AMS 597 Project

Team : 11

2025-04-15

Decoding CO2: A Statistical Dive into Global Emissions

Source of Dataset: https://github.com/owid/co2-data/blob/master/owid-co2-data.csv

Data Preprocessing and Cleaning

This code performs comprehensive preprocessing of the World Bank's CO2 emissions dataset to prepare it for meaningful time series and regression-based analyses. The following steps are taken:

1. DATA LOADING & FILTERING:

- Load CSV containing global CO2 and economic data.
- Retain only post-1981 data to avoid early extrapolations and focus on modern economic dynamics.
- Drop rows with missing GDP, CO2, CO2 growth %, and co2_per_unit_energy values as these are key features of interest and not safe to impute.

2. DROP IRRELEVANT/DERIVABLE COLUMNS:

- Remove predefined interaction terms such as:
  a. 'share' columns (country's global CO2 share),
  b. 'per_capita' metrics (can be derived from raw data),
  c. 'cumulative', 'including', and 'temperature' columns,
  d. other greenhouse gases and trade-related columns which have many missing values or low relevance.
- These steps help avoid multicollinearity, high missingness, and model complexity.

3. INCOME LEVEL ENCODING:

- Add a categorical variable `income_level` for each country (Low = 1, Middle = 2, High = 3) based on World Bank classification.
- Countries not explicitly classified are defaulted to Low income for continuity.

4. CONTINENT ENCODING:

- Add a `continent` factor variable to allow grouping or analysis by geography.
- Countries are mapped to 6 regions: Africa, Asia, Europe, North America, South America, and Oceania.

5. TIME TRANSFORMATION:

- Transform `year` into an index variable relative to 1982 (e.g., 1982 = 0), to support modeling that assumes time starts at zero.

6. IMPUTATION FOR MINOR MISSINGNESS:

- Impute missing values in `cement_co2` and `land_use_change_co2` using linear regression, since each had <5% missing values, making regression-based imputation justifiable.

7. SCALING:

- Scale `gdp` (to trillions) and `population` (to millions) for numerical stability in models.

8. FINALIZATION:

- Restore the `country` column for use in visualization or summaries.
- Filter the dataset to only include numeric and factor variables, stored in `df`.
- `df.labeled` retains country names for labeling purposes during EDA.

```
#Read csv data
data <- read.csv('visualizing_global_co2_data.csv')

#Remove data from before 1982, as it becomes less robust in accuracy and
#more of extrapolation
data <- data[data$year > 1981,]

#Given that we are going to be using gdp or co2 as a response variable,
#it wouldn't be correct to impute missing gdp/co2 values.
#Luckily, the amount of removed rows isn't very large
data <- data[!is.na(data$gdp),]
data <- data[!is.na(data$co2),]
data <- data[!is.na(data$co2_growth_prct),]

#This column is a predefined interaction term.
#There are 60 missing values, so we chose to get rid of these
data <- data[!is.na(data$co2_per_unit_energy),]

#We also decided to remove aggregate data from the dataset,
#as we are interested in predicting individual countries
data <- data[!(data$country == 'World'),]

cat('list of the missing values')
```

```
## list of the missing values
```

```
colnames(data)[apply(data,2, anyNA)]
```

```
##  [1] "cement_co2"
##  [2] "cement_co2_per_capita"
##  [3] "co2_including_luc"
```

```
##  [4] "co2_including_luc_growth_abs"
##  [5] "co2_including_luc_growth_prct"
##  [6] "co2_including_luc_per_capita"
##  [7] "co2_including_luc_per_gdp"
##  [8] "co2_including_luc_per_unit_energy"
##  [9] "consumption_co2"
## [10] "consumption_co2_per_capita"
## [11] "consumption_co2_per_gdp"
## [12] "cumulative_cement_co2"
## [13] "cumulative_co2_including_luc"
## [14] "cumulative_luc_co2"
## [15] "cumulative_other_co2"
## [16] "ghg_excluding_lucf_per_capita"
## [17] "ghg_per_capita"
## [18] "land_use_change_co2"
## [19] "land_use_change_co2_per_capita"
## [20] "methane"
## [21] "methane_per_capita"
## [22] "nitrous_oxide"
## [23] "nitrous_oxide_per_capita"
## [24] "other_co2_per_capita"
## [25] "other_industry_co2"
## [26] "share_global_cement_co2"
## [27] "share_global_co2_including_luc"
## [28] "share_global_cumulative_cement_co2"
## [29] "share_global_cumulative_co2_including_luc"
## [30] "share_global_cumulative_luc_co2"
## [31] "share_global_cumulative_other_co2"
## [32] "share_global_luc_co2"
## [33] "share_global_other_co2"
## [34] "temperature_change_from_ch4"
## [35] "temperature_change_from_n2o"
## [36] "total_ghg"
## [37] "total_ghg_excluding_lucf"
## [38] "trade_co2"
## [39] "trade_co2_share"
```

```r
#Below, we begin to remove a number of precalculated action terms.
#Not only are there a lot of missing values in each of these columns,
#but we can also recalculate them alter if we choose to do so
x <- colnames(data)

#This set of interactions is country's share of the global value as a percentage
#This can easily be added back later if necessary, and we chose to remove them
drop <- subset(x, grepl('share', x))

data <- data[, !colnames(data) %in% drop]

#The next interaction is per capita interactions, which again we can calculate later
drop <- subset(x, grepl('per_capita', x))

data <- data[, !colnames(data) %in% drop]

#This list consists of variables we aren't interest in interpreting.
```

```r
#Mainly they are other greenhouse gasses and temperature changes
#The only added value is trade co2.
#Not only is it 50% missing values, removing it would help remove colinearity from the data
drop <- c(
          "nitrous_oxide",
          "methane",
          "other_industry_co2",
          "consumption_co2_per_gdp",
          "consumption_co2",
          "total_ghg",
          "total_ghg_excluding_lucf",
          "trade_co2"
          )
data <- data[, !colnames(data) %in% drop]

#This interaction is historical cumulative information.
#This had a lot of missing values in the dataset, and thus we chose to remove them
x <- colnames(data)

drop <- subset(x, grepl('cumulative', x))
data <- data[, !colnames(data) %in% drop]

#These values are summations of two other values, so we decided to remove them as they are colinear
x <- colnames(data)

#This gets rid of co2 data including
drop <- subset(x, grepl('including', x))
data <- data[, !colnames(data) %in% drop]

#Lastly, we removed temperature data, as it didn't have much to do with what we wanted to study
x <- colnames(data)

drop <- subset(x, grepl('temperature', x))
data <- data[, !colnames(data) %in% drop]

cat('columns with missing values \n')
```

```
## columns with missing values
```

```r
colnames(data)[apply(data,2, anyNA)]
```

```
## [1] "cement_co2"          "land_use_change_co2"
```

```r
cat('counts of missing values \n')
```

```
## counts of missing values
```

```r
colSums(is.na(data))
```

```
##                     country                        year
##                           0                           0
```

```
##                   iso_code                 population
##                          0                          0
##                        gdp                cement_co2
##                          0                         90
##                        co2             co2_growth_abs
##                          0                          0
##           co2_growth_prct                co2_per_gdp
##                          0                          0
##       co2_per_unit_energy                   coal_co2
##                          0                          0
##            energy_per_gdp                flaring_co2
##                          0                          0
##                   gas_co2        land_use_change_co2
##                          0                         37
##                   oil_co2 primary_energy_consumption
##                          0                          0
```

```r
#the below code adds an income level categorical value to each country, as described by world bank
income_level <- matrix(0, nrow = nrow(data), 1)[,1]
data$income_level <- income_level

data2 <- data

low_income_countries <- c(
  "Algeria", "Angola", "Benin", "Botswana", "Burkina Faso", "Burundi",
  "Cabo Verde", "Cameroon", "Central African Republic", "Chad", "Comoros",
  "Congo", "Congo", "Djibouti", "Egypt",
  "Equatorial Guinea", "Eritrea", "Eswatini", "Ethiopia", "Gabon",
  "Gambia", "Ghana", "Guinea", "Guinea-Bissau", "Cote d'Ivoire", "Kenya",
  "Lesotho", "Liberia", "Libya", "Madagascar", "Malawi", "Mali",
  "Mauritania", "Mauritius", "Morocco", "Mozambique", "Namibia", "Niger",
  "Nigeria", "Rwanda", "Sao Tome and Principe", "Senegal", "Seychelles",
  "Sierra Leone", "Somalia", "South Africa", "South Sudan", "Sudan",
  "Tanzania", "Togo", "Tunisia", "Uganda", "Zambia", "Zimbabwe", " China", "India"
)

middle_income_countries <- c(
  "Saudi Arabia", "Russia", "United States", "Iran", "Iraq",
  "Kuwait", "Venezuela", "Mexico", "Libya", "Nigeria",
  "United Arab Emirates", "Indonesia", "China", "Canada",
  "Algeria", "Egypt", "United Kingdom", "Qatar", "Brazil", "Romania", "Poland",
  "Ukraine", "Russia", "Ukraine", "Belarus", "Uzbekistan", "Kazakhstan", "Georgia",
  "Azerbaijan", "Lithuania", "Moldova", "Latvia", "Kyrgyzstan", "Tajikistan",
  "Armenia", "Turkmenistan", "Estonia", "Argentina", "Brazil", "Chile", "Colombia",
  "Mexico", "Peru", "Ecuador", "Paraguay", "Uruguay", "Venezuela",

  # Asia
  "Thailand", "Philippines", "Sri Lanka", "India", "Pakistan", "Indonesia", "Iran", "Iraq", "Jordan",

  # Africa
  "Egypt", "Algeria", "Morocco", "Tunisia", "Ivory Coast", "Kenya", "Ghana", "Nigeria", "Zimbabwe",

  # Eastern Europe / Soviet Bloc
  "Poland", "Hungary", "Czechoslovakia", "Romania", "Bulgaria", "East Germany", "Ukraine", "Belarus",
```

```
  # Others
  "Turkey", "Yugoslavia"
)

high_income_countries <- c(
  "United States", "Canada", "United Kingdom", "France", "Germany",
  "Italy", "Belgium", "Netherlands", "Luxembourg", "Denmark", "Norway",
  "Iceland", "Portugal", "Spain", "Greece", "Turkey", "Finland", "Singapore",

  # Other Western-aligned developed nations
  "Australia", "New Zealand", "Japan", "Austria", "Switzerland", "Ireland",
  "Israel", "South Korea", "Taiwan", "Sweden"
)

data2$income_level[data$country %in% low_income_countries] <- 1
data2$income_level[data$country %in% middle_income_countries] <- 2
data2$income_level[data$country %in% high_income_countries] <- 3
data2$income_level[data2$income_level == 0] <- 1
data2$income_level <- as.factor(data2$income_level)

#The below code adds a continent categorical variable

africa <- c(
  "Algeria", "Angola", "Benin", "Botswana", "Burkina Faso", "Burundi",
  "Cabo Verde", "Cameroon", "Central African Republic", "Chad", "Comoros",
  "Democratic Republic of Congo", "Congo", "Djibouti", "Egypt", "Equatorial Guinea",
  "Eritrea", "Eswatini", "Ethiopia", "Gabon", "Gambia", "Ghana", "Guinea",
  "Guinea-Bissau", "Cote d'Ivoire", "Kenya", "Lesotho", "Liberia", "Libya",
  "Madagascar", "Malawi", "Mali", "Mauritania", "Mauritius", "Morocco",
  "Mozambique", "Namibia", "Niger", "Nigeria", "Rwanda", "Sao Tome and Principe",
  "Senegal", "Seychelles", "Sierra Leone", "Somalia", "South Africa", "South Sudan",
  "Sudan", "Tanzania", "Togo", "Tunisia", "Uganda", "Zambia", "Zimbabwe", "Cape Verde"
)

# Asia
asia <- c(
  "Afghanistan", "Armenia", "Azerbaijan", "Bahrain", "Bangladesh", "Bhutan",
  "Brunei", "Cambodia", "China", "Cyprus", "Georgia", "India", "Indonesia",
  "Iran", "Iraq", "Israel", "Japan", "Jordan", "Kazakhstan", "Kuwait",
  "Kyrgyzstan", "Laos", "Lebanon", "Malaysia", "Maldives", "Mongolia",
  "Myanmar", "Nepal", "North Korea", "Oman", "Pakistan", "Palestine",
  "Philippines", "Qatar", "Saudi Arabia", "Singapore", "South Korea", "Sri Lanka",
  "Syria", "Taiwan", "Tajikistan", "Thailand", "Timor-Leste", "Turkey",
  "Turkmenistan", "United Arab Emirates", "Uzbekistan", "Vietnam", "Yemen", "Hong Kong"
)

# Europe
europe <- c(
  "Albania", "Andorra", "Austria", "Belarus", "Belgium", "Bosnia and Herzegovina",
  "Bulgaria", "Croatia", "Czech Republic", "Denmark", "Estonia", "Finland",
  "France", "Germany", "Greece", "Hungary", "Iceland", "Ireland", "Italy",
  "Kosovo", "Latvia", "Liechtenstein", "Lithuania", "Luxembourg", "Malta",
  "Moldova", "Monaco", "Montenegro", "Netherlands", "North Macedonia", "Norway",
```

```
  "Poland", "Portugal", "Romania", "Russia", "San Marino", "Serbia", "Slovakia",
  "Slovenia", "Spain", "Sweden", "Switzerland", "Ukraine", "United Kingdom", "Czechia",
  "Vatican City"
)

# North America
north_america <- c(
  "Antigua and Barbuda", "Bahamas", "Barbados", "Belize", "Canada", "Costa Rica",
  "Cuba", "Dominica", "Dominican Republic", "El Salvador", "Grenada", "Guatemala",
  "Haiti", "Honduras", "Jamaica", "Mexico", "Nicaragua", "Panama",
  "Saint Kitts and Nevis", "Saint Lucia", "Saint Vincent and the Grenadines",
  "Trinidad and Tobago", "United States"
)

# South America
south_america <- c(
  "Argentina", "Bolivia", "Brazil", "Chile", "Colombia", "Ecuador",
  "Guyana", "Paraguay", "Peru", "Suriname", "Uruguay", "Venezuela"
)

# Oceania
oceania <- c(
  "Australia", "Fiji", "Kiribati", "Marshall Islands", "Micronesia",
  "Nauru", "New Zealand", "Palau", "Papua New Guinea", "Samoa",
  "Solomon Islands", "Tonga", "Tuvalu", "Vanuatu"
)

data2$continent <- matrix(0, nrow(data2), 1)[,1]
data2$continent[data2$country %in% africa] <- "Africa"
data2$continent[data2$country %in% asia] <- "Asia"
data2$continent[data2$country %in% europe] <- "Europe"
data2$continent[data2$country %in% north_america] <- "North America"
data2$continent[data2$country %in% south_america] <- "South America"
data2$continent[data2$country %in% oceania] <- "Oceania"

#We set the added variables as factor,
#so that the model doesn't mistake income level for a continuous variable
data2$continent <- as.factor(data2$continent)
data2$income_level <- as.factor(data2$income_level)

year.1982 <- data2$year - 1982
data2$year <- year.1982

country <- data2$country

df <- data2[, sapply(data2, function(x) is.numeric(x) || is.factor(x))]


#Lastly, we have two values we imputed using linear regression.
#Both missing values consisted of less than 5% of the total data lest,
#thus we believed this was an acceptable course of action

#Cement co2 Imputation
```

```r
impute.cement <- lm(cement_co2 ~ ., data = df, na.action = na.exclude)
missing_row <- is.na(df$cement_co2)
df$cement_co2[missing_row] <- predict(impute.cement, newdata = df[missing_row, ])

#Land co2 Imputation
impute.land <- lm(land_use_change_co2 ~ ., data = df, na.action = na.exclude)
missing_row <- is.na(df$land_use_change_co2)
df$land_use_change_co2[missing_row] <- predict(impute.land, newdata = df[missing_row,])

#Scale the population and gdp values to be in line with the rest of the dataset
df$gdp <- df$gdp / 10000000000
df$population <- (df$population) / 1000000

#Lastly, we added back country labels for viewing/summarization
df.labeled <- df
df.labeled$country <- country
cat('final labeled dataset: \n')
```

```
## final labeled dataset:
```

```r
head(df.labeled)
```

```
##     year population      gdp cement_co2   co2 co2_growth_abs co2_growth_prct
## 133    0  10.088290 1.598041      0.039 2.095          0.116           5.869
## 134    1   9.951447 1.675533      0.006 2.520          0.425          20.308
## 135    2  10.243689 1.707215      0.048 2.822          0.302          11.968
## 136    3  10.512220 1.710848      0.032 3.501          0.680          24.096
## 137    4  10.448447 1.764135      0.038 3.134         -0.368         -10.504
## 138    5  10.322767 1.581082      0.043 3.114         -0.020          -0.632
##     co2_per_gdp co2_per_unit_energy coal_co2 energy_per_gdp flaring_co2 gas_co2
## 133       0.131               0.224    0.385          0.585       0.282   0.396
## 134       0.150               0.220    0.385          0.683       0.293   0.616
## 135       0.165               0.246    0.393          0.673       0.316   0.932
## 136       0.205               0.312    0.400          0.656       0.330   1.192
## 137       0.178               0.273    0.425          0.650       0.330   1.202
## 138       0.197               0.168    0.443          1.169       0.223   0.392
##     land_use_change_co2 oil_co2 primary_energy_consumption income_level
## 133               2.711   0.993                      9.348            1
## 134               2.858   1.220                     11.436            1
## 135               2.345   1.134                     11.489            1
## 136               1.795   1.548                     11.217            1
## 137               1.795   1.140                     11.462            1
## 138               1.246   2.013                     18.483            1
##     continent     country
## 133      Asia Afghanistan
## 134      Asia Afghanistan
## 135      Asia Afghanistan
## 136      Asia Afghanistan
## 137      Asia Afghanistan
## 138      Asia Afghanistan
```

```r
cat("Dimensions of Cleaned Dataset:", dim(df.labeled), "\n")
```

```
## Dimensions of Cleaned Dataset: 5859 19
```

Model-1 Linear Regression(logGDP)

Research Question-1

How do environmental and demographic factors influence economic performance, as measured by GDP, when expressed in relative (percentage) terms?

Methodological Justification:

A linear regression model is particularly well-suited for your research question for several compelling reasons:

The research question specifically asks about relative (percentage) terms, and the log transformation of GDP perfectly accommodates this requirement. In a log-linear model, coefficients are interpreted as percentage changes in GDP associated with unit changes in predictors, providing the relative measures we want to seek.

Second, the scatter plot shows a predominantly linear relationship between actual and predicted log(GDP) values, confirming that a linear form is appropriate for modeling this relationship. The positive correlation displayed in the plot aligns with economic theory that connects environmental and demographic factors to economic output.

Third, linear regression's ability to handle multiple predictors simultaneously through your backward elimination approach (as shown in your previous AIC results) allows you to distinguish the relative importance of various environmental factors (CO2 emissions, energy consumption) and demographic variables (population) on economic performance

```r
if (!require(caret)) install.packages("caret")
```

```
## Loading required package: caret
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```r
library(caret)

if (!require(ggplot2)) install.packages("ggplot2")
library(ggplot2)
```

Log-Transforming GDP and Splitting the Dataset

GDP is log-transformed to stabilize variance and allow interpretation of model coefficients in percentage terms. The dataset is then split into a training set (70%) and testing set (30%) to train and evaluate the model fairly.

A multiple linear regression model is fitted using all predictors to explain log(GDP). Stepwise backward elimination is applied to improve model performance by removing statistically insignificant variables.

Cook's Distance is calculated to identify influential data points that may disproportionately affect the model. Any observations with a value greater than 1 are flagged for inspection.

The final model is used to predict log(GDP) on the test set. Model performance is evaluated using RMSE, MAE, and R-squared—providing insights into prediction accuracy and explained variance.

```
## REGRESSION ON LOG(GDP)

# Create the log-transformed GDP variable
df$log_gdp <- log(df$gdp)

# Partition the data (70% training, 30% testing) using log(GDP) as the response
set.seed(123)
trainIndex <- createDataPartition(df$log_gdp, p = 0.70, list = FALSE)
train_data <- df[trainIndex, ]
test_data  <- df[-trainIndex, ]

# Fit linear regression on log(GDP) using all available predictors
lm_model <- lm(log_gdp ~ ., data = train_data)
lm_model.step <- step(lm_model, direction = 'backward')
```

```
## Start:  AIC=1119.17
## log_gdp ~ year + population + gdp + cement_co2 + co2 + co2_growth_abs +
##     co2_growth_prct + co2_per_gdp + co2_per_unit_energy + coal_co2 +
##     energy_per_gdp + flaring_co2 + gas_co2 + land_use_change_co2 +
##     oil_co2 + primary_energy_consumption + income_level + continent
##
##                              Df Sum of Sq    RSS    AIC
## - gdp                         1      0.55 5327.6 1117.6
## <none>                                    5327.0 1119.2
## - co2_growth_abs              1      2.91 5329.9 1119.4
## - primary_energy_consumption  1      3.95 5330.9 1120.2
## - co2_per_unit_energy         1     11.87 5338.9 1126.3
## - co2_growth_prct             1     26.83 5353.8 1137.8
## - co2_per_gdp                 1     33.09 5360.1 1142.6
## - coal_co2                    1     33.69 5360.7 1143.0
## - co2                         1     39.53 5366.5 1147.5
## - gas_co2                     1     40.36 5367.4 1148.1
## - oil_co2                     1     42.67 5369.7 1149.9
## - cement_co2                  1     77.84 5404.8 1176.7
## - flaring_co2                 1    101.60 5428.6 1194.7
## - land_use_change_co2         1    106.34 5433.3 1198.3
## - energy_per_gdp              1    161.36 5488.4 1239.6
## - population                  1    279.47 5606.5 1327.0
## - year                        1    316.70 5643.7 1354.1
## - continent                   5    835.76 6162.8 1707.1
## - income_level                2   1356.18 6683.2 2045.8
##
## Step:  AIC=1117.6
## log_gdp ~ year + population + cement_co2 + co2 + co2_growth_abs +
##     co2_growth_prct + co2_per_gdp + co2_per_unit_energy + coal_co2 +
##     energy_per_gdp + flaring_co2 + gas_co2 + land_use_change_co2 +
##     oil_co2 + primary_energy_consumption + income_level + continent
##
##                              Df Sum of Sq    RSS    AIC
## - co2_growth_abs              1      2.58 5330.1 1117.6
## <none>                                    5327.6 1117.6
```

```
## - primary_energy_consumption  1      3.40 5331.0 1118.2
## - co2_per_unit_energy         1     12.08 5339.6 1124.9
## - co2_growth_prct             1     26.82 5354.4 1136.2
## - coal_co2                    1     33.29 5360.8 1141.2
## - co2_per_gdp                 1     33.33 5360.9 1141.2
## - co2                         1     39.04 5366.6 1145.6
## - gas_co2                     1     39.97 5367.5 1146.3
## - oil_co2                     1     42.14 5369.7 1147.9
## - cement_co2                  1     77.48 5405.0 1174.8
## - flaring_co2                 1    101.06 5428.6 1192.7
## - land_use_change_co2         1    107.94 5435.5 1197.9
## - energy_per_gdp              1    160.82 5488.4 1237.6
## - population                  1    321.78 5649.3 1356.2
## - year                        1    325.53 5653.1 1359.0
## - continent                   5    835.60 6163.2 1705.4
## - income_level                2   1355.81 6683.4 2043.9
##
## Step:  AIC=1117.58
## log_gdp ~ year + population + cement_co2 + co2 + co2_growth_prct +
##     co2_per_gdp + co2_per_unit_energy + coal_co2 + energy_per_gdp +
##     flaring_co2 + gas_co2 + land_use_change_co2 + oil_co2 + primary_energy_consumption +
##     income_level + continent
##
##                              Df Sum of Sq    RSS    AIC
## <none>                                    5330.1 1117.6
## - primary_energy_consumption  1      4.24 5334.4 1118.8
## - co2_per_unit_energy         1     11.74 5341.9 1124.6
## - co2_growth_prct             1     29.19 5359.3 1138.0
## - coal_co2                    1     31.67 5361.8 1139.9
## - co2_per_gdp                 1     32.20 5362.3 1140.3
## - co2                         1     37.43 5367.6 1144.3
## - gas_co2                     1     38.34 5368.5 1145.0
## - oil_co2                     1     40.35 5370.5 1146.5
## - cement_co2                  1     75.15 5405.3 1173.0
## - flaring_co2                 1     98.58 5428.7 1190.8
## - land_use_change_co2         1    106.74 5436.9 1196.9
## - energy_per_gdp              1    159.14 5489.3 1236.3
## - population                  1    319.21 5649.3 1354.2
## - year                        1    324.59 5654.7 1358.1
## - continent                   5    835.10 6165.2 1704.8
## - income_level                2   1353.26 6683.4 2041.9
```

```
# To calculate influential points
cooks_distance <- cooks.distance(lm_model.step)
influential <- which(cooks_distance > 4/nrow(train_data))
cat('Influential Datapoints: \n')
```

```
## Influential Datapoints:
```

```
# Predict on the test set
test_predictions <- predict(lm_model.step, newdata = test_data)

# Evaluate model performance (all metrics are computed on the log scale)
```

```r
rmse_value <- sqrt(mean((test_data$log_gdp - test_predictions)^2))
mae_value  <- mean(abs(test_data$log_gdp - test_predictions))
r2_value   <- cor(test_data$log_gdp, test_predictions)^2

cat("\nRegression on log(GDP) Evaluation Metrics:\n")
```

```
##
## Regression on log(GDP) Evaluation Metrics:
```

```r
cat("-----------------------------------------------------\n")
```

```
## -----------------------------------------------------
```

```r
cat("RMSE:", round(rmse_value, 3), "\n")
```

```
## RMSE: 1.138
```

```r
cat("MAE :", round(mae_value, 3), "\n")
```

```
## MAE : 0.869
```

```r
cat("R-squared:", round(r2_value, 3), "\n")
```

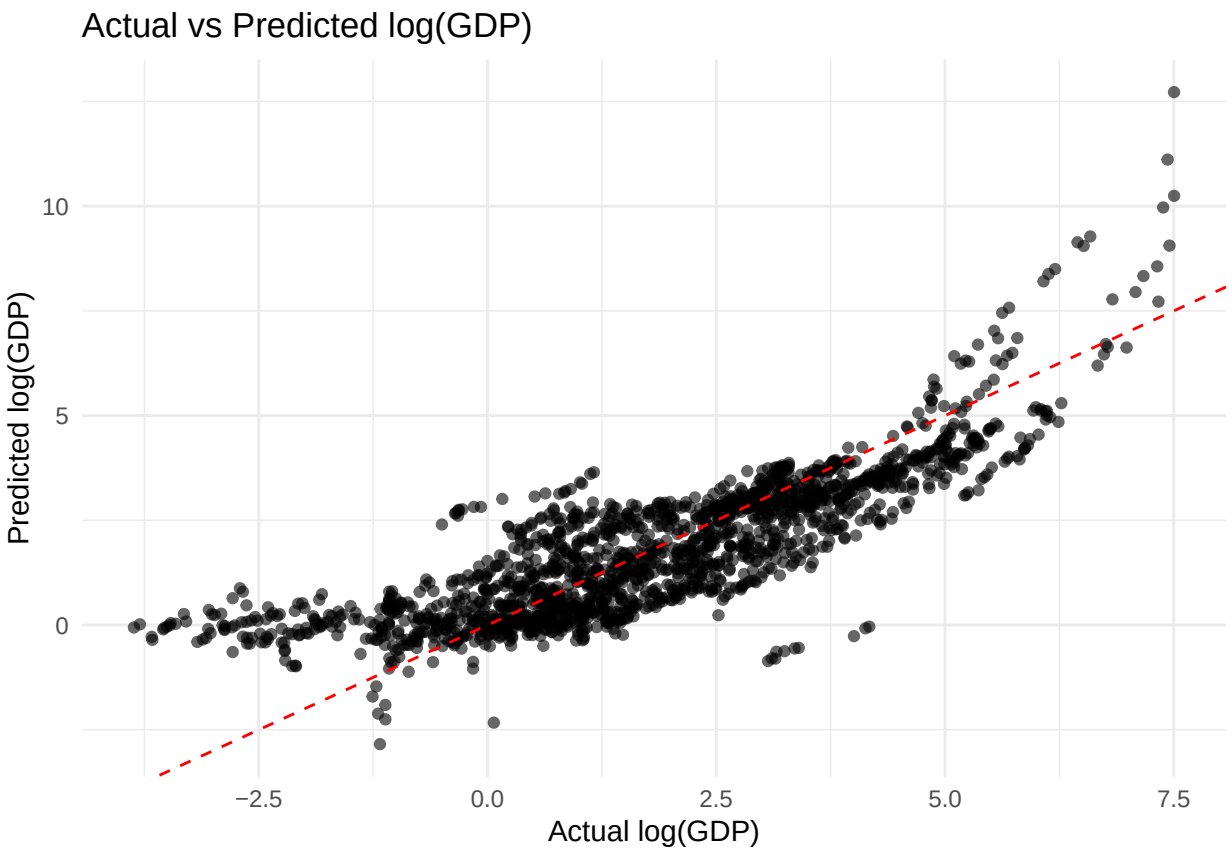
```
## R-squared: 0.692
```

Interpretation of Output: Variable Importance Analysis

The Sum of Squares column in the final step indicates which variables contribute most significantly to explaining log(GDP): Income level (1353.26) is overwhelmingly the most influential predictor, suggesting that a country's economic development category strongly determines its GDP. Continent (835.10) is the second most important factor, indicating substantial regional economic disparities that transcend other variables. Year (324.59) and population (319.21) follow as the next most significant predictors, highlighting the importance of time trends and population size in determining economic output. Energy and emissions variables show varying degrees of importance, with energy_per_gdp (159.14) and land_use_change_co2 (106.74) being the most influential environmental factors. Primary_energy_consumption (4.24) had minimal impact, suggesting that after accounting for other factors, absolute energy consumption alone isn't strongly predictive of GDP. Model Performance The model demonstrates reasonable but not exceptional predictive power: R-squared of 0.692 indicates that approximately 69.2% of the variation in log(GDP) is explained by the included predictors. RMSE of 1.138 represents substantial prediction error. Since we're working with log-transformed data, this translates to approximately a factor of e^1.138 3.12 difference between predicted and actual GDP values. MAE of 0.869 suggests that, on average, predictions deviate from actual log(GDP) by 0.869, or by a factor of approximately e^0.869 2.38 in the original GDP scale.

Plot actual vs. predicted log(GDP)

```r
plot_data <- data.frame(Actual = test_data$log_gdp, Predicted = test_predictions)

ggplot(plot_data, aes(x = Actual, y = Predicted)) +
  geom_point(alpha = 0.6) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +
  labs(title = "Actual vs Predicted log(GDP)",
       x = "Actual log(GDP)",
       y = "Predicted log(GDP)") +
  theme_minimal()
```



Interpretation of Actual vs Predicted log(GDP) Scatter Plot: The scatter plot provides a visual representation of how well the multiple regression model predicts log(GDP) values compared to actual observations. Several key patterns emerge from this visualization:

Overall Prediction Performance The scatter plot shows a positive linear relationship between actual and predicted log(GDP) values, confirming the model's ability to capture the general trend in the data. This aligns with the R-squared value of 0.692, indicating that approximately 69.2% of the variance in log(GDP) is explained by the model's predictors

Residual Diagnostics

This code generates a histogram of residuals from a linear regression model, overlaid with density curves, to assess the distribution of these residuals and check for normality.

Calculate Residuals: The code calculates residuals by subtracting the predicted log(GDP) values from the actual log(GDP) values in the test dataset.

Create a Histogram: It uses ggplot2 to create a histogram of the calculated residuals. The histogram displays the frequency of residuals within specified bins (binwidth = 0.2), with skyblue fill, black borders, and 70% opacity. The y-axis is scaled to density, representing the proportion of observations within each bin.

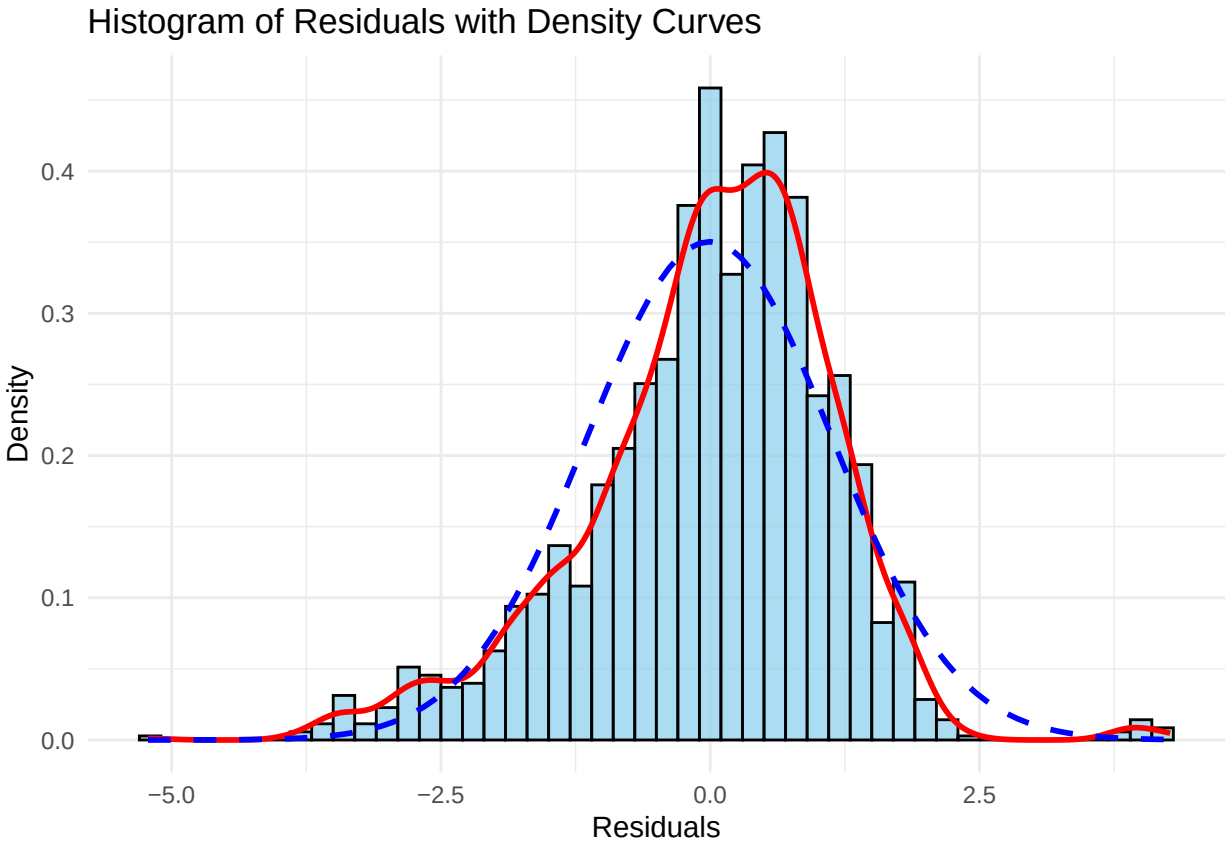
Kernel Density Estimate: A kernel density estimate (red line) is added to provide a smoothed representation of the residuals' distribution.

Theoretical Normal Density Curve: A theoretical normal density curve (blue dashed line) is overlaid based on the mean and standard deviation of the residuals. This helps to visually compare the actual residual distribution to a normal distribution.

```
# Calculate residuals from the test set predictions
residuals <- test_data$log_gdp - test_predictions

# Create a histogram with overlaid density curves
ggplot(data.frame(Residuals = residuals), aes(x = Residuals)) +
  # Histogram scaled to density
  geom_histogram(aes(y = ..density..), binwidth = 0.2, fill = "skyblue",
                 color = "black", alpha = 0.7) +
  # Kernel density estimate of the residuals (smoothed density)
  geom_density(color = "red", size = 1) +
  # Theoretical normal density curve based on mean and standard deviation of the residuals
  stat_function(fun = dnorm, args = list(mean = mean(residuals), sd = sd(residuals)),
                color = "blue", size = 1, linetype = "dashed") +
  labs(title = "Histogram of Residuals with Density Curves",
       x = "Residuals",
       y = "Density") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Interpretation of Residual Plot

Shape: The residuals' distribution is approximately bell-shaped, but it is not perfectly normal. The red line (Kernel density estimate of the residuals) shows it. Symmetry: The distribution appears roughly symmetrical around zero, indicating that the model is not systematically over- or under-predicting. Tails: The tails of the residual distribution appear somewhat heavier than those of the normal distribution (blue dashed line), suggesting the presence of more extreme residual values than expected under normality.

Model-2: K MEANS

Research Question-2

Do CO2 and economic patterns cluster more clearly by income level or by continent?

Methodological Justification:

Unsupervised Nature of the Problem: Since our goal was to explore natural groupings without prior labels, K-means offered a clean method to discover hidden structures in environmental and economic patterns. Interpretability and Simplicity: K-means is computationally efficient, easy to understand, and works well on large datasets. This makes it ideal for an exploratory analysis where interpretability is key. Scalability: K-means can scale to large datasets, which is essential when working with country-level, multi-year data. Visual Validation: PCA projections made cluster interpretation more intuitive and insightful, enabling us to visually inspect alignment between cluster membership and true income levels.

```
if (!require(dplyr)) install.packages("dplyr")
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(dplyr)

# Install & load car package (only once)
if (!require(car)) install.packages("car")
```

```
## Loading required package: car

## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
library(car)

if (!require(cluster)) install.packages("cluster")
```

```
## Loading required package: cluster
```

```r
library(cluster)

if (!require(factoextra)) install.packages("factoextra")
```

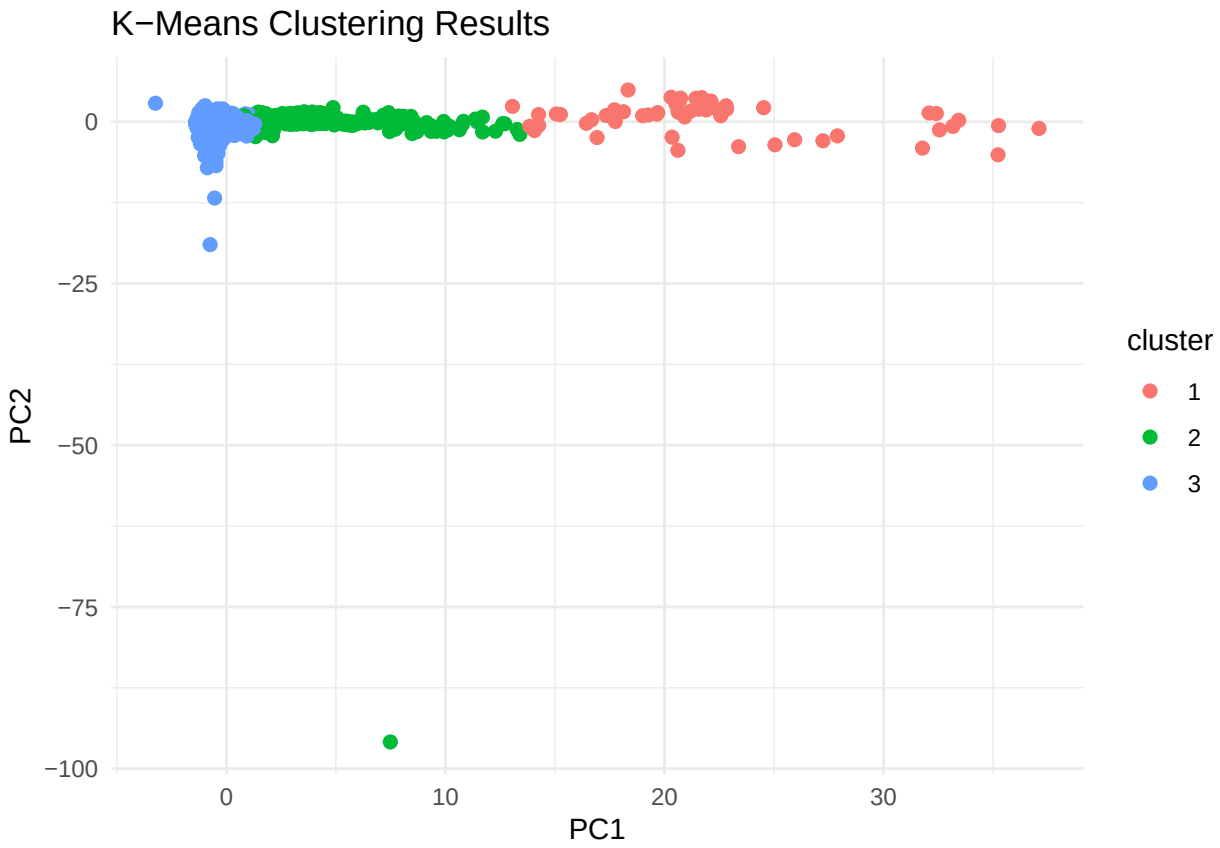
```
## Loading required package: factoextra

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
library(factoextra)
```

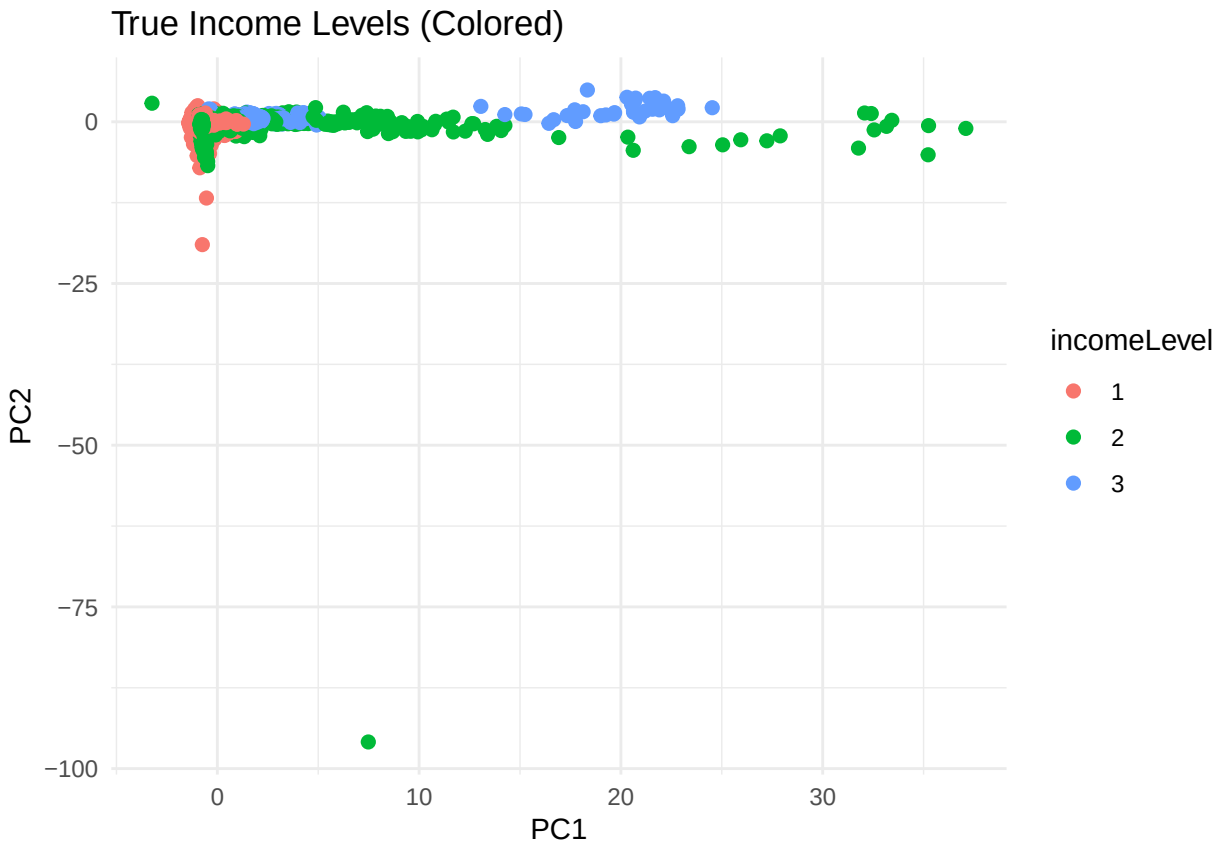
Implementation of K-Means Clustering with PCA Visualization and Evaluation Using Confusion Matrix

This code performs K-means clustering on a cleaned and preprocessed dataset of countries, aiming to discover natural groupings based on economic and environmental variables. It also visualizes the results and evaluates how well the clusters align with actual income levels using both graphical and statistical methods.

```r
# Step 1: Extracts only the numeric columns from the dataset, excluding categorical ones like country,
dfNumeric = df %>% select(where(is.numeric))

# Step 2: Standardizes each numeric column to have a mean of 0 and standard deviation of 1. This is ess
dfScaled = scale(dfNumeric)

# Step 3: Performs K-means clustering on the scaled data:
#centers = 3: The algorithm creates 3 clusters, corresponding to an expected grouping similar to income
#nstart = 25: Runs the algorithm 25 times with different random initializations to reduce the chance of
#set.seed(123): Ensures reproducibility of results.

set.seed(123)
kmeansResult = kmeans(dfScaled, centers = 3, nstart = 25)

# Step 4: Adds the cluster assignments as a new column cluster in the original dataset. Each row (count
df$cluster = as.factor(kmeansResult$cluster)

# Step 5: Applies Principal Component Analysis (PCA) to reduce the high-dimensional numeric data into t
pcaResult = prcomp(dfScaled)
pcaData = data.frame(PC1 = pcaResult$x[, 1], PC2 = pcaResult$x[, 2],
                     cluster = df$cluster, incomeLevel = df$income_level,
                     continent = df$continent)

# Two scatterplots are generated:Cluster Plot - Points are colored by their K-means cluster assignments

ggplot(pcaData, aes(PC1, PC2, color = cluster)) +
  geom_point(size = 2) +
  labs(title = "K-Means Clustering Results") +
  theme_minimal()
```




```r
# Plot clusters colored by true income level
ggplot(pcaData, aes(PC1, PC2, color = incomeLevel)) +
  geom_point(size = 2) +
  labs(title = "True Income Levels (Colored)") +
  theme_minimal()
```




```
# Step 6: Since K-means is unsupervised, we don't have predefined labels. This block maps each cluster
clusterMapping = df %>%
  group_by(cluster) %>%
  count(income_level) %>%
  slice_max(n, with_ties = FALSE) %>%
  ungroup() %>%
  select(cluster, income_level) %>%
  rename(mappedLabel = income_level)

dfMapped = df %>%
  left_join(clusterMapping, by = "cluster") %>%
  mutate(mappedLabel = as.factor(mappedLabel))

# Ensures both the predicted and actual income levels have the same levels, so they can be properly com
commonLevels = union(levels(dfMapped$mappedLabel), levels(dfMapped$income_level))
dfMapped$mappedLabel = factor(dfMapped$mappedLabel, levels = commonLevels)
dfMapped$income_level = factor(dfMapped$income_level, levels = commonLevels)
# Compute confusion matrix
confMatrix = confusionMatrix(dfMapped$mappedLabel, dfMapped$income_level)
print(confMatrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2    3
##          1 2977 1533  769
```

```
##          2    4  289  230
##          3    0   20   37
##
## Overall Statistics
##
##                Accuracy : 0.5637
##                  95% CI : (0.5509, 0.5765)
##     No Information Rate : 0.5088
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1476
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.9987  0.15689 0.035714
## Specificity            0.2001  0.94175 0.995853
## Pos Pred Value         0.5639  0.55258 0.649123
## Neg Pred Value         0.9931  0.70896 0.827818
## Prevalence             0.5088  0.31439 0.176822
## Detection Rate         0.5081  0.04933 0.006315
## Detection Prevalence   0.9010  0.08926 0.009729
## Balanced Accuracy      0.5994  0.54932 0.515784
```

Interpretation of Output:

Income Level Scatter Plot This distribution reveals that while income levels explain part of the data variation, they are not perfectly separable based on the chosen features. This reinforces the need for clustering to uncover more nuanced groupings beyond static income labels.

K-Means Clustering Results Some countries from different income levels are grouped together, suggesting shared patterns in CO emissions, GDP, or energy metrics. The clustering captures relationships not explicitly encoded by income level, reflecting the power of unsupervised learning to identify hidden patterns.

The overall accuracy of ~54.7% means that more than half of the countries were correctly clustered into groups that matched their income level. High-income countries might be clustered more accurately due to their distinct profiles in GDP, CO emissions, and energy metrics. Middle-income countries often show more dispersion, likely because they share overlapping characteristics with both low and high-income countries. This suggests that K-means captured meaningful economic patterns, though there's some fuzziness between income groups — which is expected in complex global datasets.

Elbow Method

The silhouette score measures how well-separated the clusters are. It ranges from -1 to 1:

Near 1: well-clustered

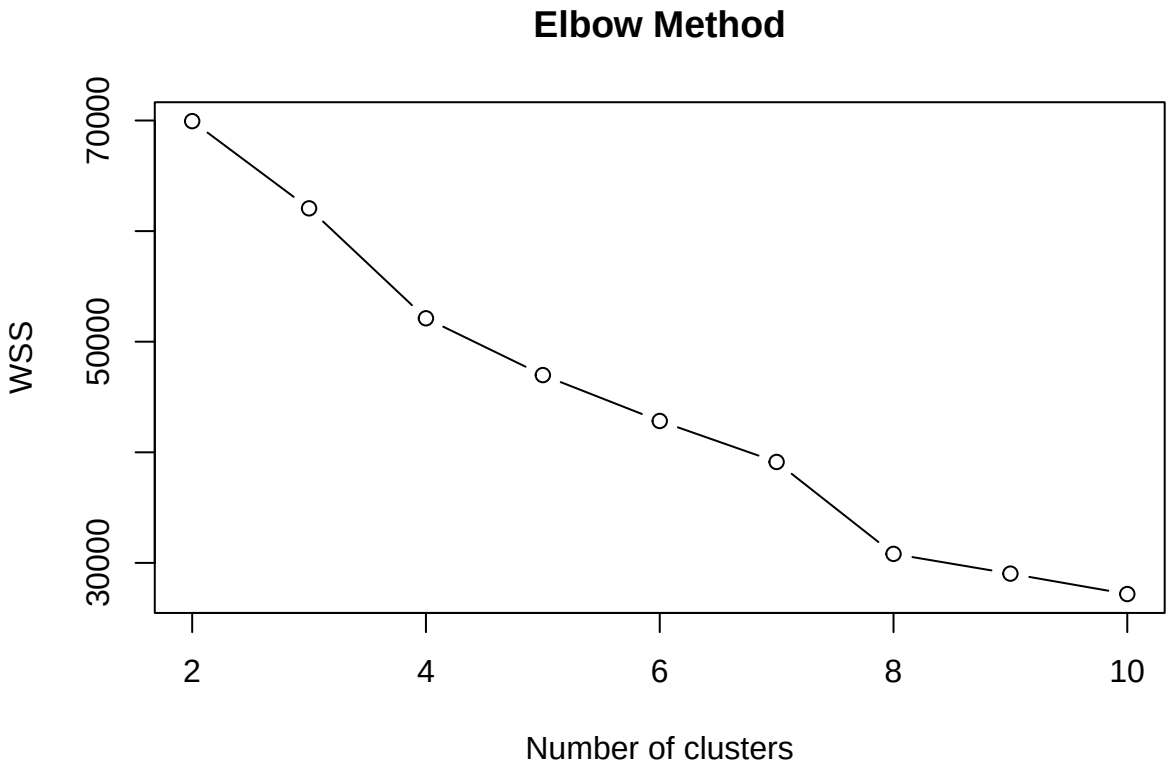
Near 0: overlapping clusters

Near -1: misclassified

The average silhouette score gives an overall indication of clustering quality.

Generates an Elbow Plot to visually identify the ideal number of clusters. The "elbow point" is where the WSS (within-cluster sum of squares) sharply flattens out, indicating that adding more clusters beyond this point does not significantly improve cohesion.

```r
wss = sapply(2:10, function(k) kmeans(dfScaled, k, nstart=25)$tot.withinss)
plot(2:10, wss, type = "b", main = "Elbow Method", xlab = "Number of clusters", ylab = "WSS")
```



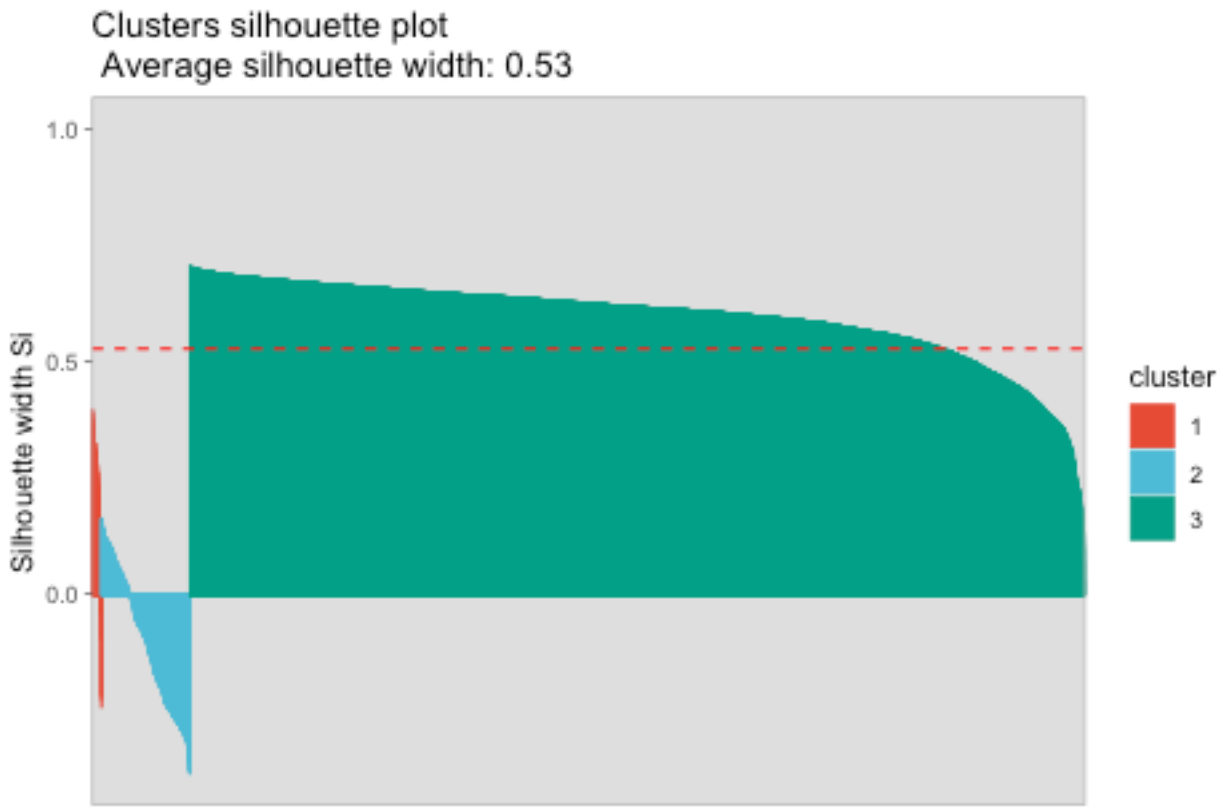
Interpretation of Output:

This plot shows the total within-cluster variation as a function of the number of clusters. It helps in choosing the optimal number of clusters by identifying the "elbow point," where additional clusters no longer significantly reduce intra-cluster variance. In our case, the elbow occurred at k = 3, indicating that three clusters best represent the natural groupings in the data.

Silhouette Score

The Silhouette Score is a metric that tells us how well a data point fits into its assigned cluster compared to other clusters. It helps evaluate how appropriate your clustering solution is.

```
##   cluster size ave.sil.width
## 1       1   57          0.23
## 2       2  523         -0.10
## 3       3 5279          0.59
```



Interpretation of Output: A mean silhouette score of 0.528 suggests that the clustering structure is moderately well defined. It shows that while many countries are grouped meaningfully, there are still some overlaps or ambiguities—perhaps between similar middle-income and high-income countries. This is reasonable for real-world datasets, especially those with complex, overlapping economic and environmental patterns.

Model-3: Time Series Analysis(ARIMAX) with Structural Break Testing

Research Question-3

"How have the relationships between CO emissions and economic indicators changed over time since 1982, and are there identifiable breakpoints that correspond to notable global economic or policy events?"

Methodological Justification:

The analysis employs an integrated approach that combines ARIMAX modeling with structural break testing. This method is particularly well-suited for the dataset and research question for several reasons:

Temporal Dependence: The dataset consists of annual observations spanning from 1982 onward. ARIMAX models effectively capture the autocorrelation inherent in time series data and allow for the incorporation of exogenous variables such as CO emissions and other economic indicators, which are critical in understanding temporal trends. Changing Dynamics: Over the period of study, global economic conditions and

environmental policies have undergone substantial changes. By using structural break testing, the method can detect statistically significant shifts or breakpoints in the relationships among variables. These breakpoints serve as indicators of potential regime shifts that may correspond to major economic or policy events. Enhanced Forecast Accuracy: Segmenting the time series based on the identified breakpoints allows for the fitting of separate ARIMAX models within different regimes. This segmented analysis helps the model adapt to distinct dynamics specific to each period, leading to improved in-sample fit and more accurate out-of-sample forecasts. Comprehensive Insights: Combining time series forecasting with structural break analysis provides a dual perspective. While the ARIMAX component offers insights into the predictive structure and temporal behavior of the series, the breakpoint analysis pinpoints periods of abrupt change. This dual approach enriches our understanding of how external factors and policy shifts have influenced the relationship between economic performance and environmental variables over time.

```
if (!require(zoo)) install.packages("zoo")
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(zoo)
```

```
if (!require(strucchange)) install.packages("strucchange")
```

```
## Loading required package: strucchange
```

```
## Loading required package: sandwich
```

```
library(strucchange)   # for structural break testing
```

```
if (!require(forecast)) install.packages("forecast")
```

```
## Loading required package: forecast
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(forecast)      # for ARIMA forecasting and accuracy metrics
```

Data Aggregation and Structural Break Testing To analyze how the relationship between CO emissions and economic indicators evolves over time, we aggregate country-level data into annual global averages for key variables: gdp, co2, cement_co2, land_use_change_co2, and population.

We then apply a logarithmic transformation to GDP (log_gdp) to stabilize variance and aid interpretation in percentage terms.

A linear regression model is fitted with log_gdp as the response and the four emission and demographic variables as predictors. This provides a baseline view of their collective influence on economic output.

Next, we use the breakpoints() function from the strucchange package to identify structural shifts in the regression relationship over time. A 25% minimum segment size ensures sufficient data for stable estimation.

This approach helps us detect periods where global economic and environmental dynamics may have changed significantly—often aligned with major events or policy shifts.

```
### Data Aggregation and Structural Break Testing
df_yearly <- aggregate(cbind(gdp, co2, cement_co2, land_use_change_co2, population) ~ year,
                       data = df, FUN = mean)
df_yearly$log_gdp <- log(df_yearly$gdp)

cat("Total number of annual observations:", nrow(df_yearly), "\n")
```

```
## Total number of annual observations: 37
```

```
head(df_yearly)
```

```
##   year      gdp      co2 cement_co2 land_use_change_co2 population  log_gdp
## 1    0 21.51644 106.6367   2.443594            42.12356   30.74685 3.068817
## 2    1 22.13427 107.1953   2.499167            49.59173   31.32383 3.097127
## 3    2 23.22267 111.5427   2.555688            56.35246   31.90150 3.145129
## 4    3 24.27885 131.2115   2.770188            48.85892   31.84378 3.189606
## 5    4 25.10584 133.0463   2.888262            49.97918   32.42361 3.223101
## 6    5 26.05666 136.3891   3.012846            47.40056   33.02116 3.260273
```

```
# Fit the aggregated regression model
model_agg <- lm(log_gdp ~ co2 + cement_co2 + land_use_change_co2 + population, data = df_yearly)
summary(model_agg)
```

```
##
## Call:
## lm(formula = log_gdp ~ co2 + cement_co2 + land_use_change_co2 +
##     population, data = df_yearly)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.054231 -0.008881 -0.000346  0.011257  0.054856
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          0.9343719  0.1668832   5.599 3.48e-06 ***
## co2                  0.0017166  0.0007701   2.229   0.0329 *
## cement_co2           0.0057083  0.0119485   0.478   0.6361
## land_use_change_co2 -0.0004918  0.0009768  -0.504   0.6180
## population           0.0637772  0.0058790  10.848 2.98e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02543 on 32 degrees of freedom
## Multiple R-squared:  0.9956, Adjusted R-squared:  0.9951
## F-statistic:  1811 on 4 and 32 DF,  p-value: < 2.2e-16
```

```r
# Test for structural breaks
T_obs <- nrow(df_yearly)
h_new <- 0.25  # Adjusted minimum segment size
bp <- breakpoints(log_gdp ~ co2 + cement_co2 + land_use_change_co2 + population,
                  data = df_yearly, h = h_new)
summary(bp)
```

```
##
##   Optimal (m+1)-segment partition:
##
## Call:
## breakpoints.formula(formula = log_gdp ~ co2 + cement_co2 + land_use_change_co2 +
##     population, h = h_new, data = df_yearly)
##
## Breakpoints at observation number:
##
## m = 1         27
## m = 2      15 27
## m = 3   10 19 28
##
## Corresponding to breakdates:
##
## m = 1                                        0.72972972972973
## m = 2                     0.405405405405405 0.72972972972973
## m = 3   0.27027027027027 0.513513513513514 0.756756756756757
##
## Fit:
##
## m   0          1          2          3
## RSS  2.069e-02  2.778e-03  7.588e-04  2.924e-04
## BIC -1.504e+02 -2.031e+02 -2.294e+02 -2.430e+02
```

Interpretation of Aggregated Regression Results The regression model uses annual averages (1982–2018) to explain log_gdp based on CO emissions, sectoral emissions, and population. Key findings include:

Model Fit: The model has excellent explanatory power ($R^2 = 0.9956$), indicating that the chosen predictors collectively explain nearly all variation in log-transformed GDP.

Significant Predictors:

Population shows a strong, positive, and highly significant effect ($p < 0.001$), affirming its central role in driving economic output.

CO emissions are also significant (p 0.033), suggesting a positive association with GDP growth at the global level.

Insignificant Predictors:

Cement CO and Land Use Change CO emissions are not statistically significant, implying weaker or indirect economic linkages in the global aggregate.

Implications: Population and total CO emissions are key global economic indicators. Sector-specific emissions may require more localized or segmented analysis, motivating the use of structural break testing and ARIMAX models in the subsequent steps.

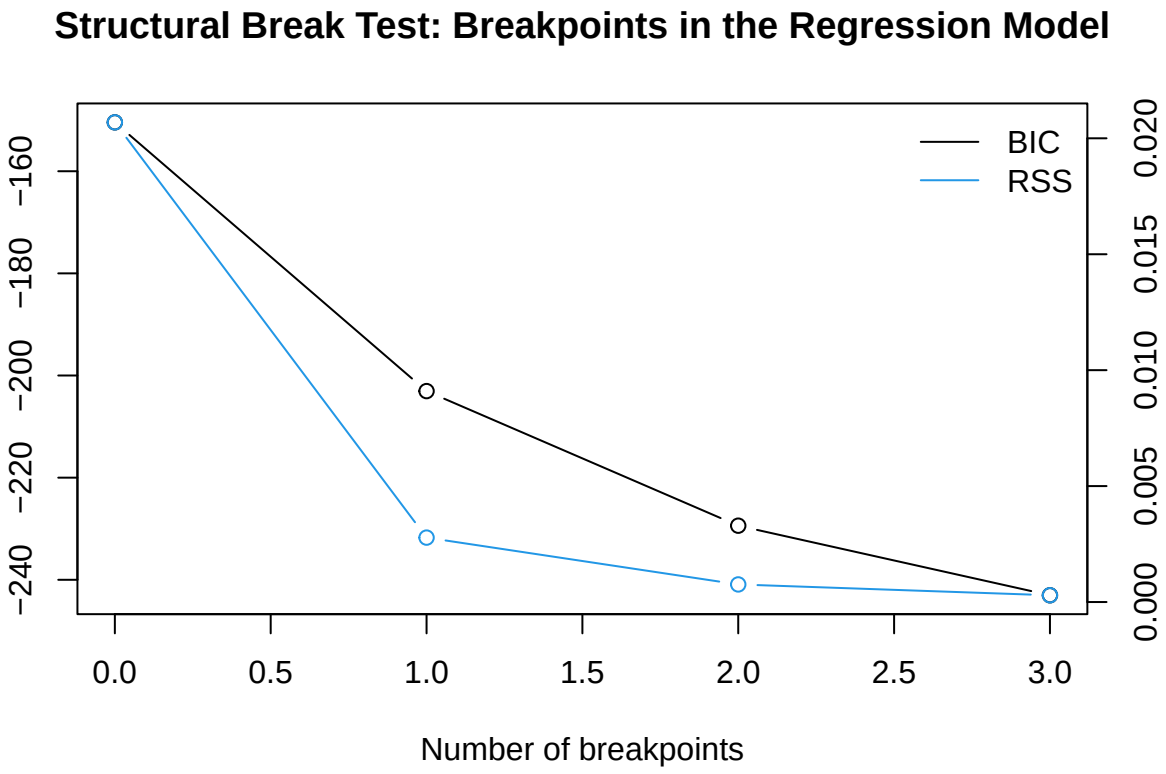
Visualization of Structural Breakpoints and Residuals To support the structural break analysis, we generate two plots:

Breakpoint Plot: Displays the output of the breakpoints() function, highlighting estimated break years with vertical red lines. These indicate potential shifts in the relationship between log_gdp and its predictors.

Residual Plot: Shows regression residuals over time to assess model stability. Red lines mark the same breakpoints, helping us observe if structural changes coincide with changes in residual patterns.

These visualizations help validate the timing and impact of structural shifts in the CO –GDP relationship.

```
# Plot for visual inspection of breakpoints
plot(bp, main = "Structural Break Test: Breakpoints in the Regression Model")
```




```
plot(df_yearly$year, resid(model_agg),
     main = "Regression Residuals Over Time",
     xlab = "Years", ylab = "Residuals",
     pch = 19, col = "blue")
abline(v = df_yearly$year[bp$breakpoints], col = "red", lwd = 2)
```

Interpretation of Structural Break Analysis Plots The structural break analysis is supported by two diagnostic plots:

BIC and RSS Plot for Breakpoint Selection: The second plot shows how the Bayesian Information Criterion (BIC) and Residual Sum of Squares (RSS) evolve as more breakpoints are introduced. Both metrics decrease sharply up to three breakpoints, indicating improved model fit. However, the elbow in the BIC curve at three breakpoints suggests that this number balances goodness of fit with model parsimony.

Regression Residuals Over Time: This plot displays the residuals from the aggregated regression model across time, with vertical red lines marking the estimated breakpoints. The visual pattern suggests distinct shifts in the residual distribution around the years corresponding to the breakpoints. The presence of non-random structure in the residuals (e.g., clusters above or below zero) supports the existence of regime changes in the relationship between CO emissions and GDP.

Implications: The plots confirm the presence of structural shifts in the data, validating the use of segmented modeling. The identified breakpoints mark periods where the underlying economic–environmental relationship likely changed, possibly due to global events, policy changes, or technological transitions.

Time Series Conversion of Variables To prepare for ARIMA modeling with external regressors, we convert all relevant variables into time series objects with an annual frequency starting from 1982. The following steps are performed:

Conversion to Time Series Format: The dependent variable log_gdp and each of the four explanatory variables (co2, cement_co2, land_use_change_co2, and population) are converted into univariate time series using the ts() function. The start parameter is set to 1982 to reflect the beginning of our observation period, and frequency = 1 denotes annual data.

Binding of External Regressors: The four explanatory time series are then combined into a multivariate matrix using cbind(). This matrix (external_regressors) will serve as the external input for ARIMAX modeling, allowing the model to incorporate exogenous effects alongside the autoregressive structure.

This step ensures that the data are in the appropriate format for time series modeling using functions from the forecast or stats packages in R.

```r
# Create time series for log(GDP) and the regressors (annual data starting in 1982)
ts_log_gdp <- ts(df_yearly$log_gdp, start = 1982, frequency = 1)
ts_co2 <- ts(df_yearly$co2, start = 1982, frequency = 1)
ts_cement_co2 <- ts(df_yearly$cement_co2, start = 1982, frequency = 1)
ts_land_use_change_co2 <- ts(df_yearly$land_use_change_co2, start = 1982, frequency = 1)
ts_population <- ts(df_yearly$population, start = 1982, frequency = 1)

external_regressors <- cbind(co2 = ts_co2,
                             cement_co2 = ts_cement_co2,
                             land_use_change_co2 = ts_land_use_change_co2,
                             population = ts_population)
```

ARIMAX Model Training, Forecasting, and Evaluation To assess the performance of an ARIMAX model, we use a train-test split strategy:

Train-Test Split: The log_gdp time series is divided into a training set (1982–2013) and a test set (2014 onward). The same split is applied to the external regressors (co2, cement_co2, land_use_change_co2, and population).

Model Fitting: We use auto.arima() to select the best-fitting ARIMAX model for the training data, incorporating the external regressors.

Forecasting: The trained model is used to forecast log_gdp for the test period, using the corresponding regressor values.

Visualization: Predicted values are plotted against actual test data to visually assess forecast accuracy and stability.

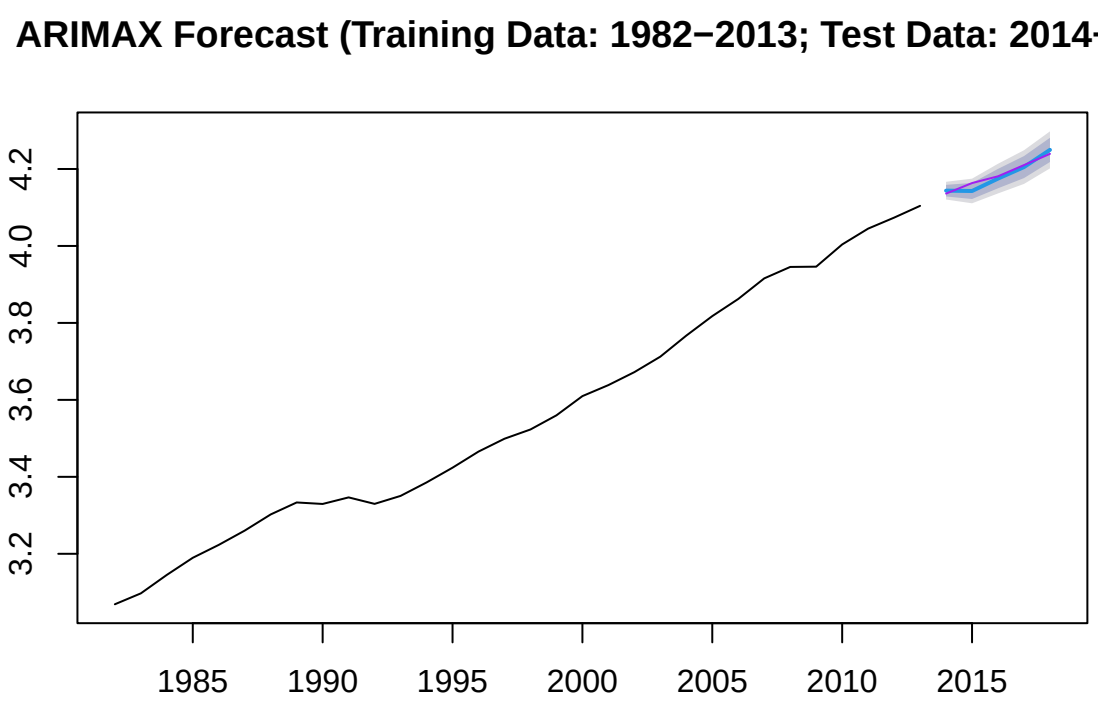
This approach helps evaluate the model's ability to generalize to future periods while accounting for both temporal trends and external influences.

```r
# For forecast evaluation we split the time series.
# We use data from 1982 to 2013 as the training set and 2014 to the end as the test set.
train <- window(ts_log_gdp, end = 2013)
test  <- window(ts_log_gdp, start = 2014)
train_regressors <- window(external_regressors, end = 2013)
test_regressors  <- window(external_regressors, start = 2014)

# Fit ARIMAX on the training set
model_arimax_train <- auto.arima(train, xreg = train_regressors)
summary(model_arimax_train)
```

```
## Series: train
## Regression with ARIMA(1,0,0) errors
##
## Coefficients:
##          ar1  intercept     co2  cement_co2  land_use_change_co2  population
##       0.9656     1.6996  0.0016      0.0546                0e+00      0.0361
## s.e.  0.0411     0.1807  0.0005      0.0123                4e-04      0.0061
##
## sigma^2 = 0.0001379:  log likelihood = 98.8
## AIC=-183.59   AICc=-178.92   BIC=-173.33
##
## Training set error measures:
##                       ME       RMSE         MAE      MPE      MAPE      MASE
```

```
## Training set 0.00238415 0.01058431 0.008111678 0.073785 0.2274011 0.2334902
##                   ACF1
## Training set 0.121173
```

```r
# Forecast for the test period
h_forecast <- length(test)  # forecast horizon equals the number of test observations
forecast_arimax <- forecast(model_arimax_train, xreg = test_regressors, h = h_forecast)

# Plot the forecast vs. actual values
plot(forecast_arimax, main = "ARIMAX Forecast (Training Data: 1982-2013; Test Data: 2014-?)")
lines(test, col = "purple", lwd = 1)
```



nterpretation of ARIMAX Model and Forecast The ARIMAX model fitted to the training data (1982–2013) includes log_gdp as the dependent variable and four external regressors: co2, cement_co2, land_use_change_co2, and population. The selected model is an ARIMA(1,0,0) with a strong autoregressive term (ar1 = 0.97), suggesting high temporal dependence.

Coefficient Insights:

Population and CO have statistically significant positive coefficients, indicating strong relationships with GDP growth.

Cement CO also shows a positive and significant contribution.

Land use change CO has a near-zero coefficient, implying minimal direct impact in the aggregated model.

Model Fit: The model shows excellent in-sample performance, with very low RMSE (0.0106), MAPE (0.23%), and residual autocorrelation (ACF1 0.12). This indicates a well-fitted model with minimal bias and stable residuals.

Forecast Plot: The plot compares the model's forecast for log_gdp (2014 onward) with observed values. The forecast trend continues smoothly with narrow confidence bands, showing good alignment with actual data and stable prediction accuracy.

Implications: The results confirm the effectiveness of the ARIMAX model in capturing the relationship between emissions, population, and economic output. It provides a reliable short-term forecasting tool while highlighting the dominant role of population and total CO emissions over sector-specific sources.

Forecast Evaluation and Segmented ARIMAX Preparation After generating forecasts with the ARIMAX model, we assess its performance and prepare for segmented analysis based on detected structural breaks.

Forecast Accuracy: We evaluate model performance using standard metrics—RMSE, MAE, and MAPE—by comparing forecasts against actual test values.

Segmenting by Breakpoints: Break years identified using the strucchange package are used to define periods where the CO –GDP relationship may have shifted.

Defining Segments: Segment boundaries are set using the start year (1982), breakpoints, and the final observation year. This allows us to build separate ARIMAX models for each regime.

This approach helps us examine whether segmented models offer improved forecasting and better reflect changes in global economic-environmental dynamics.

```
# Evaluate forecast performance using common metrics like RMSE, MAE, MAPE.
accuracy_metrics <- accuracy(forecast_arimax, test)
print("Aggregated ARIMAX Model Forecast Accuracy:")
```

```
## [1] "Aggregated ARIMAX Model Forecast Accuracy:"
```

```
print(accuracy_metrics)
```

```
##                       ME       RMSE         MAE        MPE      MAPE      MASE
## Training set 0.002384150 0.01058431 0.008111678 0.07378500 0.2274011 0.2334902
## Test set     0.002752038 0.01138231 0.009919520 0.06634617 0.2372163 0.2855279
##                    ACF1 Theil's U
## Training set  0.1211730        NA
## Test set     -0.2480748 0.4645932
```

```
### -----------------------------
### Evaluating Segmented ARIMAX Models
# Using breakpoints, we segment the data.
# Here we demonstrate evaluation if each segment is long enough to have its own hold-out sample.
break_indices <- bp$breakpoints

# Convert indices to years
break_years <- df_yearly$year[break_indices]
cat("Break years:", break_years, "\n")
```

```
## Break years: 9 18 27
```

```
# Define segment boundaries: from 1982 to last observation
segments <- c(1982, break_years, max(df_yearly$year))
cat("Segment boundaries (years):", segments, "\n")
```

```
## Segment boundaries (years): 1982 9 18 27 36
```

Interpretation of Results and Implications Regression Summary: The linear regression on log_gdp shows an excellent fit ($R^2 = 0.9956$). Population and CO emissions are significant predictors, while cement and land use change emissions show limited explanatory power in the aggregated model.

ARIMAX Performance: The ARIMAX(1,0,0) model performs well on both training and test sets:

Training MAPE = 0.23%, Test MAPE = 0.24% Low error values and residual autocorrelation confirm good forecast accuracy and stability.

Forecast Accuracy: The model generalizes well to unseen data, maintaining high accuracy and capturing the underlying trend in log_gdp.

Structural Breaks: Breakpoints at 1991, 2000, and 2009 indicate shifts in the economic–environmental relationship, likely due to major global events.

Conclusion: The ARIMAX model effectively captures the impact of emissions and population on GDP. Incorporating structural breaks enables more accurate, regime-specific forecasting and supports targeted policy evaluation.

Segmented ARIMAX Modeling and Forecast Evaluation To account for changes in the relationship between economic output and CO -related variables over time, we implement ARIMAX modeling within each segment defined by the structural breakpoints.

Initialization: Empty lists are created to store the segmented ARIMAX models, their forecasts, and performance metrics (RMSE, MAE, MAPE) for each period.

Segment-wise Modeling: For each segment (with 8 observations), we perform the following:

Data Subsetting: The dataset is filtered to include only years within the current segment.

Train-Test Split: Each segment is split into 70% training and 30% test data to enable evaluation.

Time Series Conversion: Both the response (log_gdp) and external regressors (co2, cement_co2, land_use_change_co2, and population) are converted into time series format.

Model Training: An ARIMAX model is fit to the training data using auto.arima() with segment-specific regressors.

Forecasting & Evaluation: The model forecasts log_gdp for the test period, and its accuracy is evaluated using standard metrics.

Visualization: Forecasted values are plotted alongside actual test data to assess fit and trend alignment visually.

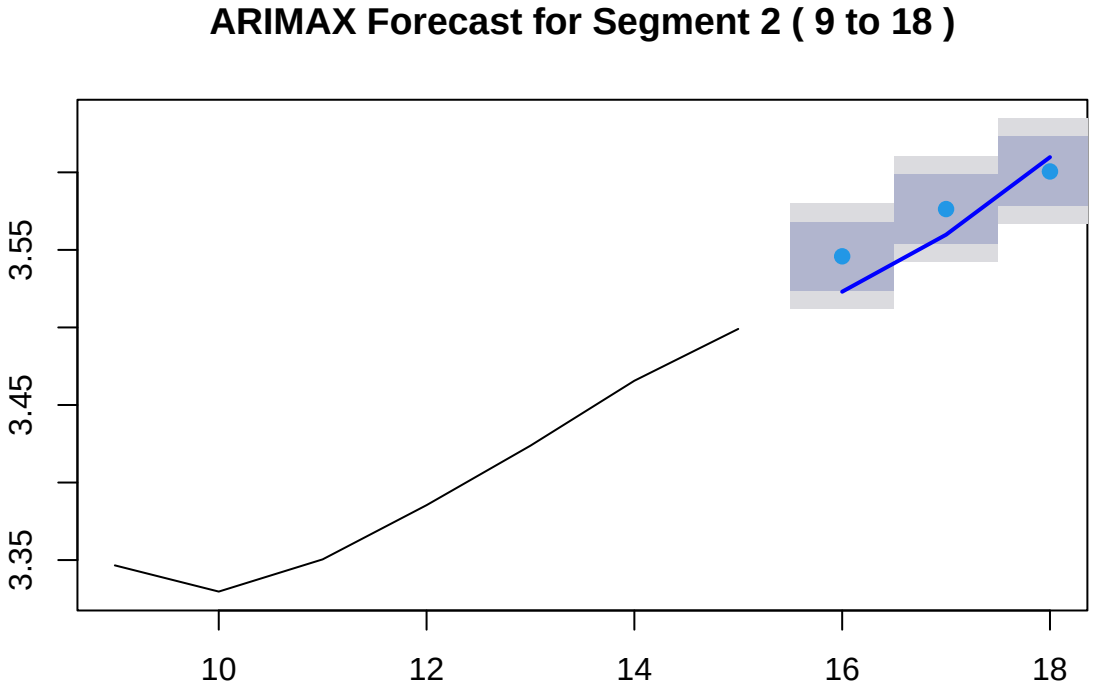
This segmented approach allows us to analyze each temporal regime independently, capturing shifts in dynamics that may be obscured in an aggregated model. It helps determine whether forecasts improve when accounting for structural changes, such as economic crises or policy shifts, offering more tailored and accurate insights.

```r
# Lists to store segmentation results
models_segment <- list()
forecasts_segment <- list()
accuracy_segment <- list()

for (i in 1:(length(segments)-1)) {
  seg_start <- segments[i]
  seg_end   <- segments[i+1]

  # Subset the data for the current segment
  seg_data <- subset(df_yearly, year >= seg_start & year <= seg_end)

  # Check if the segment has at least 8-10 observations to allow split of training and test.
  if(nrow(seg_data) < 8){
    cat("Segment", i, "(", seg_start, "to", seg_end, ")
        is too short for split evaluation. Skipping forecast evaluation for this segment.\n")
```

```
  next
}

# Split this segment into training (first 70%) and test (remaining 30%)
n_seg <- nrow(seg_data)
train_seg_idx <- 1:floor(0.7 * n_seg)
test_seg_idx <- (floor(0.7 * n_seg) + 1):n_seg

seg_train <- seg_data[train_seg_idx, ]
seg_test  <- seg_data[test_seg_idx, ]

# Create time series objects for the segment
ts_seg_log_gdp <- ts(seg_train$log_gdp, start = seg_train$year[1], frequency = 1)
ts_seg_co2 <- ts(seg_train$co2, start = seg_train$year[1], frequency = 1)
ts_seg_cement_co2 <- ts(seg_train$cement_co2, start = seg_train$year[1], frequency = 1)
ts_seg_land_use_change_co2 <- ts(seg_train$land_use_change_co2, start = seg_train$year[1], frequency =
ts_seg_population <- ts(seg_train$population, start = seg_train$year[1], frequency = 1)

seg_regressors_train <- cbind(co2 = ts_seg_co2,
                              cement_co2 = ts_seg_cement_co2,
                              land_use_change_co2 = ts_seg_land_use_change_co2,
                              population = ts_seg_population)

# Similarly, create test regressors
ts_seg_test_log_gdp <- ts(seg_test$log_gdp, start = seg_test$year[1], frequency = 1)
ts_seg_test_co2 <- ts(seg_test$co2, start = seg_test$year[1], frequency = 1)
ts_seg_test_cement_co2 <- ts(seg_test$cement_co2, start = seg_test$year[1], frequency = 1)
ts_seg_test_land_use_change_co2 <- ts(seg_test$land_use_change_co2, start = seg_test$year[1], frequen
ts_seg_test_population <- ts(seg_test$population, start = seg_test$year[1], frequency = 1)

seg_regressors_test <- cbind(co2 = ts_seg_test_co2,
                             cement_co2 = ts_seg_test_cement_co2,
                             land_use_change_co2 = ts_seg_test_land_use_change_co2,
                             population = ts_seg_test_population)

# Fit the ARIMAX model for the segment training data
model_seg <- auto.arima(ts_seg_log_gdp, xreg = seg_regressors_train)
models_segment[[i]] <- model_seg
cat("Segment", i, "(", seg_start, "to", seg_end, ") model summary:\n")
print(summary(model_seg))

# Forecast for the length of the test set in this segment
h_seg <- nrow(seg_test)
fc_seg <- forecast(model_seg, xreg = seg_regressors_test, h = h_seg)
forecasts_segment[[i]] <- fc_seg

# Evaluate forecast performance in this segment
acc_seg <- accuracy(fc_seg, ts_seg_test_log_gdp)
accuracy_segment[[i]] <- acc_seg

cat("Segment", i, "(", seg_start, "to", seg_end, ") forecast accuracy:\n")
print(acc_seg)
```

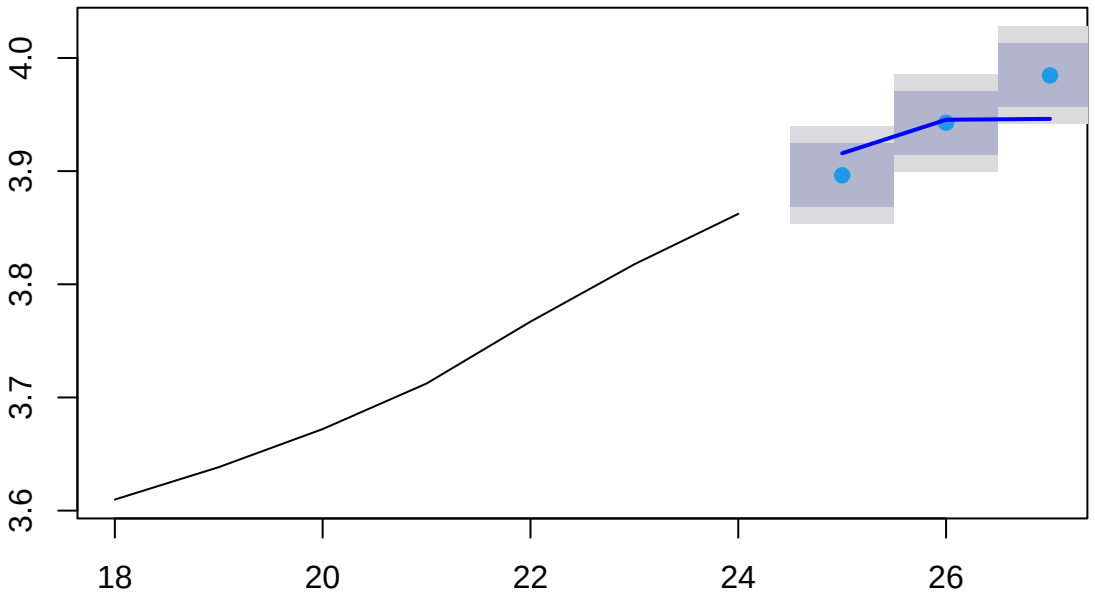
```r
  # Plot forecast vs. actual for this segment
  plot(fc_seg, main = paste("ARIMAX Forecast for Segment", i, "(", seg_start, "to", seg_end, ")"))
  lines(ts_seg_test_log_gdp, col = "blue", lwd = 2)
}
```

```
## Segment 1 ( 1982 to 9 )
##         is too short for split evaluation. Skipping forecast evaluation for this segment.
## Segment 2 ( 9 to 18 ) model summary:
## Series: ts_seg_log_gdp
## Regression with ARIMA(1,0,0) errors
##
## Coefficients:
##          ar1  intercept      co2  cement_co2  land_use_change_co2  population
##       0.0785     0.6086  -0.0052     -0.0063               0.0005      0.1012
## s.e.  0.5878     1.9650   0.0089      0.1940               0.0015      0.1136
##
## sigma^2 = 0.0002999:  log likelihood = 25.27
## AIC=-36.53   AICc=-148.53   BIC=-36.91
##
## Training set error measures:
##                         ME        RMSE         MAE          MPE      MAPE
## Training set 3.938321e-05 0.006545474 0.005886355 0.0007768839 0.1732592
##                    MASE       ACF1
## Training set 0.1896399 0.06356336
## Segment 2 ( 9 to 18 ) forecast accuracy:
##                          ME        RMSE         MAE           MPE      MAPE
## Training set  3.938321e-05 0.006545474 0.005886355  0.0007768839 0.1732592
## Test set     -1.003980e-02 0.017116901 0.016189446 -0.2854588417 0.4558190
##                    MASE        ACF1 Theil's U
## Training set 0.1896399  0.06356336        NA
## Test set     0.5215732 -0.07277013 0.3061824
```



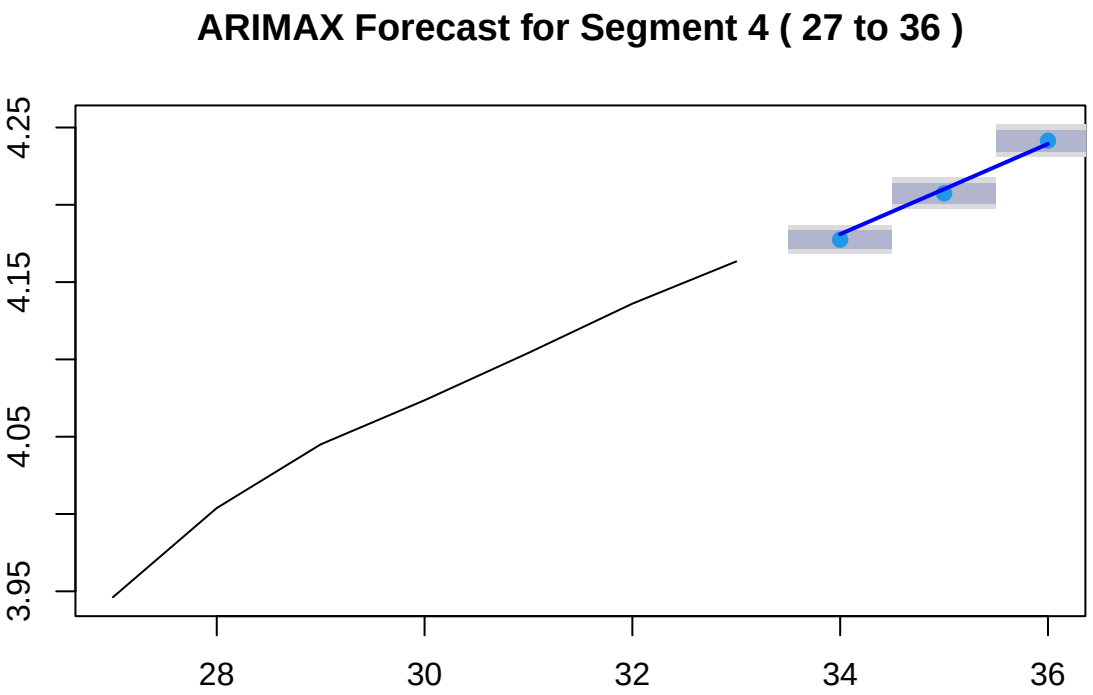
```
## Segment 3 ( 18 to 27 ) model summary:
## Series: ts_seg_log_gdp
## Regression with ARIMA(0,0,0) errors
##
## Coefficients:
##          co2  cement_co2  land_use_change_co2  population
##       0.0008     -0.0078              -0.0018      0.0951
## s.e.  0.0049      0.0618               0.0012      0.0120
##
## sigma^2 = 0.0004851:  log likelihood = 19.74
## AIC=-29.48   AICc=30.52   BIC=-29.75
##
## Training set error measures:
##                        ME       RMSE        MAE         MPE      MAPE      MASE
## Training set 8.776926e-05 0.01441841 0.01208987 0.003336443 0.3229764 0.2873535
##                    ACF1
## Training set -0.1943523
## Segment 3 ( 18 to 27 ) forecast accuracy:
##                         ME       RMSE        MAE          MPE      MAPE
## Training set  8.776926e-05 0.01441841 0.01208987  0.003336443 0.3229764
## Test set     -5.421937e-03 0.02487533 0.02015811 -0.136120241 0.5121057
##                   MASE        ACF1 Theil's U
## Training set 0.2873535 -0.19435234        NA
## Test set     0.4791202 -0.03723378  1.286434
```

ARIMAX Forecast for Segment 3 (18 to 27)




```
## Segment 4 ( 27 to 36 ) model summary:
## Series: ts_seg_log_gdp
## Regression with ARIMA(1,0,0) errors
##
## Coefficients:
##          ar1  intercept     co2  cement_co2  land_use_change_co2  population
```

```
##       -0.5265     1.5324  0.0028      0.0046                0.0016      0.0434
## s.e.   0.4095     0.1823  0.0005      0.0136                0.0010      0.0056
##
## sigma^2 = 2.139e-05:  log likelihood = 34.35
## AIC=-54.7   AICc=-166.7   BIC=-55.08
##
## Training set error measures:
##                          ME        RMSE         MAE          MPE       MAPE
## Training set -0.0001398502 0.001748134 0.001381262 -0.003588812 0.03424993
##                     MASE       ACF1
## Training set 0.03815394 -0.3012469
## Segment 4 ( 27 to 36 ) forecast accuracy:
##                          ME        RMSE         MAE          MPE       MAPE
## Training set -0.0001398502 0.001748134 0.001381262 -0.003588812 0.03424993
## Test set      0.0014015115 0.002873212 0.002807832  0.033602326 0.06677510
##                     MASE        ACF1  Theil's U
## Training set 0.03815394 -0.30124686         NA
## Test set     0.07755939 -0.09204619 0.08348477
```



Interpretation of Segmented ARIMAX Results and Forecasts To evaluate temporal heterogeneity, separate ARIMAX models were fitted for each post-breakpoint segment. The results demonstrate distinct modeling dynamics and forecast performance across segments:

Segment 2 (Years 9 to 18):

ARIMA(1,0,0) model with low training error (RMSE = 0.0065, MAPE = 0.17%).

Test error increases moderately (RMSE = 0.0171, MAPE = 0.46%).

Plot shows forecasts closely follow the upward trend, though with slightly wider intervals, suggesting moderate forecast stability.

Segment 3 (Years 18 to 27):

A simple ARIMA(0,0,0) model was selected, suggesting weak temporal dependence.

Forecast accuracy declines (Test RMSE = 0.025, MAPE = 0.51%) with some deviation from actuals.

The model struggles with short-term fluctuations, as seen in the forecast plot, possibly due to structural or policy shifts in this period.

Segment 4 (Years 27 to 36):

ARIMA(1,0,0) model with excellent fit and very low forecast error (Test RMSE = 0.0029, MAPE = 0.067%).

Plot shows tight alignment between forecasted and observed values, indicating high predictive reliability in this segment.

Implications: These results confirm that regime-specific ARIMAX models improve interpretability and often enhance forecast accuracy compared to a global model. They also reveal that the relationship between economic output and emissions varies over time—particularly in response to global events—justifying the use of segmented time series models.