FA 582 - Foundations of Data Science
Fall Semester
EUR/USD Exchange Rate

Gutium, Andrian

agutium@stevens.edu

Naumov, Vasil

vnaumov@stevens.edu

Malamisura, Federica

fmalamis@stevens.edu

de Alarcon, Lucia

ldealarc@stevens.edu

Abstract

This report presents a comprehensive analysis of the EUR/USD exchange rate using statistical and machine learning methods. The analysis follows a structured approach, starting from data preprocessing, exploratory analysis, and modeling to advanced clustering techniques. Each step is explained in detail, accompanied by relevant R code snippets, outputs, and visualizations. The findings are insightful for understanding exchange rate dynamics and highlight the importance of macroeconomic indicators.

Keywords: EUR/USD, regression, machine learning, clustering, feature importance, inflation, GDP, interest rates

Contents

1 Introduction

The EUR/USD exchange rate is one of the most traded currency pairs in the world, reflecting the economic relationship between the Eurozone and the United States. As a benchmark for global financial markets, fluctuations in the EUR/USD rate can significantly impact international trade, investment decisions, and economic stability. Understanding the dynamics of this exchange rate is essential for traders, policymakers, and economists, as it provides valuable insights into market sentiment and economic conditions.

This report aims to analyze the factors influencing the EUR/USD exchange rate, addressing key questions about the economic, political, and social elements that contribute to its fluctuations. The euro functions as a common currency for a diverse group of countries within the Eurozone, each with varying levels of economic development, fiscal policies, and political stability. This diversity presents both advantages and challenges, particularly during times of economic stress or political uncertainty.

The analysis will explore how macroeconomic indicators—such as interest rates, inflation, and employment figures—affect the EUR/USD exchange rate. Additionally, geopolitical developments and central bank policies, including those from the European Central Bank (ECB) and the Federal Reserve, play a crucial role in shaping investor confidence and market dynamics. Unexpected events, such as political unrest or economic crises, can further influence these fluctuations, affecting the euro's value and its relationship with the US dollar.

The primary objective of this report is to provide a comprehensive analysis of the EUR/USD exchange rate over a defined period, identifying the key determinants that drive its movements. By employing advanced statistical techniques and economic modeling, this analysis aims to enhance understanding of the factors behind exchange rate changes. Furthermore, the report will consider the broader implications of having a single currency in the Eurozone, especially in light of the economic and political diversity among member states.

Ultimately, the findings of this report will serve to equip stakeholders with the knowledge needed to navigate the complexities of currency markets and make informed decisions in an evolving economic landscape.

2 Research Questions

2.1 What are the Key Factors Influencing the EUR/USD Exchange Rate?

Understanding the factors that contribute to fluctuations in the EUR/USD exchange rate is essential for analyzing its movements and predicting future trends. Various economic, political, and market-related elements significantly impact exchange rate dynamics.

2.1.1 Macroeconomic Indicators

Key macroeconomic indicators, such as interest rates, inflation rates, and employment statistics, have a profound effect on the EUR/USD exchange rate. For instance, changes in interest rates set by the European Central Bank (ECB) or the Federal Reserve can shift investor sentiment and affect capital flows. Higher interest rates in one currency zone can attract foreign investment, increasing demand for that currency and leading to appreciation against the other.

2.1.2 Geopolitical Events

The geopolitical landscape also plays a critical role in shaping currency fluctuations. Events such as elections, trade negotiations, and international conflicts can create uncertainty in the markets, prompting investors to reassess their positions. Significant political developments in either the Eurozone or the U.S. can lead to sudden shifts in the EUR/USD rate as traders react to perceived risks.

2.1.3 Market Sentiment and Speculation

Market sentiment and speculative trading further contribute to exchange rate movements. Traders often base decisions on news and economic forecasts, leading to rapid changes in demand for the euro or the dollar. Speculative activities can amplify price movements, resulting in increased fluctuations during certain periods.

In summary, a comprehensive understanding of these factors—macroeconomic indicators, geopolitical events, and market sentiment—is essential for evaluating the dynamics of the EUR/USD exchange rate and its implications for global financial markets.

3 Data Preparation

3.1 Data Loading

The first step involves loading several datasets using the `read_excel` function for Excel files and `read.csv` for a CSV file. The datasets include:

- EUR/USD exchange rates
- GDP data for EUR and USD
- Interest rates for EUR and USD
- Inflation rates for EUR and USD

These datasets form the foundation for subsequent analysis.

3.2 Column Naming

After loading the USD inflation data, the code renames its columns for clarity, ensuring consistency in referencing during later operations.

3.3 Data Cleaning Function

A reusable function `clean_data` is defined to streamline the cleaning process. This function takes several parameters:

- `df`: The dataframe to clean.
- `remove_rows`: The number of rows to remove from the top.
- `select_cols`: The specific columns to retain.
- `new_col_names`: New names for the selected columns.
- `start_date`: The starting date for generating a sequence.
- `by`: The increment for the date sequence.

```r
clean_data <- function(df, remove_rows, select_cols, new_col_names,
    start_date, by) {
  df <- df[-c(1:remove_rows), select_cols]
  colnames(df) <- new_col_names
  df$Date <- seq(from = as.Date(start_date), by = by, length.out =
      nrow(df))
  return(as.data.frame(df))
}
```

Listing 1: R Code: Data Cleaning Function

This function was applied to all datasets to ensure uniformity. The cleaned datasets were then merged, and outliers were handled appropriately.

Within this function:

1. Unnecessary rows are removed, and relevant columns are selected.
2. Columns are renamed for consistency.
3. A date sequence is generated based on the provided parameters.
4. An error is raised if the number of rows does not match the length of the date sequence.

3.4 Cleaning Individual Datasets

The `clean_data` function is applied to various datasets:

- **EUR/USD Exchange Rates**: Cleaned for daily frequency.
- **EUR Inflation**: Processed for monthly data.
- **GDP Data**: Both EUR and USD GDP data are cleaned and adjusted for quarterly data.
- **Interest Rates**: Cleaned similarly to align with other datasets.

3.5 Expanding Monthly to Daily Data

For the USD inflation data, the code expands monthly values to daily values by replicating the monthly inflation rate across each day of the month. This is achieved using the `mutate` and `unnest` functions from the `dplyr` package.

3.6 Aligning Dates

To ensure coherence across datasets, a common date range is determined (from the earliest to the latest date available). The function `filter_dates` filters each dataset to this common date range.

3.7 Merging Data

All cleaned datasets are merged into a single dataframe called `complete_data` using the `reduce` function from `purrr` and `full_join` from `dplyr`. This approach maintains all available data aligned by date.

3.8 Data Finalization

Finally, potential issues arising from merging are handled:

- Duplicate rows based on the date are eliminated.
- Column names are ensured to be unique.
- Character columns are converted to numeric types.
- Numeric values are rounded to two decimal places for clarity.

The result of this extensive data preparation process is the `complete_data` dataframe, which contains aligned and cleaned economic indicators for EUR and USD, ready for further analysis or visualization. The first few rows of this dataset have been inspected to verify the outcome of the preparation efforts.

Table 1: Print of Head(complete_data)

Date	EUR_USD	GDP_EUR	GDP_USD	INT_EUR	INT_USD	INFL_EUR	INFL_USD
2006-05-25	1.06	0.9	3.2	3	4.75	1.3	2.66
2006-05-26	1.07	0.9	3.2	3	4.75	1.3	2.66
2006-05-27	1.07	0.9	3.2	3	4.75	1.3	2.66
2006-05-28	1.07	0.9	3.2	3	4.75	1.3	2.66
2006-05-29	1.07	0.9	3.2	3	4.75	1.3	2.66
2006-05-30	1.07	0.9	3.2	3	4.75	1.3	2.66

4 Exploratory Data Analysis

In this section, we conducted a thorough Exploratory Data Analysis (EDA) to uncover important insights from the dataset and identify potential relationships between the key macroeconomic indicators and the EUR/USD exchange rate.

4.1 Summary Statistics

The analysis begins with the computation of summary statistics for each of the variables, including GDP, interest rates, and inflation rates, for both the Eurozone and the U.S. using the `summary()` function. This provides a quick overview of the dataset, including measures such as the minimum, maximum, mean, median, and quartiles for each numerical variable. This step is crucial for identifying outliers, central tendencies, and the distribution and behavior of these economic factors over time.

Table 2: Summary Statistics of Key Variables

Variable	Mean	Median	Min	Max
EUR/USD	1.19	1.18	0.83	1.60
GDP (EUR)	0.27	0.40	-11.1	11.7
GDP (USD)	1.96	2.20	-7.5	12.2
Interest Rate (EUR)	1.61	1.00	0.00	4.75
Interest Rate (USD)	1.99	1.25	0.25	6.50

Summary statistics (Table 2) provide an overview of the key variables, including their mean, median, minimum, and maximum values.

4.2 Correlation Analysis

Next, correlation analysis is performed to explore relationships between numeric variables. A subset of the dataset is created by excluding the `Date` column, allowing for focused analysis of numerical relationships.

```
numeric_cols <- complete_data %>% select(-Date)
```

The correlation matrix is computed using the `cor()` function, which calculates pairwise correlations between numeric columns, using only complete observations.

```
cor_matrix <- cor(numeric_cols, use = "complete.obs")
```

To visualize these correlations, the `corrplot` package is utilized, creating a color-coded plot of the correlation matrix.

```
corrplot(cor_matrix, method = "color", type = "upper", tl.col =
    "black", tl.srt = 45)
```

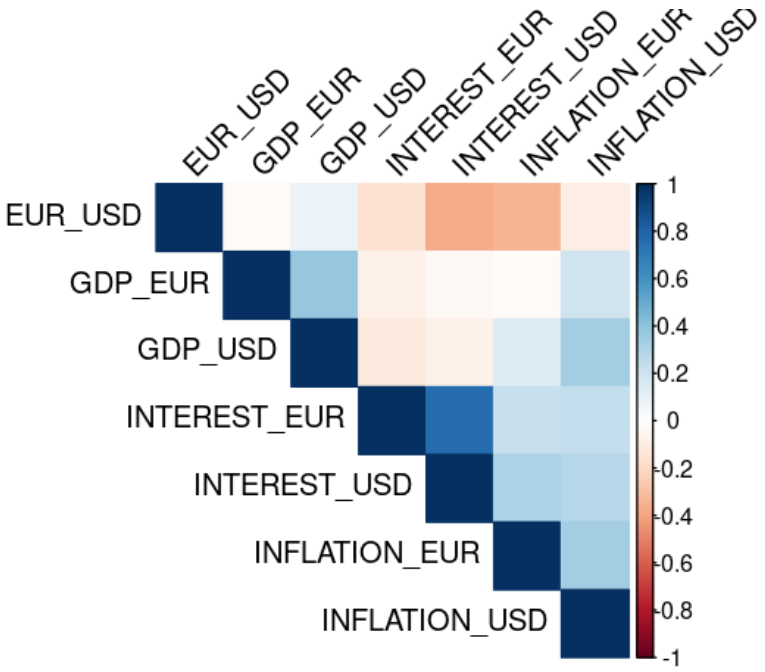


Figure 1: Correlation Matrix of Variables

Correlation analysis (Figure 1) highlights relationships between variables. High correlations between certain indicators (e.g., interest rates and exchange rates) provide valuable insights.

- **EUR/USD vs. Eurozone Inflation (-0.21981):** The correlation between the EUR/USD exchange rate and Eurozone inflation is weakly negative, suggesting a slight inverse relationship. When Eurozone inflation increases, the EUR/USD exchange rate tends to decrease, though the effect is minimal, indicating limited impact of Eurozone inflation on the exchange rate.

- **EUR/USD vs. US Inflation (0.65335):** There is a strong positive correlation between EUR/USD and US inflation, indicating that an increase in US inflation is associated with a rise in the EUR/USD exchange rate. This suggests that US inflation has a significant influence on the volatility of the EUR/USD exchange rate, likely due to changes in dollar demand as US inflation affects purchasing power.

- **EUR/USD vs. Eurozone GDP (-0.16781):** The correlation between the EUR/USD exchange rate and Eurozone GDP is weakly negative, implying that Eurozone GDP changes have little impact on the exchange rate. This suggests that fluctuations in Eurozone economic output do not significantly influence the EUR/USD exchange rate.

- **EUR/USD vs. US GDP (-0.32826):** The correlation between US GDP and the EUR/USD exchange rate is moderately negative, suggesting that when the US economy grows, the EUR/USD exchange rate tends to decline. This may be explained by increased demand for US dollars when the US economy strengthens, causing the exchange rate to drop.

- **EUR/USD vs. Eurozone Interest Rate (0.65335):** The correlation between the EUR/USD exchange rate and Eurozone interest rates is strongly positive. This indicates that higher interest rates in the Eurozone tend to strengthen the Euro, resulting in a higher EUR/USD exchange rate. Investors may be drawn to higher yields, increasing demand for the Euro.

- **EUR/USD vs. US Interest Rate (0.28968):** The correlation between the EUR/USD exchange rate and US interest rates is moderately positive. Although higher US interest rates are associated with an increase in the EUR/USD exchange rate, the effect is weaker compared to Eurozone interest rates.

4.3 EUR/USD Exchange Rate Over Time

The first plot visualizes the EUR/USD exchange rate over time using `ggplot2`. A line plot is created with points highlighted for the maximum and minimum exchange rates, aiding in the identification of significant fluctuations.

```
ggplot(complete_data, aes(x = Date, y = EUR_USD)) +
  geom_line(color = "blue") +
  geom_point(data = complete_data %>%
               filter(EUR_USD == max(EUR_USD, na.rm = TRUE) |
                      EUR_USD == min(EUR_USD, na.rm = TRUE)),
             aes(x = Date, y = EUR_USD), color = "red", size = 3) +
  labs(title = "EUR/USD Exchange Rate Over Time",
       x = "Date",
       y = "EUR/USD Exchange Rate") +
  theme_minimal()
```

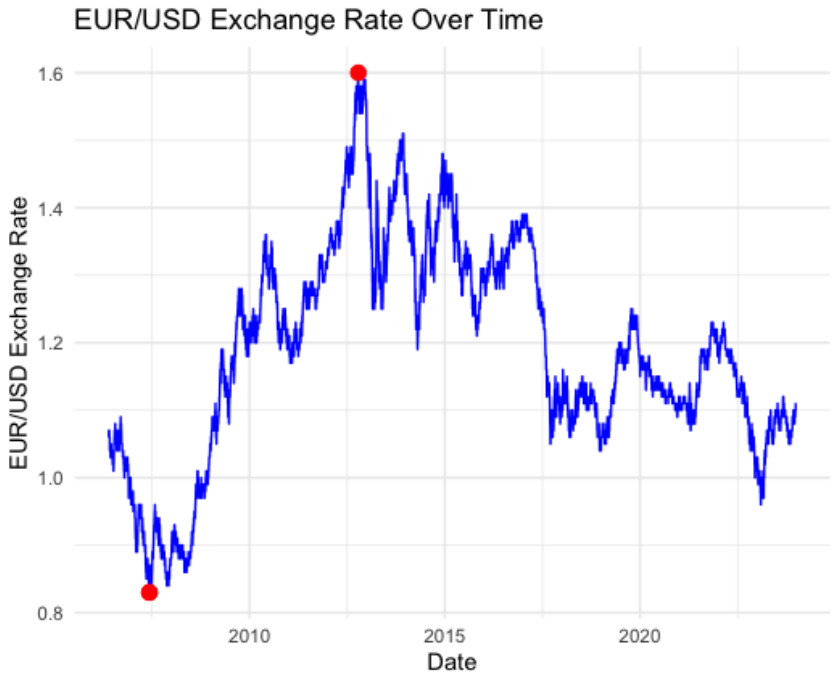


Figure 2: EUR/USD Exchange Rate Over Time

4.4 Trends in GDP, Inflation, and Interest Rates

Separate plots are created for GDP, inflation, and interest rates, each using a similar approach.

GDP Trends A line plot illustrates the trends in both EUR and USD GDP over time, using distinct colors for differentiation.

```
gdp_plot <- ggplot(complete_data, aes(x = Date)) +
  geom_line(aes(y = GDP_EUR, color = "GDP_EUR")) +
  geom_line(aes(y = GDP_USD, color = "GDP_USD")) +
  labs(title = "GDP Trends Over Time",
       x = "Date",
       y = "GDP") +
  theme_minimal() +
  scale_color_manual(values = c("GDP_EUR" = "green", "GDP_USD" =
     "purple"))
```

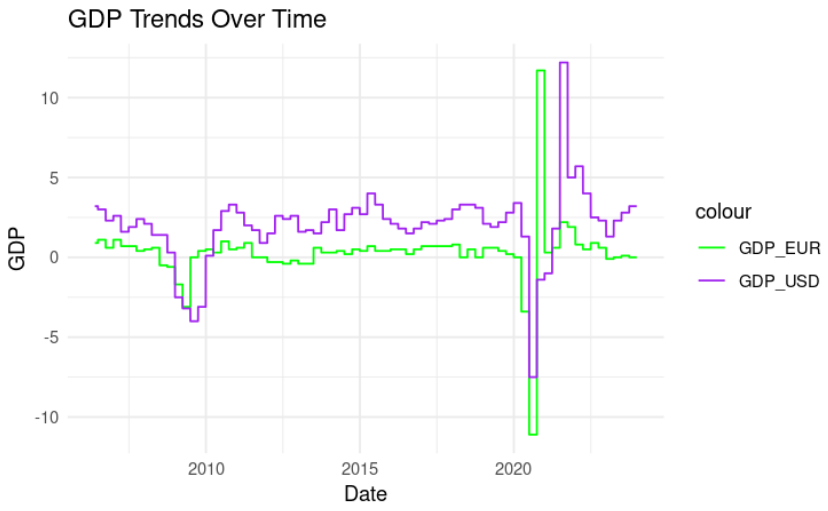


Figure 3: GDP Trends Over Time

Inflation Trends The inflation rates for EUR and USD are plotted similarly, facilitating visual comparison of inflation trends.

```
inflation_plot <- ggplot(complete_data, aes(x = Date)) +
  geom_line(aes(y = INFLATION_EUR, color = "INFLATION_EUR")) +
  geom_line(aes(y = INFLATION_USD, color = "INFLATION_USD")) +
  labs(title = "Inflation Trends Over Time",
       x = "Date",
       y = "Inflation Rate") +
  theme_minimal() +
  scale_color_manual(values = c("INFLATION_EUR" = "orange",
     "INFLATION_USD" = "brown"))
```

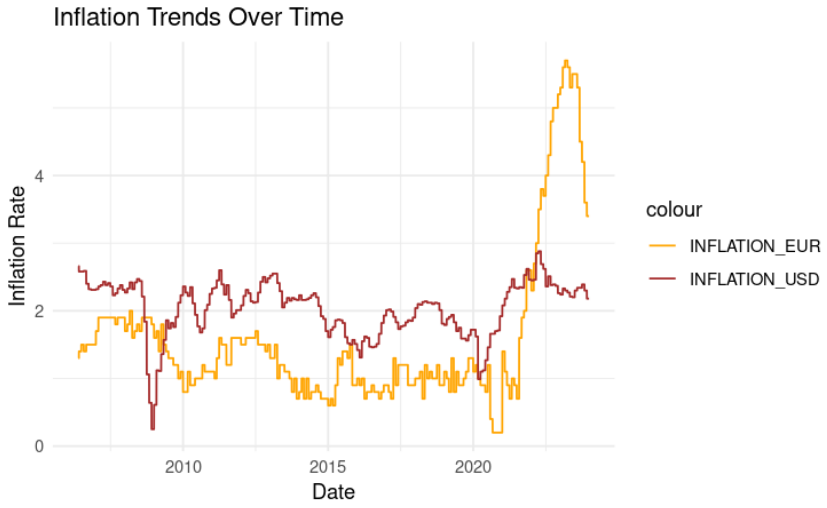


Figure 4: Inflation Rate Trends Over Time

Interest Rate Trends Finally, interest rates are visualized similarly, providing a comprehensive view of monetary conditions.

```
interest_plot <- ggplot(complete_data, aes(x = Date)) +
  geom_line(aes(y = INTEREST_EUR, color = "INTEREST_EUR")) +
  geom_line(aes(y = INTEREST_USD, color = "INTEREST_USD")) +
  labs(title = "Interest Rate Trends Over Time",
       x = "Date",
       y = "Interest Rate") +
  theme_minimal() +
  scale_color_manual(values = c("INTEREST_EUR" = "cyan",
     "INTEREST_USD" = "magenta"))
```

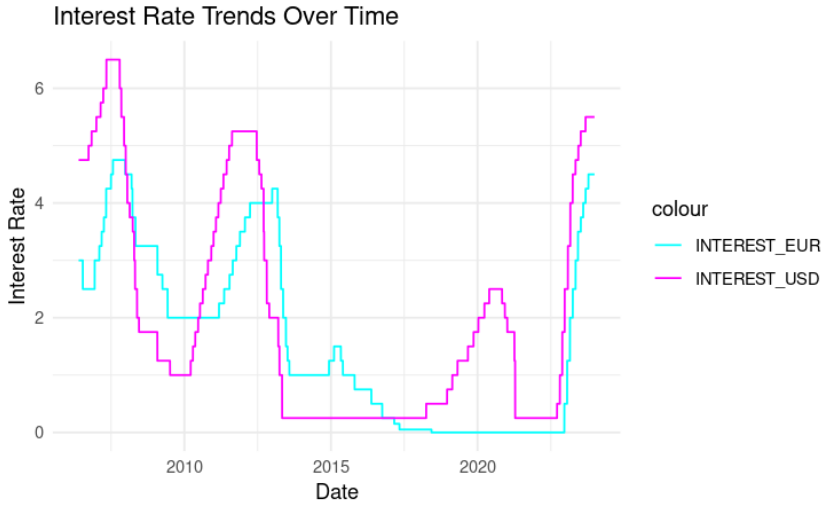


Figure 5: Interest Rate Trends Over Time

The trend plots are displayed to visualize the economic indicators over time.

4.5 Distribution Analysis

The final part of the exploratory analysis examines the distribution of numeric variables. The numeric data is melted into a long format, which is suitable for plotting distributions.

```
melted_data <- melt(numeric_cols, variable.name = "Variable",
    value.name = "Value")
```

Using `ggplot2`, histograms for each variable are created to visualize their distributions, aiding in the identification of shapes, skewness, and outliers.

```
ggplot(melted_data, aes(x = Value)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "black") +
  facet_wrap(~Variable, scales = "free") +
  labs(title = "Variable Distributions", x = "Value", y = "Frequency")
     +
  theme_minimal()
```

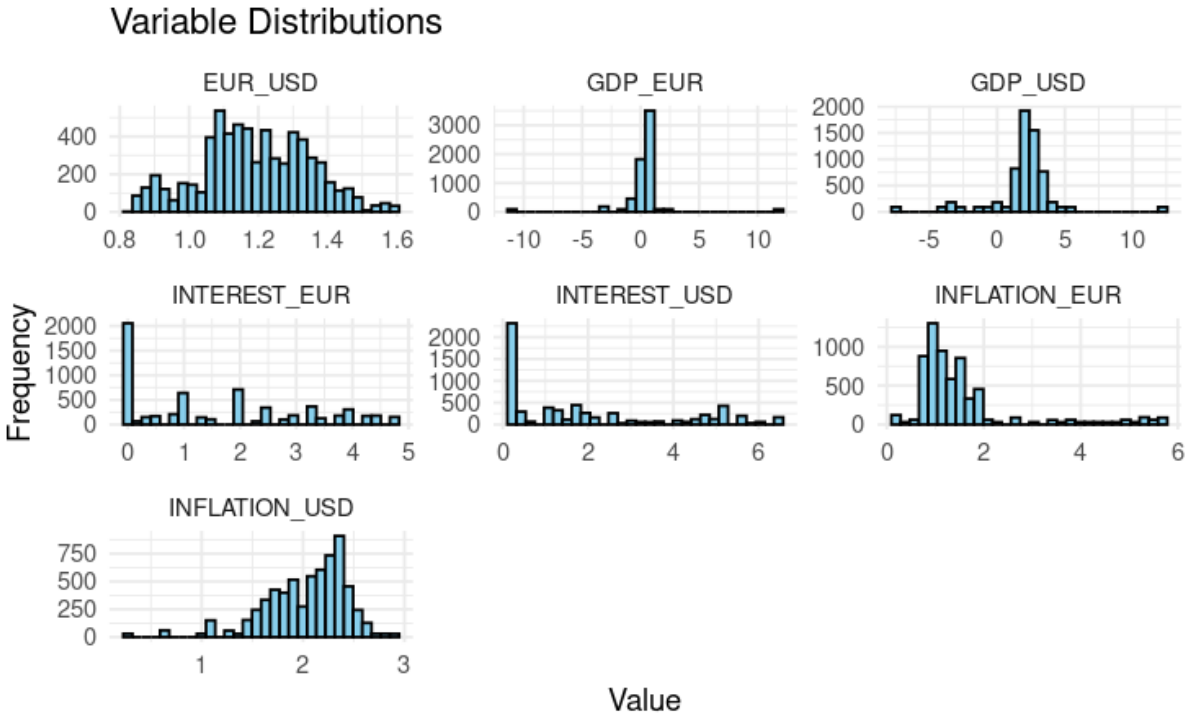


Figure 6: Visualization of the Variable Distribuition

Overall, our EDA provided a structured approach to understanding the dataset, revealing important trends and relationships that will serve as a foundation for our further analysis of the EUR/USD exchange rate.

5 Modeling and Predictions

5.1 Data Scaling

The analysis begins with scaling the data, a crucial preprocessing step to ensure that all features contribute equally to the model. The `preProcess` function from the `caret` package is used to center and scale the numeric features of the `complete_data` dataset, excluding the date column.

```
scaled_data <- preProcess(complete_data[, -1], method = c("center",
    "scale"))
complete_data_scaled <- predict(scaled_data, complete_data[, -1])
complete_data_scaled <- cbind(Date = complete_data$Date,
    complete_data_scaled)
```

This scaling helps normalize the data, making it suitable for linear models and regularization techniques that are sensitive to the scale of input features.

5.2 Preparing Predictors

Next, a new dataset, `predictors_data`, is created, which excludes the `EUR_USD` column since it will be the target variable for prediction.

```
predictors_data <- complete_data_scaled %>% select(-EUR_USD)
```

5.3 Linear Regression Model

Linear regression was used to model the relationship between the EUR/USD exchange rate and macroeconomic indicators. The code then fits a linear regression model using all predictors except the date. The formula `EUR_USD ~ .` indicates that EUR/USD is regressed on all other available variables.

```
lm_model <- lm(EUR_USD ~ ., data = complete_data_scaled %>%
    select(-Date))
summary(lm_model)
```

Call:

```
lm(formula = EUR\_USD ~ ., data = complete\_data\_scaled \%>\% select(-Date))
```

Residuals:

Min	1Q	Median	3Q	Max
-2.36363	-0.64682	0.06115	0.52635	2.27287

The `summary()` function provides insights into the model's coefficients, statistical significance, and overall fit.

Performance Evaluation To evaluate the performance of the linear regression model, predictions are made, and the Mean Squared Error (MSE) is calculated. The MSE quantifies the average squared difference between actual and predicted values, serving as a measure of the model's accuracy.

```
lm_pred <- predict(lm_model, complete_data_scaled %>% select(-EUR_USD))
mse_lm <- mse(complete_data_scaled$EUR_USD, lm_pred)
cat("Linear Regression MSE:", mse_lm, "\n")
```

Linear Regression MSE: 0.7462606

5.4 Lasso and Ridge Regression Models

We then explored Lasso and Ridge regression, both of which are regularization techniques that help prevent overfitting by adding penalties to the loss function.

Lasso Regression For Lasso, the `cv.glmnet` function is used to perform cross-validated Lasso regression, allowing for feature selection by potentially shrinking some coefficients to zero (helped in feature selection by penalizing less important variables).

```
X <- as.matrix(predictors_data[, -1])
y <- complete_data_scaled$EUR_USD
lasso_model <- cv.glmnet(X, y, alpha = 1)
lasso_pred <- predict(lasso_model, s = "lambda.min", newx = X)
mse_lasso <- mse(y, lasso_pred)
cat("Lasso Regression MSE:", mse_lasso, "\n")
```

The coefficients of the Lasso model are examined to identify which features are deemed important.

Lasso Regression MSE: 0.7462731

Ridge Regression Similar steps are followed for Ridge regression, using `cv.glmnet` with `alpha = 0`. This model penalizes large coefficients but does not reduce them to zero.

```
ridge_model <- cv.glmnet(X, y, alpha = 0)
ridge_pred <- predict(ridge_model, s = "lambda.min", newx = X)
mse_ridge <- mse(y, ridge_pred)
cat("Ridge Regression MSE:", mse_ridge, "\n")
```

Ridge Regression MSE: 0.7478506

5.5 Residual Analysis

Residuals from the Ridge regression model are calculated and plotted to assess the model's fit. This step helps in diagnosing potential issues with the model, such as non-linearity or heteroscedasticity.

```
residuals_ridge <- y - ridge_pred
plot(residuals_ridge, main = "Residuals for Ridge Regression", ylab =
    "Residuals", xlab = "Index")
abline(h = 0, col = "red")
```

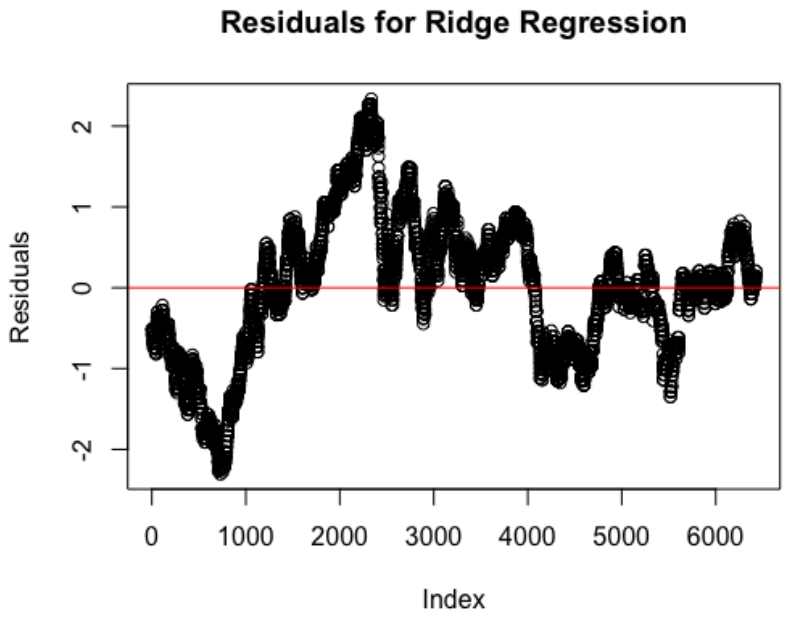


Figure 7: Residuals Ridge for Regression

5.6 Interaction Model

An interaction model is then constructed to capture the combined effects of GDP and interest rates on the EUR/USD exchange rate. This model includes interaction terms between the variables.

```
interaction_model <- lm(EUR_USD ~ GDP_USD * INTEREST_USD + GDP_EUR *
    INTEREST_EUR, data = complete_data_scaled)
summary(interaction_model)
```

5.7 Decision Tree Model

We fitted a decision tree model using the `rpart` function, which can capture non-linear relationships between predictors and the target variable.

```
tree_model <- rpart(EUR_USD ~ ., data = complete_data_scaled %>%
    select(-Date), method = "anova")
```

After fitting the model, predictions are made, and MSE is computed to evaluate performance.

```
tree_pred <- predict(tree_model, complete_data_scaled %>%
    select(-Date))
mse_tree <- mse(complete_data_scaled$EUR_USD, tree_pred)
cat("Decision Tree MSE:", mse_tree, "\n")
```

Decision Tree MSE: 0.1352307

5.8 Model Performance Summary

A summary of the performance of all models (Linear, Lasso, Ridge, Decision Tree) is compiled into a data frame, allowing for easy comparison.

Model	MSE
Linear Regression	0.7462606
Lasso Regression	0.7462731
Ridge Regression	0.7478506
Decision Tree	0.1352307

```
performance_summary <- data.frame(
  Model = c("Linear Regression", "Lasso Regression", "Ridge
      Regression", "Decision Tree"),
  MSE = c(mse_lm, mse_lasso, mse_ridge, mse_tree)
)
print(performance_summary)
```

5.9 Tree Validation

To ensure the robustness of our decision tree model, you split the dataset into training and testing sets. This validation helps assess whether the model is overfitting the training data.

```
set.seed(123)
train_index <- createDataPartition(complete_data_scaled$EUR_USD, p =
    0.8, list = FALSE)
train_data <- complete_data_scaled[train_index, ]
test_data <- complete_data_scaled[-train_index, ]
```

We fitted the decision tree model on the training data and evaluate its performance on the test set, calculating the MSE.

```
tree_model <- rpart(EUR_USD ~ ., data = train_data %>% select(-Date),
    method = "anova")
tree_test_pred <- predict(tree_model, test_data %>% select(-Date))
test_mse_tree <- mse(test_data$EUR_USD, tree_test_pred)
cat("Decision Tree Test MSE:", test_mse_tree, "\n")
```

Decision Tree Test MSE: 0.1359242

5.10 Pruning the Tree

To combat overfitting, you prune the decision tree based on the complexity parameter that minimizes cross-validated error.

```
pruned_tree <- prune(tree_model, cp =
    tree_model$cptable[which.min(tree_model$cptable[,"xerror"]), "CP"])
rpart.plot(pruned_tree)
```

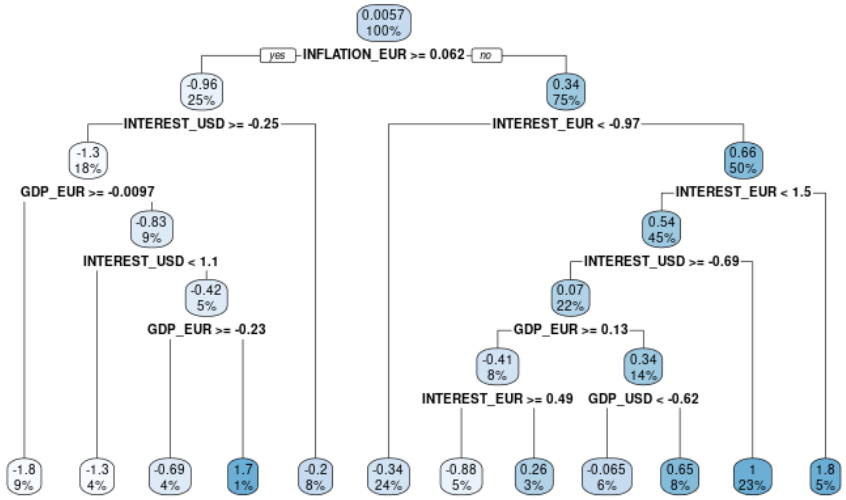


Figure 8: Pruned Decision Tree

5.11 Ensemble Methods: Random Forest

We then implemented a Random Forest model, an ensemble method that aggregates predictions from multiple decision trees to improve accuracy and control overfitting.

```
rf_model <- randomForest(EUR_USD ~ ., data = train_data %>%
    select(-Date))
rf_test_pred <- predict(rf_model, test_data %>% select(-Date))
rf_test_mse <- mse(test_data$EUR_USD, rf_test_pred)
cat("Random Forest Test MSE:", rf_test_mse, "\n")
```

Random Forest Test MSE: 0.01101849

5.12 Nonlinear Relationships

You explore nonlinear relationships by fitting polynomial models and expanded interaction models, allowing for more complex interactions between the features.

```
polynomial_model <- lm(EUR_USD ~ poly(GDP_EUR, 2) + poly(INTEREST_EUR,
    2) +
                         poly(GDP_USD, 2) + poly(INTEREST_USD, 2) +
                         GDP_EUR:INTEREST_EUR + GDP_USD:INTEREST_USD,
                         data = train_data)
```

Feature	Estimate	Interpretation
GDP_EUR	-0.3412079	Negative effect on EUR/USD
INTEREST_EUR	0.1965838	Positive effect on EUR/USD
GDP_USD	0.1728594	Positive effect on EUR/USD
INTEREST_USD	-0.5138366	Negative effect on EUR/USD
GDP_EUR:INTEREST_EUR	-0.2753846	Negative interaction effect
GDP_USD:INTEREST_USD	0.1042721	Positive interaction effect

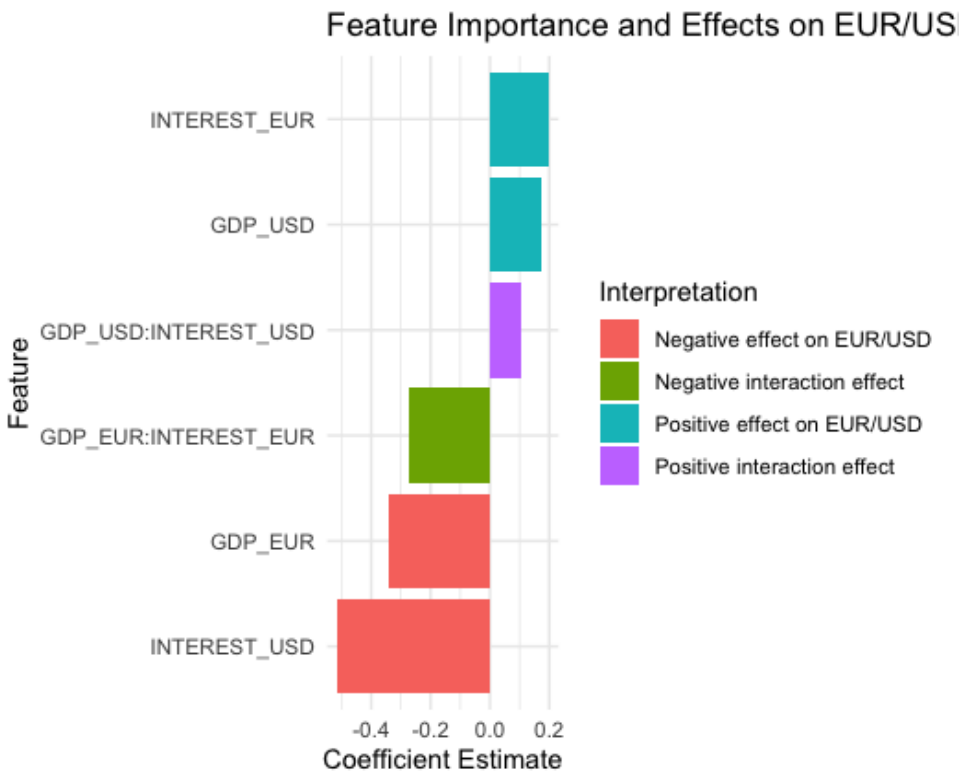


Figure 9: Feature Importance and Effects on EUR/USD

5.13 Final Model Selection

The final model is fitted based on the best features identified throughout the analysis, including interaction terms.

```
final_model <- lm(EUR_USD ~ GDP_EUR + INTEREST_EUR + GDP_USD +
    INTEREST_USD +
                  GDP_EUR:INTEREST_EUR + GDP_USD:INTEREST_USD,
                  data = train_data)
summary(final_model)
```

5.14 Data Engineering: Outliers and Clustering

The last part of your code involves K-Means clustering to investigate the presence of outliers and to segment the data into clusters based on similar characteristics.

```
kmeans_result <- kmeans(complete_data_scaled[, -1], centers = 3)
```

We summarized and visualized the clusters, examining the average values of different economic indicators within each cluster.

```
cluster_summary <- complete_data_scaled %>%
  group_by(Cluster) %>%
  summarise(
    Avg_EUR_USD = mean(EUR_USD, na.rm = TRUE),
    Avg_GDP_EUR = mean(GDP_EUR, na.rm = TRUE),
    Avg_GDP_USD = mean(GDP_USD, na.rm = TRUE),
    Avg_INTEREST_EUR = mean(INTEREST_EUR, na.rm = TRUE),
    Avg_INTEREST_USD = mean(INTEREST_USD, na.rm = TRUE),
    Avg_INFLATION_EUR = mean(INFLATION_EUR, na.rm = TRUE),
    Avg_INFLATION_USD = mean(INFLATION_USD, na.rm = TRUE)
  )
print(cluster_summary)
```

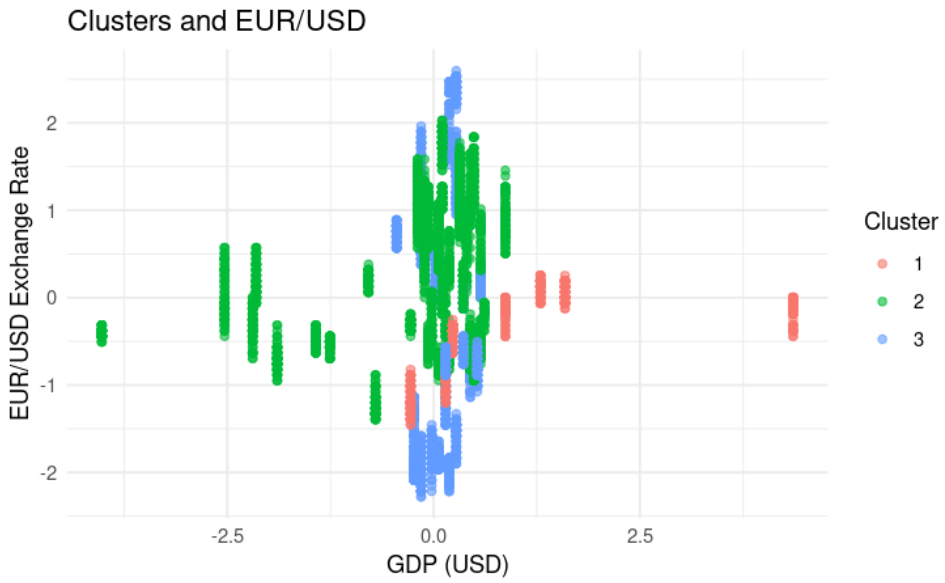


Figure 10: Clusters and EUR/USD

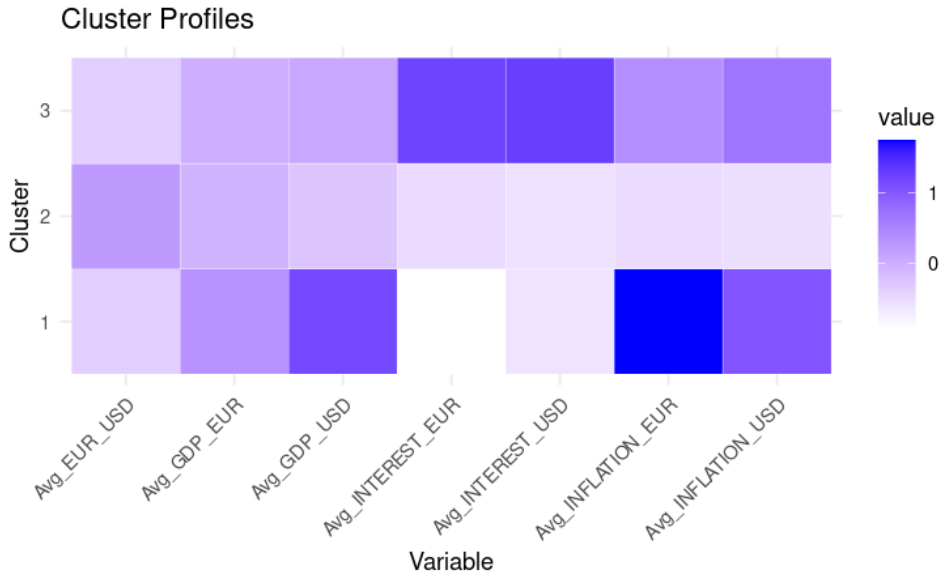


Figure 11: Clusters of EUR/USD Exchange Rate

6 Conclusion

The analysis of the EUR/USD exchange rate presented in this project helped us to address ours research questions. In examining the interplay between macroeconomic indicators, geopolitical events, and market sentiment, we gain a richer understanding of the dynamics that drive currency fluctuations.

Macroeconomic Indicators

Macroeconomic indicators such as interest rates, inflation, and GDP play a pivotal role in shaping the EUR/USD exchange rate. Our findings underscore that higher interest rates in the Eurozone typically enhance the Euro's value relative to the Dollar. Conversely, the strong correlation between U.S. inflation and the exchange rate suggests that shifts in American economic conditions can significantly impact the Euro's performance. This highlights the interconnectedness of economies in today's globalized financial landscape, where decisions made by central banks in one region can reverberate across borders.

Geopolitical Events

The impact of geopolitical events further complicates the narrative. Political stability, trade negotiations, and international relations are critical in shaping investor confidence and market behavior. For instance, elections or significant policy shifts can lead to immediate reactions in currency markets as traders reassess risks and opportunities. This volatility emphasizes the need for investors to remain attuned not only to economic data but also to the broader political environment.

Market Sentiment

Market sentiment, often driven by speculative trading, adds another layer of complexity. The analysis illustrates how traders' perceptions and reactions to news can create significant volatility in the currency market. This phenomenon raises questions about the extent to which rational economic indicators govern currency values versus the influence of psychological factors and herd behavior among traders.

Methodological Insights

From a methodological perspective, the project utilized a variety of analytical techniques, ranging from traditional linear regression to more complex models like decision trees and Random Forests. The superior performance of the decision tree model in capturing non-linear relationships exemplifies the value of employing diverse modeling approaches in financial data science. Moreover, the integration of machine learning techniques not only enhances predictive accuracy but also reveals intricate patterns that might be overlooked by conventional methods.

Implications for Stakeholders

The insights gleaned from this analysis are crucial for a range of stakeholders, including traders, policymakers, and economists. For traders, understanding the nuanced relationships between economic indicators and the currency pair can inform more strategic decision-making. Policymakers can leverage these insights to anticipate market reactions to economic policies, while economists may find value in further researching the implications of macroeconomic stability on currency values.

Final Thoughts

In conclusion, the EUR/USD exchange rate reflects economic health and investor sentiment worldwide. This project not only answers the critical research questions but also opens avenues for further exploration into the intricate relationships that define currency dynamics. As the financial landscape continues to evolve, ongoing research and analysis will be essential in navigating the complexities of currency markets and understanding their broader implications for global economic stability.

7 References

- ECB Data Portal. [online] Available at: https://data.ecb.europa.eu/.
- Federal Reserve Bank of St. Louis (2023). Inflation, Consumer Prices for the United States. [online] Stlouisfed.org. Available at: https://fred.stlouisfed.org/series/FPCPITOTLZGUSA.

8 Team Members Contribution

The entirety of the job was carried out by Federica and Lucia, with Vasil and Andrian helping.

Appendix A: R Scripts and Output

Export of the R Script Implemented and its Output

```
#
#                         EUR/GDP Analysis

# Group: Lucia de Alarcon, Federica Malamisura, Vasil Naumov and Andrian Gutium
```

```
# (0) SET WORKING DIRECTORY AND LOAD PACKAGES
```

```
setwd("/cloud/project")
print(getwd())
```

```
## [1] "/cloud/project"
```

```
# Load the necessary packages
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(dplyr)
library(lubridate)
library(ggplot2)
library(readr)
library(stringr)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(zoo)
```

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
```

```
##
##     as.Date, as.Date.numeric
```

```
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
library(readxl)
library(conflicted)
library(caret)
```

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

```
library(Metrics)
library(rpart)
library(rpart.plot)
library(randomForest)
```

```
## randomForest 4.7-1.2
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(boot)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(cluster)
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
# Resolve conflicts by preferring dplyr's filter over stats' filter
conflicts_prefer(dplyr::filter)
```

```
## [conflicted] Will prefer dplyr::filter over any other package.
```

```
# (1) PREPARE DATA
```

```
# Load all the data
eur_usd <- read_excel("eurusd.xlsx")
gdp_eur <- read_excel("gdpeur.xlsx")
```

```
## New names:
## * `` -> `...3`
```

```
eur_interest <- read_excel("interesteur.xlsx")
```

```
## New names:
## * `` -> `...3`
```

```r
eur_inflation <- read_excel("inflationeur.xlsx")
```

```
## New names:
## * `` -> `...3`
```

```r
gdp_usd <- read_excel("gdpusa.xlsx")
```

```
## New names:
## * `` -> `...3`
```

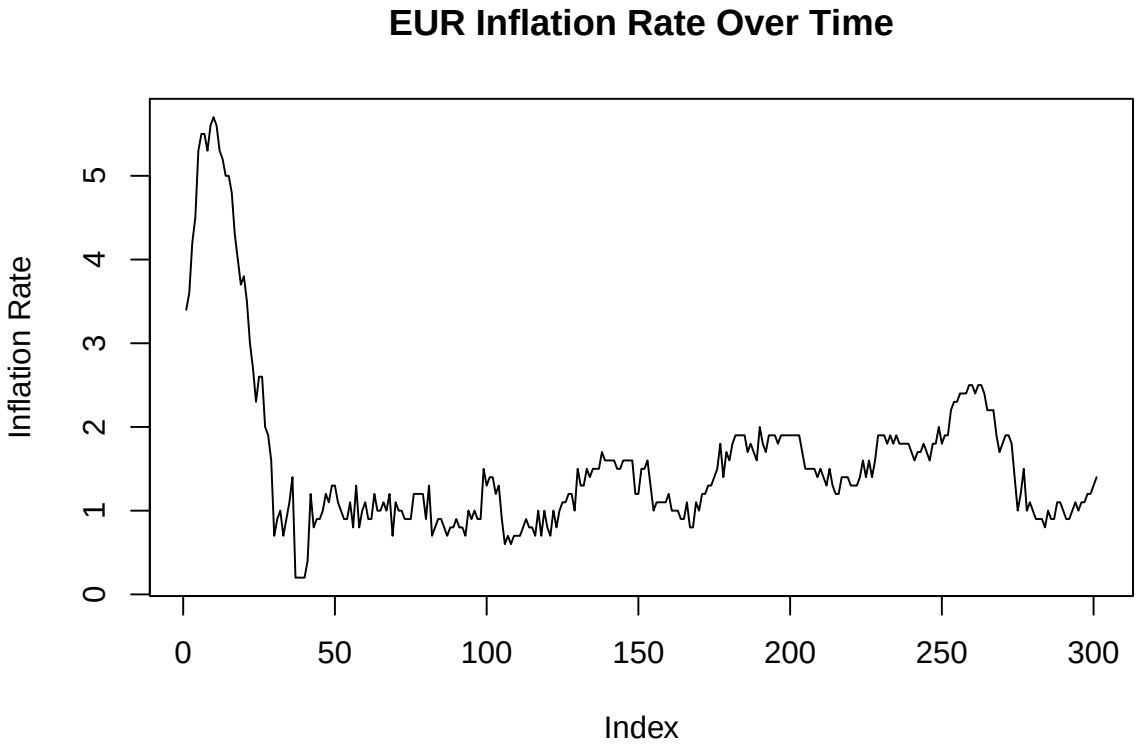
```r
usd_interest <- read_excel("interestusd.xlsx")
```

```
## New names:
## * `` -> `...3`
```

```r
usd_inflation <- read.csv("T10YIEM.csv")

# Assign proper column names for USD Inflation
colnames(usd_inflation) <- c("Date", "Inflation_Rate")

# Data Cleaning Function to avoid repetition
clean_data <- function(df, remove_rows, select_cols, new_col_names, start_date, by) {
  # Remove specified header rows and select desired columns
  df <- df[-c(1:remove_rows), select_cols]

  # Rename columns
  colnames(df) <- new_col_names

  # Calculate the number of rows after removal
  length_out <- nrow(df)

  # Generate dates based on the specified start_date and frequency
  dates <- seq(
    from = as.Date(start_date, format = "%Y-%m-%d"),
    by = by,
    length.out = length_out
  )

  # Assign the dates to the Date column
  df$Date <- dates

  # Verify that the length of dates matches the number of rows
  if(length(dates) != nrow(df)) {
    stop(paste("Length of dates (", length(dates),
               ") does not match number of rows in dataframe (",
               nrow(df), ").", sep = ""))
  }

  return(as.data.frame(df))
}

# Clean EUR/USD data
eur_usd <- clean_data(
  df = eur_usd,
  remove_rows = 7,
  select_cols = 1:2,
```

```
  new_col_names = c("Date", "EUR_USD"),
  start_date = "2023-12-28",
  by = "-1 day"
)

# Clean EUR Inflation data
eur_inflation <- clean_data(
  df = eur_inflation,
  remove_rows = 5,
  select_cols = 1:2,
  new_col_names = c("Date", "Inflation_Rate"),
  start_date = "2023-12-01",
  by = "-1 month"
)
rownames(eur_inflation) <- eur_inflation$Date

# Clean GDP EUR data (Quarterly)
gdp_eur <- clean_data(
  df = gdp_eur,
  remove_rows = 5,
  select_cols = 1:2,
  new_col_names = c("Date", "GDP_EUR"),
  start_date = "2023-12-31",
  by = "-3 month"
)
rownames(gdp_eur) <- gdp_eur$Date

# Clean EUR Interest data
eur_interest <- clean_data(
  df = eur_interest,
  remove_rows = 5,
  select_cols = 1:2,
  new_col_names = c("Date", "INTEREST_EUR"),
  start_date = "2023-12-29",
  by = "-1 day"
)

# Clean USD Inflation data
# Remove specified header rows (first 5 rows)
usd_inflation <- usd_inflation[-c(1:5), c(1,2)]
colnames(usd_inflation) <- c("Date", "Inflation_Rate")  # Ensure correct naming

# Convert Date column to Date type
usd_inflation$Date <- as.Date(usd_inflation$Date, format = "%Y-%m-%d")

# Handle any potential NA values in Date
usd_inflation <- usd_inflation %>% filter(!is.na(Date))

# Ensure no remaining NA in Inflation_Rate
if(all(is.na(usd_inflation$Inflation_Rate))) {
  stop("All values in usd_inflation$Inflation_Rate are NA.")
}
```

```r
# Generate daily dates for USD Inflation
# Assuming 'Inflation_Rate' is monthly, expand to daily by carrying forward the monthly rate
daily_usd_inflation <- usd_inflation %>%
  mutate(
    DailyDate = map(Date, ~ seq(.x, by = "day", length.out = days_in_month(.x)))
  ) %>%
  unnest(DailyDate) %>%
  select(DailyDate, Inflation_Rate) %>%
  rename(Date = DailyDate)

# Clean GDP USD data (Quarterly)
gdp_usd <- clean_data(
  df = gdp_usd,
  remove_rows = 5,
  select_cols = 1:2,
  new_col_names = c("Date", "GDP_USD"),
  start_date = "2023-12-31",
  by = "-3 month"
)
rownames(gdp_usd) <- gdp_usd$Date

# Clean USD Interest data
usd_interest <- clean_data(
  df = usd_interest,
  remove_rows = 5,
  select_cols = 1:2,
  new_col_names = c("Date", "INTEREST_USD"),
  start_date = "2023-12-29",
  by = "-1 day"
)

# Data Cleaning for EUR Inflation: Ensure 'Inflation_Rate' exists
colnames(eur_inflation) <- c("Date", "Inflation_Rate")

# Plotting Inflation Rates (After Correcting Column Names)
# Before plotting, ensure that 'Inflation_Rate' contains finite values
if(all(is.na(eur_inflation$Inflation_Rate))){
  stop("All values in eur_inflation$Inflation_Rate are NA.")
} else {
  plot(eur_inflation$Inflation_Rate, type = "l", main = "EUR Inflation Rate Over Time",
       xlab = "Index", ylab = "Inflation Rate")
}
```




```r
# Create a daily dataset for EUR Inflation
eur_inflation$Date <- as.Date(eur_inflation$Date, format = "%Y-%m-%d")
daily_eur_inflation <- eur_inflation %>%
  mutate(
    DailyDate = map(Date, ~ seq(.x, by = "day", length.out = days_in_month(.x)))
  ) %>%
  unnest(DailyDate) %>%
  select(DailyDate, Inflation_Rate) %>%
  rename(Date = DailyDate)

# Ensure daily_usd_inflation has correct Date type
daily_usd_inflation$Date <- as.Date(daily_usd_inflation$Date, format = "%Y-%m-%d")

# Function to expand quarterly GDP data to daily
expand_quarterly_to_daily <- function(df, value_col) {
  df <- df %>% arrange(Date)

  # Ensure the value_col is numeric
  if(!is.numeric(df[[value_col]])) {
    # Attempt to convert to numeric after removing commas and other non-numeric characters
    df[[value_col]] <- as.numeric(gsub(",", "", df[[value_col]]))

    # Check for conversion issues
    if(any(is.na(df[[value_col]]))) {
      warning(paste("Conversion to numeric introduced NA values in", value_col))
    }
  }

  daily_df <- data.frame(Date = as.Date(character()), Value = numeric())
```

```
  for(i in 1:(nrow(df)-1)) {
    start_date <- df$Date[i]
    end_date <- df$Date[i+1] - days(1)
    daily_dates <- seq(from = start_date, to = end_date, by = "day")

    # Replicate the value_col for each day
    temp_data <- data.frame(Date = daily_dates, Value = rep(df[[value_col]][i], length(daily_dates)))

    # Append to daily_df
    daily_df <- bind_rows(daily_df, temp_data)
  }

  # Handle the last quarter by extending to the endpoint (assuming 3 months)
  last_start_date <- df$Date[nrow(df)]
  end_date_last <- last_start_date + months(3) - days(1)
  daily_dates_last <- seq(from = last_start_date, to = end_date_last, by = "day")

  temp_data_last <- data.frame(Date = daily_dates_last, Value = rep(df[[value_col]][nrow(df)], length(da

  # Append the last set of data
  daily_df <- bind_rows(daily_df, temp_data_last)

  return(daily_df)
}

# Expand GDP EUR and GDP USD to daily data
daily_gdp_eur <- expand_quarterly_to_daily(gdp_eur, "GDP_EUR")
daily_gdp_usd <- expand_quarterly_to_daily(gdp_usd, "GDP_USD")

# Rename columns for clarity
colnames(eur_usd) <- c("Date", "EUR_USD")
colnames(gdp_eur) <- c("Date", "GDP_EUR")
colnames(gdp_usd) <- c("Date", "GDP_USD")
colnames(eur_interest) <- c("Date", "INTEREST_EUR")
colnames(usd_interest) <- c("Date", "INTEREST_USD")

# Ensure all Date columns are Date type
complete_data_frames <- list(
  eur_usd,
  daily_gdp_eur,
  daily_gdp_usd,
  eur_interest,
  usd_interest,
  daily_eur_inflation,
  daily_usd_inflation
)

complete_data_frames <- lapply(complete_data_frames, function(df) {
  df$Date <- as.Date(df$Date)
  return(df)
})

# Assign back to variables
```

```r
eur_usd <- complete_data_frames[[1]]
daily_gdp_eur <- complete_data_frames[[2]]
daily_gdp_usd <- complete_data_frames[[3]]
eur_interest <- complete_data_frames[[4]]
usd_interest <- complete_data_frames[[5]]
daily_eur_inflation <- complete_data_frames[[6]]
daily_usd_inflation <- complete_data_frames[[7]]

# Align all datasets to the same date range
start_point <- max(
  min(eur_usd$Date, na.rm = TRUE),
  min(daily_gdp_eur$Date, na.rm = TRUE),
  min(daily_gdp_usd$Date, na.rm = TRUE),
  min(eur_interest$Date, na.rm = TRUE),
  min(usd_interest$Date, na.rm = TRUE),
  min(daily_eur_inflation$Date, na.rm = TRUE),
  min(daily_usd_inflation$Date, na.rm = TRUE)
)

end_point <- min(
  max(eur_usd$Date, na.rm = TRUE),
  max(daily_gdp_eur$Date, na.rm = TRUE),
  max(daily_gdp_usd$Date, na.rm = TRUE),
  max(eur_interest$Date, na.rm = TRUE),
  max(usd_interest$Date, na.rm = TRUE),
  max(daily_eur_inflation$Date, na.rm = TRUE),
  max(daily_usd_inflation$Date, na.rm = TRUE)
)

# Function to filter dataframes based on date range
filter_dates <- function(df, start, end) {
  df %>% filter(Date >= start & Date <= end)
}

# Apply date filtering
eur_usd <- filter_dates(eur_usd, start_point, end_point)
daily_gdp_eur <- filter_dates(daily_gdp_eur, start_point, end_point)
daily_gdp_usd <- filter_dates(daily_gdp_usd, start_point, end_point)
eur_interest <- filter_dates(eur_interest, start_point, end_point)
usd_interest <- filter_dates(usd_interest, start_point, end_point)
daily_eur_inflation <- filter_dates(daily_eur_inflation, start_point, end_point)
daily_usd_inflation <- filter_dates(daily_usd_inflation, start_point, end_point)

# Merge all the data into a single dataframe
complete_data <- reduce(
  list(
    eur_usd,
    daily_gdp_eur,
    daily_gdp_usd,
    eur_interest,
    usd_interest,
    daily_eur_inflation,
    daily_usd_inflation
```

```r
  ),
  full_join, by = "Date"
)

# Handle any duplicated columns due to merging
complete_data <- complete_data %>%
  arrange(Date) %>%
  distinct(Date, .keep_all = TRUE)

# Make column names unique (if necessary)
names(complete_data) <- make.unique(names(complete_data))

# Convert all character columns to numeric
complete_data <- complete_data %>%
  mutate(across(where(is.character), ~ as.numeric(.)))

# Round all numeric columns to 2 decimal places
complete_data <- complete_data %>%
  mutate(across(where(is.numeric), ~ round(.x, 2)))

# Rename columns after rounding for clarity
colnames(complete_data) <- c(
  "Date",
  "EUR_USD",
  "GDP_EUR",
  "GDP_USD",
  "INTEREST_EUR",
  "INTEREST_USD",
  "INFLATION_EUR",
  "INFLATION_USD"
)

# Check the first few rows
print(head(complete_data))
```

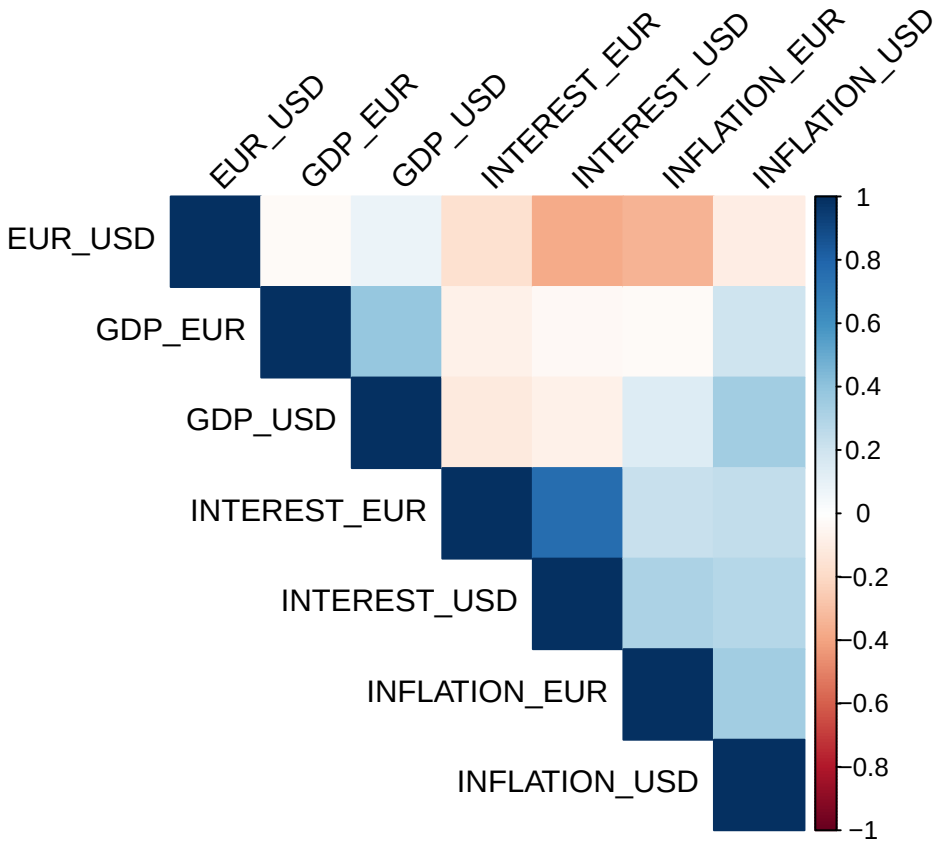
```
##         Date EUR_USD GDP_EUR GDP_USD INTEREST_EUR INTEREST_USD INFLATION_EUR
## 1 2006-05-25    1.06     0.9     3.2            3         4.75           1.3
## 2 2006-05-26    1.07     0.9     3.2            3         4.75           1.3
## 3 2006-05-27    1.07     0.9     3.2            3         4.75           1.3
## 4 2006-05-28    1.07     0.9     3.2            3         4.75           1.3
## 5 2006-05-29    1.07     0.9     3.2            3         4.75           1.3
## 6 2006-05-30    1.07     0.9     3.2            3         4.75           1.3
##   INFLATION_USD
## 1          2.66
## 2          2.66
## 3          2.66
## 4          2.66
## 5          2.66
## 6          2.66
```

```r
# (2) EXPLORATORY ANALYSIS
```

```r
# Summary Statistics
print(summary(complete_data))
```

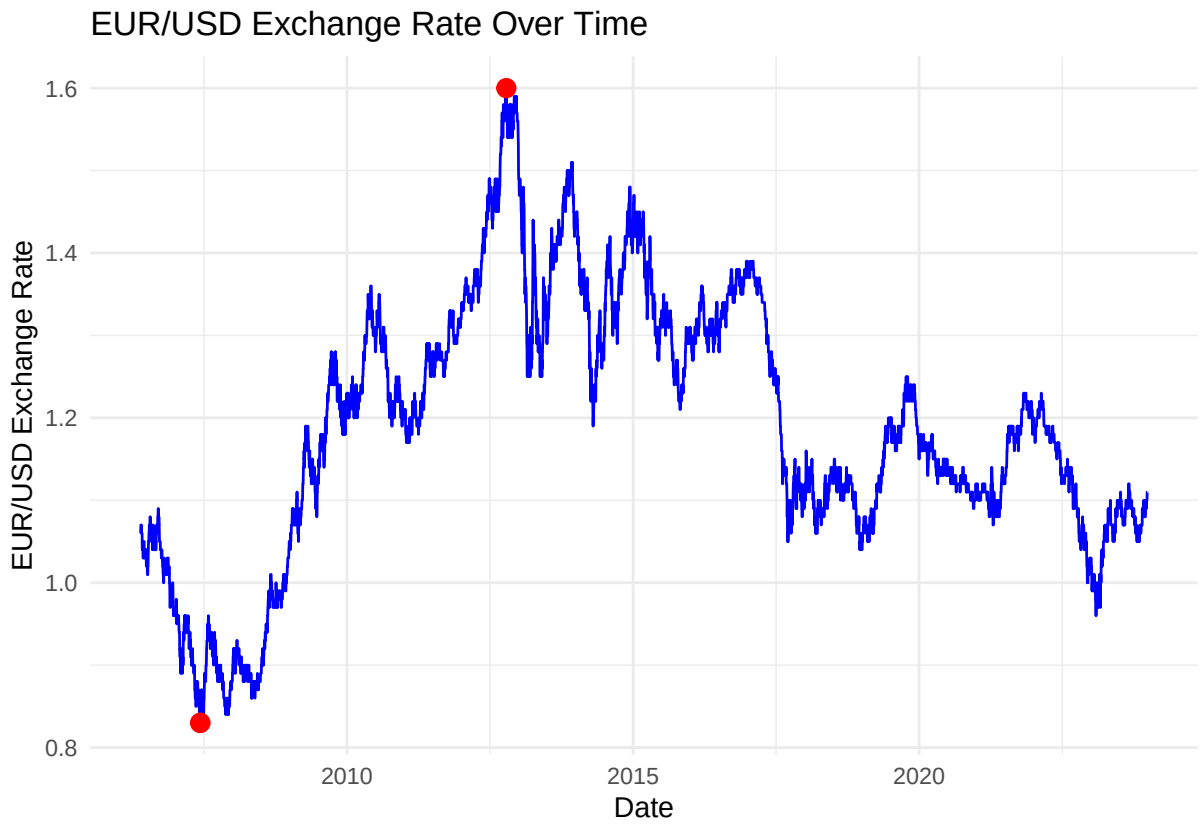
```
##       Date                EUR_USD         GDP_EUR             GDP_USD
##  Min.   :2006-05-25   Min.   :0.83   Min.   :-11.1000   Min.   :-7.500
##  1st Qu.:2010-10-17   1st Qu.:1.09   1st Qu.:  0.0000   1st Qu.: 1.600
##  Median :2015-03-12   Median :1.18   Median :  0.4000   Median : 2.200
##  Mean   :2015-03-12   Mean   :1.19   Mean   :  0.2702   Mean   : 1.956
##  3rd Qu.:2019-08-04   3rd Qu.:1.31   3rd Qu.:  0.6000   3rd Qu.: 2.800
##  Max.   :2023-12-28   Max.   :1.60   Max.   : 11.7000   Max.   :12.200
##   INTEREST_EUR     INTEREST_USD    INFLATION_EUR    INFLATION_USD
##  Min.   :0.000   Min.   :0.250   Min.   :0.200   Min.   :0.250
##  1st Qu.:0.000   1st Qu.:0.250   1st Qu.:0.900   1st Qu.:1.780
##  Median :1.000   Median :1.250   Median :1.200   Median :2.130
##  Mean   :1.611   Mean   :1.995   Mean   :1.578   Mean   :2.037
##  3rd Qu.:3.000   3rd Qu.:3.500   3rd Qu.:1.700   3rd Qu.:2.330
##  Max.   :4.750   Max.   :6.500   Max.   :5.700   Max.   :2.880
```

```r
# Correlation Analysis
# Subset numeric columns for correlation
numeric_cols <- complete_data %>% select(-Date)

# Compute correlation matrix
cor_matrix <- cor(numeric_cols, use = "complete.obs")

# Plot the correlation matrix
corrplot(cor_matrix, method = "color", type = "upper", tl.col = "black", tl.srt = 45)
```



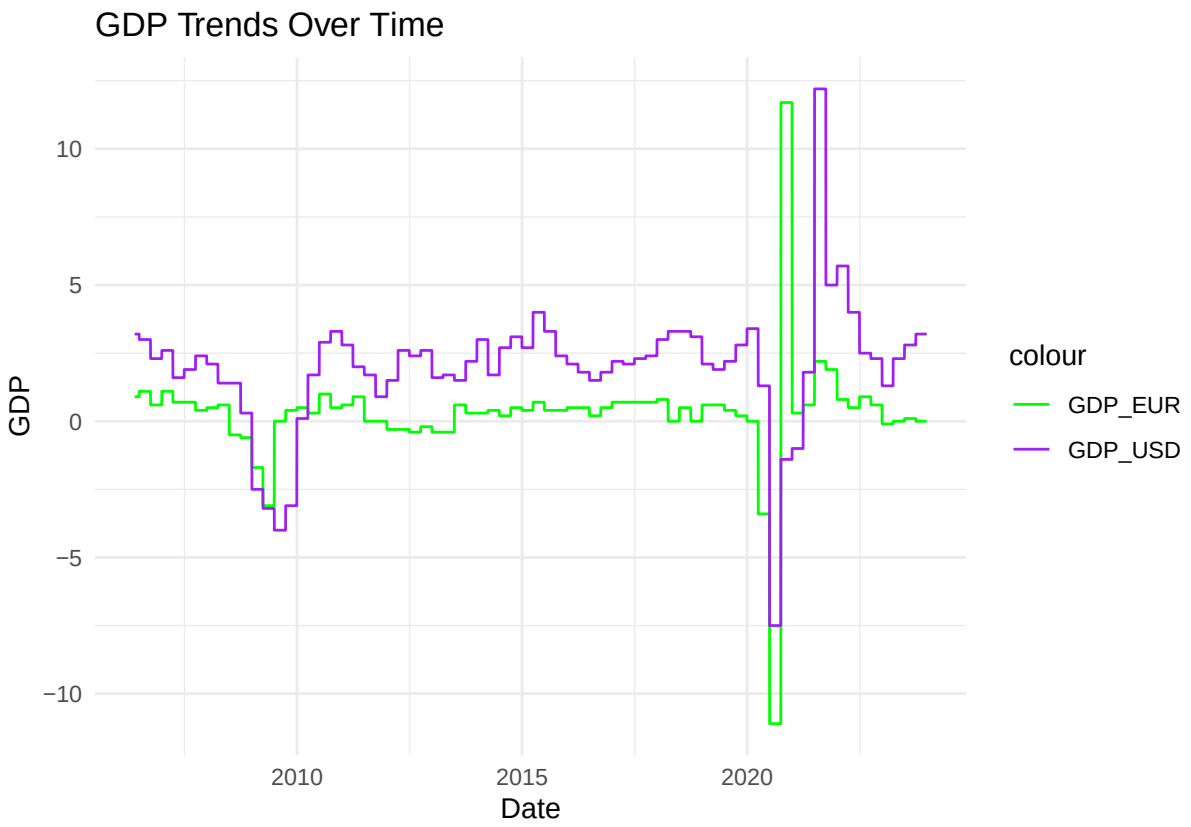

```r
# General plots
```

```r
# Plot EUR/USD Over Time
ggplot(complete_data, aes(x = Date, y = EUR_USD)) +
  geom_line(color = "blue") +
  geom_point(data = complete_data %>%
               filter(EUR_USD == max(EUR_USD, na.rm = TRUE) |
                        EUR_USD == min(EUR_USD, na.rm = TRUE)),
             aes(x = Date, y = EUR_USD), color = "red", size = 3) +
  labs(title = "EUR/USD Exchange Rate Over Time",
       x = "Date",
       y = "EUR/USD Exchange Rate") +
  theme_minimal()
```

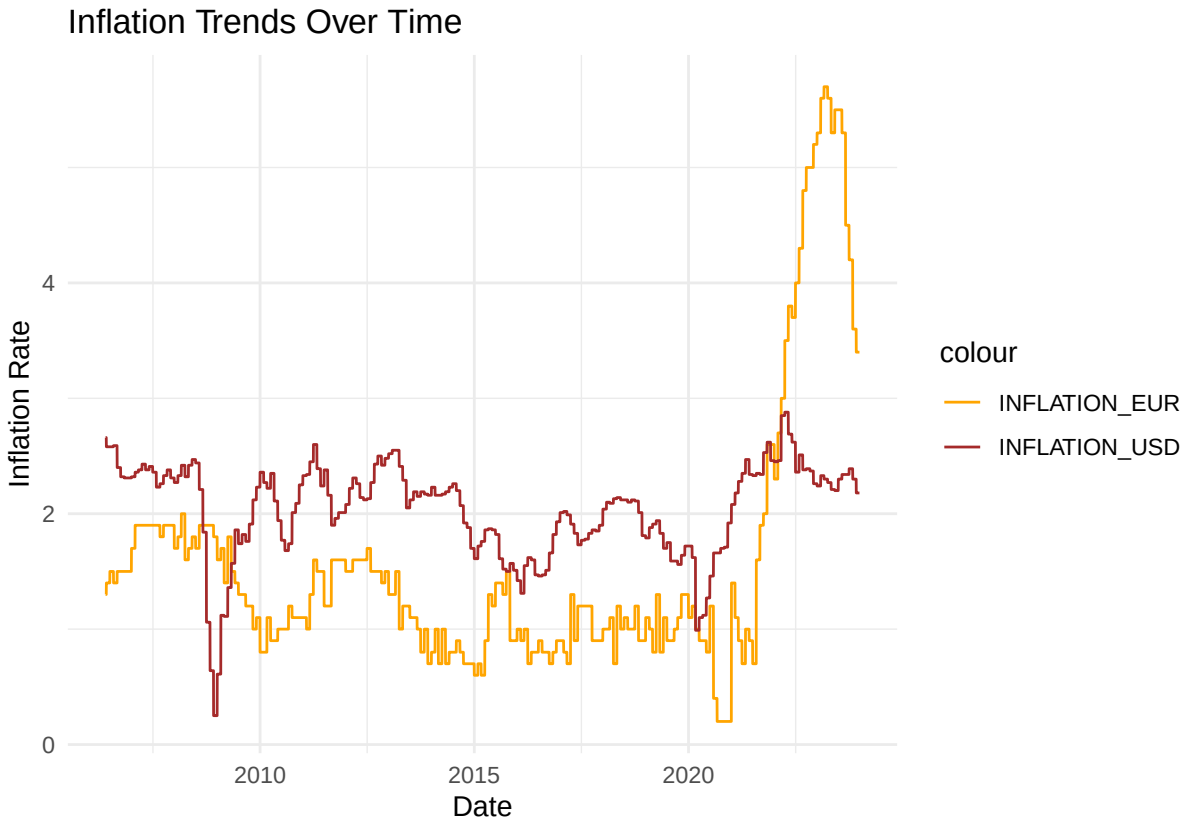



```r
# Trends in GDP, Inflation, and Interest Rates
gdp_plot <- ggplot(complete_data, aes(x = Date)) +
  geom_line(aes(y = GDP_EUR, color = "GDP_EUR")) +
  geom_line(aes(y = GDP_USD, color = "GDP_USD")) +
  labs(title = "GDP Trends Over Time",
       x = "Date",
       y = "GDP") +
  theme_minimal() +
  scale_color_manual(values = c("GDP_EUR" = "green", "GDP_USD" = "purple"))

inflation_plot <- ggplot(complete_data, aes(x = Date)) +
  geom_line(aes(y = INFLATION_EUR, color = "INFLATION_EUR")) +
  geom_line(aes(y = INFLATION_USD, color = "INFLATION_USD")) +
  labs(title = "Inflation Trends Over Time",
       x = "Date",
```

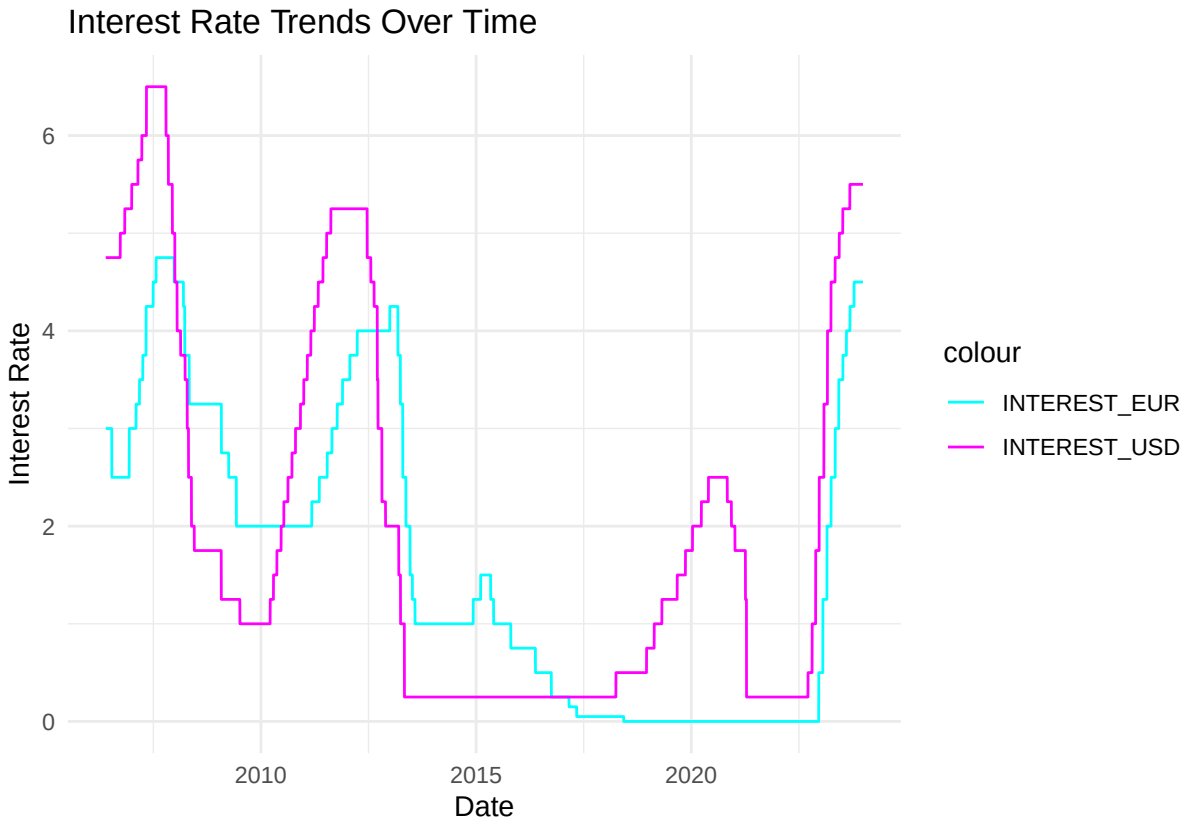
```
       y = "Inflation Rate") +
  theme_minimal() +
  scale_color_manual(values = c("INFLATION_EUR" = "orange", "INFLATION_USD" = "brown"))

interest_plot <- ggplot(complete_data, aes(x = Date)) +
  geom_line(aes(y = INTEREST_EUR, color = "INTEREST_EUR")) +
  geom_line(aes(y = INTEREST_USD, color = "INTEREST_USD")) +
  labs(title = "Interest Rate Trends Over Time",
       x = "Date",
       y = "Interest Rate") +
  theme_minimal() +
  scale_color_manual(values = c("INTEREST_EUR" = "cyan", "INTEREST_USD" = "magenta"))

# Display the trend plots
print(gdp_plot)
```




```
print(inflation_plot)
```



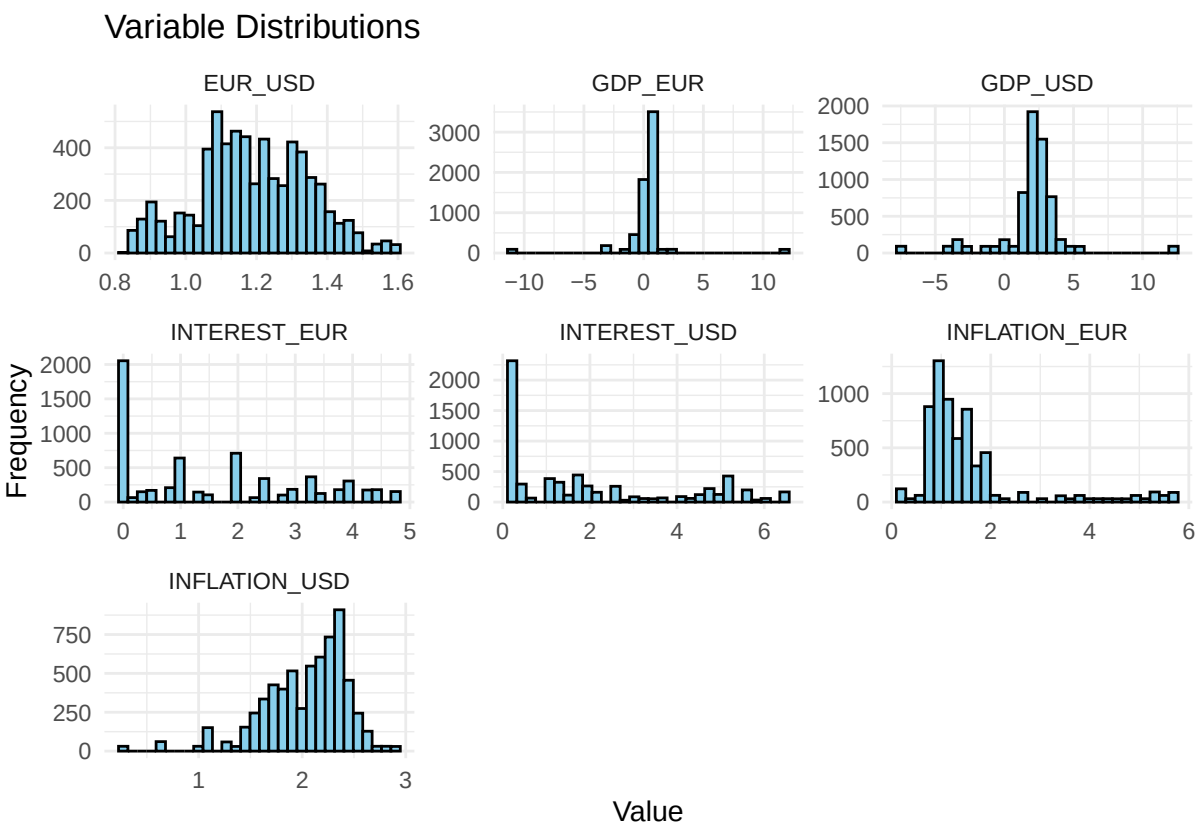

```
print(interest_plot)
```




```
# Distribution Analysis
# Melt the numeric data for distribution plotting
melted_data <- melt(numeric_cols, variable.name = "Variable", value.name = "Value")
```

```
## No id variables; using all as measure variables
```

```
# Plot the distributions
ggplot(melted_data, aes(x = Value)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "black") +
  facet_wrap(~Variable, scales = "free") +
  labs(title = "Variable Distributions", x = "Value", y = "Frequency") +
  theme_minimal()
```




```
# (4) MODELS
```

```
# Scale the data
scaled_data <- preProcess(complete_data[, -1], method = c("center", "scale"))
complete_data_scaled <- predict(scaled_data, complete_data[, -1])
complete_data_scaled <- cbind(Date = complete_data$Date, complete_data_scaled)

# Create a new dataset excluding the EUR_USD column
predictors_data <- complete_data_scaled %>% select(-EUR_USD)

# Linear regression with predictors excluding EUR_USD
lm_model <- lm(EUR_USD ~ ., data = complete_data_scaled %>% select(-Date))
summary(lm_model)
```

```
##
## Call:
## lm(formula = EUR_USD ~ ., data = complete_data_scaled %>% select(-Date))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.36363 -0.64682  0.06115  0.52635  2.27287
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.154e-15  1.078e-02   0.000        1
## GDP_EUR       -8.984e-02  1.182e-02  -7.602 3.33e-14 ***
## GDP_USD        1.423e-01  1.254e-02  11.350  < 2e-16 ***
## INTEREST_EUR   2.913e-01  1.688e-02  17.259  < 2e-16 ***
```

```
## INTEREST_USD  -5.124e-01  1.727e-02 -29.666  < 2e-16 ***
## INFLATION_EUR -2.884e-01  1.195e-02 -24.126  < 2e-16 ***
## INFLATION_USD  4.958e-02  1.267e-02   3.914 9.16e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8643 on 6420 degrees of freedom
## Multiple R-squared:  0.2536, Adjusted R-squared:  0.2529
## F-statistic: 363.6 on 6 and 6420 DF,  p-value: < 2.2e-16
```

```
# Evaluate performance
lm_pred <- predict(lm_model, complete_data_scaled %>% select(-EUR_USD))
mse_lm <- mse(complete_data_scaled$EUR_USD, lm_pred)
cat("Linear Regression MSE:", mse_lm, "\n")
```

```
## Linear Regression MSE: 0.7462606
```

```
# Lasso regression
# Prepare data for glmnet
X <- as.matrix(predictors_data[, -1])  # Exclude Date
y <- complete_data_scaled$EUR_USD

# Lasso regression
lasso_model <- cv.glmnet(X, y, alpha = 1)
lasso_pred <- predict(lasso_model, s = "lambda.min", newx = X)
mse_lasso <- mse(y, lasso_pred)
cat("Lasso Regression MSE:", mse_lasso, "\n")
```

```
## Lasso Regression MSE: 0.7462731
```

```
# Examine selected features
important_features <- coef(lasso_model, s = "lambda.min")
print(important_features)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                           s1
## (Intercept)    1.793729e-15
## GDP_EUR       -8.824936e-02
## GDP_USD        1.409424e-01
## INTEREST_EUR   2.872845e-01
## INTEREST_USD  -5.085888e-01
## INFLATION_EUR -2.870973e-01
## INFLATION_USD  4.829226e-02
```

```
# Ridge regression
ridge_model <- cv.glmnet(X, y, alpha = 0)
ridge_pred <- predict(ridge_model, s = "lambda.min", newx = X)
mse_ridge <- mse(y, ridge_pred)
cat("Ridge Regression MSE:", mse_ridge, "\n")
```

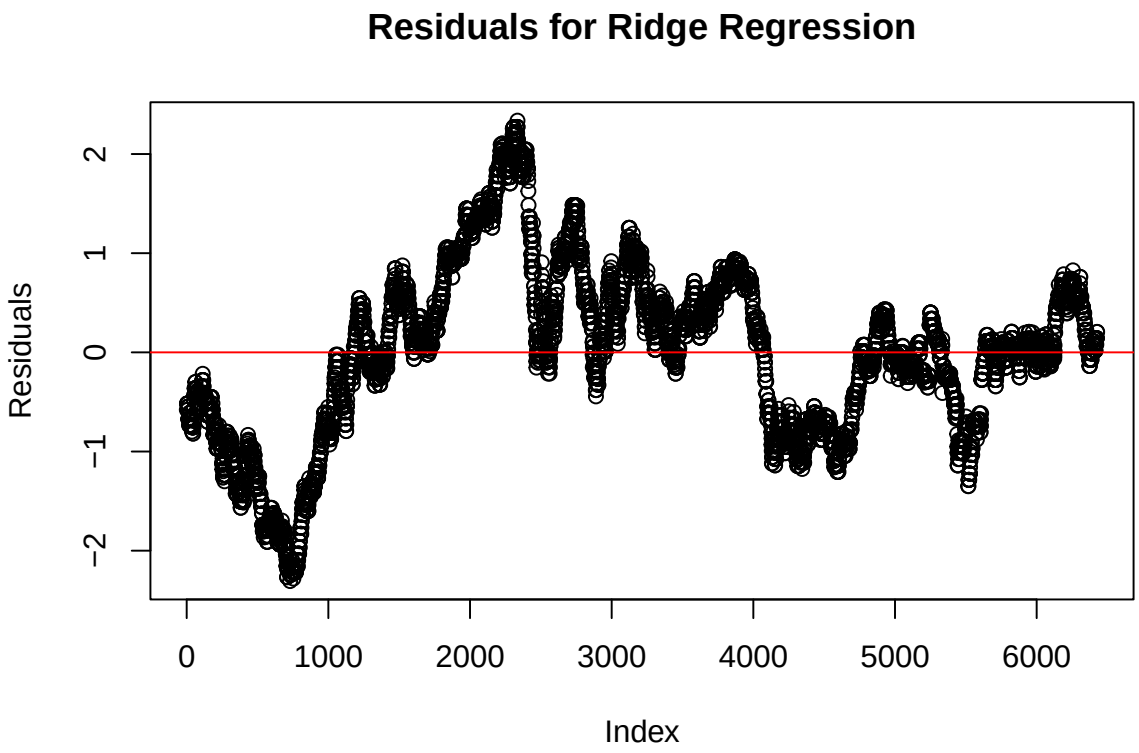
```
## Ridge Regression MSE: 0.7478506
```

```
# Examine Ridge coefficients
ridge_coefficients <- coef(ridge_model, s = "lambda.min")
print(ridge_coefficients)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                           s1
```

```
## (Intercept)    1.646254e-15
## GDP_EUR       -8.402401e-02
## GDP_USD        1.339780e-01
## INTEREST_EUR   2.383975e-01
## INTEREST_USD  -4.562821e-01
## INFLATION_EUR -2.802910e-01
## INFLATION_USD  4.380957e-02
```

```r
# Residual analysis
residuals_ridge <- y - ridge_pred
plot(residuals_ridge, main = "Residuals for Ridge Regression", ylab = "Residuals", xlab = "Index")
abline(h = 0, col = "red")
```




```r
# Interaction Model
interaction_model <- lm(EUR_USD ~ GDP_USD * INTEREST_USD + GDP_EUR * INTEREST_EUR, data = complete_data_
summary(interaction_model)
```

```
##
## Call:
## lm(formula = EUR_USD ~ GDP_USD * INTEREST_USD + GDP_EUR * INTEREST_EUR,
##     data = complete_data_scaled)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.34888 -0.66038 -0.04283  0.59381  2.34069
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -0.01218    0.01150  -1.060    0.289
## GDP_USD              0.16651    0.01585  10.508  < 2e-16 ***
## INTEREST_USD        -0.52673    0.01899 -27.735  < 2e-16 ***
```

```
## GDP_EUR              -0.31653    0.02941 -10.761  < 2e-16 ***
## INTEREST_EUR          0.20603    0.02010  10.253  < 2e-16 ***
## GDP_USD:INTEREST_USD  0.09613    0.02386   4.029 5.66e-05 ***
## GDP_EUR:INTEREST_EUR -0.25312    0.02741  -9.236  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8963 on 6420 degrees of freedom
## Multiple R-squared:  0.1975, Adjusted R-squared:  0.1967
## F-statistic: 263.3 on 6 and 6420 DF,  p-value: < 2.2e-16
```

```
# Fit Decision Tree Model
tree_model <- rpart(EUR_USD ~ ., data = complete_data_scaled %>% select(-Date), method = "anova")

# Predict using the Decision Tree model
tree_pred <- predict(tree_model, complete_data_scaled %>% select(-Date))

# Calculate MSE for Decision Tree
mse_tree <- mse(complete_data_scaled$EUR_USD, tree_pred)
cat("Decision Tree MSE:", mse_tree, "\n")
```

```
## Decision Tree MSE: 0.1352307
```

```
# General summary:
# Summarize model performance, including Decision Tree
performance_summary <- data.frame(
  Model = c("Linear Regression", "Lasso Regression", "Ridge Regression", "Decision Tree"),
  MSE = c(mse_lm, mse_lasso, mse_ridge, mse_tree)
)

# Print performance summary
print(performance_summary)
```

```
##               Model       MSE
## 1 Linear Regression 0.7462606
## 2  Lasso Regression 0.7462731
## 3  Ridge Regression 0.7478506
## 4     Decision Tree 0.1352307
```

```
# The best model is the decision tree but it can probably be because of
# overfitting so let's check if this is the case
```

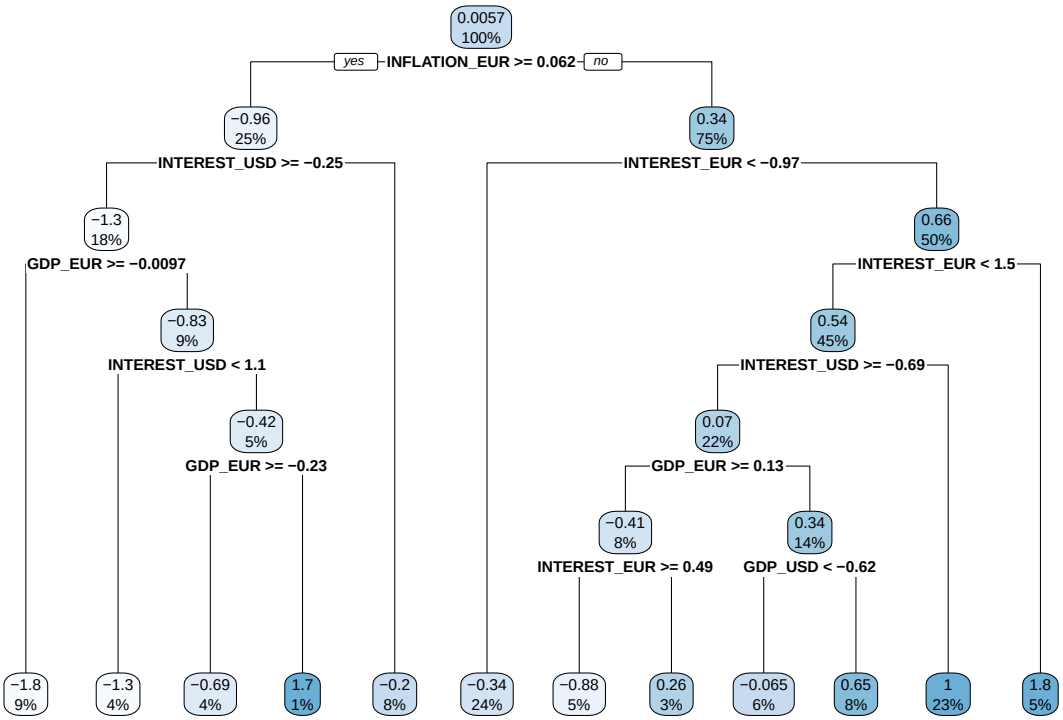
```
# (4.2) TREE VALIDATION
```

```
set.seed(123)

# Split data into training and testing sets
train_index <- createDataPartition(complete_data_scaled$EUR_USD, p = 0.8, list = FALSE)
train_data <- complete_data_scaled[train_index, ]
test_data <- complete_data_scaled[-train_index, ]

# Fit Decision Tree on training data
tree_model <- rpart(EUR_USD ~ ., data = train_data %>% select(-Date), method = "anova")

# Predict on test data
tree_test_pred <- predict(tree_model, test_data %>% select(-Date))
test_mse_tree <- mse(test_data$EUR_USD, tree_test_pred)
```

```r
cat("Decision Tree Test MSE:", test_mse_tree, "\n")
```

```
## Decision Tree Test MSE: 0.1359242
```

```r
# Prune the tree
pruned_tree <- prune(tree_model, cp = tree_model$cptable[which.min(tree_model$cptable[,"xerror"]), "CP"])
rpart.plot(pruned_tree)
```




```r
# (5) ENSEMBLE METHODS
```

```r
# Fit Random Forest Model on training data
rf_model <- randomForest(EUR_USD ~ ., data = train_data %>% select(-Date))
rf_test_pred <- predict(rf_model, test_data %>% select(-Date))
rf_test_mse <- mse(test_data$EUR_USD, rf_test_pred)
cat("Random Forest Test MSE:", rf_test_mse, "\n")
```

```
## Random Forest Test MSE: 0.01101849
```

```r
# (6) NONLINEAR RELATIONSHIPS
```

```r
# Interaction Model on training data
interaction_model <- lm(EUR_USD ~ GDP_EUR * INTEREST_EUR + GDP_USD * INTEREST_USD, data = train_data)
summary(interaction_model)
```

```
##
## Call:
## lm(formula = EUR_USD ~ GDP_EUR * INTEREST_EUR + GDP_USD * INTEREST_USD,
##     data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.35323 -0.66269 -0.02826  0.60476  2.33311
##
```

```
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -0.01278    0.01301  -0.982 0.326029
## GDP_EUR              -0.34121    0.03306 -10.321  < 2e-16 ***
## INTEREST_EUR          0.19658    0.02259   8.704  < 2e-16 ***
## GDP_USD               0.17286    0.01807   9.565  < 2e-16 ***
## INTEREST_USD         -0.51384    0.02138 -24.031  < 2e-16 ***
## GDP_EUR:INTEREST_EUR -0.27538    0.03076  -8.952  < 2e-16 ***
## GDP_USD:INTEREST_USD  0.10427    0.02681   3.889 0.000102 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9057 on 5137 degrees of freedom
## Multiple R-squared:  0.1925, Adjusted R-squared:  0.1916
## F-statistic: 204.1 on 6 and 5137 DF,  p-value: < 2.2e-16
```

```
# Polynomial Model on training data
polynomial_model <- lm(EUR_USD ~ poly(GDP_EUR, 2) + poly(INTEREST_EUR, 2) +
                         poly(GDP_USD, 2) + poly(INTEREST_USD, 2) +
                         GDP_EUR:INTEREST_EUR + GDP_USD:INTEREST_USD,
                       data = train_data)
summary(polynomial_model)
```

```
##
## Call:
## lm(formula = EUR_USD ~ poly(GDP_EUR, 2) + poly(INTEREST_EUR,
##     2) + poly(GDP_USD, 2) + poly(INTEREST_USD, 2) + GDP_EUR:INTEREST_EUR +
##     GDP_USD:INTEREST_USD, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5215 -0.5697 -0.0239  0.5514  2.5519
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -0.02254    0.01229  -1.834   0.0668 .
## poly(GDP_EUR, 2)1       -45.96079    2.38665 -19.257  < 2e-16 ***
## poly(GDP_EUR, 2)2        24.52389    1.50725  16.271  < 2e-16 ***
## poly(INTEREST_EUR, 2)1   14.86430    1.58945   9.352  < 2e-16 ***
## poly(INTEREST_EUR, 2)2 -23.95835    0.94647 -25.313  < 2e-16 ***
## poly(GDP_USD, 2)1        30.58837    1.90884  16.025  < 2e-16 ***
## poly(GDP_USD, 2)2       -18.45310    1.27759 -14.444  < 2e-16 ***
## poly(INTEREST_USD, 2)1 -34.43288    1.55901 -22.086  < 2e-16 ***
## poly(INTEREST_USD, 2)2   7.62503    0.96819   7.876 4.11e-15 ***
## GDP_EUR:INTEREST_EUR    -0.42063    0.02971 -14.157  < 2e-16 ***
## GDP_USD:INTEREST_USD     0.04994    0.03788   1.318   0.1874
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8402 on 5133 degrees of freedom
## Multiple R-squared:  0.3055, Adjusted R-squared:  0.3042
## F-statistic: 225.8 on 10 and 5133 DF,  p-value: < 2.2e-16
```

```
# Expanded Interaction Model on training data
expanded_interaction_model <- lm(EUR_USD ~ (GDP_EUR + INTEREST_EUR + GDP_USD + INTEREST_USD)^2,
```

```
                                  data = train_data)
summary(expanded_interaction_model)
```

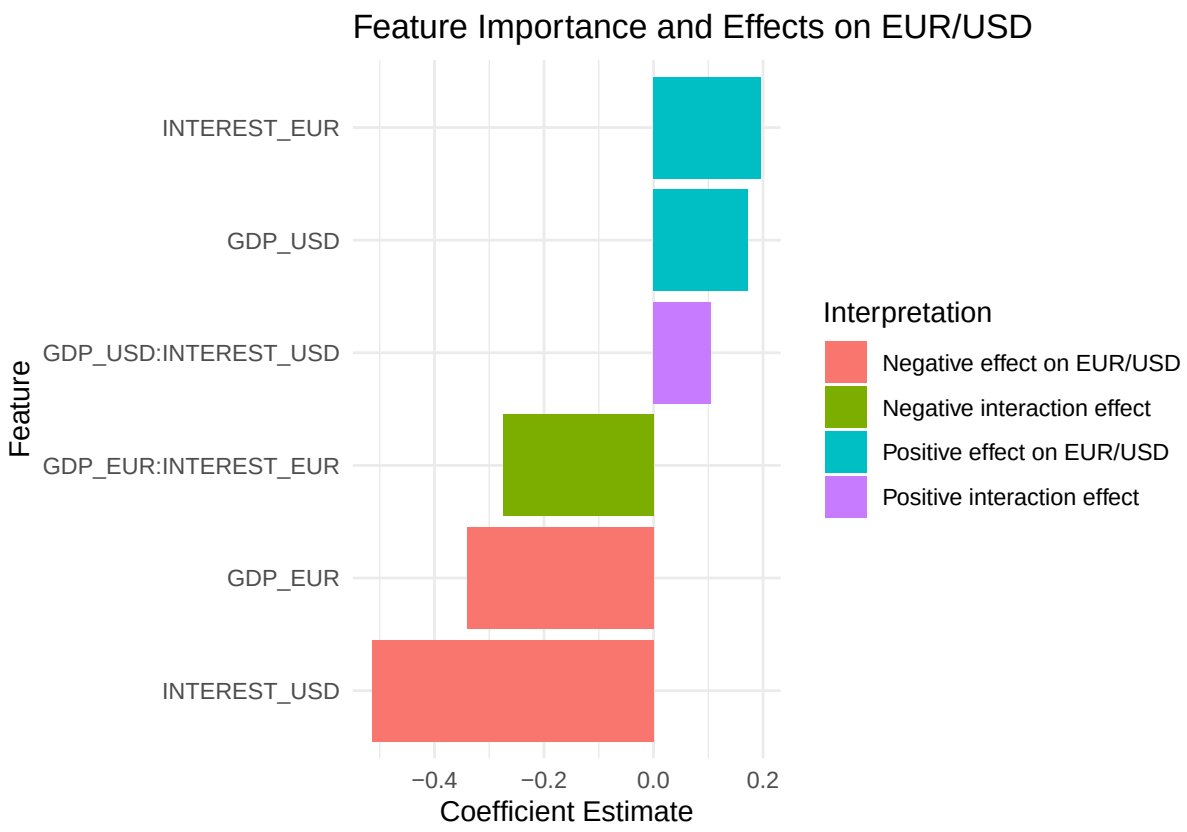
```
##
## Call:
## lm(formula = EUR_USD ~ (GDP_EUR + INTEREST_EUR + GDP_USD + INTEREST_USD)^2,
##     data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.66049 -0.52285  0.06456  0.54248  1.68990
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 0.20860    0.01756  11.882   <2e-16 ***
## GDP_EUR                    -0.51346    0.03716 -13.819   <2e-16 ***
## INTEREST_EUR                0.34354    0.02039  16.849   <2e-16 ***
## GDP_USD                     0.28104    0.02142  13.118   <2e-16 ***
## INTEREST_USD               -0.46314    0.02203 -21.027   <2e-16 ***
## GDP_EUR:INTEREST_EUR       -0.40751    0.03082 -13.223   <2e-16 ***
## GDP_EUR:GDP_USD            -0.13915    0.01109 -12.547   <2e-16 ***
## GDP_EUR:INTEREST_USD       -1.11535    0.05113 -21.816   <2e-16 ***
## INTEREST_EUR:GDP_USD        0.46016    0.02000  23.012   <2e-16 ***
## INTEREST_EUR:INTEREST_USD -0.23706    0.01677 -14.136   <2e-16 ***
## GDP_USD:INTEREST_USD       -0.09170    0.03642  -2.518   0.0118 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7981 on 5133 degrees of freedom
## Multiple R-squared:  0.3735, Adjusted R-squared:  0.3722
## F-statistic:   306 on 10 and 5133 DF,  p-value: < 2.2e-16
```

```
# Summary of the key features
features_table <- data.frame(
  Feature = c(
    "GDP_EUR",
    "INTEREST_EUR",
    "GDP_USD",
    "INTEREST_USD",
    "GDP_EUR:INTEREST_EUR",
    "GDP_USD:INTEREST_USD"
  ),
  Estimate = c(
    coef(interaction_model)["GDP_EUR"],
    coef(interaction_model)["INTEREST_EUR"],
    coef(interaction_model)["GDP_USD"],
    coef(interaction_model)["INTEREST_USD"],
    coef(interaction_model)["GDP_EUR:INTEREST_EUR"],
    coef(interaction_model)["GDP_USD:INTEREST_USD"]
  ),
  Interpretation = c(
    "Negative effect on EUR/USD",
    "Positive effect on EUR/USD",
    "Positive effect on EUR/USD",
```

```
    "Negative effect on EUR/USD",
    "Negative interaction effect",
    "Positive interaction effect"
  )
)

# Print the table
print(features_table)
```

```
##                                   Feature   Estimate
## GDP_EUR                           GDP_EUR -0.3412079
## INTEREST_EUR                 INTEREST_EUR  0.1965838
## GDP_USD                           GDP_USD  0.1728594
## INTEREST_USD                 INTEREST_USD -0.5138366
## GDP_EUR:INTEREST_EUR GDP_EUR:INTEREST_EUR -0.2753846
## GDP_USD:INTEREST_USD GDP_USD:INTEREST_USD  0.1042721
##                                   Interpretation
## GDP_EUR                Negative effect on EUR/USD
## INTEREST_EUR           Positive effect on EUR/USD
## GDP_USD                Positive effect on EUR/USD
## INTEREST_USD           Negative effect on EUR/USD
## GDP_EUR:INTEREST_EUR Negative interaction effect
## GDP_USD:INTEREST_USD Positive interaction effect
```

```
# Create a visualization for the feature importance
ggplot(features_table, aes(x = reorder(Feature, Estimate), y = Estimate, fill = Interpretation)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Feature Importance and Effects on EUR/USD",
       x = "Feature",
       y = "Coefficient Estimate") +
  theme_minimal()
```




```
# (7) FINAL MODEL
```

```
# Fit the final model on training data
final_model <- lm(EUR_USD ~ GDP_EUR + INTEREST_EUR + GDP_USD + INTEREST_USD +
                    GDP_EUR:INTEREST_EUR + GDP_USD:INTEREST_USD,
                  data = train_data)
summary(final_model)
```

```
##
## Call:
## lm(formula = EUR_USD ~ GDP_EUR + INTEREST_EUR + GDP_USD + INTEREST_USD +
##     GDP_EUR:INTEREST_EUR + GDP_USD:INTEREST_USD, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.35323 -0.66269 -0.02826  0.60476  2.33311
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -0.01278    0.01301  -0.982 0.326029
## GDP_EUR              -0.34121    0.03306 -10.321  < 2e-16 ***
## INTEREST_EUR          0.19658    0.02259   8.704  < 2e-16 ***
## GDP_USD               0.17286    0.01807   9.565  < 2e-16 ***
## INTEREST_USD         -0.51384    0.02138 -24.031  < 2e-16 ***
## GDP_EUR:INTEREST_EUR -0.27538    0.03076  -8.952  < 2e-16 ***
## GDP_USD:INTEREST_USD  0.10427    0.02681   3.889 0.000102 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

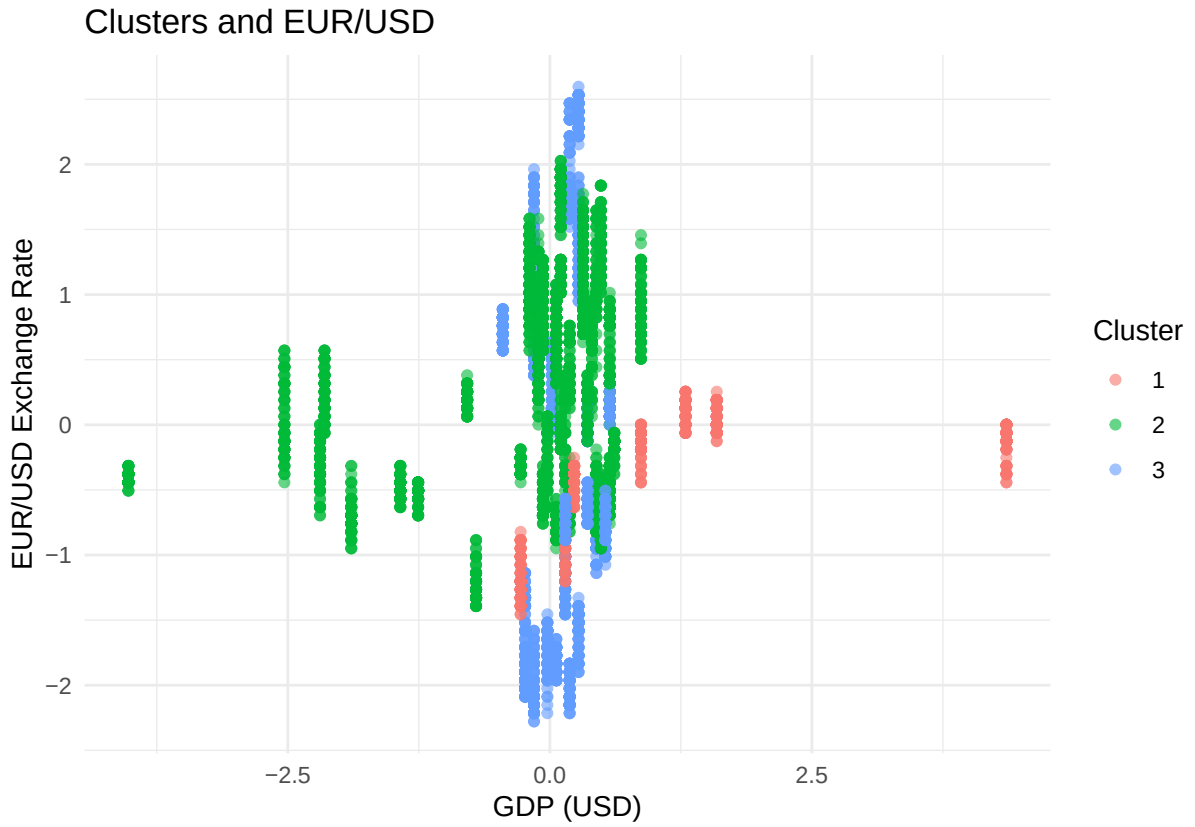
```
##
## Residual standard error: 0.9057 on 5137 degrees of freedom
## Multiple R-squared:  0.1925, Adjusted R-squared:  0.1916
## F-statistic: 204.1 on 6 and 5137 DF,  p-value: < 2.2e-16
```

```
# (8) DATA ENGINEERING: OUTLIERS
```

```
# Perform K-Means Clustering
kmeans_result <- kmeans(complete_data_scaled[, -1], centers = 3)
complete_data_scaled$Cluster <- as.factor(kmeans_result$cluster)

# Investigate the composition of clusters
cluster_summary <- complete_data_scaled %>%
  group_by(Cluster) %>%
  summarise(
    Avg_EUR_USD = mean(EUR_USD, na.rm = TRUE),
    Avg_GDP_EUR = mean(GDP_EUR, na.rm = TRUE),
    Avg_GDP_USD = mean(GDP_USD, na.rm = TRUE),
    Avg_INTEREST_EUR = mean(INTEREST_EUR, na.rm = TRUE),
    Avg_INTEREST_USD = mean(INTEREST_USD, na.rm = TRUE),
    Avg_INFLATION_EUR = mean(INFLATION_EUR, na.rm = TRUE),
    Avg_INFLATION_USD = mean(INFLATION_USD, na.rm = TRUE)
  )
print(cluster_summary)
```

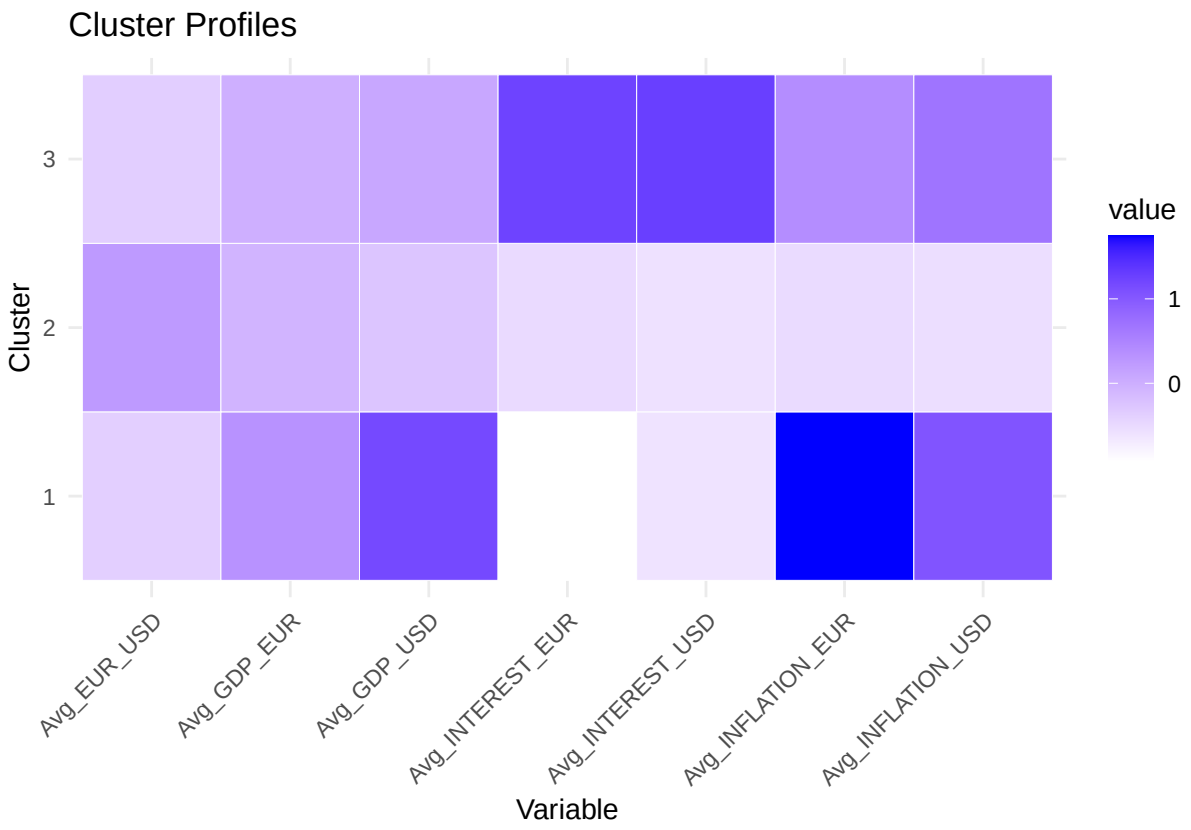
```
## # A tibble: 3 x 8
##   Cluster Avg_EUR_USD Avg_GDP_EUR Avg_GDP_USD Avg_INTEREST_EUR Avg_INTEREST_USD
##   <fct>         <dbl>       <dbl>       <dbl>            <dbl>            <dbl>
## 1 1            -0.367     0.336        1.17             -0.907           -0.589
## 2 2             0.246    -0.0549      -0.248            -0.490           -0.568
## 3 3            -0.353    -0.00316      0.0966            1.23             1.28
## # i 2 more variables: Avg_INFLATION_EUR <dbl>, Avg_INFLATION_USD <dbl>
```

```
# Visualize clusters
ggplot(complete_data_scaled, aes(x = GDP_USD, y = EUR_USD, color = Cluster)) +
  geom_point(alpha = 0.6) +
  labs(title = "Clusters and EUR/USD", x = "GDP (USD)", y = "EUR/USD Exchange Rate") +
  theme_minimal()
```




```
# Pairwise Scatter Plots
ggpairs(
  complete_data_scaled,
  columns = c("EUR_USD", "GDP_EUR", "GDP_USD", "INTEREST_EUR", "INTEREST_USD", "INFLATION_EUR", "INFLAT
  aes(color = Cluster, alpha = 0.7)
)
```


```r
# Cluster Profiles - Heatmap
cluster_summary_long <- melt(cluster_summary, id.vars = "Cluster")

ggplot(cluster_summary_long, aes(x = variable, y = Cluster, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(title = "Cluster Profiles", x = "Variable", y = "Cluster") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Analyze Cluster Significance
anova_results <- aov(EUR_USD ~ Cluster, data = complete_data_scaled)
summary(anova_results)
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## Cluster        2    563  281.38   308.3 <2e-16 ***
## Residuals   6424   5863    0.91
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```