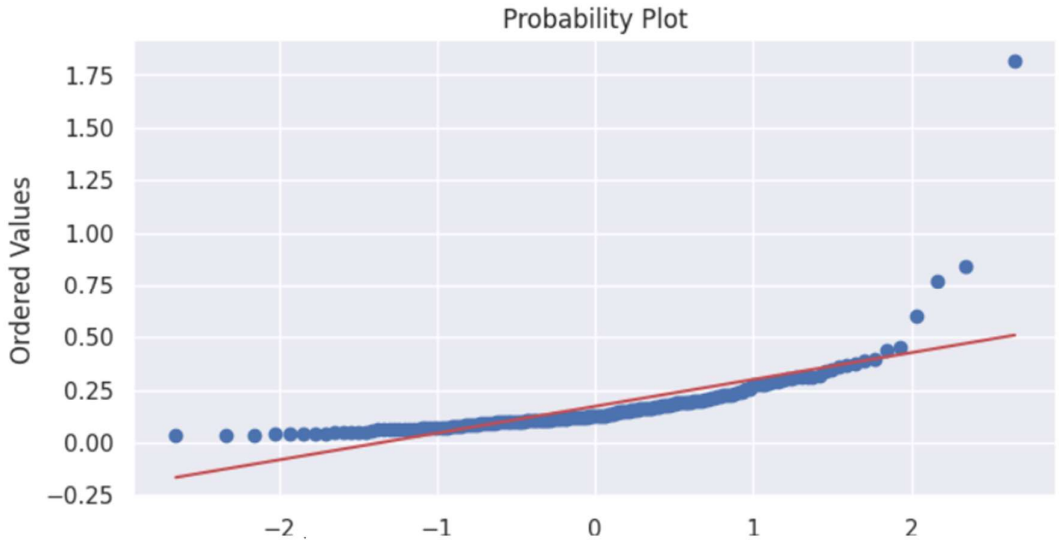



```
]: error_df['normalized_error'].hist(bins=100)
```

```
]: <Axes: >
```



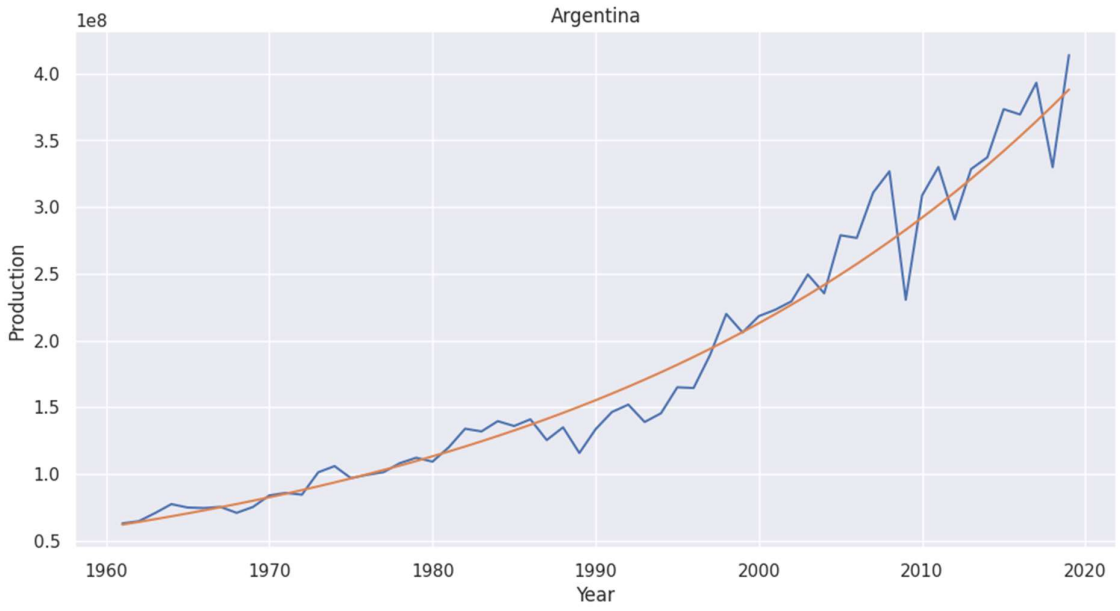
n [ ]:
```python
# Q-Q plot
import scipy.stats as stats
import pylab

stats.probplot(error_df['normalized_error'], dist="norm", plot=pylab)
pylab.show()

# Shapiro-Wilk Test
print(stats.shapiro(error_df['normalized_error']))
```




```
ShapiroResult(statistic=0.5486217737197876, pvalue=4.749914223143136e-21)
```

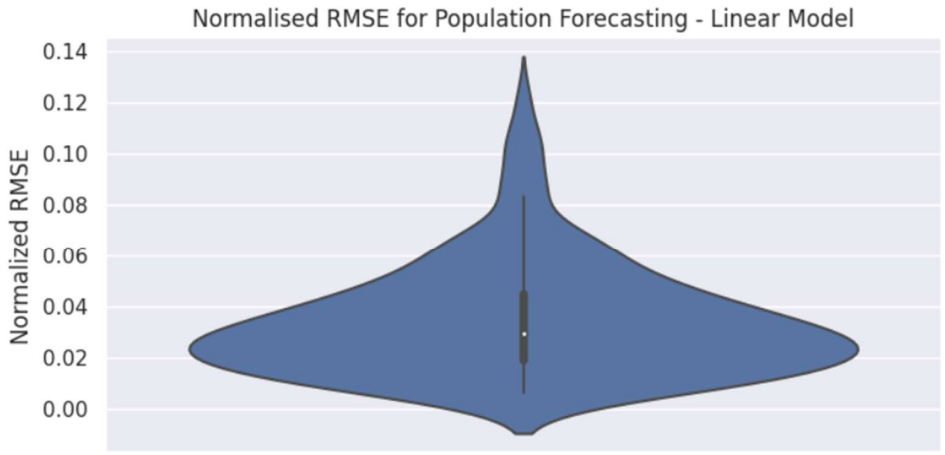
[ ]:
```python
# Production Forecast based on Exponential Model
production_forecast = pd.pivot_table(df, values='Production', index=['Area'], columns=['Year'], aggfunc=np.sum)
production_forecast.reset_index(inplace=True)
production_forecast[list(range(2020, 2034))] = production_forecast.apply(lambda x: np.exp(models[x['Area']].predict(np.arange(2020, 2034).reshape(-1, 1))), axis=1, result_type='expand')
production_forecast
```

[ ]:
```python
groups = df.groupby(['Area', 'Year'])['Production'].sum()
for country in groups.index.levels[0][:10]:
    model = LinearRegression()
    X = groups[country].index.values.reshape(-1, 1)
    y = np.log(groups[country].values)
    # train_index = np.random.choice(len(X), int(len(X) * 0.8), replace=False)
    # X_train, y_train = X[train_index], y[train_index]
    model.fit(X, y)
    plt.figure(figsize=(12, 6))
    plt.plot(groups[country].index.values, groups[country].values)
    plt.plot(X, np.exp(model.predict(X)))
    plt.title(country)
    plt.xlabel('Year')
    plt.ylabel('Production')
    plt.show()
```



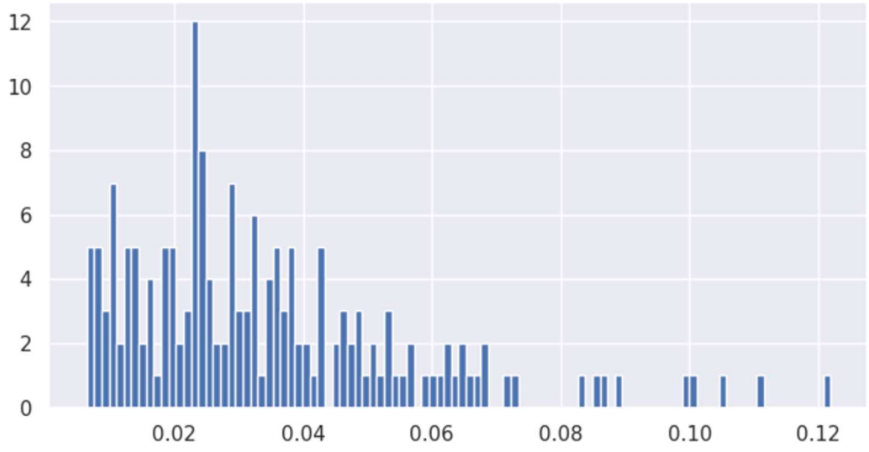
In [ ]:
```
# Linear Model Population
groups = demographics.groupby(['Country Name', 'Year'])['Population'].sum()
n_folds = 5
errors = {}
models = {}
for country in groups.index.levels[0]:
    X = groups[country].index.values.reshape(-1, 1)
    mean = groups[country].values.mean()
    values = groups[country].values
    tscv = TimeSeriesSplit(n_splits=n_folds)
    error = 0
    if len(x) > n_folds:
        for fold_index, (train_index, test_index) in enumerate(tscv.split(x)):
            x_train, x_test = X[train_index], X[test_index]
            y_train, y_test = values[train_index], values[test_index]
            model = LinearRegression()
            model.fit(x_train.reshape(-1, 1), y_train)
            pred = model.predict(x_test.reshape(-1, 1))
            error += np.sqrt(np.mean((pred - y_test)**2))
        error /= n_folds
        model = LinearRegression()
        model.fit(X, values)
        errors[country] = {'error': error, 'mean': mean}
        models[country] = model
error_df = pd.DataFrame.from_dict(errors, orient='index').reset_index()
error_df['normalized error'] = error_df['error'] / error_df['mean']
```

In [ ]:
```
sns.set(rc={'figure.figsize':(8, 4)})

# Plot the violin plot
sns.violinplot(error_df, y="normalized_error")

# Set labels and title
plt.xlabel("")
plt.ylabel("Normalized RMSE")
plt.title("Normalised RMSE for Population Forecasting - Linear Model")


# Show the plot
plt.show()
```

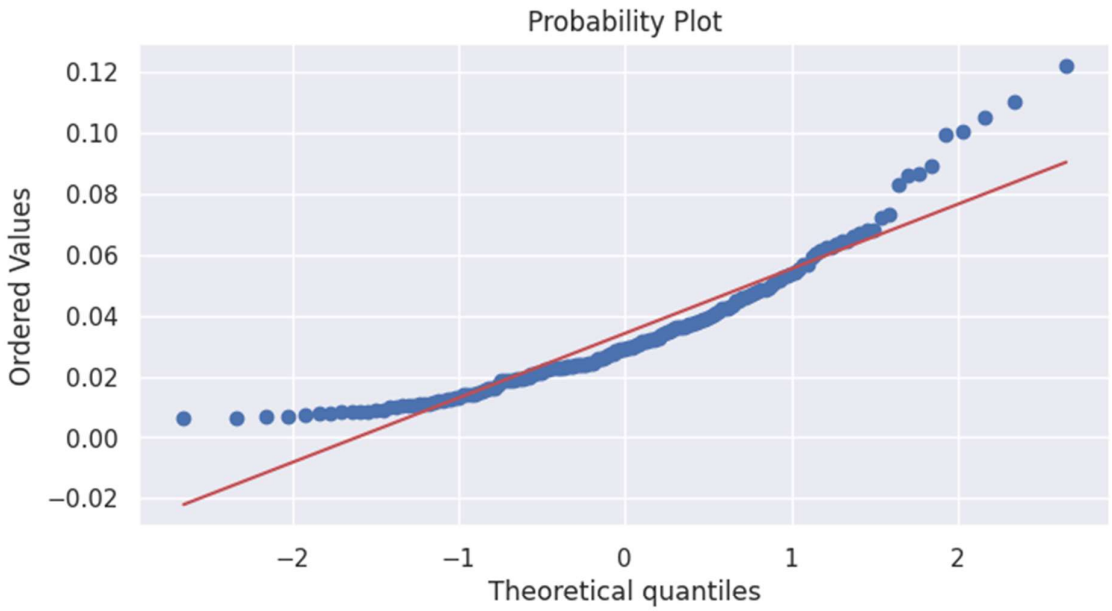


In [ ]:
```
error_df['normalized_error'].hist(bins=100)
```

Out[ ]:
```
<Axes: >
```



In [ ]:
```
stats.probplot(error_df['normalized_error'], dist="norm", plot=pylab)
pylab.show()
```

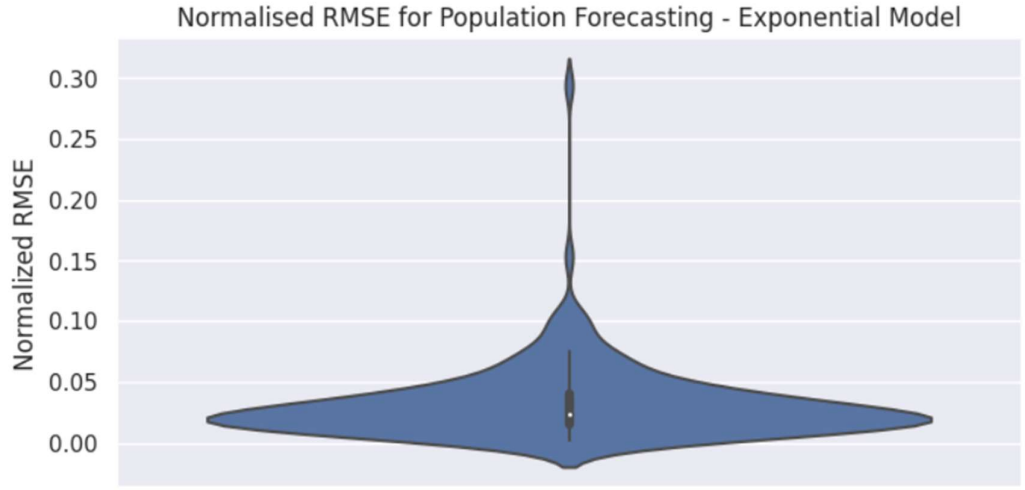


In [ ]:
```
# Shapiro-Wilk Test
print(stats.shapiro(error_df['normalized_error']))
```

```
ShapiroResult(statistic=0.8902348875999451, pvalue=5.37045796722424e-10)
```

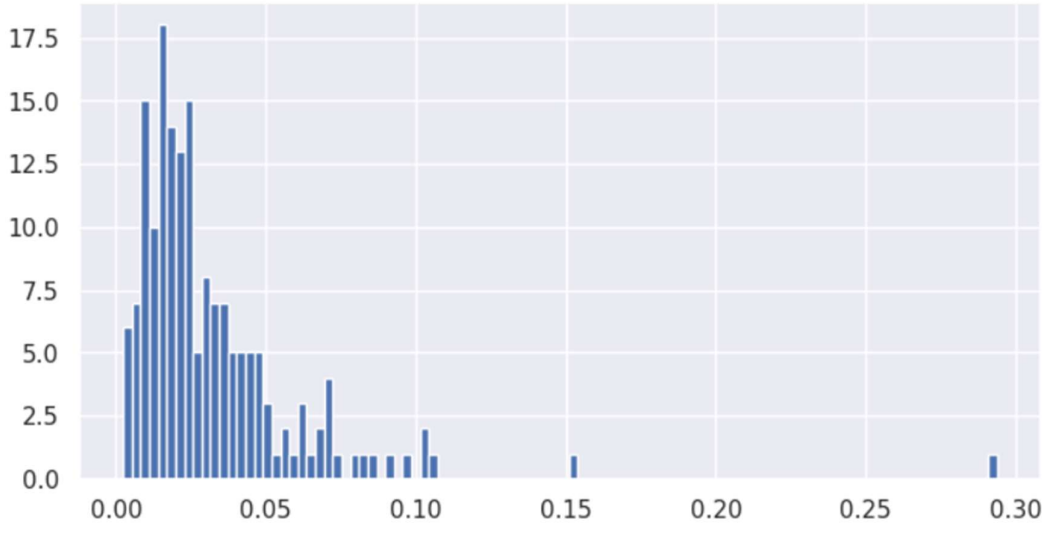
In [ ]:
```
# Exponential Model Population
groups = demographics.groupby(['Country Name', 'Year'])['Population'].sum()
n_folds = 5
errors = {}
models = {}
for country in groups.index.levels[0]:
    X = groups[country].index.values.reshape(-1, 1)
    mean = groups[country].values.mean()
    values = np.log(groups[country].values)
    tscv = TimeSeriesSplit(n_splits=n_folds)
    error = 0
    if len(x) > n_folds:
        for fold_index, (train_index, test_index) in enumerate(tscv.split(x)):
            x_train, x_test = X[train_index], X[test_index]
            y_train, y_test = values[train_index], values[test_index]
            model = LinearRegression()
            model.fit(x_train.reshape(-1, 1), y_train)
            pred = model.predict(x_test.reshape(-1, 1))
            pred = np.exp(pred)
            error += np.sqrt(np.mean((pred - np.exp(y_test))**2))
        error /= n_folds
        model = LinearRegression()
        model.fit(X, values)
        errors[country] = {'error': error, 'mean': mean}
        models[country] = model
error_df = pd.DataFrame.from_dict(errors, orient='index').reset_index()
error_df['normalized_error'] = error_df['error'] / error_df['mean']
```

```
n [ ]: sns.set(rc={'figure.figsize':(8, 4)})

       # Plot the violin plot
       sns.violinplot(error_df, y="normalized_error")

       # Set labels and title
       plt.xlabel("")
       plt.ylabel("Normalized RMSE")
       plt.title("Normalised RMSE for Population Forecasting - Exponential Model")


       # Show the plot
       plt.show()
```



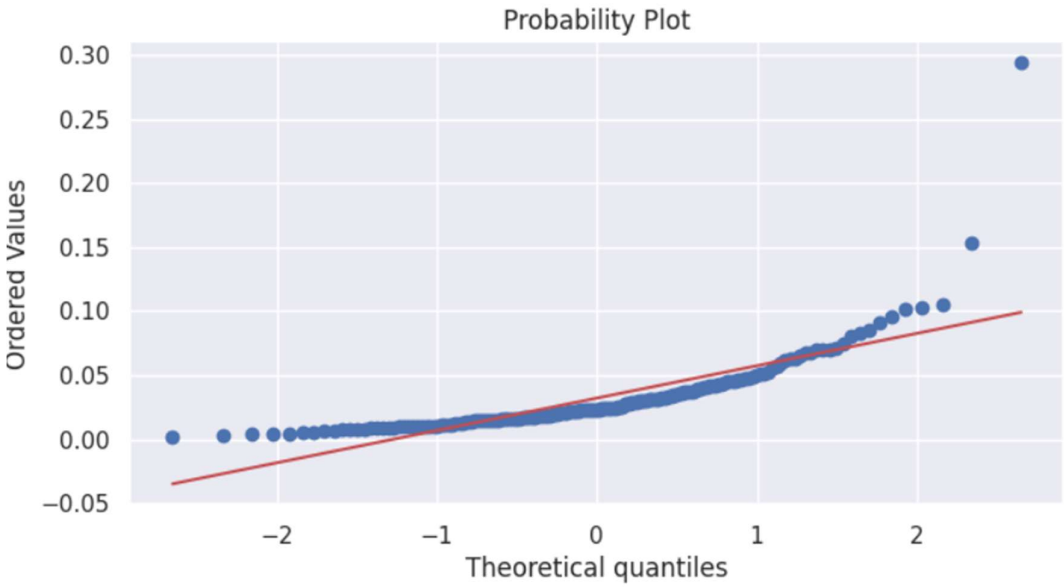

```
]: error_df['normalized_error'].hist(bins=100)
```

```
]: <Axes: >
```




```
]: stats.probplot(error_df['normalized_error'], dist="norm", plot=pylab)
   pylab.show()
```

```
stats.probplot(error_df[ normalized_error ], dist= norm , plot=pylab)
pylab.show()
```



In [ ]:
```
# Shapiro-Wilk Test
print(stats.shapiro(error_df['normalized_error']))
```

```
ShapiroResult(statistic=0.6638813614845276, pvalue=2.487397862805652e-18)
```

In [ ]:
```
# Population Forecasting
pivot = pd.pivot_table(demographics, values='Population', index=['Country Name'], columns=['Year'], aggfunc=np.sum)
pivot.reset_index(inplace=True)
pivot[list(range(2022, 2034))] = pivot.apply(lambda x: models[x['Country Name']].predict(np.arange(2022, 2034).reshape(-1, 1)), axis=1, result_type='expand')
```

In [ ]:
```
# Plot line graph for top 10 countries
for country in pivot.sort_values(2033, ascending=False)['Country Name'].values[:10]:
    plt.figure(figsize=(12, 6))
    plt.plot(pivot.columns[1:], pivot[pivot['Country Name'] == country].values[0][1:])
    plt.title(country)
    plt.xlabel('Year')
    plt.ylabel('Population')
    plt.show()
```