# Code:

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression

# Load the investment data

investment\_file = 'finvestment.xlsx'

df = pd.read\_excel(investment\_file)

# Convert columns starting with 'YR' to numerical type

for col in df.columns:

if col.startswith('YR'):

df[col] = pd.to\_numeric(df[col], errors='coerce')

# Initialize lists to store results

mean\_investments = []

median\_investments = []

growth\_rates = []

# Function to calculate growth rate using linear regression

def calculate\_growth\_rate(values, years):

values = np.array(values)

years = np.array(years)

# Check lengths

if len(years) != len(values):

print(f"Length of years: {len(years)}, Length of values: {len(values)}")

raise ValueError("Mismatch between years and values lengths.")

# Create a boolean mask to filter out NaNs

valid = ~np.isnan(values)

# Filter valid data

valid\_years = years[valid]

valid\_values = values[valid]

# Check if there are enough valid data points

if len(valid\_years) < 2:

return np.nan

model = LinearRegression()

model.fit(valid\_years.reshape(-1, 1), valid\_values)

growth\_rate = model.coef\_[0] \* 100 # Convert to percentage

return growth\_rate

# Group by country and calculate statistics

grouped = df.groupby('Country Name')

for country, group in grouped:

# Drop non-numeric columns

numeric\_data = group.drop(columns=['Series Name', 'Country Name'])

# Check that numeric\_data contains the correct number of columns

year\_columns = [col for col in numeric\_data.columns if col.startswith('YR')]

years = [int(col.replace('YR', '')) for col in year\_columns]

for idx, row in numeric\_data.iterrows():

values = row[year\_columns].values

print(f"Country: {country}, Years: {years}, Values: {values}")

# Calculate mean and median for each country

mean\_investments.append(numeric\_data[year\_columns].mean(axis=1).mean())

median\_investments.append(numeric\_data[year\_columns].median(axis=1).median())

# Calculate growth rate

growth\_rate = calculate\_growth\_rate(values, years)

growth\_rates.append(growth\_rate)

# Create a summary DataFrame

summary\_df = pd.DataFrame({

'Country Name': grouped.size().index,

'Mean Investment': mean\_investments,

'Median Investment': median\_investments,

'Growth Rate (%)': growth\_rates

})

# Save the summary DataFrame to an Excel file

summary\_df.to\_excel('investment\_analysis\_summary.xlsx', index=False)

print("Investment analysis completed and saved to 'investment\_analysis\_summary.xlsx'.")

print(summary\_df)

# output:

Country: brazil, Years: [1990, 2000, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023], Values: [0.000000e+00 1.616400e+09 3.374923e+10 1.752800e+09 8.961000e+08

5.496000e+08 1.041400e+09 2.763010e+09 2.821600e+08 6.527030e+09

1.041998e+10 0.000000e+00]

Country: british virgin islands, Years: [1990, 2000, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023], Values: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

Country: china, Years: [1990, 2000, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023], Values: [1.730000e+08 1.558500e+09 5.009410e+09 2.913740e+09 3.469630e+09

1.258744e+10 2.596480e+10 2.251550e+10 2.788890e+09 6.504820e+09

2.843359e+10 0.000000e+00]

Country: colombia, Years: [1990, 2000, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023], Values: [0.00000e+00 1.04770e+09 4.24790e+09 4.49590e+09 5.99854e+09 3.68500e+08

1.86474e+09 2.64500e+09 1.00000e+09 5.86220e+08 2.07000e+09 0.00000e+00]

Country: india, Years: [1990, 2000, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023], Values: [1.90000e+06 9.64000e+07 2.93165e+09 2.04722e+09 2.62312e+09 3.48290e+09

6.60752e+09 6.46984e+09 1.86186e+09 5.82331e+09 9.27164e+09 0.00000e+00]

Country: latin america & caribbean, Years: [1990, 2000, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023], Values: [0.000000e+00 4.488000e+09 3.944584e+10 1.478370e+10 8.453240e+09

0.000000e+00 4.087500e+09 0.000000e+00 2.357860e+09 0.000000e+00

0.000000e+00 0.000000e+00]

Country: mexico, Years: [1990, 2000, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023], Values: [4.6034e+09 7.9360e+08 6.2940e+08 7.8710e+08 4.2270e+08 1.0069e+09

5.8936e+08 2.6000e+08 1.3770e+08 6.5540e+08 2.2800e+08 0.0000e+00]

Country: south asia, Years: [1990, 2000, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023], Values: [1.90000e+06 9.64000e+07 2.93165e+09 2.39722e+09 2.62312e+09 3.78248e+09

6.60752e+09 7.61910e+09 2.72286e+09 6.47331e+09 9.64917e+09 0.00000e+00]

Country: turkiye, Years: [1990, 2000, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023], Values: [0.00000e+00 0.00000e+00 4.74340e+09 4.31032e+10 7.79000e+08 9.91880e+08

6.92390e+09 2.37800e+08 0.00000e+00 8.22200e+09 3.00000e+07 0.00000e+00]

Investment analysis completed and saved to 'investment\_analysis\_summary.xlsx'.

Country Name Mean Investment Median Investment \

0 brazil 4.966476e+09 1.328900e+09

1 british virgin islands 0.000000e+00 0.000000e+00

2 china 9.326610e+09 4.239520e+09

3 colombia 2.027042e+09 1.456220e+09

4 india 3.434780e+09 2.777385e+09

5 latin america & caribbean 6.134678e+09 1.178930e+09

6 mexico 8.427967e+08 6.093800e+08

7 south asia 3.742061e+09 2.827255e+09

8 turkiye 5.419265e+09 5.084000e+08

Growth Rate (%)

0 9.167063e+09

1 0.000000e+00

2 4.392850e+10

3 3.427516e+09

4 1.682851e+10

5 -4.139683e+09

6 -1.063170e+10

7 1.849819e+10

8 9.153676e+09

# Code:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Load the mortality data

mortality\_file = 'fmortality.xlsx'

df = pd.read\_excel(mortality\_file)

# Convert columns starting with 'YR' to numerical type

for col in df.columns:

if col.startswith('YR'):

df[col] = pd.to\_numeric(df[col], errors='coerce')

# Calculate average mortality rate for each country

df['Average Mortality Rate'] = df.filter(like='YR').mean(axis=1)

# Initialize lists to store results

countries = df['Country Name'].unique()

trends = []

# Analyze trends over time

for country in countries:

country\_data = df[df['Country Name'] == country]

years = [int(col.replace('YR', '')) for col in df.columns if col.startswith('YR')]

mortality\_rates = country\_data.filter(like='YR').values.flatten()

# Remove NaN values

valid\_indices = ~np.isnan(mortality\_rates)

valid\_years = np.array(years \* len(country\_data))[valid\_indices]

valid\_mortality\_rates = mortality\_rates[valid\_indices]

# Plot trend

plt.figure(figsize=(10, 6))

plt.plot(valid\_years, valid\_mortality\_rates, marker='o', linestyle='-', label=country)

plt.xlabel('Year')

plt.ylabel('Mortality Rate (per 100,000 population)')

plt.title(f'Mortality Rate Trends for {country}')

plt.legend()

plt.grid(True)

plt.savefig(f'{country}\_mortality\_trends.png')

plt.close()

# Print average mortality rates

print("Average Mortality Rates:")

print(df[['Country Name', 'Average Mortality Rate']])

# Save the results to an Excel file

df[['Country Name', 'Average Mortality Rate']].to\_excel('average\_mortality\_rates.xlsx', index=False)

print("Mortality rates analysis completed and saved to 'average\_mortality\_rates.xlsx'.")

# output:

Average Mortality Rates:

Country Name Average Mortality Rate

0 brazil 11.316667

1 british virgin islands 0.000000

2 china 10.775000

3 colombia 10.200000

4 india 9.183333

5 mexico 7.800000

6 turkiye 5.200000

7 latin america & caribbean 10.475106

8 south asia 8.972203

Mortality rates analysis completed and saved to 'average\_mortality\_rates.xlsx'.

# Code:

import pandas as pd

import numpy as np

# Load the SPI data

spi\_file = 'fSPI.xlsx'

df = pd.read\_excel(spi\_file)

# Ensure that all SPI columns are numeric

# Assuming all columns are SPI columns starting with 'YR'

for col in df.columns:

if col.startswith('YR'):

df[col] = pd.to\_numeric(df[col], errors='coerce')

# Drop non-numeric columns for SPI calculations

spi\_data = df.drop(columns=['Series Name', 'Country Name'])

# Calculate summary statistics

mean\_spi = spi\_data.mean(axis=1)

median\_spi = spi\_data.median(axis=1)

range\_spi = spi\_data.max(axis=1) - spi\_data.min(axis=1)

# Add summary statistics to DataFrame

df['Mean SPI'] = mean\_spi

df['Median SPI'] = median\_spi

df['Range SPI'] = range\_spi

# Print summary statistics

print("Summary Statistics for SPI Scores:")

print(f"Mean:\n{df['Mean SPI'].describe()}")

print(f"Median:\n{df['Median SPI'].describe()}")

print(f"Range:\n{df['Range SPI'].describe()}")

# Save the results to an Excel file

df[['Country Name', 'Mean SPI', 'Median SPI', 'Range SPI']].to\_excel('spi\_summary\_statistics.xlsx', index=False)

print("SPI performance analysis completed and saved to 'spi\_summary\_statistics.xlsx'.")

# output:

Summary Statistics for SPI Scores:

Mean:

count 9.000000

mean 25.694444

std 20.573378

min 0.000000

25% 0.000000

50% 30.833333

75% 38.333333

max 55.000000

Name: Mean SPI, dtype: float64

Median:

count 9.000000

mean 38.333333

std 30.669610

min 0.000000

25% 0.000000

50% 47.500000

75% 55.000000

max 82.500000

Name: Median SPI, dtype: float64

Range:

count 9.000000

mean 50.000000

std 39.370039

min 0.000000

25% 0.000000

50% 65.000000

75% 75.000000

max 100.000000

Name: Range SPI, dtype: float64

SPI performance analysis completed and saved to 'spi\_summary\_statistics.xlsx'.

# Code :

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Load the investment data

investment\_file = 'finvestment.xlsx'

df = pd.read\_excel(investment\_file)

# Convert columns starting with 'YR' to numerical type

for col in df.columns:

if col.startswith('YR'):

df[col] = pd.to\_numeric(df[col], errors='coerce')

# Extract the year columns

year\_columns = [col for col in df.columns if col.startswith('YR')]

years = [int(col.replace('YR', '')) for col in year\_columns]

# Prepare to collect descriptive statistics

descriptive\_stats = []

# Plot investment trends over time for each country

countries = df['Country Name'].unique()

plt.figure(figsize=(14, 10))

for country in countries:

country\_data = df[df['Country Name'] == country]

investments = country\_data[year\_columns].values.flatten()

# Remove NaN values

valid\_indices = ~np.isnan(investments)

valid\_years = np.array(years \* len(country\_data))[valid\_indices]

valid\_investments = investments[valid\_indices]

# Calculate descriptive statistics

mean\_investment = np.mean(valid\_investments)

median\_investment = np.median(valid\_investments)

std\_dev\_investment = np.std(valid\_investments)

min\_investment = np.min(valid\_investments)

max\_investment = np.max(valid\_investments)

descriptive\_stats.append({

'Country': country,

'Mean Investment': mean\_investment,

'Median Investment': median\_investment,

'Standard Deviation': std\_dev\_investment,

'Min Investment': min\_investment,

'Max Investment': max\_investment

})

# Plot

plt.plot(valid\_years, valid\_investments, marker='o', linestyle='-', label=country)

plt.xlabel('Year')

plt.ylabel('Investment (current US$)')

plt.title('Investment Trends Over Time by Country')

plt.legend(loc='upper left', bbox\_to\_anchor=(1, 1))

plt.grid(True)

plt.tight\_layout()

# Save the plot

plt.savefig('investment\_trends.png')

plt.show()

# Print descriptive statistics

stats\_df = pd.DataFrame(descriptive\_stats)

print("Descriptive Statistics for Investment Trends:")

print(stats\_df)

print("Investment trends plotted and saved to 'investment\_trends.png'.")

# output :

Descriptive Statistics for Investment Trends:

Country Mean Investment Median Investment \

0 brazil 4.966476e+09 1.328900e+09

1 british virgin islands 0.000000e+00 0.000000e+00

2 china 9.326610e+09 4.239520e+09

3 colombia 2.027042e+09 1.456220e+09

4 india 3.434780e+09 2.777385e+09

5 mexico 8.427967e+08 6.093800e+08

6 turkiye 5.419265e+09 5.084000e+08

7 latin america & caribbean 6.134678e+09 1.178930e+09

8 south asia 3.742061e+09 2.827255e+09

Standard Deviation Min Investment Max Investment

0 9.173270e+09 0 33749230000

1 0.000000e+00 0 0

2 1.000989e+10 0 28433590000

3 1.876805e+09 0 5998540000

4 2.873626e+09 0 9271640000

5 1.169795e+09 0 4603400000

6 1.170881e+10 0 43103200000

7 1.094222e+10 0 39445840000

8 3.050088e+09 0 9649170000

Investment trends plotted and saved to 'investment\_trends.png'.

# Code:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

# Load the mortality data

mortality\_file = 'fmortality.xlsx'

df = pd.read\_excel(mortality\_file)

# Convert columns starting with 'YR' to numerical type

for col in df.columns:

if col.startswith('YR'):

df[col] = pd.to\_numeric(df[col], errors='coerce')

# Extract year columns

year\_columns = [col for col in df.columns if col.startswith('YR')]

years = [int(col.replace('YR', '')) for col in year\_columns]

# Prepare to collect descriptive statistics and trends

descriptive\_stats = []

increasing\_countries = []

decreasing\_countries = []

# Plot mortality trends over time for each country

countries = df['Country Name'].unique()

plt.figure(figsize=(14, 10))

for country in countries:

country\_data = df[df['Country Name'] == country]

mortality\_rates = country\_data[year\_columns].values.flatten()

# Remove NaN values

valid\_indices = ~np.isnan(mortality\_rates)

valid\_years = np.array(years \* len(country\_data))[valid\_indices]

valid\_mortality\_rates = mortality\_rates[valid\_indices]

# Calculate descriptive statistics

mean\_mortality = np.mean(valid\_mortality\_rates)

median\_mortality = np.median(valid\_mortality\_rates)

std\_dev\_mortality = np.std(valid\_mortality\_rates)

min\_mortality = np.min(valid\_mortality\_rates)

max\_mortality = np.max(valid\_mortality\_rates)

descriptive\_stats.append({

'Country': country,

'Mean Mortality Rate': mean\_mortality,

'Median Mortality Rate': median\_mortality,

'Standard Deviation': std\_dev\_mortality,

'Min Mortality Rate': min\_mortality,

'Max Mortality Rate': max\_mortality

})

# Plot

plt.plot(valid\_years, valid\_mortality\_rates, marker='o', linestyle='-', label=country)

# Calculate trend slope using linear regression

if len(valid\_years) > 1: # Ensure there's more than one data point

years\_array = np.array(valid\_years).reshape(-1, 1)

mortality\_array = np.array(valid\_mortality\_rates).reshape(-1, 1)

model = LinearRegression()

model.fit(years\_array, mortality\_array)

trend\_slope = model.coef\_[0][0]

if trend\_slope > 0:

increasing\_countries.append(country)

elif trend\_slope < 0:

decreasing\_countries.append(country)

plt.xlabel('Year')

plt.ylabel('Mortality Rate (per 100,000 population)')

plt.title('Mortality Rate Trends Over Time by Country')

plt.legend(loc='upper left', bbox\_to\_anchor=(1, 1))

plt.grid(True)

plt.tight\_layout()

# Save the plot

plt.savefig('mortality\_trends.png')

plt.show()

# Print descriptive statistics

stats\_df = pd.DataFrame(descriptive\_stats)

print("Descriptive Statistics for Mortality Rates:")

print(stats\_df)

print("Mortality rate trends plotted and saved to 'mortality\_trends.png'.")

# Print countries with increasing or decreasing mortality rates

print("Countries with increasing mortality rates:")

print(increasing\_countries)

print("Countries with decreasing mortality rates:")

print(decreasing\_countries)

# output :

Descriptive Statistics for Mortality Rates:

Country Mean Mortality Rate Median Mortality Rate \

0 brazil 11.316667 16.550000

1 british virgin islands 0.000000 0.000000

2 china 10.775000 17.500000

3 colombia 10.200000 15.850000

4 india 9.183333 15.250000

5 mexico 7.800000 12.900000

6 turkiye 5.200000 7.550000

7 latin america & caribbean 10.475106 17.363751

8 south asia 8.972203 15.085419

Standard Deviation Min Mortality Rate Max Mortality Rate

0 9.725382 0.0 23.600000

1 0.000000 0.0 0.000000

2 9.158068 0.0 21.400000

3 8.688019 0.0 20.000000

4 7.772369 0.0 16.900000

5 6.600631 0.0 14.200000

6 4.461502 0.0 9.900000

7 8.862897 0.0 19.001251

8 7.586245 0.0 15.937456

Mortality rate trends plotted and saved to 'mortality\_trends.png'.

Countries with increasing mortality rates:

[]

Countries with decreasing mortality rates:

['brazil', 'china', 'colombia', 'india', 'mexico', 'turkiye', 'latin america & caribbean', 'south asia']

# Code:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Load the SPI data

spi\_file = 'fSPI.xlsx'

df = pd.read\_excel(spi\_file)

# Convert columns starting with 'YR' to numerical type

for col in df.columns:

if col.startswith('YR'):

df[col] = pd.to\_numeric(df[col], errors='coerce')

# Extract year columns

year\_columns = [col for col in df.columns if col.startswith('YR')]

years = [int(col.replace('YR', '')) for col in year\_columns]

# Prepare to collect descriptive statistics

descriptive\_stats = []

# Plot infrastructure performance scores over time for each country

countries = df['Country Name'].unique()

plt.figure(figsize=(14, 10))

for country in countries:

country\_data = df[df['Country Name'] == country]

spi\_scores = country\_data[year\_columns].values.flatten()

# Remove NaN values

valid\_indices = ~np.isnan(spi\_scores)

valid\_years = np.array(years \* len(country\_data))[valid\_indices]

valid\_spi\_scores = spi\_scores[valid\_indices]

# Calculate descriptive statistics

mean\_spi = np.mean(valid\_spi\_scores)

median\_spi = np.median(valid\_spi\_scores)

std\_dev\_spi = np.std(valid\_spi\_scores)

min\_spi = np.min(valid\_spi\_scores)

max\_spi = np.max(valid\_spi\_scores)

descriptive\_stats.append({

'Country': country,

'Mean SPI Score': mean\_spi,

'Median SPI Score': median\_spi,

'Standard Deviation': std\_dev\_spi,

'Min SPI Score': min\_spi,

'Max SPI Score': max\_spi

})

# Plot

plt.plot(valid\_years, valid\_spi\_scores, marker='o', linestyle='-', label=country)

plt.xlabel('Year')

plt.ylabel('SPI Score')

plt.title('Infrastructure Performance Scores Over Time by Country')

plt.legend(loc='upper left', bbox\_to\_anchor=(1, 1))

plt.grid(True)

plt.tight\_layout()

# Save the plot

plt.savefig('infrastructure\_performance\_trends.png')

plt.show()

# Print descriptive statistics

stats\_df = pd.DataFrame(descriptive\_stats)

print("Descriptive Statistics for Infrastructure Performance Scores:")

print(stats\_df)

# print("Infrastructure performance trends plotted and saved to 'infrastructure\_performance\_trends.png'.") output:

Descriptive Statistics for Infrastructure Performance Scores:

Country Mean SPI Score Median SPI Score \

0 brazil 38.333333 55.0

1 british virgin islands 0.000000 0.0

2 china 30.833333 47.5

3 colombia 34.583333 47.5

4 india 30.833333 50.0

5 mexico 41.666667 62.5

6 turkiye 55.000000 82.5

7 latin america & caribbean 0.000000 0.0

8 south asia 0.000000 0.0

Standard Deviation Min SPI Score Max SPI Score

0 32.871805 0 75

1 0.000000 0 0

2 26.444386 0 65

3 30.987789 0 80

4 26.127359 0 55

5 35.843022 0 75

6 46.948553 0 100

7 0.000000 0 0

8 0.000000 0 0

Infrastructure performance trends plotted and saved to 'infrastructure\_performance\_trends.png'.

# Code :

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Load the investment and mortality data

investment\_file = 'finvestment.xlsx'

mortality\_file = 'fmortality.xlsx'

df\_investment = pd.read\_excel(investment\_file)

df\_mortality = pd.read\_excel(mortality\_file)

# Clean column names

df\_investment.columns = df\_investment.columns.str.strip()

df\_mortality.columns = df\_mortality.columns.str.strip()

# Drop unnecessary columns (e.g., 'Unnamed' columns)

df\_investment = df\_investment.loc[:, ~df\_investment.columns.str.contains('^Unnamed')]

df\_mortality = df\_mortality.loc[:, ~df\_mortality.columns.str.contains('^Unnamed')]

# Extract year columns

year\_columns\_investment = [col for col in df\_investment.columns if col.startswith('YR')]

year\_columns\_mortality = [col for col in df\_mortality.columns if col.startswith('YR')]

# Create long format DataFrames

df\_investment\_long = df\_investment.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_investment, var\_name='Year', value\_name='Investment')

df\_mortality\_long = df\_mortality.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_mortality, var\_name='Year', value\_name='Mortality')

# Remove 'YR' prefix from 'Year' column and convert to numeric

df\_investment\_long['Year'] = df\_investment\_long['Year'].str.replace('YR', '').astype(int)

df\_mortality\_long['Year'] = df\_mortality\_long['Year'].str.replace('YR', '').astype(int)

# Merge the long format DataFrames on 'Country Name' and 'Year'

merged\_long = pd.merge(df\_investment\_long, df\_mortality\_long, on=['Country Name', 'Year'])

# Calculate correlation for each year

correlation\_per\_year = merged\_long.groupby('Year').apply(lambda x: x[['Investment', 'Mortality']].corr().iloc[0, 1])

# Plot the correlations over time

plt.figure(figsize=(12, 6))

plt.plot(correlation\_per\_year.index, correlation\_per\_year.values, marker='o', linestyle='-', color='b')

plt.xlabel('Year')

plt.ylabel('Correlation between Investment and Mortality')

plt.title('Correlation between Transport Investment and Road Traffic Mortality Rates Over Time')

plt.grid(True)

plt.tight\_layout()

# Save the plot

plt.savefig('investment\_vs\_mortality\_correlation.png')

plt.show()

# Display average correlation

average\_correlation = correlation\_per\_year.mean()

print("Correlation analysis completed and plot saved to 'investment\_vs\_mortality\_correlation.png'.")

print("Average Correlation:", average\_correlation)

# Descriptive statistics for correlations

stats = {

'Mean Correlation': average\_correlation,

'Median Correlation': correlation\_per\_year.median(),

'Standard Deviation': correlation\_per\_year.std(),

'Min Correlation': correlation\_per\_year.min(),

'Max Correlation': correlation\_per\_year.max()

}

print("\nDescriptive Statistics for Correlation:")

for key, value in stats.items():

print(f"{key}: {value:.4f}")

# output:

Correlation analysis completed and plot saved to 'investment\_vs\_mortality\_correlation.png'.

Average Correlation: 0.3624883722747718

Descriptive Statistics for Correlation:

Mean Correlation: 0.3625

Median Correlation: 0.4394

Standard Deviation: 0.2427

Min Correlation: -0.1367

Max Correlation: 0.5842

# Code :

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Load the infrastructure performance and mortality data

infrastructure\_file = 'fspi.xlsx'

mortality\_file = 'fmortality.xlsx'

df\_infrastructure = pd.read\_excel(infrastructure\_file)

df\_mortality = pd.read\_excel(mortality\_file)

# Clean column names

df\_infrastructure.columns = df\_infrastructure.columns.str.strip()

df\_mortality.columns = df\_mortality.columns.str.strip()

# Drop unnecessary columns (e.g., 'Unnamed' columns)

df\_infrastructure = df\_infrastructure.loc[:, ~df\_infrastructure.columns.str.contains('^Unnamed')]

df\_mortality = df\_mortality.loc[:, ~df\_mortality.columns.str.contains('^Unnamed')]

# Extract year columns

year\_columns\_infrastructure = [col for col in df\_infrastructure.columns if col.startswith('YR')]

year\_columns\_mortality = [col for col in df\_mortality.columns if col.startswith('YR')]

# Create long format DataFrames

df\_infrastructure\_long = df\_infrastructure.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_infrastructure, var\_name='Year', value\_name='Infrastructure')

df\_mortality\_long = df\_mortality.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_mortality, var\_name='Year', value\_name='Mortality')

# Remove 'YR' prefix from 'Year' column and convert to numeric

df\_infrastructure\_long['Year'] = df\_infrastructure\_long['Year'].str.replace('YR', '').astype(int)

df\_mortality\_long['Year'] = df\_mortality\_long['Year'].str.replace('YR', '').astype(int)

# Merge the long format DataFrames on 'Country Name' and 'Year'

merged\_long = pd.merge(df\_infrastructure\_long, df\_mortality\_long, on=['Country Name', 'Year'])

# Calculate correlation for each year

correlation\_per\_year = merged\_long.groupby('Year').apply(lambda x: x[['Infrastructure', 'Mortality']].corr().iloc[0, 1])

# Plot the correlations over time

plt.figure(figsize=(12, 6))

plt.plot(correlation\_per\_year.index, correlation\_per\_year.values, marker='o', linestyle='-', color='b')

plt.xlabel('Year')

plt.ylabel('Correlation between Infrastructure and Mortality')

plt.title('Correlation between Infrastructure Performance and Road Traffic Mortality Rates Over Time')

plt.grid(True)

plt.tight\_layout()

# Save the plot

plt.savefig('infrastructure\_vs\_mortality\_correlation.png')

plt.show()

print("Correlation analysis completed and plot saved to 'infrastructure\_vs\_mortality\_correlation.png'.")

# Display average correlation

average\_correlation = correlation\_per\_year.mean()

print("Average Correlation:", average\_correlation)

# Descriptive statistics for correlations

stats = {

'Mean Correlation': average\_correlation,

'Median Correlation': correlation\_per\_year.median(),

'Standard Deviation': correlation\_per\_year.std(),

'Min Correlation': correlation\_per\_year.min(),

'Max Correlation': correlation\_per\_year.max()

}

print("\nDescriptive Statistics for Correlation:")

for key, value in stats.items():

print(f"{key}: {value:.4f}")

# output :

Correlation analysis completed and plot saved to 'infrastructure\_vs\_mortality\_correlation.png'.

Average Correlation: 0.15808626403756137

Descriptive Statistics for Correlation:

Mean Correlation: 0.1581

Median Correlation: 0.1419

Standard Deviation: 0.0705

Min Correlation: 0.0929

Max Correlation: 0.2556

# code:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Load the data from the three files

investment\_file = 'finvestment.xlsx'

mortality\_file = 'fmortality.xlsx'

infrastructure\_file = 'fspi.xlsx'

df\_investment = pd.read\_excel(investment\_file)

df\_mortality = pd.read\_excel(mortality\_file)

df\_infrastructure = pd.read\_excel(infrastructure\_file)

# Clean column names

df\_investment.columns = df\_investment.columns.str.strip()

df\_mortality.columns = df\_mortality.columns.str.strip()

df\_infrastructure.columns = df\_infrastructure.columns.str.strip()

# Drop unnecessary columns (e.g., 'Unnamed' columns)

df\_investment = df\_investment.loc[:, ~df\_investment.columns.str.contains('^Unnamed')]

df\_mortality = df\_mortality.loc[:, ~df\_mortality.columns.str.contains('^Unnamed')]

df\_infrastructure = df\_infrastructure.loc[:, ~df\_infrastructure.columns.str.contains('^Unnamed')]

# Extract year columns

year\_columns\_investment = [col for col in df\_investment.columns if col.startswith('YR')]

year\_columns\_mortality = [col for col in df\_mortality.columns if col.startswith('YR')]

year\_columns\_infrastructure = [col for col in df\_infrastructure.columns if col.startswith('YR')]

# Create long format DataFrames

df\_investment\_long = df\_investment.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_investment, var\_name='Year', value\_name='Investment')

df\_mortality\_long = df\_mortality.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_mortality, var\_name='Year', value\_name='Mortality')

df\_infrastructure\_long = df\_infrastructure.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_infrastructure, var\_name='Year', value\_name='Infrastructure')

# Remove 'YR' prefix from 'Year' column and convert to numeric

df\_investment\_long['Year'] = df\_investment\_long['Year'].str.replace('YR', '').astype(int)

df\_mortality\_long['Year'] = df\_mortality\_long['Year'].str.replace('YR', '').astype(int)

df\_infrastructure\_long['Year'] = df\_infrastructure\_long['Year'].str.replace('YR', '').astype(int)

# Descriptive statistics

def print\_descriptive\_stats(df\_long, column\_name):

stats = df\_long.groupby('Country Name')[column\_name].agg(['mean', 'median', 'std', 'min', 'max'])

print(f"\nDescriptive Statistics for {column\_name}:")

print(stats)

print\_descriptive\_stats(df\_investment\_long, 'Investment')

print\_descriptive\_stats(df\_mortality\_long, 'Mortality')

print\_descriptive\_stats(df\_infrastructure\_long, 'Infrastructure')

# Plot Investment, Mortality, and Infrastructure over time

fig, ax = plt.subplots(3, 1, figsize=(14, 18), sharex=True)

# Plot Investment over time

for country in df\_investment\_long['Country Name'].unique():

country\_data = df\_investment\_long[df\_investment\_long['Country Name'] == country]

ax[0].plot(country\_data['Year'], country\_data['Investment'], label=country)

ax[0].set\_title('Investment Over Time')

ax[0].set\_ylabel('Investment')

ax[0].legend(loc='upper right', bbox\_to\_anchor=(1.15, 1.02), title='Country')

ax[0].grid(True)

# Plot Mortality over time

for country in df\_mortality\_long['Country Name'].unique():

country\_data = df\_mortality\_long[df\_mortality\_long['Country Name'] == country]

ax[1].plot(country\_data['Year'], country\_data['Mortality'], label=country)

ax[1].set\_title('Mortality Over Time')

ax[1].set\_ylabel('Mortality')

ax[1].legend(loc='upper right', bbox\_to\_anchor=(1.15, 1.02), title='Country')

ax[1].grid(True)

# Plot Infrastructure over time

for country in df\_infrastructure\_long['Country Name'].unique():

country\_data = df\_infrastructure\_long[df\_infrastructure\_long['Country Name'] == country]

ax[2].plot(country\_data['Year'], country\_data['Infrastructure'], label=country)

ax[2].set\_title('Infrastructure Over Time')

ax[2].set\_ylabel('Infrastructure')

ax[2].set\_xlabel('Year')

ax[2].legend(loc='upper right', bbox\_to\_anchor=(1.15, 1.02), title='Country')

ax[2].grid(True)

plt.tight\_layout()

plt.savefig('time\_series\_plots.png')

plt.show()

print("Time series plots completed and saved to 'time\_series\_plots.png'.")

# output:

Descriptive Statistics for Investment:

mean median std min \

Country Name

brazil 4.966476e+09 1.328900e+09 9.581168e+09 0

british virgin islands 0.000000e+00 0.000000e+00 0.000000e+00 0

china 9.326610e+09 4.239520e+09 1.045499e+10 0

colombia 2.027042e+09 1.456220e+09 1.960259e+09 0

india 3.434780e+09 2.777385e+09 3.001404e+09 0

latin america & caribbean 6.134678e+09 1.178930e+09 1.142878e+10 0

mexico 8.427967e+08 6.093800e+08 1.221811e+09 0

south asia 3.742061e+09 2.827255e+09 3.185713e+09 0

turkiye 5.419265e+09 5.084000e+08 1.222945e+10 0

max

Country Name

brazil 33749230000

british virgin islands 0

china 28433590000

colombia 5998540000

india 9271640000

latin america & caribbean 39445840000

mexico 4603400000

south asia 9649170000

turkiye 43103200000

Descriptive Statistics for Mortality:

mean median std min max

Country Name

brazil 11.316667 16.550000 10.157830 0.0 23.600000

british virgin islands 0.000000 0.000000 0.000000 0.0 0.000000

china 10.775000 17.500000 9.565290 0.0 21.400000

colombia 10.200000 15.850000 9.074339 0.0 20.000000

india 9.183333 15.250000 8.117975 0.0 16.900000

latin america & caribbean 10.475106 17.363751 9.256994 0.0 19.001251

mexico 7.800000 12.900000 6.894135 0.0 14.200000

south asia 8.972203 15.085419 7.923575 0.0 15.937456

turkiye 5.200000 7.550000 4.659887 0.0 9.900000

Descriptive Statistics for Infrastructure:

mean median std min max

Country Name

brazil 38.333333 55.0 34.333480 0 75

british virgin islands 0.000000 0.0 0.000000 0 0

china 30.833333 47.5 27.620260 0 65

colombia 34.583333 47.5 32.365690 0 80

india 30.833333 50.0 27.289136 0 55

latin america & caribbean 0.000000 0.0 0.000000 0 0

mexico 41.666667 62.5 37.436815 0 75

south asia 0.000000 0.0 0.000000 0 0

turkiye 55.000000 82.5 49.036165 0 100

# code :

import pandas as pd

import matplotlib.pyplot as plt

# Load the data from the files

investment\_file = 'finvestment.xlsx'

mortality\_file = 'fmortality.xlsx'

df\_investment = pd.read\_excel(investment\_file)

df\_mortality = pd.read\_excel(mortality\_file)

# Clean column names

df\_investment.columns = df\_investment.columns.str.strip()

df\_mortality.columns = df\_mortality.columns.str.strip()

# Drop unnecessary columns (e.g., 'Unnamed' columns)

df\_investment = df\_investment.loc[:, ~df\_investment.columns.str.contains('^Unnamed')]

df\_mortality = df\_mortality.loc[:, ~df\_mortality.columns.str.contains('^Unnamed')]

# Extract year columns

year\_columns\_investment = [col for col in df\_investment.columns if col.startswith('YR')]

year\_columns\_mortality = [col for col in df\_mortality.columns if col.startswith('YR')]

# Create long format DataFrames

df\_investment\_long = df\_investment.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_investment, var\_name='Year', value\_name='Investment')

df\_mortality\_long = df\_mortality.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_mortality, var\_name='Year', value\_name='Mortality')

# Remove 'YR' prefix from 'Year' column and convert to numeric

df\_investment\_long['Year'] = df\_investment\_long['Year'].str.replace('YR', '').astype(int)

df\_mortality\_long['Year'] = df\_mortality\_long['Year'].str.replace('YR', '').astype(int)

# Calculate average values for each country, handling missing data

avg\_investment = df\_investment\_long.groupby('Country Name')['Investment'].mean()

avg\_mortality = df\_mortality\_long.groupby('Country Name')['Mortality'].mean()

# Merge the average values into a single DataFrame

comparison\_df = pd.DataFrame({

'Average Investment': avg\_investment,

'Average Mortality': avg\_mortality

}).reset\_index()

# Descriptive statistics

def print\_descriptive\_stats(df):

print("\nDescriptive Statistics:")

print(df.describe())

print("Descriptive Statistics for Average Investment:")

print\_descriptive\_stats(avg\_investment)

print("Descriptive Statistics for Average Mortality:")

print\_descriptive\_stats(avg\_mortality)

# Plot average investment and mortality

fig, ax = plt.subplots(figsize=(14, 8))

# Bar chart for Average Investment

bar\_width = 0.35

index = range(len(comparison\_df))

bars1 = ax.bar(index, comparison\_df['Average Investment'], bar\_width, label='Average Investment')

bars2 = ax.bar([i + bar\_width for i in index], comparison\_df['Average Mortality'], bar\_width, label='Average Mortality')

# Add labels and title

ax.set\_xlabel('Country')

ax.set\_ylabel('Average Value')

ax.set\_title('Comparison of Average Investment and Mortality Rates Across Countries')

ax.set\_xticks([i + bar\_width / 2 for i in index])

ax.set\_xticklabels(comparison\_df['Country Name'], rotation=90)

ax.legend()

plt.tight\_layout()

plt.savefig('average\_comparison.png')

plt.show()

# print("Bar charts completed and saved to 'average\_comparison.png'.") output:

Descriptive Statistics for Average Investment:

Descriptive Statistics:

count 9.000000e+00

mean 3.988190e+09

std 2.874962e+09

min 0.000000e+00

25% 2.027042e+09

50% 3.742061e+09

75% 5.419265e+09

max 9.326610e+09

Name: Investment, dtype: float64

Descriptive Statistics for Average Mortality:

Descriptive Statistics:

count 9.000000

mean 8.213590

std 3.596560

min 0.000000

25% 7.800000

50% 9.183333

75% 10.475106

max 11.316667

Name: Mortality, dtype: float64

# Code:

import pandas as pd

import matplotlib.pyplot as plt

# Load the data from the files

investment\_file = 'finvestment.xlsx'

mortality\_file = 'fmortality.xlsx'

infrastructure\_file = 'fspi.xlsx'

df\_investment = pd.read\_excel(investment\_file)

df\_mortality = pd.read\_excel(mortality\_file)

df\_infrastructure = pd.read\_excel(infrastructure\_file)

# Clean column names

df\_investment.columns = df\_investment.columns.str.strip()

df\_mortality.columns = df\_mortality.columns.str.strip()

df\_infrastructure.columns = df\_infrastructure.columns.str.strip()

# Drop unnecessary columns (e.g., 'Unnamed' columns)

df\_investment = df\_investment.loc[:, ~df\_investment.columns.str.contains('^Unnamed')]

df\_mortality = df\_mortality.loc[:, ~df\_mortality.columns.str.contains('^Unnamed')]

df\_infrastructure = df\_infrastructure.loc[:, ~df\_infrastructure.columns.str.contains('^Unnamed')]

# Extract year columns

year\_columns\_investment = [col for col in df\_investment.columns if col.startswith('YR')]

year\_columns\_mortality = [col for col in df\_mortality.columns if col.startswith('YR')]

year\_columns\_infrastructure = [col for col in df\_infrastructure.columns if col.startswith('YR')]

# Create long format DataFrames

df\_investment\_long = df\_investment.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_investment, var\_name='Year', value\_name='Investment')

df\_mortality\_long = df\_mortality.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_mortality, var\_name='Year', value\_name='Mortality')

df\_infrastructure\_long = df\_infrastructure.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_infrastructure, var\_name='Year', value\_name='Infrastructure')

# Remove 'YR' prefix from 'Year' column and convert to numeric

df\_investment\_long['Year'] = df\_investment\_long['Year'].str.replace('YR', '').astype(int)

df\_mortality\_long['Year'] = df\_mortality\_long['Year'].str.replace('YR', '').astype(int)

df\_infrastructure\_long['Year'] = df\_infrastructure\_long['Year'].str.replace('YR', '').astype(int)

# Merge the long format DataFrames on 'Country Name' and 'Year'

merged\_investment\_mortality = pd.merge(df\_investment\_long, df\_mortality\_long, on=['Country Name', 'Year'])

merged\_infrastructure\_mortality = pd.merge(df\_infrastructure\_long, df\_mortality\_long, on=['Country Name', 'Year'])

# Average values per country

avg\_investment = merged\_investment\_mortality.groupby('Country Name')['Investment'].mean()

avg\_mortality\_investment = merged\_investment\_mortality.groupby('Country Name')['Mortality'].mean()

avg\_infrastructure = merged\_infrastructure\_mortality.groupby('Country Name')['Infrastructure'].mean()

avg\_mortality\_infrastructure = merged\_infrastructure\_mortality.groupby('Country Name')['Mortality'].mean()

# Combine average values into DataFrames for plotting

df\_investment\_mortality = pd.DataFrame({

'Average Investment': avg\_investment,

'Average Mortality': avg\_mortality\_investment

}).reset\_index()

df\_infrastructure\_mortality = pd.DataFrame({

'Average Infrastructure': avg\_infrastructure,

'Average Mortality': avg\_mortality\_infrastructure

}).reset\_index()

# Print description of the data used for plotting

print("Average Investment vs Mortality Rates Data:")

print(df\_investment\_mortality.describe())

print("\nAverage Infrastructure vs Mortality Rates Data:")

print(df\_infrastructure\_mortality.describe())

# Create scatter plot for Investment vs Mortality

plt.figure(figsize=(16, 8))

plt.subplot(1, 2, 1)

plt.scatter(df\_investment\_mortality['Average Investment'], df\_investment\_mortality['Average Mortality'], alpha=0.7)

plt.xlabel('Average Investment')

plt.ylabel('Average Mortality')

plt.title('Investment vs Mortality Rates')

plt.grid(True)

# Create scatter plot for Infrastructure vs Mortality

plt.subplot(1, 2, 2)

plt.scatter(df\_infrastructure\_mortality['Average Infrastructure'], df\_infrastructure\_mortality['Average Mortality'], alpha=0.7, color='orange')

plt.xlabel('Average Infrastructure')

plt.ylabel('Average Mortality')

plt.title('Infrastructure Scores vs Mortality Rates')

plt.grid(True)

# Save and show plots

plt.tight\_layout()

plt.savefig('scatter\_plots.png')

plt.show()

print("Scatter plots completed and saved to 'scatter\_plots.png'.")

# OUTPUT:

Average Investment vs Mortality Rates Data:

Average Investment Average Mortality

count 9.000000e+00 9.000000

mean 3.988190e+09 8.213590

std 2.874962e+09 3.596560

min 0.000000e+00 0.000000

25% 2.027042e+09 7.800000

50% 3.742061e+09 9.183333

75% 5.419265e+09 10.475106

max 9.326610e+09 11.316667

Average Infrastructure vs Mortality Rates Data:

Average Infrastructure Average Mortality

count 9.000000 9.000000

mean 25.694444 8.213590

std 20.573378 3.596560

min 0.000000 0.000000

25% 0.000000 7.800000

50% 30.833333 9.183333

75% 38.333333 10.475106

max 55.000000 11.316667

# code :

import pandas as pd

import numpy as np

import statsmodels.api as sm

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error, r2\_score

# Load the data from the files

investment\_file = 'finvestment.xlsx'

mortality\_file = 'fmortality.xlsx'

infrastructure\_file = 'fspi.xlsx'

df\_investment = pd.read\_excel(investment\_file)

df\_mortality = pd.read\_excel(mortality\_file)

df\_infrastructure = pd.read\_excel(infrastructure\_file)

# Clean column names

df\_investment.columns = df\_investment.columns.str.strip()

df\_mortality.columns = df\_mortality.columns.str.strip()

df\_infrastructure.columns = df\_infrastructure.columns.str.strip()

# Drop unnecessary columns (e.g., 'Unnamed' columns)

df\_investment = df\_investment.loc[:, ~df\_investment.columns.str.contains('^Unnamed')]

df\_mortality = df\_mortality.loc[:, ~df\_mortality.columns.str.contains('^Unnamed')]

df\_infrastructure = df\_infrastructure.loc[:, ~df\_infrastructure.columns.str.contains('^Unnamed')]

# Extract year columns

year\_columns\_investment = [col for col in df\_investment.columns if col.startswith('YR')]

year\_columns\_mortality = [col for col in df\_mortality.columns if col.startswith('YR')]

year\_columns\_infrastructure = [col for col in df\_infrastructure.columns if col.startswith('YR')]

# Create long format DataFrames

df\_investment\_long = df\_investment.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_investment, var\_name='Year', value\_name='Investment')

df\_mortality\_long = df\_mortality.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_mortality, var\_name='Year', value\_name='Mortality')

df\_infrastructure\_long = df\_infrastructure.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_infrastructure, var\_name='Year', value\_name='Infrastructure')

# Remove 'YR' prefix from 'Year' column and convert to numeric

df\_investment\_long['Year'] = df\_investment\_long['Year'].str.replace('YR', '').astype(int)

df\_mortality\_long['Year'] = df\_mortality\_long['Year'].str.replace('YR', '').astype(int)

df\_infrastructure\_long['Year'] = df\_infrastructure\_long['Year'].str.replace('YR', '').astype(int)

# Merge the long format DataFrames on 'Country Name' and 'Year'

merged\_df = pd.merge(df\_investment\_long, df\_mortality\_long, on=['Country Name', 'Year'])

merged\_df = pd.merge(merged\_df, df\_infrastructure\_long, on=['Country Name', 'Year'])

# Pivot data to get average values per country

average\_df = merged\_df.groupby('Country Name').agg({

'Investment': 'mean',

'Mortality': 'mean',

'Infrastructure': 'mean'

}).reset\_index()

# Define features and target variable

X = average\_df[['Investment', 'Infrastructure']]

y = average\_df['Mortality']

# Add constant term for intercept

X = sm.add\_constant(X)

# Perform regression analysis using statsmodels

model = sm.OLS(y, X).fit()

print(model.summary())

# For scikit-learn

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Linear Regression Model

lin\_reg = LinearRegression()

lin\_reg.fit(X\_train, y\_train)

# Predictions

y\_pred = lin\_reg.predict(X\_test)

# Model Evaluation

print("Mean Squared Error:", mean\_squared\_error(y\_test, y\_pred))

print("R^2 Score:", r2\_score(y\_test, y\_pred))

# OUTPUT:

OLS Regression Results

==============================================================================

Dep. Variable: Mortality R-squared: 0.332

Model: OLS Adj. R-squared: 0.109

Method: Least Squares F-statistic: 1.491

Date: Sun, 21 Jul 2024 Prob (F-statistic): 0.298

Time: 18:58:32 Log-Likelihood: -21.945

No. Observations: 9 AIC: 49.89

Df Residuals: 6 BIC: 50.48

Df Model: 2

Covariance Type: nonrobust

==================================================================================

coef std err t P>|t| [0.025 0.975]

----------------------------------------------------------------------------------

const 4.9534 2.387 2.075 0.083 -0.887 10.794

Investment 6.861e-10 4.21e-10 1.628 0.155 -3.45e-10 1.72e-09

Infrastructure 0.0204 0.059 0.346 0.741 -0.124 0.164

==============================================================================

Omnibus: 2.247 Durbin-Watson: 1.794

Prob(Omnibus): 0.325 Jarque-Bera (JB): 1.383

Skew: -0.880 Prob(JB): 0.501

Kurtosis: 2.229 Cond. No. 1.02e+10

==============================================================================

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 1.02e+10. This might indicate that there are

strong multicollinearity or other numerical problems.

Mean Squared Error: 68.27654602015656

R^2 Score: -2.392605436244054

# Code:

import pandas as pd

import numpy as np

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

# Load the data from the files

investment\_file = 'finvestment.xlsx'

mortality\_file = 'fmortality.xlsx'

infrastructure\_file = 'fspi.xlsx'

df\_investment = pd.read\_excel(investment\_file)

df\_mortality = pd.read\_excel(mortality\_file)

df\_infrastructure = pd.read\_excel(infrastructure\_file)

# Clean column names

df\_investment.columns = df\_investment.columns.str.strip()

df\_mortality.columns = df\_mortality.columns.str.strip()

df\_infrastructure.columns = df\_infrastructure.columns.str.strip()

# Drop unnecessary columns (e.g., 'Unnamed' columns)

df\_investment = df\_investment.loc[:, ~df\_investment.columns.str.contains('^Unnamed')]

df\_mortality = df\_mortality.loc[:, ~df\_mortality.columns.str.contains('^Unnamed')]

df\_infrastructure = df\_infrastructure.loc[:, ~df\_infrastructure.columns.str.contains('^Unnamed')]

# Extract year columns

year\_columns\_investment = [col for col in df\_investment.columns if col.startswith('YR')]

year\_columns\_mortality = [col for col in df\_mortality.columns if col.startswith('YR')]

year\_columns\_infrastructure = [col for col in df\_infrastructure.columns if col.startswith('YR')]

# Create long format DataFrames

df\_investment\_long = df\_investment.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_investment, var\_name='Year', value\_name='Investment')

df\_mortality\_long = df\_mortality.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_mortality, var\_name='Year', value\_name='Mortality')

df\_infrastructure\_long = df\_infrastructure.melt(id\_vars=['Country Name'], value\_vars=year\_columns\_infrastructure, var\_name='Year', value\_name='Infrastructure')

# Remove 'YR' prefix from 'Year' column and convert to numeric

df\_investment\_long['Year'] = df\_investment\_long['Year'].str.replace('YR', '').astype(int)

df\_mortality\_long['Year'] = df\_mortality\_long['Year'].str.replace('YR', '').astype(int)

df\_infrastructure\_long['Year'] = df\_infrastructure\_long['Year'].str.replace('YR', '').astype(int)

# Merge the long format DataFrames on 'Country Name' and 'Year'

merged\_df = pd.merge(df\_investment\_long, df\_mortality\_long, on=['Country Name', 'Year'])

merged\_df = pd.merge(merged\_df, df\_infrastructure\_long, on=['Country Name', 'Year'])

# Pivot data to get average values per country

average\_df = merged\_df.groupby('Country Name').agg({

'Investment': 'mean',

'Mortality': 'mean',

'Infrastructure': 'mean'

}).reset\_index()

# Print statistics about the average data used for PCA

print("Average Data Used for PCA:")

print(average\_df.describe())

# Prepare data for PCA

X = average\_df[['Investment', 'Infrastructure']]

y = average\_df['Mortality']

# Standardize the data

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Apply PCA

pca = PCA(n\_components=2)

principal\_components = pca.fit\_transform(X\_scaled)

# Create DataFrame for PCA results

pca\_df = pd.DataFrame(data=principal\_components, columns=['Principal Component 1', 'Principal Component 2'])

pca\_df['Country Name'] = average\_df['Country Name']

pca\_df['Mortality'] = y

# Plot PCA results

plt.figure(figsize=(12, 6))

plt.scatter(pca\_df['Principal Component 1'], pca\_df['Principal Component 2'], c=pca\_df['Mortality'], cmap='viridis', s=100)

plt.colorbar(label='Mortality Rate')

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.title('PCA of Transport Investment and Infrastructure Performance')

plt.grid(True)

plt.tight\_layout()

# Save the plot

plt.savefig('pca\_analysis.png')

plt.show()

# Print explained variance ratio

print("Explained Variance Ratio:", pca.explained\_variance\_ratio\_)

# Print the PCA components

print("\nPCA Components:")

print(pd.DataFrame(pca.components\_, columns=['Investment', 'Infrastructure'], index=['PC1', 'PC2']))

OUTPUT:  
Average Data Used for PCA:

Investment Mortality Infrastructure

count 9.000000e+00 9.000000 9.000000

mean 3.988190e+09 8.213590 25.694444

std 2.874962e+09 3.596560 20.573378

min 0.000000e+00 0.000000 0.000000

25% 2.027042e+09 7.800000 0.000000

50% 3.742061e+09 9.183333 30.833333

75% 5.419265e+09 10.475106 38.333333

max 9.326610e+09 11.316667 55.000000

Explained Variance Ratio: [0.56854329 0.43145671]

PCA Components:

Investment Infrastructure

PC1 -0.707107 -0.707107

PC2 0.707107 -0.707107

[ ]: