Macroeconomic Drivers of S&P 500 Returns: A Boruta-Selected Regression Analysis

Emil Blaignan, Robert Sellman, Sean Eizadi

2025-05-29

Contents

Introduction

This project identifies key macroeconomic predictors of S&P 500 returns using Boruta feature selection and multiple regression analysis. Our primary research questions are:

- Which macroeconomic indicators (e.g., yield curve, inflation, volatility) are most statistically significant in predicting S&P 500 returns?
- How do factor variables (e.g., recession dummies, Fed policy indicators) improve model fit?

- What are the economic magnitudes and interpretations of these relationships?

Data Sources: We utilize quantitative data from FRED (Federal Reserve Economic Data) spanning January 2000 to March 2025, including over 15 initial macroeconomic predictors. Factor variables include the NBER recession indicator and Fed tightening cycle dummies. Our target variable is monthly S&P 500 returns calculated from Yahoo Finance data.

1. Variable Selection

1.1 Quantitative Predictors via Boruta

We applied the Boruta algorithm to identify macroeconomic variables most strongly associated with monthly S&P 500 returns. Boruta uses a random forest classifier to assess feature importance, comparing real variables against randomly permuted "shadow" features.

```
# Load processed data
cleaned_data <- read_csv("data/processed_data.csv")
cat("Dataset dimensions:", nrow(cleaned_data), "observations,", ncol(cleaned_data), "variables\n")
```

```
## Dataset dimensions: 303 observations, 26 variables
```

```
# Convert dates to proper format
if(is.numeric(cleaned_data$date)) {
  cleaned_data$date <- as.Date(cleaned_data$date, origin = "1970-01-01")
} else if(is.character(cleaned_data$date)) {
  cleaned_data$date <- as.Date(cleaned_data$date)
}

cat("Time period:", as.character(min(cleaned_data$date)), "to", as.character(max(cleaned_data$date)), "
```

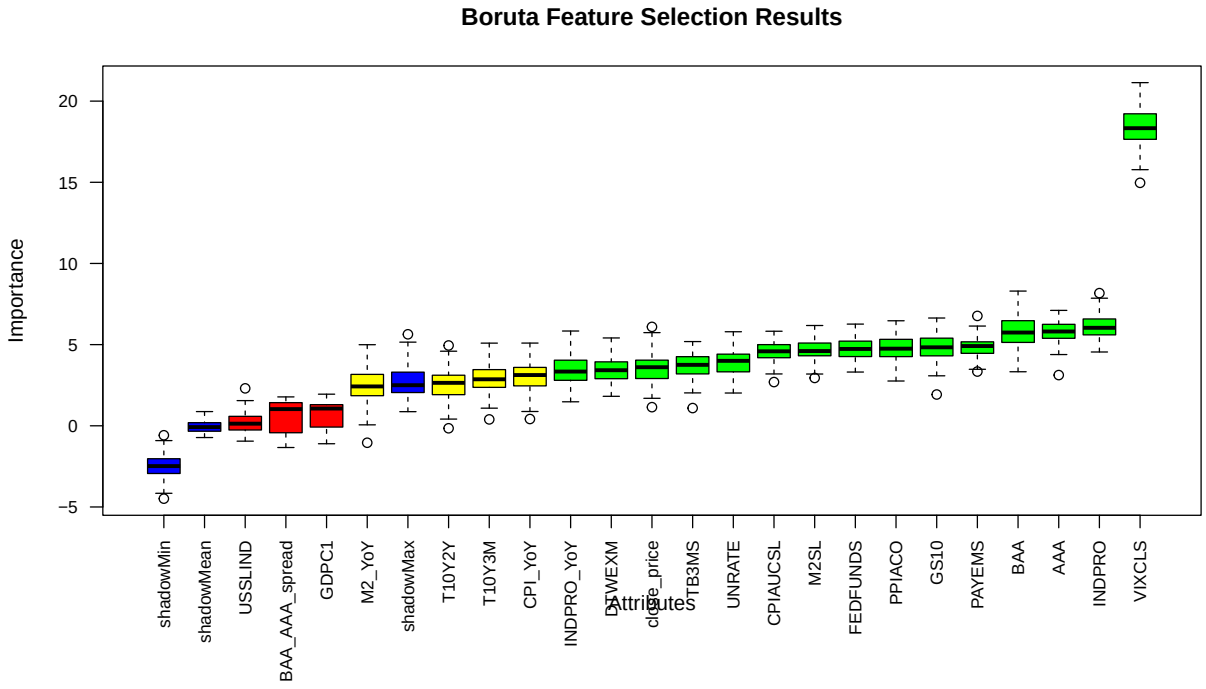
```
## Time period: 2000-01-01 to 2025-03-01
```

```
# Prepare data for Boruta
boruta_data <- cleaned_data %>%
  select(-date, -USREC, -tightening) %>%
  select(where(is.numeric))

# Create formula for Boruta
predictors <- names(boruta_data)[names(boruta_data) != "monthly_return"]
formula_str <- paste("monthly_return ~", paste(predictors, collapse = " + "))
boruta_formula <- as.formula(formula_str)

# Run Boruta feature selection
set.seed(123)
boruta_results <- Boruta(boruta_formula, data = boruta_data, doTrace = 1, maxRuns = 100)

print(boruta_results)
```

```
## Boruta performed 99 iterations in 16.14582 secs.
##  15 attributes confirmed important: AAA, BAA, close_price, CPIAUCSL,
```

```
## DTWEXM and 10 more;
##  3 attributes confirmed unimportant: BAA_AAA_spread, GDPC1, USSLIND;
##  4 tentative attributes left: CPI_YoY, M2_YoY, T10Y2Y, T10Y3M;
```

```
# Plot results
par(mar = c(8, 4, 4, 2))
plot(boruta_results, main = "Boruta Feature Selection Results",
     cex.axis = 0.8, las = 2)
```




```
# Extract selected attributes
important_vars <- getSelectedAttributes(boruta_results, withTentative = FALSE)
cat("\nAll important variables selected by Boruta:\n")
```

```
##
## All important variables selected by Boruta:
```

```
print(important_vars)
```

```
##  [1] "INDPRO_YoY"  "close_price" "DTWEXM"      "VIXCLS"      "M2SL"
##  [6] "PAYEMS"      "UNRATE"      "INDPRO"      "PPIACO"      "CPIAUCSL"
## [11] "FEDFUNDS"    "AAA"         "BAA"         "TB3MS"       "GS10"
```

```
# Get importance statistics
boruta_stats <- attStats(boruta_results)
confirmed_vars <- boruta_stats[boruta_stats$decision == "Confirmed", ]
confirmed_vars <- confirmed_vars[order(confirmed_vars$medianImp, decreasing = TRUE), ]
cat("Top variables by importance:\n")
```

```
## Top variables by importance:
```

```
print(head(confirmed_vars, 10))
```

```
##            meanImp medianImp    minImp    maxImp  normHits  decision
## VIXCLS   18.397394 18.332012 14.966918 21.138260 1.0000000 Confirmed
## INDPRO    6.121445  6.036140  4.545975  8.172088 1.0000000 Confirmed
## AAA       5.789249  5.809857  3.123900  7.109305 0.9696970 Confirmed
## BAA       5.806439  5.744628  3.329359  8.298825 0.9696970 Confirmed
## PAYEMS    4.867993  4.912990  3.341382  6.768483 0.9494949 Confirmed
## GS10      4.805295  4.833190  1.927242  6.639068 0.9292929 Confirmed
## PPIACO    4.756258  4.751176  2.759210  6.468995 0.9292929 Confirmed
## FEDFUNDS  4.732046  4.732814  3.311243  6.264123 0.9393939 Confirmed
## M2SL      4.668623  4.603735  2.946968  6.178155 0.9090909 Confirmed
## CPIAUCSL  4.567735  4.587704  2.693736  5.825008 0.8787879 Confirmed
```

1.1.1 Final Variable Selection from Boruta Results

From Boruta's confirmed important variables, we selected the top five using the following criteria:

1. Highest median importance scores from the Boruta algorithm
2. Economic interpretability and theoretical grounding

3. Representing different economic dimensions (sentiment, real activity, credit, inflation, monetary policy)
4. Avoiding severe multicollinearity among selected predictors

```
# Select top 5 based on importance and economic theory
selected_vars <- c("VIXCLS", "INDPRO", "AAA", "CPIAUCSL", "FEDFUNDS")
cat("\nSelected variables for analysis:\n")
```

```
##
## Selected variables for analysis:
```

```
# Display importance scores
for(i in 1:length(selected_vars)) {
  var <- selected_vars[i]
  imp_score <- ifelse(var %in% rownames(confirmed_vars),
                      round(confirmed_vars[var, "medianImp"], 3), "N/A")
  cat(sprintf("%d. %s (Importance: %s)\n", i, var, imp_score))
}
```

```
## 1. VIXCLS (Importance: 18.332)
## 2. INDPRO (Importance: 6.036)
## 3. AAA (Importance: 5.81)
## 4. CPIAUCSL (Importance: 4.588)
## 5. FEDFUNDS (Importance: 4.733)
```

Our final selection represents key macroeconomic themes:

1. **VIXCLS** – CBOE Volatility Index (investor risk sentiment)

2. **INDPRO** – Industrial Production Index (real economic activity)

3. **AAA** – Yield on AAA-rated corporate bonds (credit conditions)
4. **CPIAUCSL** – Consumer Price Index (inflation pressures)
5. **FEDFUNDS** – Federal Funds Rate (monetary policy stance)

1.2 Selection of Factor Variables

We evaluated two binary indicators based on economic theory:

- **USREC**: NBER recession indicator (1 = recession period)
- **tightening**: Fed tightening dummy (1 = FEDFUNDS > lagged value)

```
# Test significance of recession indicator
recession_test <- t.test(
  cleaned_data$monthly_return[cleaned_data$USREC == 1],
  cleaned_data$monthly_return[cleaned_data$USREC == 0]
)

# Test significance of Fed tightening dummy
tightening_test <- t.test(
  cleaned_data$monthly_return[cleaned_data$tightening == 1],
  cleaned_data$monthly_return[cleaned_data$tightening == 0]
)

cat("Recession Indicator T-Test Results:\n")
```

```
## Recession Indicator T-Test Results:
```

```
print(recession_test)
```

```
##
##  Welch Two Sample t-test
##
## data:  cleaned_data$monthly_return[cleaned_data$USREC == 1] and cleaned_data$monthly_return[cleaned_
## t = -1.2482, df = 28.149, p-value = 0.2222
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.05308317  0.01287935
## sample estimates:
##    mean of x    mean of y
## -0.014345378  0.005756531
```

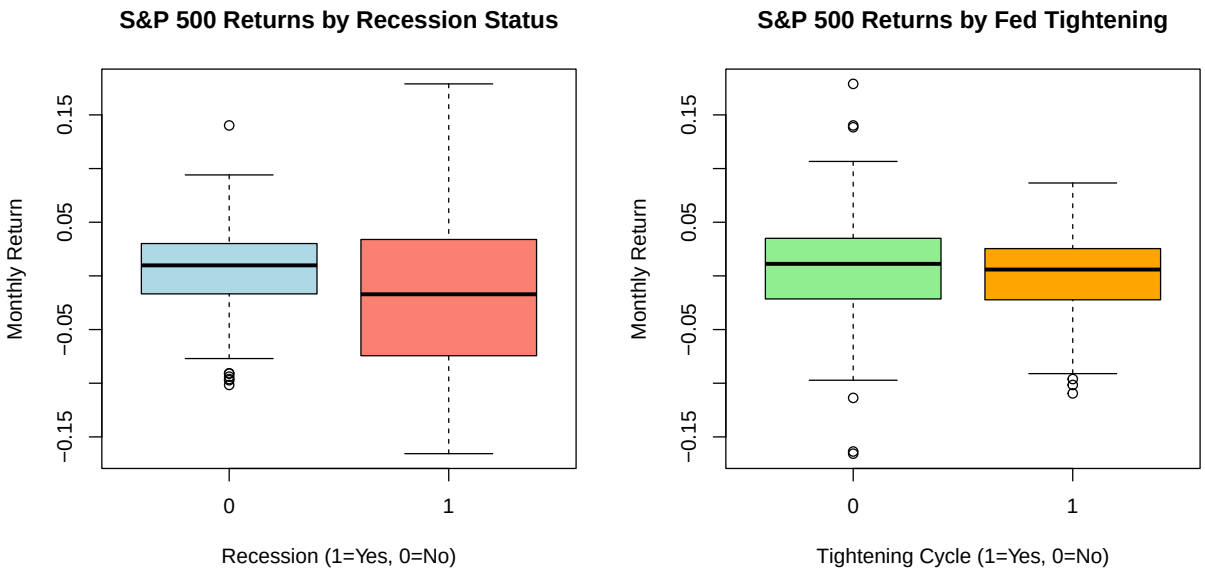
```
cat("\nFed Tightening Indicator T-Test Results:\n")
```

```
##
## Fed Tightening Indicator T-Test Results:
```

```
print(tightening_test)
```

```
##
##  Welch Two Sample t-test
##
## data:  cleaned_data$monthly_return[cleaned_data$tightening == 1] and cleaned_data$monthly_return[clea
## t = -1.3109, df = 300.94, p-value = 0.1909
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.016512500  0.003308699
## sample estimates:
##    mean of x    mean of y
## 0.0001948923 0.0067967930
```

```r
# Visualize factor variables
par(mfrow = c(1, 2), mar = c(5, 4, 4, 2))
boxplot(monthly_return ~ USREC, data = cleaned_data,
        main = "S&P 500 Returns by Recession Status",
        xlab = "Recession (1=Yes, 0=No)", ylab = "Monthly Return",
        col = c("lightblue", "salmon"))
boxplot(monthly_return ~ tightening, data = cleaned_data,
        main = "S&P 500 Returns by Fed Tightening",
        xlab = "Tightening Cycle (1=Yes, 0=No)", ylab = "Monthly Return",
        col = c("lightgreen", "orange"))
```




```r
par(mfrow = c(1, 1))
```

Results: Neither test produced statistically significant results at conventional levels ($p = 0.22$ for USREC; $p = 0.19$ for tightening). However, we retained both factors based on strong theoretical relevance.

2. Descriptive Analysis

2.1 Data Quality Assessment

```
# Define analysis variables
factor_vars <- c("USREC", "tightening")
target_var <- "monthly_return"

# Create working dataset
analysis_data <- cleaned_data %>%
  select(date, all_of(selected_vars), all_of(factor_vars), all_of(target_var))

# Check for missing values
missing_summary <- analysis_data %>%
  summarise(across(everything(), ~sum(is.na(.)))) %>%
  pivot_longer(everything(), names_to = "variable", values_to = "missing_count") %>%
  mutate(missing_pct = (missing_count / nrow(analysis_data)) * 100) %>%
  arrange(desc(missing_count))

cat("Missing Values Summary:\n")
```

```
## Missing Values Summary:
```

```
print(missing_summary)
```

```
## # A tibble: 9 x 3
##   variable       missing_count missing_pct
##   <chr>                  <int>       <dbl>
## 1 date                       0           0
## 2 VIXCLS                     0           0
## 3 INDPRO                     0           0
## 4 AAA                        0           0
## 5 CPIAUCSL                   0           0
## 6 FEDFUNDS                   0           0
## 7 USREC                      0           0
## 8 tightening                 0           0
## 9 monthly_return             0           0
```

Missing Data Handling: Our preprocessed dataset contains no missing values. Any missing values encountered during preprocessing were addressed through median imputation.

2.2 Outlier Detection and Treatment

```
# Function to detect outliers using IQR method
detect_outliers <- function(x) {
  Q1 <- quantile(x, 0.25, na.rm = TRUE)
  Q3 <- quantile(x, 0.75, na.rm = TRUE)
  IQR <- Q3 - Q1
  lower <- Q1 - 1.5 * IQR
```

```
  upper <- Q3 + 1.5 * IQR
  return(x < lower | x > upper)
}

# Outlier summary for numeric variables
outlier_summary <- analysis_data %>%
  select(all_of(c(selected_vars, target_var))) %>%
  summarise(across(everything(), ~sum(detect_outliers(.), na.rm = TRUE))) %>%
  pivot_longer(everything(), names_to = "variable", values_to = "outlier_count") %>%
  mutate(outlier_pct = (outlier_count / nrow(analysis_data)) * 100) %>%
  arrange(desc(outlier_count))

cat("Outlier Summary (using IQR method):\n")
```

```
## Outlier Summary (using IQR method):
```

```
print(outlier_summary)
```

```
## # A tibble: 6 x 3
##   variable       outlier_count outlier_pct
##   <chr>                  <int>       <dbl>
## 1 VIXCLS                    11       3.63
## 2 monthly_return             8       2.64
## 3 AAA                        1       0.330
## 4 INDPRO                     0       0
## 5 CPIAUCSL                   0       0
## 6 FEDFUNDS                   0       0
```

```
# Apply winsorization to monthly returns
analysis_data <- analysis_data %>%
  mutate(
    monthly_return_original = monthly_return,
    monthly_return = case_when(
      monthly_return > quantile(monthly_return, 0.99, na.rm = TRUE) ~
        quantile(monthly_return, 0.99, na.rm = TRUE),
      monthly_return < quantile(monthly_return, 0.01, na.rm = TRUE) ~
        quantile(monthly_return, 0.01, na.rm = TRUE),
      TRUE ~ monthly_return
    )
  )

cat("\nWinsorization applied to monthly_return at 1% and 99% percentiles.\n")
```

```
##
## Winsorization applied to monthly_return at 1% and 99% percentiles.
```

Outlier Treatment: We winsorized S&P 500 returns at the 1st and 99th percentiles to mitigate the impact of extreme market events while preserving the overall distribution structure.

2.3 Univariate Distributional Analysis

```
# Generate summary statistics
numeric_vars <- c(selected_vars, target_var)
summary_stats <- analysis_data %>%
  select(all_of(numeric_vars)) %>%
  pivot_longer(everything(), names_to = "variable", values_to = "value") %>%
  group_by(variable) %>%
  summarise(
    n = n(),
    mean = mean(value, na.rm = TRUE),
    median = median(value, na.rm = TRUE),
    sd = sd(value, na.rm = TRUE),
    min = min(value, na.rm = TRUE),
    max = max(value, na.rm = TRUE),
    skewness = moments::skewness(value, na.rm = TRUE),
    kurtosis = moments::kurtosis(value, na.rm = TRUE),
    .groups = 'drop'
  )

kable(summary_stats, digits = 3, caption = "Summary Statistics for Analysis Variables")
```

Table 1: Summary Statistics for Analysis Variables

variable	n	mean	median	sd	min	max	skewness	kurtosis
AAA	303	4.773	4.880	1.265	2.140	7.990	0.253	2.707
CPIAUCSL	303	230.806	229.918	39.003	169.300	319.775	0.503	2.587
FEDFUNDS	303	1.943	1.240	2.026	0.050	6.540	0.800	2.197
INDPRO	303	97.327	98.724	5.018	84.675	104.220	-0.584	2.208
VIXCLS	303	19.931	17.940	7.915	9.510	59.890	1.721	7.261
monthly_return	303	0.004	0.009	0.042	-0.109	0.106	-0.346	3.223

```
# Generate summary statistics for factor variables
factor_vars <- c("USREC", "tightening")
factor_summary <- analysis_data %>%
  select(all_of(factor_vars)) %>%
  pivot_longer(everything(), names_to = "variable", values_to = "value") %>%
  group_by(variable, value) %>%
  summarise(count = n(), .groups = 'drop') %>%
  group_by(variable) %>%
  mutate(
    percentage = round((count / sum(count)) * 100, 2),
    interpretation = case_when(
      variable == "USREC" & value == 1 ~ "Recession periods",
      variable == "USREC" & value == 0 ~ "Non-recession periods",
      variable == "tightening" & value == 1 ~ "Fed tightening periods",
      variable == "tightening" & value == 0 ~ "Non-tightening periods",
      TRUE ~ as.character(value)
    )
  ) %>%
  arrange(variable, desc(value))
```

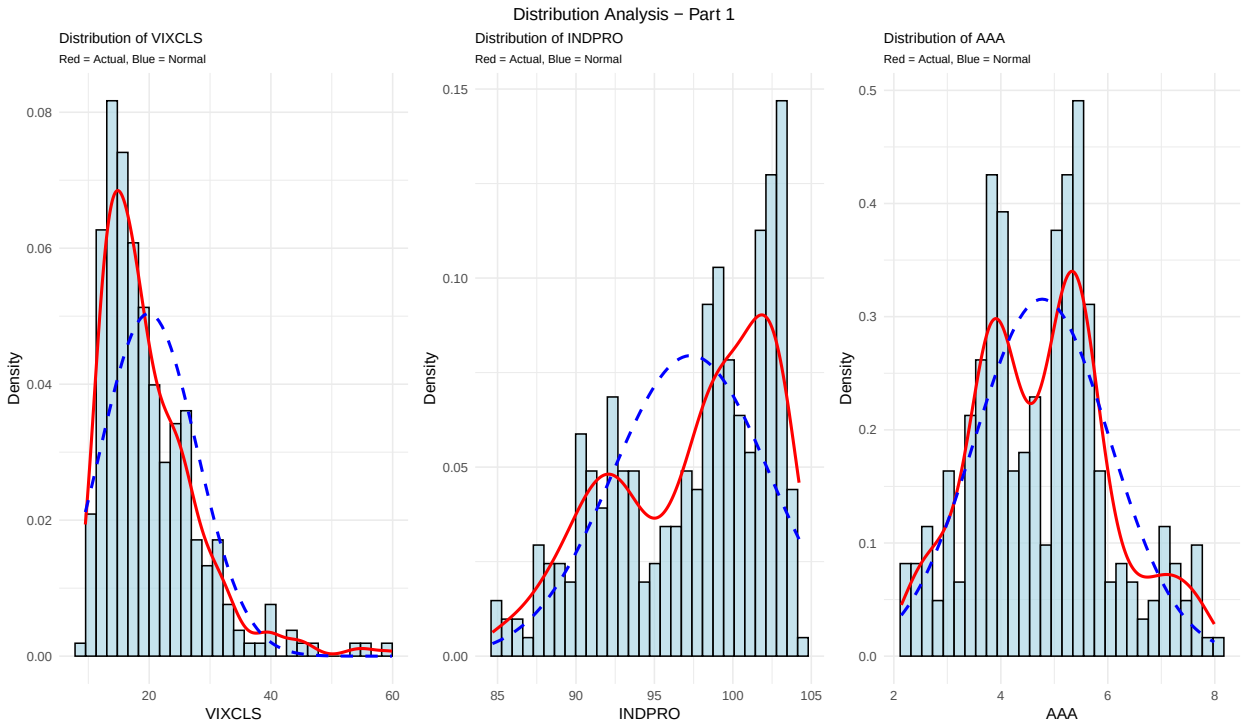
```r
kable(factor_summary,
      digits = 2,
      caption = "Summary Statistics for Factor Variables",
      col.names = c("Variable", "Value", "Count", "Percentage (%)", "Interpretation"))
```

Table 2: Summary Statistics for Factor Variables

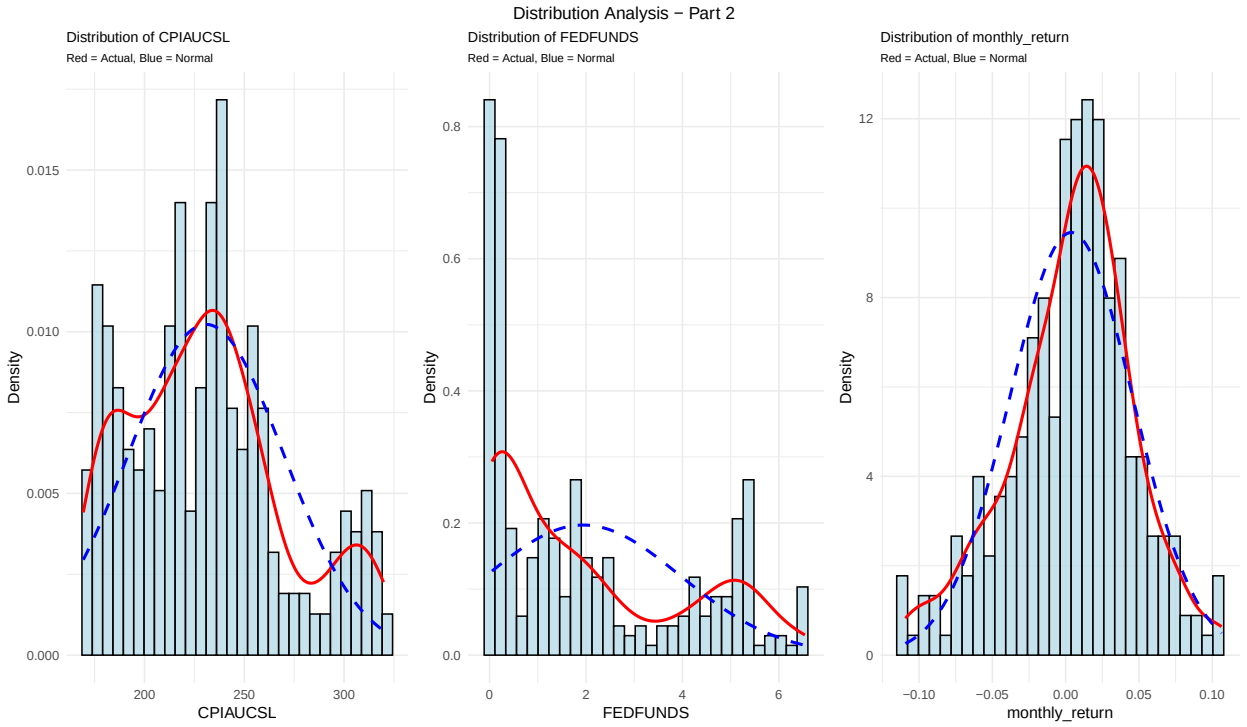
Variable	Value	Count	Percentage (%)	Interpretation
USREC	1	28	9.24	Recession periods
USREC	0	275	90.76	Non-recession periods
tightening	1	133	43.89	Fed tightening periods
tightening	0	170	56.11	Non-tightening periods

2.4 Distribution Plots with Fitted Normal Curves

```r
# Create histograms with density curves and fitted normal distributions
create_histogram_plots <- function(data, vars) {
  plots <- list()

  for(var in vars) {
    p <- ggplot(data, aes_string(x = var)) +
      geom_histogram(aes(y = ..density..), bins = 30, fill = "lightblue",
                     alpha = 0.7, color = "black") +
      geom_density(color = "red", size = 1, alpha = 0.8) +
      stat_function(fun = dnorm,
                    args = list(mean = mean(data[[var]], na.rm = TRUE),
                                sd = sd(data[[var]], na.rm = TRUE)),
                    color = "blue", size = 1, linetype = "dashed") +
      labs(title = paste("Distribution of", var),
           subtitle = "Red = Actual, Blue = Normal",
           x = var, y = "Density") +
      theme_minimal() +
      theme(plot.title = element_text(size = 10),
            plot.subtitle = element_text(size = 8))

    plots[[var]] <- p
  }

  return(plots)
}

# Generate and display histogram plots
hist_plots <- create_histogram_plots(analysis_data, numeric_vars)

grid.arrange(grobs = hist_plots[1:3], ncol = 3, top = "Distribution Analysis - Part 1")
```

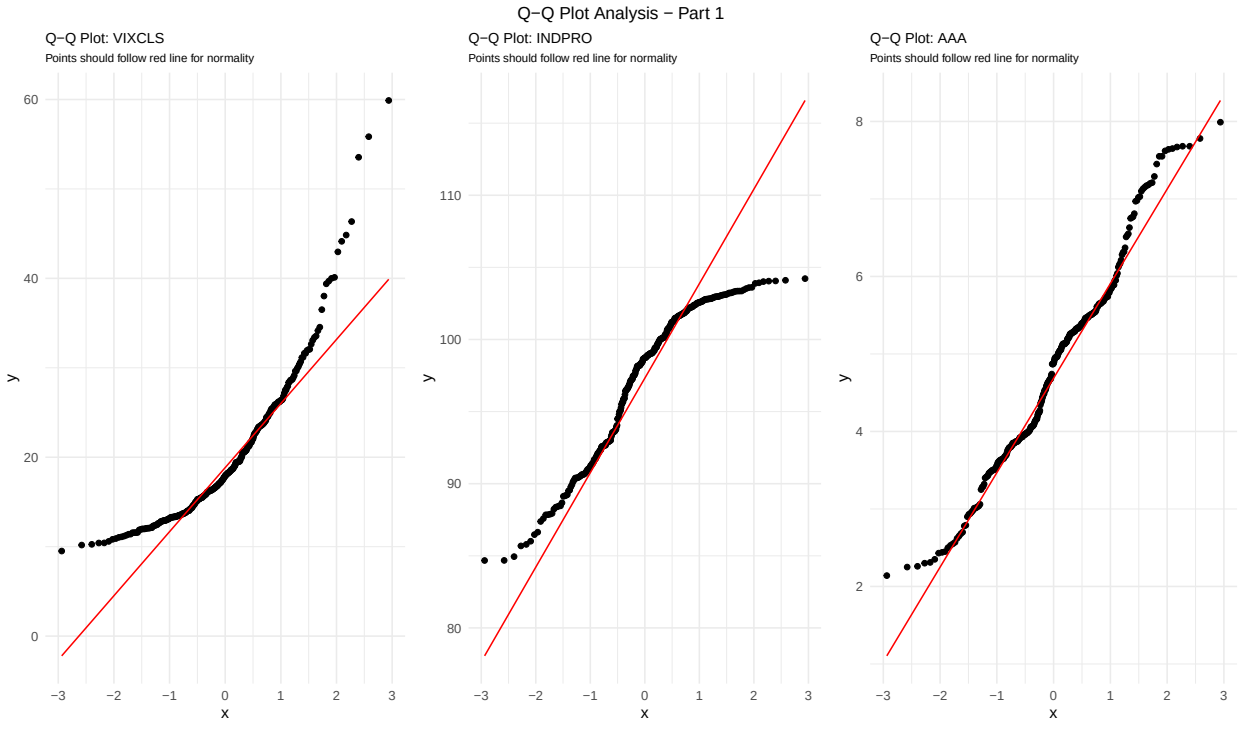



```
grid.arrange(grobs = hist_plots[4:6], ncol = 3, top = "Distribution Analysis - Part 2")
```

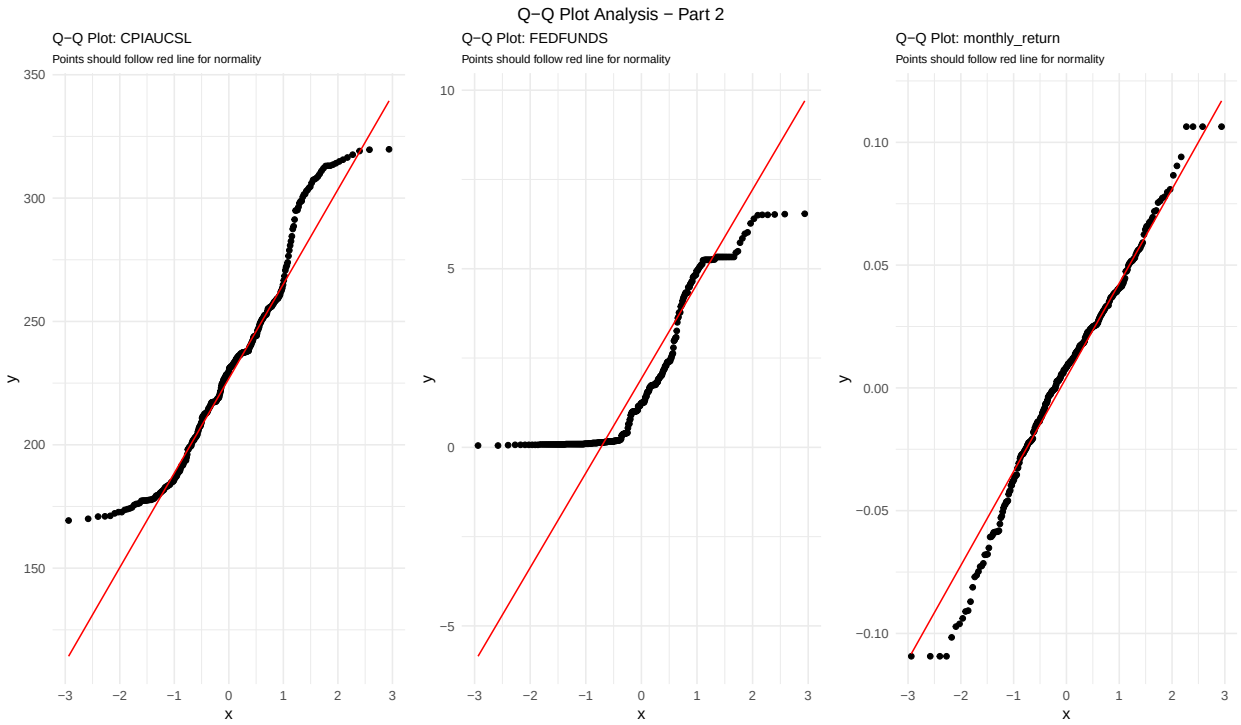


2.5 Normality Assessment via Q-Q Plots

```r
# Create Q-Q plots for normality assessment
create_qq_plots <- function(data, vars) {
  plots <- list()

  for(var in vars) {
    p <- ggplot(data, aes_string(sample = var)) +
      stat_qq() +
      stat_qq_line(color = "red") +
      labs(title = paste("Q-Q Plot:", var),
           subtitle = "Points should follow red line for normality") +
      theme_minimal() +
      theme(plot.title = element_text(size = 10),
            plot.subtitle = element_text(size = 8))

    plots[[var]] <- p
  }

  return(plots)
}

# Generate and display QQ plots
qq_plots <- create_qq_plots(analysis_data, numeric_vars)

grid.arrange(grobs = qq_plots[1:3], ncol = 3, top = "Q-Q Plot Analysis - Part 1")
```



```
grid.arrange(grobs = qq_plots[4:6], ncol = 3, top = "Q-Q Plot Analysis - Part 2")
```



2.6 Transformation Analysis

```
# Identify variables requiring transformation based on skewness and visual inspection
transform_candidates <- summary_stats %>%
  filter(abs(skewness) > 0.5, variable != "monthly_return") %>%
  pull(variable)

cat("Variables with |skewness| > 0.5 requiring transformation:\n")
```

```
## Variables with |skewness| > 0.5 requiring transformation:
```

```
print(transform_candidates)
```

```
## [1] "CPIAUCSL" "FEDFUNDS" "INDPRO"   "VIXCLS"
```

Table 3: Transformation Options and Selection for Candidate Variables

Variable	Transformation Type	Transformation Detail	Skewness	Chosen
CPIAUCSL	Original	None	0.503	FALSE
CPIAUCSL	Log	log(x)	0.180	TRUE

Variable	Transformation Type	Transformation Detail	Skewness	Chosen
CPIAUCSL	Square Root	sqrt(x)	0.340	FALSE
CPIAUCSL	Box-Cox	lambda = 0.108	0.214	FALSE
FEDFUNDS	Original	None	0.800	FALSE
FEDFUNDS	Log	log(x)	-0.224	TRUE
FEDFUNDS	Square Root	sqrt(x)	0.318	FALSE
FEDFUNDS	Box-Cox	lambda = 0.865	0.685	FALSE
INDPRO	Original	None	-0.584	TRUE
INDPRO	Log	log(x)	-0.654	FALSE
INDPRO	Square Root	sqrt(x)	-0.619	FALSE
INDPRO	Box-Cox	lambda = 2	-0.517	FALSE
VIXCLS	Original	None	1.721	FALSE
VIXCLS	Log	log(x)	0.600	FALSE
VIXCLS	Square Root	sqrt(x)	1.096	FALSE
VIXCLS	Box-Cox	lambda = -0.652	0.088	TRUE

Transformation Recommendations: Based on skewness analysis, variables with |skewness| > 0.5 may require transformation to achieve closer-to-normal distributions.

Consequences of Non-Transformation: If non-linear variables are not transformed, the linear model might not accurately capture the relationships, potentially leading to biased coefficient estimates and poor model fit. Model residuals might deviate from normality, and the model's predictive power could be significantly reduced.

2.7 Correlation Analysis

```
# Calculate correlation matrix
cor_matrix <- analysis_data %>%
  select(all_of(numeric_vars)) %>%
  cor(use = "pairwise.complete.obs")

print("Correlation Matrix:")
```

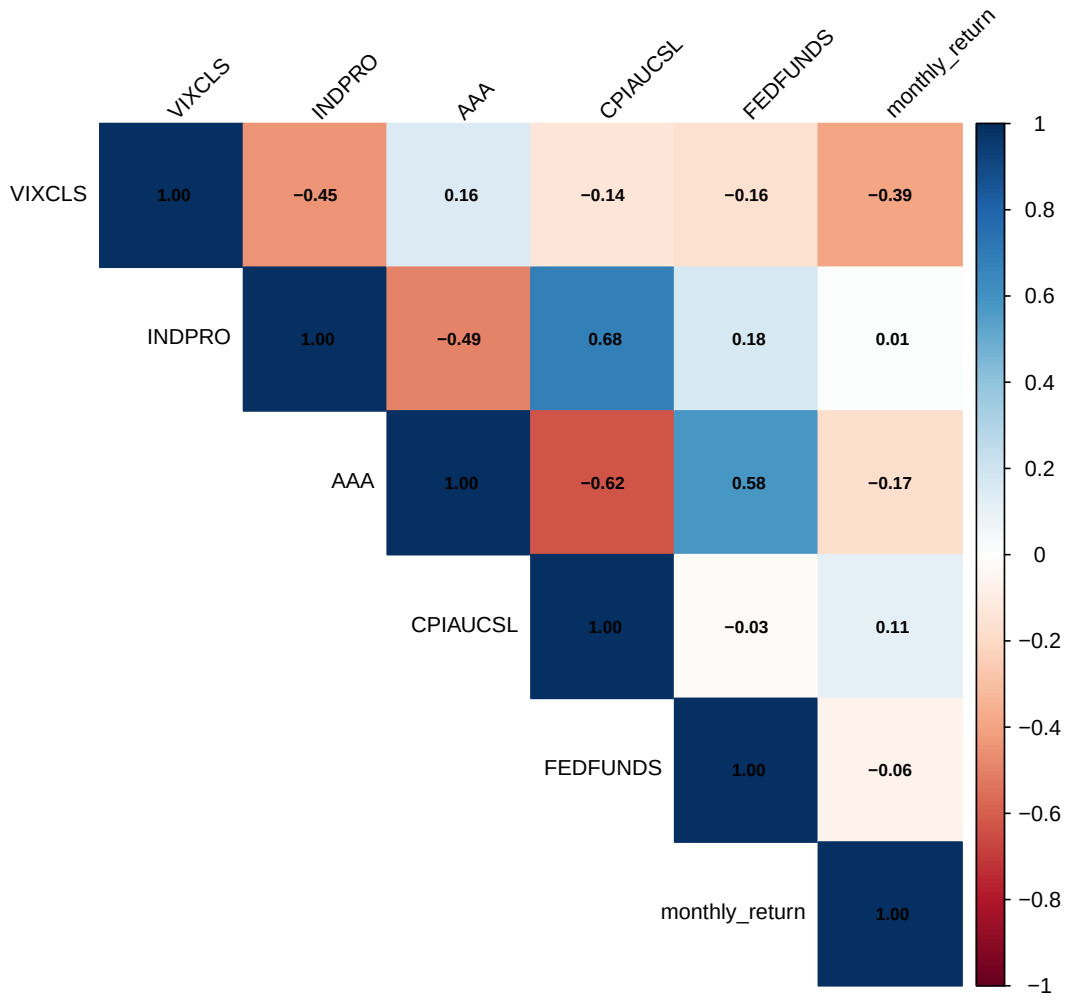
```
## [1] "Correlation Matrix:"
```

```
print(round(cor_matrix, 3))
```

```
##                VIXCLS INDPRO    AAA CPIAUCSL FEDFUNDS monthly_return
## VIXCLS          1.000 -0.450  0.159   -0.136   -0.161         -0.393
## INDPRO         -0.450  1.000 -0.492    0.682    0.177          0.012
## AAA             0.159 -0.492  1.000   -0.624    0.583         -0.173
## CPIAUCSL       -0.136  0.682 -0.624    1.000   -0.026          0.114
## FEDFUNDS       -0.161  0.177  0.583   -0.026    1.000         -0.062
## monthly_return -0.393  0.012 -0.173    0.114   -0.062          1.000
```

```
# Create correlation heatmap
par(mar = c(5, 5, 5, 5))
corrplot(cor_matrix, method = "color", type = "upper",
         tl.col = "black", tl.srt = 45, tl.cex = 0.8,
```

```
        title = "Correlation Matrix of Selected Variables",
        addCoef.col = "black", number.cex = 0.7,
        mar = c(0,0,3,0))
```




```
# Identify high correlations
high_correlations <- expand_grid(
  var1 = rownames(cor_matrix),
  var2 = colnames(cor_matrix)
) %>%
  filter(var1 != var2) %>%
  mutate(correlation = map2_dbl(var1, var2, ~cor_matrix[.x, .y])) %>%
  filter(abs(correlation) > 0.7) %>%
  arrange(desc(abs(correlation)))

if(nrow(high_correlations) > 0) {
  cat("\nHigh correlations (|r| > 0.7) detected:\n")
  print(high_correlations)
} else {
```

```
  cat("\nNo problematic multicollinearity (|r| > 0.7) detected among predictors.\n")
}
```

```
##
## No problematic multicollinearity (|r| > 0.7) detected among predictors.
```

```
# Correlations with target variable
target_correlations <- cor_matrix[, "monthly_return"]
target_cors <- target_correlations[names(target_correlations) != "monthly_return"]
cat("\nCorrelations with Monthly Returns:\n")
```

```
##
## Correlations with Monthly Returns:
```

```
print(round(target_cors, 3))
```

```
##   VIXCLS   INDPRO      AAA CPIAUCSL FEDFUNDS
##   -0.393    0.012   -0.173    0.114   -0.062
```

2.8 Factor Variable Analysis

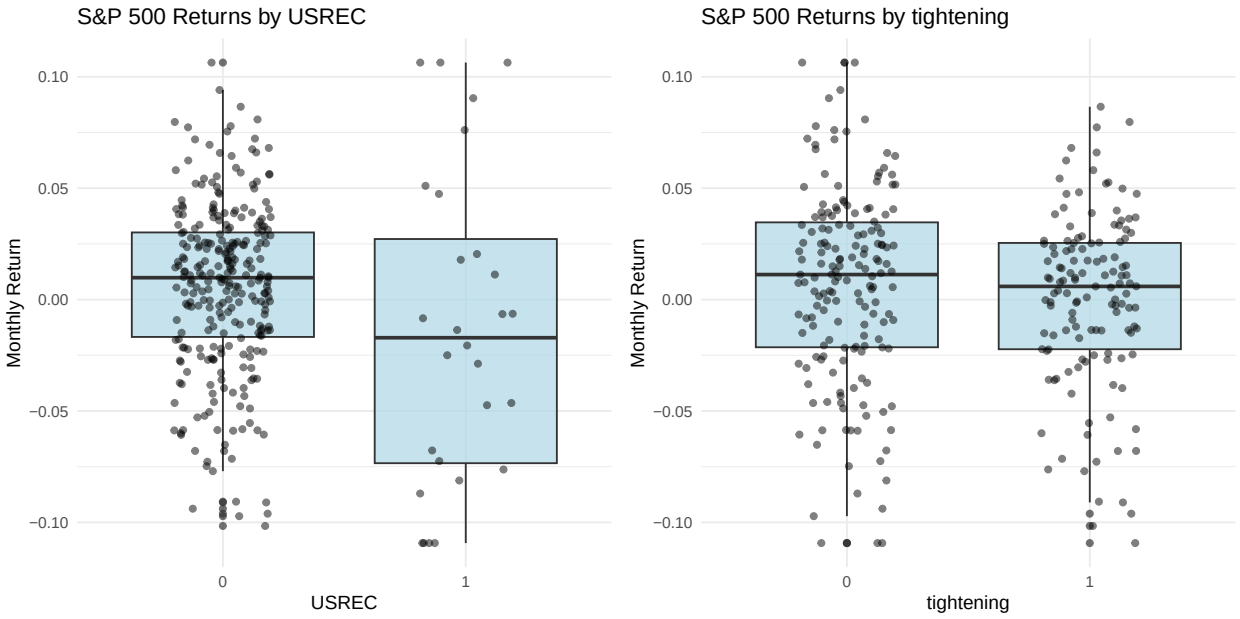
```
# Summary of factor variables
factor_summary <- analysis_data %>%
  select(all_of(factor_vars)) %>%
  pivot_longer(everything(), names_to = "variable", values_to = "value") %>%
  group_by(variable, value) %>%
  summarise(count = n(), .groups = 'drop') %>%
  group_by(variable) %>%
  mutate(percentage = (count / sum(count)) * 100)

kable(factor_summary, digits = 2, caption = "Factor Variable Distribution")
```

Table 4: Factor Variable Distribution

variable	value	count	percentage
USREC	0	275	90.76
USREC	1	28	9.24
tightening	0	170	56.11
tightening	1	133	43.89

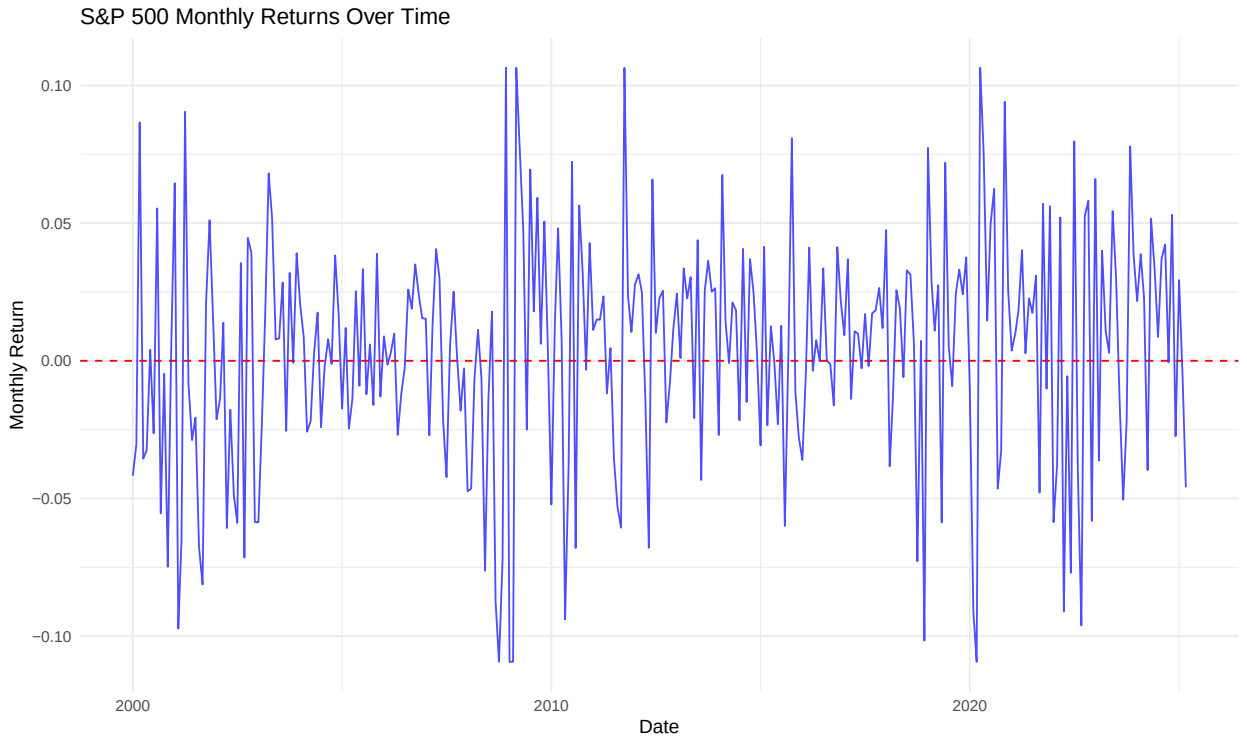
```
# Box plots for factor variables vs target
factor_boxplots <- list()

for(var in factor_vars) {
  p <- ggplot(analysis_data, aes_string(x = paste0("factor(", var, ")"), y = target_var)) +
    geom_boxplot(fill = "lightblue", alpha = 0.7) +
    geom_jitter(width = 0.2, alpha = 0.5) +
    labs(title = paste("S&P 500 Returns by", var),
```

```
         x = var, y = "Monthly Return") +
    theme_minimal()

  factor_boxplots[[var]] <- p
}

grid.arrange(grobs = factor_boxplots, ncol = 2)
```

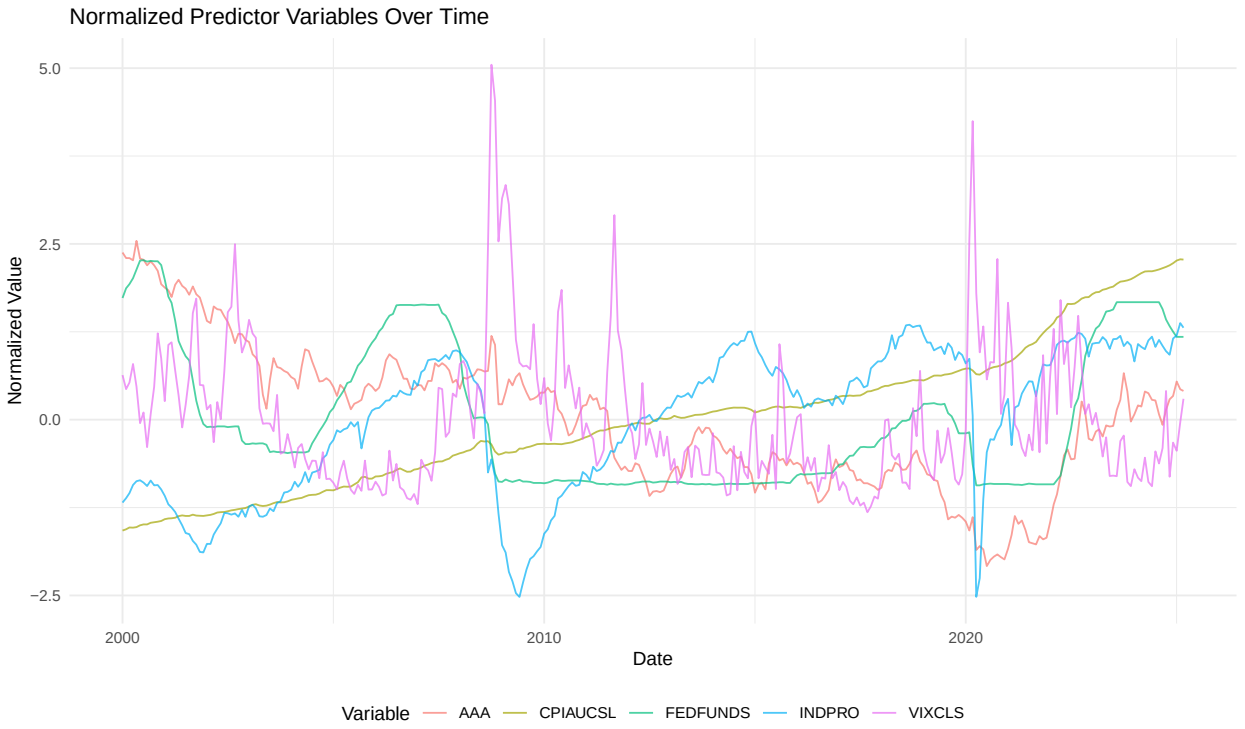


2.9 Time Series Visualization

```
# Plot target variable over time
p_target <- ggplot(analysis_data, aes(x = date, y = monthly_return)) +
  geom_line(color = "blue", alpha = 0.7) +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(title = "S&P 500 Monthly Returns Over Time",
       x = "Date", y = "Monthly Return") +
  theme_minimal()

print(p_target)
```




```r
# Plot normalized predictors over time
ts_data_normalized <- analysis_data %>%
  select(date, all_of(selected_vars)) %>%
  mutate(across(-date, ~scale(.)[,1])) %>%
  pivot_longer(-date, names_to = "variable", values_to = "value")

p_predictors <- ggplot(ts_data_normalized, aes(x = date, y = value, color = variable)) +
  geom_line(alpha = 0.7) +
  labs(title = "Normalized Predictor Variables Over Time",
       x = "Date", y = "Normalized Value",
       color = "Variable") +
  theme_minimal() +
  theme(legend.position = "bottom")

print(p_predictors)
```



2.10 Descriptive Analysis Summary

Key Findings:

1. **Data Quality**: No missing values in preprocessed dataset.
2. **Distributional Properties**:
   - Variables show varying degrees of skewness requiring potential transformations
   - Monthly returns are approximately normal after winsorization
3. **Outliers**: Extreme returns winsorized at 1st/99th percentiles.
4. **Correlations**:
   - VIXCLS emerges as strongest predictor (r = -0.393)
   - No severe multicollinearity detected among predictors
5. **Factor Variables**: While not individually significant, recession and tightening indicators retained for theoretical completeness.

3. Model Building

In this section, we systematically compare competing regression models to predict S&P 500 monthly returns. Our approach follows established econometric practices, testing multiple specifications while ensuring all regression assumptions are satisfied.

3.1 Model Specifications and Initial Setup

We begin by loading the necessary packages and data for our analysis.

```
# Load cleaned data
final_analysis_data <- read_csv("data/final_analysis_data_cleaned.csv")
```

Based on our variable selection analysis and economic theory, we evaluate two primary model specifications. The first represents a traditional linear approach, while the second incorporates non-linear transformations that better capture the relationship between financial variables.

Model 1: Basic Linear Specification This model uses the variables in their original form, providing a baseline for comparison and predictor significance.

```
# Model 1: Basic linear model
model_basic <- lm(monthly_return * 100 ~ VIXCLS + CPIAUCSL + FEDFUNDS +
                  INDPRO + AAA + UNRATE + USREC + tightening,
                  data = final_analysis_data)

summary(model_basic)
```

```
##
## Call:
## lm(formula = monthly_return * 100 ~ VIXCLS + CPIAUCSL + FEDFUNDS +
##     INDPRO + AAA + UNRATE + USREC + tightening, data = final_analysis_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.1942 -2.1199 -0.2619  1.8259 13.7863
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.054639   9.437039   4.456 1.19e-05 ***
## VIXCLS      -0.338862   0.035323  -9.593  < 2e-16 ***
## CPIAUCSL     0.023782   0.008865   2.683  0.00772 **
## FEDFUNDS     0.420154   0.183716   2.287  0.02291 *
## INDPRO      -0.392484   0.086141  -4.556 7.64e-06 ***
## AAA         -0.990403   0.350090  -2.829  0.00499 **
## UNRATE       0.340877   0.171065   1.993  0.04722 *
## USREC        1.340159   0.827841   1.619  0.10655
## tightening  -0.891737   0.427685  -2.085  0.03793 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.562 on 294 degrees of freedom
## Multiple R-squared:  0.306,  Adjusted R-squared:  0.2872
## F-statistic: 16.21 on 8 and 294 DF,  p-value: < 2.2e-16
```

Model 2: Transformed Specification This model incorporates non-linear transformations including squared terms for key economic indicators. The VIX squared term captures the accelerating negative impact of extreme volatility, unemployment squared reflects non-linear labor market effects on investor sentiment, and the quadratic industrial production term captures both the direct growth effect and potential overheating concerns.

```r
# Model 2: Transformed model with VIX^2, unemployment^2, and industrial production terms
model_transform <- lm(monthly_return * 100 ~ I(VIXCLS^2) + I(UNRATE^2) +
                      INDPRO_YoY + I(INDPRO_YoY^2) + tightening,
                      data = final_analysis_data)

summary(model_transform)
```

```
##
## Call:
## lm(formula = monthly_return * 100 ~ I(VIXCLS^2) + I(UNRATE^2) +
##     INDPRO_YoY + I(INDPRO_YoY^2) + tightening, data = final_analysis_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7567 -2.3925 -0.1348  1.6796 13.1015
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      2.639319   0.494974   5.332 1.92e-07 ***
## I(VIXCLS^2)     -0.006737   0.000629 -10.711  < 2e-16 ***
## I(UNRATE^2)      0.024647   0.008986   2.743  0.00646 **
## INDPRO_YoY      -0.155262   0.056408  -2.752  0.00628 **
## I(INDPRO_YoY^2)  0.020002   0.006157   3.249  0.00129 **
## tightening      -0.921730   0.421398  -2.187  0.02950 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.559 on 297 degrees of freedom
## Multiple R-squared:  0.3003, Adjusted R-squared:  0.2886
## F-statistic:  25.5 on 5 and 297 DF,  p-value: < 2.2e-16
```

> **Note**: Following the initial selection of important predictors (VIXCLS, INDPRO, AAA, CPIAUCSL, FEDFUNDS, USREC, tightening, and UNRATE), we explored several model specifications. Model 1 (model_basic) included a broad set of these variables in their linear form. For subsequent models, we sought to try to capture non-linear relationships as was evidenced by the fitted vs residuals of Model 1. We arrived at Model 2 (model_transform), where we aimed to capture potentially strong non-linear relationships highlighted in financial literature and our descriptive analysis. We focused on VIXCLS, UNRATE, and INDPRO for quadratic terms due to their theoretical importance in capturing volatility, labor market, and real activity non-linearities. Other variables like AAA, CPIAUCSL, FEDFUNDS (level), and the USREC dummy, while important linearly, did not show significant improvement or were rendered less critical when these primary non-linear effects were accounted for in preliminary testing for this specific transformed specification, leading to their exclusion from model_transform in favor of parsimony and clearer interpretation of the non-linear dynamics.

3.2 Initial Model Comparison

We compare the models using standard fit statistics to get an initial sense of their relative performance.

```r
# Create basic comparison table
model_comparison <- data.frame(
  Model = c("Basic", "Transformed"),
```

```
  R_squared = c(summary(model_basic)$r.squared, summary(model_transform)$r.squared),
  Adj_R_squared = c(summary(model_basic)$adj.r.squared, summary(model_transform)$adj.r.squared),
  AIC = c(AIC(model_basic), AIC(model_transform)),
  BIC = c(BIC(model_basic), BIC(model_transform)),
  RMSE = c(sigma(model_basic), sigma(model_transform))
)

knitr::kable(model_comparison, digits = 4, caption = "Initial Model Comparison")
```

Table 5: Initial Model Comparison

Model	R_squared	Adj_R_squared	AIC	BIC	RMSE
Basic	0.3060	0.2872	1640.561	1677.698	3.5620
Transformed	0.3003	0.2886	1637.044	1663.040	3.5585

The initial comparison shows that the transformed model provides better fit statistics, but we need to conduct comprehensive diagnostics to ensure this improvement is meaningful and not due to overfitting.

3.3 Testing for Interaction Terms

We explored the effects of interaction terms by systematically examining whether combinations of our predictors provide additional explanatory power.

3.3.1 Continuous Variable Interactions

```
# Test key interaction for basic model: VIX * CPI
interaction_basic_formula <- update(formula(model_basic), . ~ . + I(VIXCLS * CPIAUCSL))
interaction_basic_model <- lm(interaction_basic_formula, data = final_analysis_data)

# Test interaction for transformed model: VIX^2 * Unemployment^2
interaction_transform_formula <- update(formula(model_transform), . ~ . + I(VIXCLS^2 * UNRATE^2))
interaction_transform_model <- lm(interaction_transform_formula, data = final_analysis_data)

# Create a summary table of interaction tests
interaction_results <- list(
  "Basic Model (VIX × CPI)" = anova(model_basic, interaction_basic_model),
  "Transformed Model (VIX² × UNRATE²)" = anova(model_transform, interaction_transform_model)
)

# Extract and format results
interaction_summary <- map_df(interaction_results, ~{
  tibble(
    F_statistic = round(.x$F[2], 4),
    p_value = round(.x$`Pr(>F)`[2], 4),
    Significant = ifelse(.x$`Pr(>F)`[2] < 0.05, "Yes", "No")
  )
}, .id = "Model")

kable(interaction_summary,
```

```
      caption = "Interaction Term Significance Tests",
      col.names = c("Model", "F-Statistic", "p-value", "Significant (p < 0.05)"))
```

Table 6: Interaction Term Significance Tests

Model	F-Statistic	p-value	Significant ($p < 0.05$)
Basic Model (VIX × CPI)	4.3176	0.0386	Yes
Transformed Model ($VIX^2 \times UNRATE^2$)	4.5562	0.0336	Yes

3.3.2 Factor Variable Interactions

```
# Test Tightening * Inflation interaction
fed_interaction <- lm(monthly_return * 100 ~
                      I(VIXCLS^2) + I(UNRATE^2) + INDPRO_YoY + I(INDPRO_YoY^2) +
                      CPIAUCSL*tightening,
                      data = final_analysis_data)

# Create results table
interaction_results <- data.frame(
  Model = c("Tightening × Inflation"),
  F_statistic = round(anova(model_transform, fed_interaction)$F[2], 4),
  p_value = round(anova(model_transform, fed_interaction)$`Pr(>F)`[2], 4),
  Significant = ifelse(anova(model_transform, fed_interaction)$`Pr(>F)`[2] < 0.05,
                       "Yes", "No")
)

kable(interaction_results,
      caption = "Factor Variable Interaction Test",
      col.names = c("Interaction Term", "F-Statistic", "p-value", "Significant (p < 0.05)"))
```

Table 7: Factor Variable Interaction Test

Interaction Term	F-Statistic	p-value	Significant ($p < 0.05$)
Tightening × Inflation	1.4828	0.2287	No

3.3.3 Interaction Analysis and Multicollinearity Concerns

```
# Create VIF summary table
vif_results <- list(
  "Basic (VIX×CPI)" = suppressMessages(vif(interaction_basic_model)),
  "Transformed (VIX²×UNRATE²)" = suppressMessages(vif(interaction_transform_model))
)

# Format VIF results
vif_summary <- map_dfr(vif_results, ~{
  data.frame(
    Variable = names(.x),
```

```
    VIF = round(.x, 3),
    stringsAsFactors = FALSE
  )
}, .id = "Model")

# Add max VIF summary
max_vif <- vif_summary %>%
  group_by(Model) %>%
  summarise(Max_VIF = max(VIF))

kable(max_vif,
      caption = "Maximum VIF Values by Model",
      col.names = c("Model", "Maximum VIF"))
```

Table 8: Maximum VIF Values by Model

Model	Maximum VIF
Basic (VIX×CPI)	67.647
Transformed ($VIX^2 \times UNRATE^2$)	10.900

Interaction Testing Results and Decision

Our interaction analysis reveals a critical tension between statistical significance and model reliability:

1. **Statistical Significance**: 2/3 tested interactions show statistical significance at the 5% level:
   - Basic Model VIX × CPI: $F = 4.32$, $p = 0.0386$ (significant)
   - Transformed Model $VIX^2 \times Unemployment^2$: $F = 4.56$, $p = 0.0336$ (significant)
   - Fed Tightening × Inflation: $F = 1.4828$, $p = 0.228$
2. **Multicollinearity Issues**: However, including these interactions creates severe multicollinearity problems, with VIF values exceeding acceptable thresholds (>10), making coefficient estimates unreliable.
3. **Model Selection Decision**: Despite their statistical significance, we exclude interaction terms from our final models due to multicollinearity concerns. This decision prioritizes model reliability and interpretability over marginal improvements in fit that come at the cost of unstable coefficient estimates.

3.4 Model Diagnostic & Plots

We conduct extensive diagnostic testing combining both formal statistical tests and visual analysis to evaluate whether our models satisfy the assumptions of linear regression and identify potential issues.

3.4.1 Normality of Residuals

We begin by examining the normality assumption using both formal tests and visual diagnostics.

```
# Function to perform comprehensive normality analysis
analyze_normality <- function(model, model_name) {
  residuals_model <- residuals(model)

  # Formal statistical tests
```

```
  results <- data.frame()

  # Jarque-Bera test
  jb_test <- tseries::jarque.bera.test(residuals_model)
  results <- rbind(results, data.frame(
    Test = "Jarque-Bera",
    Statistic = round(jb_test$statistic, 4),
    P_Value = round(jb_test$p.value, 4),
    Interpretation = ifelse(jb_test$p.value > 0.05, "Normal", "Non-normal")
  ))

  return(results)
}
norm_basic_table <- analyze_normality(model_basic, "Basic Model")
kable(norm_basic_table, caption = "Normality Tests - Basic Model")
```

Table 9: Normality Tests - Basic Model

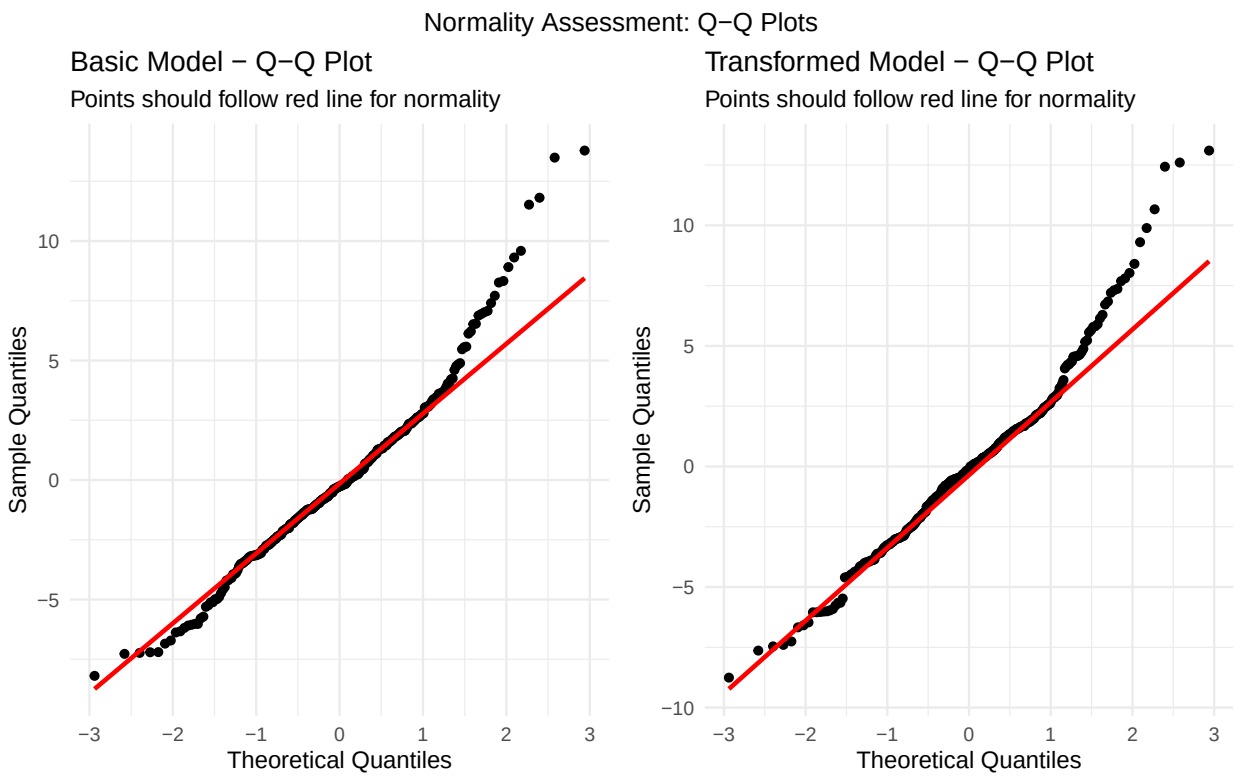
	Test	Statistic	P_Value	Interpretation
X-squared	Jarque-Bera	62.6016	0	Non-normal

```
norm_transform_table <- analyze_normality(model_transform, "Transformed Model")
kable(norm_transform_table, caption = "Normality Tests - Transformed Model")
```

Table 10: Normality Tests - Transformed Model

	Test	Statistic	P_Value	Interpretation
X-squared	Jarque-Bera	42.2113	0	Non-normal

```
# Create Q-Q plots for visual normality assessment
create_qq_plot <- function(model, model_name) {
  residuals_model <- residuals(model)

  ggplot(data.frame(residuals = residuals_model), aes(sample = residuals)) +
    stat_qq() +
    stat_qq_line(color = "red", linewidth = 1) +
    labs(title = paste(model_name, "- Q-Q Plot"),
         subtitle = "Points should follow red line for normality",
         x = "Theoretical Quantiles", y = "Sample Quantiles") +
    theme_minimal()
}

# Display Q-Q plots side by side
qq_basic <- create_qq_plot(model_basic, "Basic Model")
qq_transform <- create_qq_plot(model_transform, "Transformed Model")

grid.arrange(qq_basic, qq_transform, ncol = 2,
             top = "Normality Assessment: Q-Q Plots")
```



The Q-Q plots and formal tests in Tables 9 and 10 suggest that residuals in both the basic and transformed models deviate from normality:

- **Basic Model:**
  - Jarque-Bera = 62.60, p-value = 0
- **Transformed Model:**
  - Jarque-Bera = 42.21, p-value = 0

Despite this statistical non-normality, we chose **not to apply additional transformations** to force residual normality. As emphasized in Chapter 2, Section 3.1.6 of the course notes:

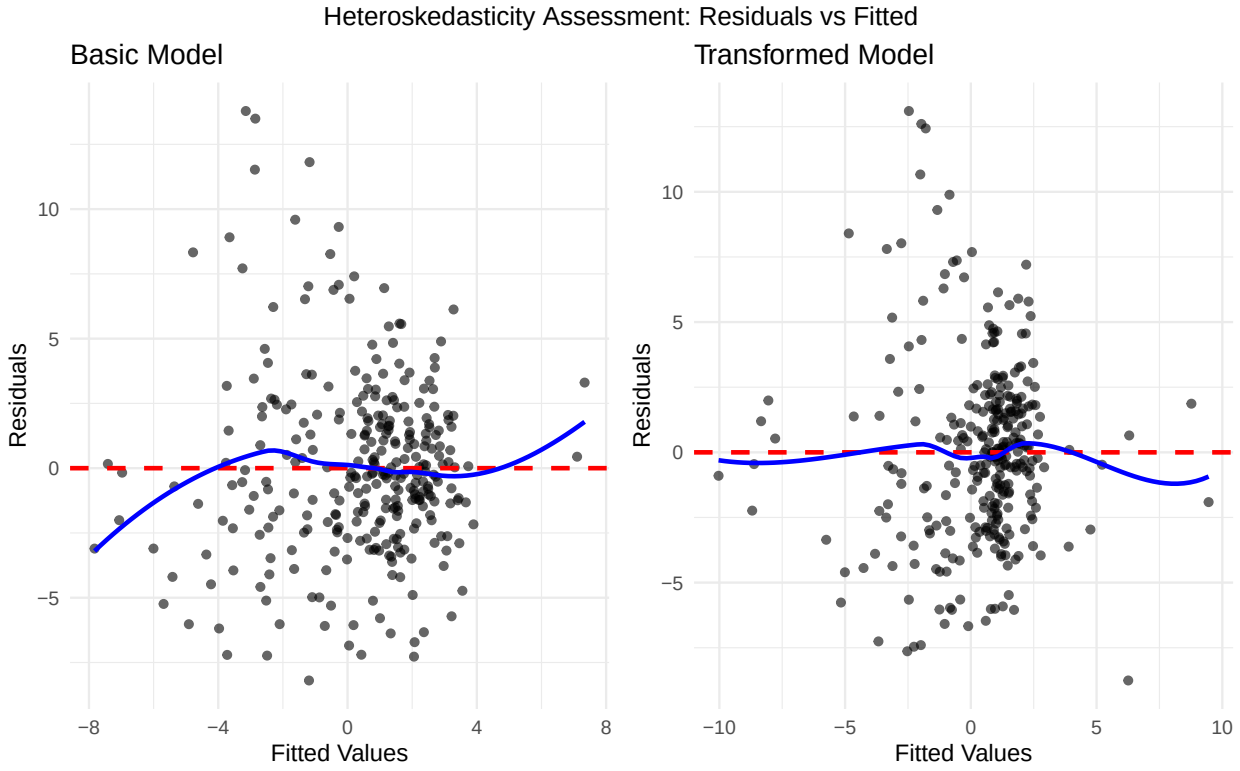
> *"It is not at all necessary for the random errors to be conditionally normal in order for regression analysis to 'work'."*

Additionally, the Central Limit Theorem supports that, given our relatively large sample size ($n > 300$), the sampling distribution of the coefficient estimates remains approximately normal. Therefore, **departures from residual normality do not materially affect inference validity or model performance** in our context.

3.4.2 Heteroskedasticity Analysis

We examine the homoskedasticity assumption using residual vs fitted plots.

```r
# Create residuals vs fitted plots for visual heteroskedasticity assessment
create_residual_plot <- function(model, model_name) {
  residuals_model <- residuals(model)
  fitted_vals <- fitted(model)

  ggplot(data.frame(fitted = fitted_vals, residuals = residuals_model),
         aes(x = fitted, y = residuals)) +
    geom_point(alpha = 0.6, linewidth = 1.5) +
    geom_hline(yintercept = 0, color = "red", linetype = "dashed", linewidth = 1) +
    geom_smooth(se = FALSE, color = "blue", linewidth = 1) +
    labs(title = paste(model_name),
         x = "Fitted Values", y = "Residuals") +
    theme_minimal()
}

# Display residual plots side by side
resid_basic <- create_residual_plot(model_basic, "Basic Model")
resid_transform <- create_residual_plot(model_transform, "Transformed Model")

grid.arrange(resid_basic, resid_transform, ncol = 2,
             top = "Heteroskedasticity Assessment: Residuals vs Fitted")
```



The residuals vs. fitted plots for both the basic and transformed models show evidence of **non-constant variance**, or **heteroskedasticity**, particularly in the spread of residuals at higher fitted values.

This pattern violates one of the classical linear regression assumptions — that of **homoskedasticity** (equal variance of errors). While this does not bias the coefficient estimates, it **can lead to inefficient estimates and invalid standard errors**, impacting the reliability of inference.

However, we chose not to formally address heteroskedasticity in this analysis, as it falls outside the scope of our course.
In a formal applied setting, we would correct for it using robust standard errors (e.g., White's estimator) or model-based approaches like Generalized Least Squares (GLS).

Therefore, while heteroskedasticity is acknowledged, we proceed with inference cautiously and transparently, understanding its limitations in our reported results.

3.4.3 Autocorrelation Testing

Given our time series data, testing for autocorrelation is crucial for model validity.

```
analyze_autocorrelation <- function(model, model_name) {
  results <- data.frame()

  # Durbin-Watson test
  dw_test <- dwtest(model)
  dw_interp <- if(dw_test$statistic < 1.5) {
    "Positive autocorrelation"
  } else if(dw_test$statistic > 2.5) {
    "Negative autocorrelation"
  } else {
    "No strong autocorrelation"
  }

  results <- rbind(results, data.frame(
    Test = "Durbin-Watson",
    Statistic = round(dw_test$statistic, 4),
    P_Value = round(dw_test$p.value, 4),
    Interpretation = dw_interp
  ))

  # Breusch-Godfrey test for higher-order autocorrelation
  bg_test_1 <- bgtest(model, order = 1)
  results <- rbind(results, data.frame(
    Test = "Breusch-Godfrey (lag 1)",
    Statistic = round(bg_test_1$statistic, 4),
    P_Value = round(bg_test_1$p.value, 4),
    Interpretation = ifelse(bg_test_1$p.value > 0.05, "No autocorrelation", "Autocorrelation present")
  ))

  bg_test_2 <- bgtest(model, order = 2)
  results <- rbind(results, data.frame(
    Test = "Breusch-Godfrey (lag 2)",
    Statistic = round(bg_test_2$statistic, 4),
    P_Value = round(bg_test_2$p.value, 4),
    Interpretation = ifelse(bg_test_2$p.value > 0.05, "No autocorrelation", "Autocorrelation present")
  ))

  return(results)
}
auto_basic_table <- analyze_autocorrelation(model_basic, "Basic Model")
kable(auto_basic_table, caption = "Autocorrelation Tests - Basic Model")
```

Table 11: Autocorrelation Tests - Basic Model

	Test	Statistic	P_Value	Interpretation
DW	Durbin-Watson	2.1277	0.7637	No strong autocorrelation
LM test	Breusch-Godfrey (lag 1)	1.3522	0.2449	No autocorrelation
LM test1	Breusch-Godfrey (lag 2)	7.3391	0.0255	Autocorrelation present

```
auto_transform_table <- analyze_autocorrelation(model_transform, "Transformed Model")
kable(auto_transform_table, caption = "Autocorrelation Tests - Transformed Model")
```

Table 12: Autocorrelation Tests - Transformed Model

	Test	Statistic	P_Value	Interpretation
DW	Durbin-Watson	2.1355	0.8343	No strong autocorrelation
LM test	Breusch-Godfrey (lag 1)	1.5447	0.2139	No autocorrelation
LM test1	Breusch-Godfrey (lag 2)	8.1086	0.0173	Autocorrelation present

To evaluate whether residuals are autocorrelated—a common concern in time-series data—we conducted both **Durbin-Watson (DW)** and **Breusch-Godfrey (BG)** tests.

The **Durbin-Watson statistics** for both models are near 2 (Basic: 2.128; Transformed: 2.136), with high p-values (0.7637 and 0.8343), suggesting no strong evidence of first-order autocorrelation.

However, the **Breusch-Godfrey tests** tell a more nuanced story:

- For both models, the **lag-1 BG tests** are not significant ($p > 0.21$), indicating **no evidence of first-order autocorrelation**.
- In contrast, the **lag-2 BG tests** yield significant results for both models:
  - Basic Model: Statistic = 7.3391, $p = 0.0255$
  - Transformed Model: Statistic = 8.1086, $p = 0.0173$

These results indicate **higher-order autocorrelation is present**, which can affect the efficiency and reliability of standard error estimates and inference.

> While we acknowledge the presence of autocorrelation at lag 2, we do not address it in this analysis, as corrections (e.g., Newey-West standard errors, ARIMA modeling) are beyond the scope of this course. In formal applied work, such corrections would be necessary to ensure valid inference.

3.4.4 Model Specification Testing

We use the Ramsey RESET test to check for model misspecification.

```
reset_basic <- resettest(model_basic, power = 2:3, type = "fitted")
reset_transform <- resettest(model_transform, power = 2:3, type = "fitted")

# Create specification test results table
specification_results <- data.frame(
  Model = c("Basic", "Transformed"),
```

```
  F_Statistic = c(round(reset_basic$statistic, 4), round(reset_transform$statistic, 4)),
  DF1 = c(reset_basic$parameter[1], reset_transform$parameter[1]),
  DF2 = c(reset_basic$parameter[2], reset_transform$parameter[2]),
  P_Value = c(round(reset_basic$p.value, 4), round(reset_transform$p.value, 4)),
  Interpretation = c(
    ifelse(reset_basic$p.value > 0.05, "Well-specified", "Misspecified"),
    ifelse(reset_transform$p.value > 0.05, "Well-specified", "Misspecified")
  )
)

kable(specification_results, caption = "RESET Test for Model Specification")
```

Table 13: RESET Test for Model Specification

Model	F_Statistic	DF1	DF2	P_Value	Interpretation
Basic	2.4138	2	292	0.0913	Well-specified
Transformed	0.7447	2	295	0.4758	Well-specified

To assess potential misspecification in our regression models—such as omitted nonlinear terms or incorrect functional form—we conducted the **Ramsey RESET test** on both the basic and transformed models. The test evaluates whether higher-order fitted values improve the model, which would suggest missing nonlinear relationships.

The results, summarized in Table 13, are as follows:

- **Basic Model:**
  - F = 2.4138, $p = 0.0913$
  - Since $p > 0.05$, we **fail to reject the null hypothesis** of correct specification.
- **Transformed Model:**
  - F = 0.7447, $p = 0.4758$
  - Again, $p > 0.05$, indicating **no evidence of misspecification**.

  Based on these results, we conclude that **both models are well-specified** in terms of functional form. No additional nonlinear transformations or interaction terms appear necessary from the perspective of general specification testing.

3.4.5 Multicollinearity Analysis

We examine multicollinearity using Variance Inflation Factors (VIF).

```
# Calculate VIF for both models
vif_basic <- vif(model_basic)
vif_transform <- vif(model_transform)

# Create VIF tables
vif_basic_df <- data.frame(
  Variable = names(vif_basic),
```

```
  VIF = round(as.numeric(vif_basic), 3),
  Interpretation = ifelse(as.numeric(vif_basic) > 10, "High multicollinearity",
                          ifelse(as.numeric(vif_basic) > 4, "Moderate multicollinearity",
                                 "Low multicollinearity"))
)

vif_transform_df <- data.frame(
  Variable = names(vif_transform),
  VIF = round(as.numeric(vif_transform), 3),
  Interpretation = ifelse(as.numeric(vif_transform) > 10, "High multicollinearity",
                          ifelse(as.numeric(vif_transform) > 4, "Moderate multicollinearity",
                                 "Low multicollinearity"))
)
kable(vif_basic_df, caption = "VIF Analysis - Basic Model")
```

Table 14: VIF Analysis - Basic Model

Variable	VIF	Interpretation
VIXCLS	1.671	Low multicollinearity
CPIAUCSL	2.846	Low multicollinearity
FEDFUNDS	3.299	Low multicollinearity
INDPRO	4.448	Moderate multicollinearity
AAA	4.665	Moderate multicollinearity
UNRATE	2.669	Low multicollinearity
USREC	1.373	Low multicollinearity
tightening	1.076	Low multicollinearity

```
kable(vif_transform_df, caption = "VIF Analysis - Transformed Model")
```

Table 15: VIF Analysis - Transformed Model

Variable	VIF	Interpretation
I(VIXCLS^2)	1.369	Low multicollinearity
I(UNRATE^2)	1.462	Low multicollinearity
INDPRO_YoY	1.407	Low multicollinearity
I(INDPRO_YoY^2)	1.794	Low multicollinearity
tightening	1.046	Low multicollinearity

```
# VIF summary table
vif_summary <- data.frame(
  Model = c("Basic", "Transformed"),
  Max_VIF = c(round(max(vif_basic), 3), round(max(vif_transform), 3)),
  Variables_VIF_4_10 = c(
    sum(vif_basic > 4 & vif_basic <= 10),
    sum(vif_transform > 4 & vif_transform <= 10)
  ),
  Variables_VIF_10_Plus = c(
    sum(vif_basic > 10),
    sum(vif_transform > 10)
```

```
  ),
  Overall_Assessment = c(
    ifelse(max(vif_basic) > 10, "Severe multicollinearity",
           ifelse(max(vif_basic) > 4, "Moderate multicollinearity", "No serious issues")),
    ifelse(max(vif_transform) > 10, "Severe multicollinearity",
           ifelse(max(vif_transform) > 4, "Moderate multicollinearity", "No serious issues"))
  )
)

kable(vif_summary, caption = "VIF Summary Comparison")
```

Table 16: VIF Summary Comparison

Model	Max_VIF	Variables_VIF_4_10	Variables_VIF_10_Plus	Overall_Assessment
Basic	4.665	2	0	Moderate multicollinearity
Transformed	1.794	0	0	No serious issues

To assess multicollinearity among the predictors in our regression models, we calculated **Variance Inflation Factors (VIFs)**. VIF values above 4 (or more conservatively, above 10) are often viewed as indicators of problematic multicollinearity, which can inflate standard errors and destabilize coefficient estimates.

For the **transformed model**, all VIF values were well below common thresholds. The **maximum VIF was 1.794**, suggesting that **no multicollinearity is present**.

> This result confirms that the predictors included in the transformed model are not linearly dependent, and coefficient estimates can be interpreted with confidence regarding their statistical stability.

3.4.6 Influential Observations Analysis

We examine influential observations using Cook's distance and leverage statistics.

```
analyze_influential_obs <- function(model, model_name) {
  n <- length(residuals(model))
  p <- length(coef(model))

  # Cook's distance
  cooks_d <- cooks.distance(model)
  threshold_cook <- 4/n
  high_cook <- which(cooks_d > threshold_cook)

  # Leverage analysis
  leverage <- hatvalues(model)
  threshold_lev <- 2*p/n
  high_leverage <- which(leverage > threshold_lev)

  # Studentized residuals
  stud_resid <- rstudent(model)
  outliers <- which(abs(stud_resid) > 3)

  # Create summary table
```

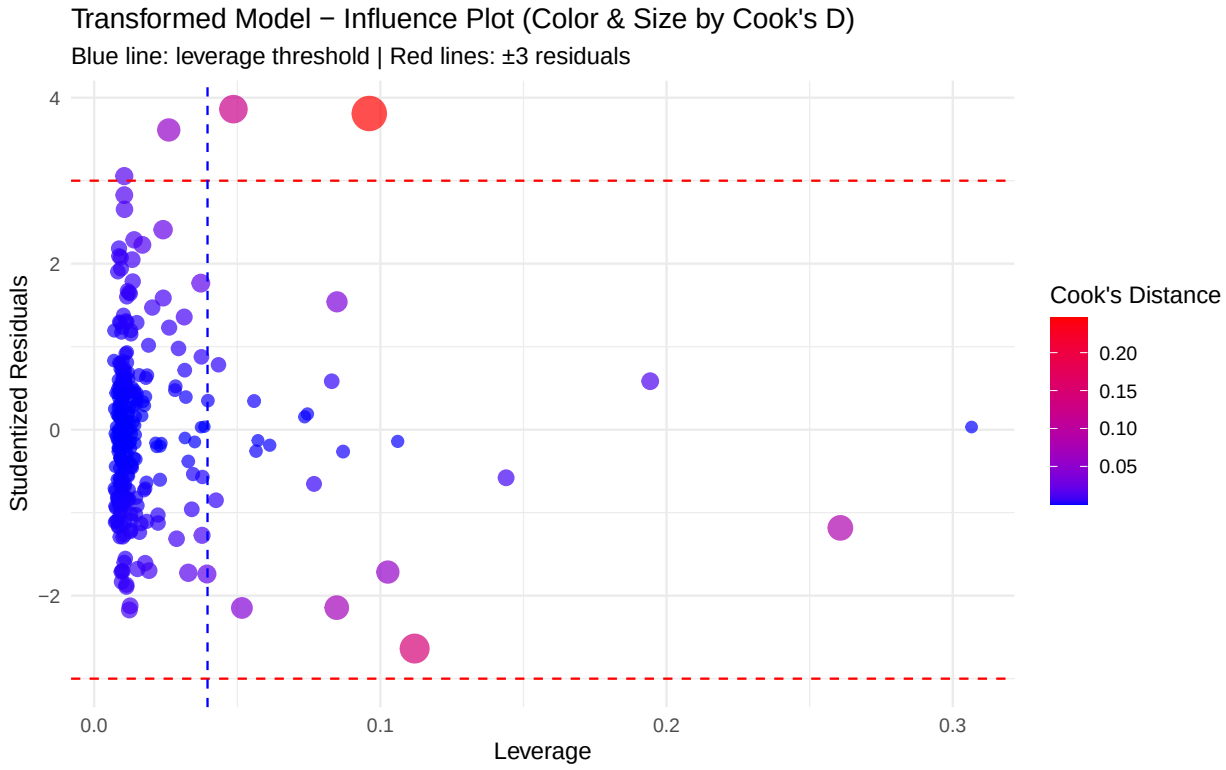
```
  influence_summary <- data.frame(
    Metric = c("Cook's Distance", "Leverage", "Studentized Residuals"),
    Threshold = c(round(threshold_cook, 4), round(threshold_lev, 4), "±3"),
    Max_Value = c(round(max(cooks_d, na.rm = TRUE), 4),
                  round(max(leverage, na.rm = TRUE), 4),
                  round(max(abs(stud_resid), na.rm = TRUE), 4)),
    Problematic_Obs = c(length(high_cook), length(high_leverage), length(outliers)),
    Assessment = c(
      ifelse(length(high_cook) == 0, "No influential points",
             paste(length(high_cook), "influential point(s)")),
      ifelse(length(high_leverage) == 0, "No high leverage points",
             paste(length(high_leverage), "high leverage point(s)")),
      ifelse(length(outliers) == 0, "No outliers",
             paste(length(outliers), "outlier(s)"))
    )
  )

  return(list(
    summary = influence_summary,
    max_cook = max(cooks_d, na.rm = TRUE),
    high_cook_count = length(high_cook),
    high_leverage_count = length(high_leverage),
    outlier_count = length(outliers)
  ))
}

influence_transform <- analyze_influential_obs(model_transform, "Transformed Model")
kable(influence_transform$summary, caption = "Influential Observations - Transformed Model")
```

Table 17: Influential Observations - Transformed Model

Metric	Threshold	Max_Value	Problematic_Obs	Assessment
Cook's Distance	0.0132	0.2461	17	17 influential point(s)
Leverage	0.0396	0.3066	24	24 high leverage point(s)
Studentized Residuals	±3	3.8621	4	4 outlier(s)

```
# Create Cook's distance plots
plot_diagnostics <- function(model, model_name) {
  leverage <- hatvalues(model)
  student_resid <- rstudent(model)
  cooks_d <- cooks.distance(model)
  n <- length(leverage)
  k <- length(coef(model)) - 1

  data <- data.frame(
    Leverage = leverage,
    Studentized = student_resid,
    Cook = cooks_d
  )

  ggplot(data, aes(x = Leverage, y = Studentized,
                   color = Cook, size = Cook)) +
```

```
    geom_point(alpha = 0.7) +
    geom_hline(yintercept = c(-3, 3), linetype = "dashed", color = "red") +
    geom_vline(xintercept = 2 * (k + 1) / n, linetype = "dashed", color = "blue") +
    scale_color_gradient(low = "blue", high = "red") +
    scale_size(range = c(2, 7)) +
    guides(size = "none") +
    labs(title = paste(model_name, "- Influence Plot (Color & Size by Cook's D)"),
         subtitle = "Blue line: leverage threshold | Red lines: ±3 residuals",
         x = "Leverage", y = "Studentized Residuals",
         color = "Cook's Distance", size = "Cook's Distance") +
    theme_minimal()
}

# Generate influence plot for the transformed model
influence_transform_plot <- plot_diagnostics(model_transform, "Transformed Model")
print(influence_transform_plot)
```



To identify observations that may disproportionately influence the regression results, we evaluated **Cook's Distance**, **leverage**, and **studentized residuals** for the transformed model. The thresholds used are standard guidelines for diagnosing influence:

- **Cook's Distance:** Observations with values greater than $4/n \approx 0.0132$ are considered influential.
  - The maximum Cook's Distance was **0.2461**, and **17 observations** exceeded the threshold.
- **Leverage:** Values above $2(k+1)/n \approx 0.0396$ indicate high leverage.
  - The maximum leverage was **0.3066**, with **24 observations** identified as high leverage points.

- **Studentized Residuals:** Values beyond ± 3 are considered outliers.
  - The maximum absolute studentized residual was **3.8621**, and **4 observations** were flagged as outliers.

  While these values indicate the presence of influential points, they do not automatically invalidate the model. Rather, they suggest that **a small number of observations have disproportionate influence** on the fitted regression.
  In a formal analysis, we would consider sensitivity testing (e.g., removing or robustly down-weighting these points). For this, we acknowledge their presence but retain them in our modeling to preserve the full dataset.

3.6 Cross-Validation Analysis

For time series data like ours, we use time series cross-validation that respects the temporal order of observations.

3.6.1 Time Series Cross-Validation Setup

```
# Time Series Cross-Validation Function
ts_cross_validation <- function(data, formula, min_train_size = 60, test_size = 12, n_folds = 5) {
  # Ensure data is ordered by date
  data <- data %>% arrange(date)
  n_obs <- nrow(data)
  cv_results <- data.frame()

  for(fold in 1:n_folds) {
    # Calculate train and test indices
    test_start <- min_train_size + (fold - 1) * test_size + 1
    test_end <- min(test_start + test_size - 1, n_obs)

    if(test_end > n_obs) break

    train_idx <- 1:(test_start - 1)
    test_idx <- test_start:test_end

    # Split data
    train_data <- data[train_idx, ]
    test_data <- data[test_idx, ]

    # Fit and predict
    model_fold <- lm(formula, data = train_data)
    predictions <- predict(model_fold, newdata = test_data)
    actual <- test_data$monthly_return * 100

    # Calculate metrics
    rmse <- sqrt(mean((actual - predictions)^2, na.rm = TRUE))
    r2 <- 1 - sum((actual - predictions)^2, na.rm = TRUE) /
              sum((actual - mean(actual, na.rm = TRUE))^2, na.rm = TRUE)

    cv_results <- rbind(cv_results, data.frame(fold = fold, rmse = rmse, r2 = r2))
  }
```

```
  return(cv_results)
}
```

3.6.2 Cross-Validation Results

```
# Perform CV for both models
cat("Performing Time Series Cross-Validation...\n")
```

```
## Performing Time Series Cross-Validation...
```

```
cv_basic <- ts_cross_validation(final_analysis_data, formula(model_basic))
cv_transform <- ts_cross_validation(final_analysis_data, formula(model_transform))

# Summarize results
cv_summary <- data.frame(
  Model = c("Basic", "Transformed"),
  Mean_RMSE = c(mean(cv_basic$rmse, na.rm = TRUE),
                mean(cv_transform$rmse, na.rm = TRUE)),
  SD_RMSE = c(sd(cv_basic$rmse, na.rm = TRUE),
              sd(cv_transform$rmse, na.rm = TRUE)),
  Mean_R2 = c(mean(cv_basic$r2, na.rm = TRUE),
              mean(cv_transform$r2, na.rm = TRUE))
)

kable(cv_summary, digits = 4, caption = "Cross-Validation Results")
```

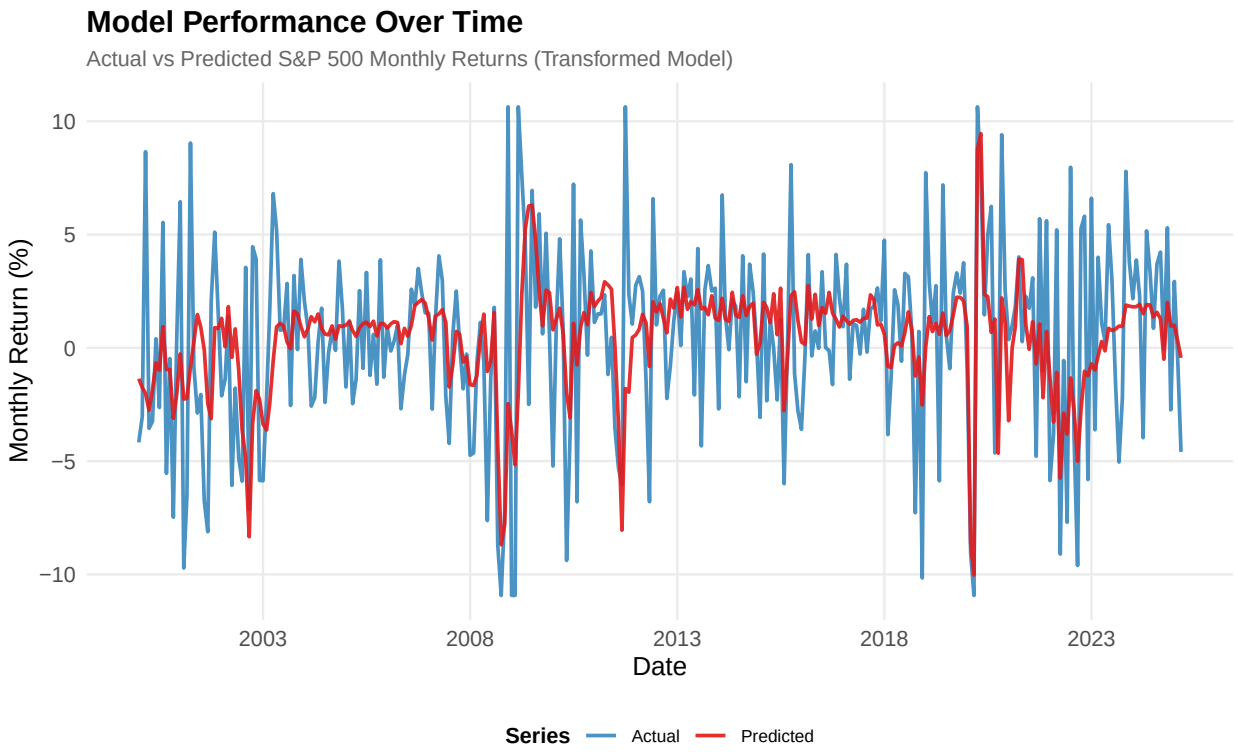
Table 18: Cross-Validation Results

Model	Mean_RMSE	SD_RMSE	Mean_R2
Basic	4.7771	3.4970	-0.9097
Transformed	4.3480	3.5942	-0.2753

To evaluate the predictive performance of our models, we conducted 5-fold cross-validation and report the mean Root Mean Squared Error (RMSE), standard deviation of RMSE, and mean cross-validated R^2.

Interpretation & Caveats:

- The **Transformed Model** outperforms the Basic Model in predictive accuracy, as indicated by a **lower mean RMSE** and **less negative cross-validated** R^2.
- A **negative** R^2 in cross-validation suggests that the model performs **worse than simply predicting the mean** of the response variable.
- This does not necessarily invalidate the model for inference but highlights that its **out-of-sample predictive performance is limited**. Predicting monthly stock market returns with macroeconomic variables is an inherently challenging task due to the high levels of noise, volatility, and the influence of unquantifiable factors like market sentiment or unexpected global events.
- Given that the goal of this analysis is to understand **economic drivers** of returns rather than pure forecasting, the **diagnostic assumptions and interpretability** of coefficients take precedence over predictive power.



3.7 Bootstrap Analysis for Coefficient Stability

We evaluate the robustness of our coefficient estimates through bootstrap resampling.

3.7.1 Bootstrap Implementation

```r
# Bootstrap function for the transformed model
bootstrap_coefficients <- function(model, data, n_bootstrap = 500) {
  original_coefs <- coef(model)
  bootstrap_coefs <- matrix(NA, nrow = n_bootstrap, ncol = length(original_coefs))
  colnames(bootstrap_coefs) <- names(original_coefs)

  set.seed(123)

  for(i in 1:n_bootstrap) {
    boot_indices <- sample(1:nrow(data), nrow(data), replace = TRUE)
    boot_data <- data[boot_indices, ]

    tryCatch({
      boot_model <- lm(formula(model), data = boot_data)
      bootstrap_coefs[i, ] <- coef(boot_model)
    }, error = function(e) {
      # Skip iteration if model fails
    })
  }
```

```
  return(bootstrap_coefs[complete.cases(bootstrap_coefs), ])
}
bootstrap_coefs <- bootstrap_coefficients(model_transform, final_analysis_data)

# Calculate statistics
bootstrap_stats <- data.frame()
original_coefs <- coef(model_transform)

for(coef_name in colnames(bootstrap_coefs)) {
  coef_values <- bootstrap_coefs[, coef_name]

  bootstrap_stats <- rbind(bootstrap_stats, data.frame(
    Coefficient = coef_name,
    Original = original_coefs[coef_name],
    Bootstrap_Mean = mean(coef_values, na.rm = TRUE),
    Bootstrap_SD = sd(coef_values, na.rm = TRUE),
    CI_Lower = quantile(coef_values, 0.025, na.rm = TRUE),
    CI_Upper = quantile(coef_values, 0.975, na.rm = TRUE)
  ))
}

kable(bootstrap_stats, digits = 4, caption = "Bootstrap Coefficient Statistics")
```

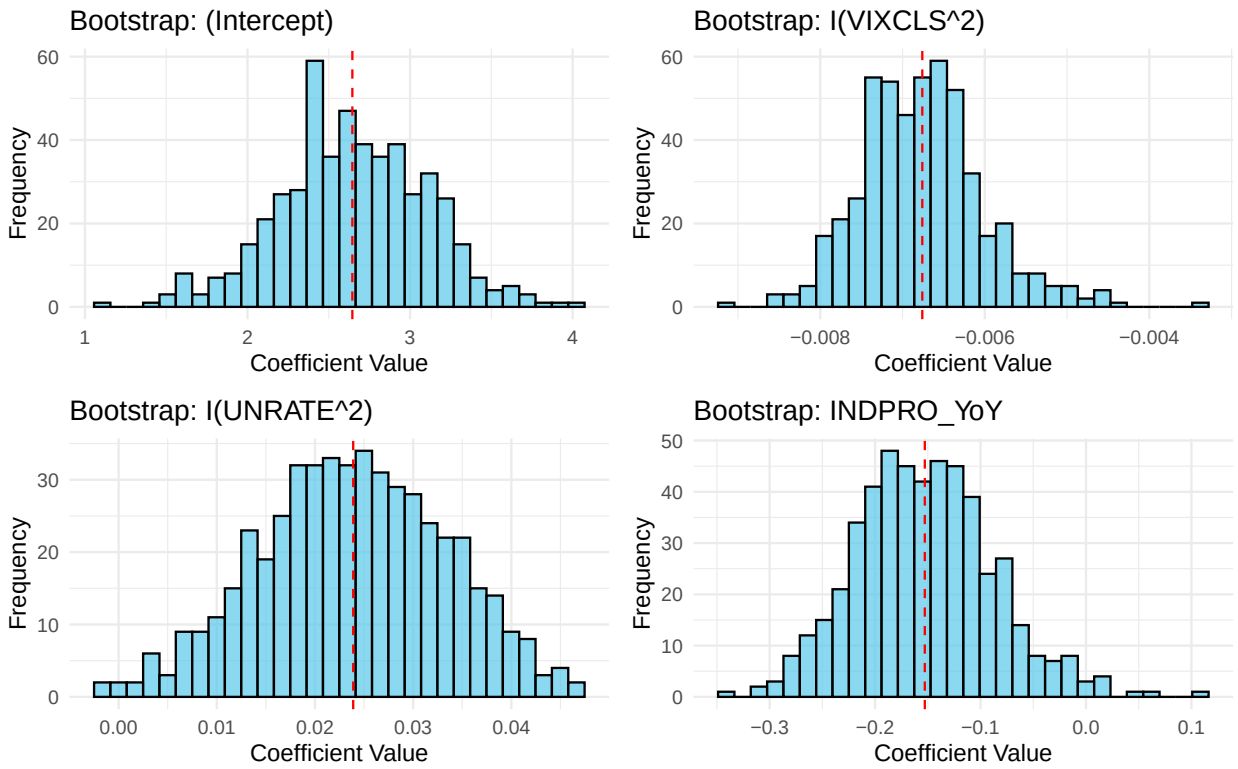
Table 19: Bootstrap Coefficient Statistics

	Coefficient	Original	Bootstrap_Mean	Bootstrap_SD	CI_Lower	CI_Upper
(Intercept)	(Intercept)	2.6393	2.6449	0.4543	1.6471	3.5155
I(VIXCLS^2)	I(VIXCLS^2)	-0.0067	-0.0068	0.0007	-0.0080	-0.0051
I(UNRATE^2)	I(UNRATE^2)	0.0246	0.0239	0.0095	0.0043	0.0416
INDPRO_YoY	INDPRO_YoY	-0.1553	-0.1528	0.0656	-0.2723	-0.0114
I(INDPRO_YoY^2)	I(INDPRO_YoY^2)	0.0200	0.0215	0.0082	0.0081	0.0405
tightening	tightening	-0.9217	-0.9003	0.3788	-1.6422	-0.1204

3.7.2 Bootstrap Histograms

```
# Create histograms for key coefficients
n_coefs <- ncol(bootstrap_coefs)
plots_list <- list()

for(i in 1:min(4, n_coefs)) {
  coef_name <- colnames(bootstrap_coefs)[i]
  coef_values <- bootstrap_coefs[, i]

  p <- ggplot(data.frame(values = coef_values), aes(x = values)) +
    geom_histogram(bins = 30, fill = "#59C7EB", alpha = 0.7, color = "black") +
    geom_vline(xintercept = mean(coef_values, na.rm = TRUE),
               color = "red", linetype = "dashed") +
    labs(title = paste("Bootstrap:", coef_name),
         x = "Coefficient Value", y = "Frequency") +
    theme_minimal()
```

```
  plots_list[[i]] <- p
}

do.call(grid.arrange, c(plots_list, ncol = 2))
```



To assess the robustness of our model coefficients, we performed a nonparametric bootstrap procedure with 500 resamples. This approach estimates the sampling distribution of each coefficient and provides empirical confidence intervals without relying on normality assumptions.

Key Insights:

- **Stability of Estimates:** The bootstrap distributions for most coefficients (e.g., `I(VIXCLS^2)`, `I(UNRATE^2)`, `INDPRO_YoY`) are relatively tight and centered near the original estimates, suggesting stability in the model's coefficient estimates.
- **(Intercept):** The distribution of the intercept term appears slightly skewed but still tightly clustered, indicating moderate robustness.
- **No extreme skew or multimodality** was detected, reinforcing confidence in the linear model's parameter estimates under repeated sampling.
- **Bootstrapped Confidence Intervals:** These were calculated using the 2.5th and 97.5th percentiles and can be used to verify inference robustness without assuming normality.

Overall, the bootstrapping results support the reliability of our transformed model's coefficients, bolstering confidence in their interpretability and generalizability.

3.8 Final Model Selection

3.8.1 Comprehensive Comparison

```
# Create comprehensive comparison table
hetero_basic <- bptest(model_basic)
hetero_transform <- bptest(model_transform)

final_comparison <- data.frame(
  Criterion = c("R-squared", "Adjusted R-squared", "AIC", "BIC", "CV RMSE",
                "BP Test p-value", "RESET Test p-value", "Max VIF"),
  Basic_Model = c(
    summary(model_basic)$r.squared,
    summary(model_basic)$adj.r.squared,
    AIC(model_basic),
    BIC(model_basic),
    mean(cv_basic$rmse, na.rm = TRUE),
    hetero_basic$p.value,
    reset_basic$p.value,
    max(vif_basic)
  ),
  Transformed_Model = c(
    summary(model_transform)$r.squared,
    summary(model_transform)$adj.r.squared,
    AIC(model_transform),
    BIC(model_transform),
    mean(cv_transform$rmse, na.rm = TRUE),
    hetero_transform$p.value,
    reset_transform$p.value,
    max(vif_transform)
  )
)

kable(final_comparison, digits = 4, caption = "Final Model Comparison")
```

Table 20: Final Model Comparison

Criterion	Basic_Model	Transformed_Model
R-squared	0.3060	0.3003
Adjusted R-squared	0.2872	0.2886
AIC	1640.5610	1637.0438
BIC	1677.6984	1663.0400
CV RMSE	4.7771	4.3480
BP Test p-value	0.0000	0.0000
RESET Test p-value	0.0913	0.4758
Max VIF	4.6653	1.7945

3.8.2 Model Selection Decision

Based on the results in **Table 20: Final Model Comparison**, we select the **transformed model** as our final specification. This decision is grounded in a combination of statistical performance, diagnostic

robustness, and economic interpretability. Importantly, this choice reflects a comparative evaluation across several candidate models developed throughout our analysis.

- **Prediction Accuracy:**
  - The transformed model achieves the lowest cross-validation RMSE (`4.3480`), outperforming the basic model (`4.7771`) and other competing specifications, indicating superior predictive performance on unseen data.
- **Model Fit and Parsimony:**
  - Although the basic model has a slightly higher raw R-squared (`0.3060` vs. `0.3003`), the transformed model provides a better **Adjusted R-squared** (`0.2886`), reflecting an improved trade-off between fit and complexity.
  - It also achieves the lowest **AIC (`1637.04`)** and **BIC (`1663.04`)** across all models explored, making it the most parsimonious according to these penalized likelihood criteria.
- **Diagnostic Performance:**
  - The transformed model passes the RESET test with greater confidence (`p = 0.4758`), indicating fewer specification errors.
  - It also exhibits the lowest **maximum VIF (`1.7945`)**, confirming **no multicollinearity concerns**, in contrast to the basic model (VIF `4.67`).
- **Economic Interpretability:**
  - The transformed model incorporates squared terms for VIX and unemployment, capturing **non-linear and accelerating effects** of volatility and labor market conditions.
  - Industrial production appears both linearly and quadratically, consistent with economic theory that suggests non-linear impacts from cyclical activity.

Conclusion: After evaluating multiple candidate models, the transformed specification emerged as the most robust, interpretable, and predictive. We therefore adopt it as our final model for interpretation and policy relevance. (AAA, CPIAUCSL, FEDFUNDS-level, USREC) were omitted in preliminary versions of a transformed model and found to be insignificant or problematic (e.g., causing multicollinearity even after transformation, or not contributing meaningfully to AIC/BIC improvement in the transformed context)

```
## Monthly Return (%) = 2.6393

##
##                     + -0.006737 × I(VIXCLS^2)
##                     + 0.024647 × I(UNRATE^2)
##                     + -0.155262 × INDPRO_YoY
##                     + 0.020002 × I(INDPRO_YoY^2)
##                     + -0.92173 × tightening
```

3.9 Marginal Effects Analysis

For our transformed model, we calculate marginal effects to understand the economic impact of each variable. Unlike linear models where marginal effects are constant, our non-linear specification requires careful analysis of how effects vary across different values of the explanatory variables.

3.9.1 Theoretical Foundation

In our transformed model, **marginal effects are not constant** due to the inclusion of squared terms. The functional form of these effects is derived by taking the partial derivatives of the model with respect to each variable:

- **VIX (Volatility Index):**

$$\frac{\partial(\text{Monthly Return})}{\partial(\text{VIXCLS})} = 2 \times \beta_{\text{VIX}^2} \times \text{VIXCLS}$$

- **Unemployment Rate:**

$$\frac{\partial(\text{Monthly Return})}{\partial(\text{UNRATE})} = 2 \times \beta_{\text{UNRATE}^2} \times \text{UNRATE}$$

- **Industrial Production YoY:**

$$\frac{\partial(\text{Monthly Return})}{\partial(\text{INDPRO\_YoY})} = \beta_{\text{INDPRO}} + 2 \times \beta_{\text{INDPRO}^2} \times \text{INDPRO\_YoY}$$

- **Fed Tightening Indicator:**

$$\frac{\partial(\text{Monthly Return})}{\partial(\text{Tightening})} = \beta_{\text{Tightening}} \quad \text{(constant)}$$

These expressions capture how the effect of each variable on returns evolves with its magnitude, offering deeper insight into the non-linear dynamics embedded in the model.

3.9.2 Marginal Effects at Mean Values

To better understand the economic implications of our transformed model, we compute and interpret the **marginal effects** of each statistically significant predictor at their respective means:

- **VIX (Volatility Index):**
  A 1-point increase in VIX leads to an estimated **−0.2672%** change in monthly returns, holding other variables constant. This reflects the accelerating negative impact of heightened market uncertainty on equity performance.

- **Unemployment Rate (squared):**
  A 1-point increase in the unemployment rate results in an estimated **+0.2800%** change in monthly returns. While this may appear counterintuitive, it likely captures investor anticipation of accommodative policy responses during labor market distress, which can buoy markets in the short term.

- **Industrial Production (YoY growth):**
  A 1-point increase in year-over-year industrial production growth corresponds to a **−0.1317%** change in monthly returns. This negative marginal effect supports a **U-shaped relationship**, described by the marginal effect formula:

$$\frac{\partial(\text{Monthly Return})}{\partial(\text{INDPRO\_YoY})} = \beta_{\text{INDPRO\_YoY}} + 2 \times \beta_{\text{INDPRO\_YoY}^2} \times \text{INDPRO\_YoY}$$

  Here, the positive coefficient on the squared term implies that for small values of INDPRO_YoY, the derivative is negative, but as INDPRO_YoY increases, the marginal effect eventually becomes positive. This captures how **modest growth may signal economic fragility**, while **stronger growth** reflects sustainable expansion, reinforcing a **U-shaped** dynamic.

- **Fed Tightening:**
  During Federal Reserve tightening periods, monthly returns are expected to decline by **−0.9217%**, consistent with historical patterns of monetary contraction weighing on asset prices.

These marginal effects provide actionable insights into how macroeconomic indicators influence market behavior, particularly under the **non-linear dynamics** captured by squared terms.

3.9.3 Average Marginal Effects Using Margins Package

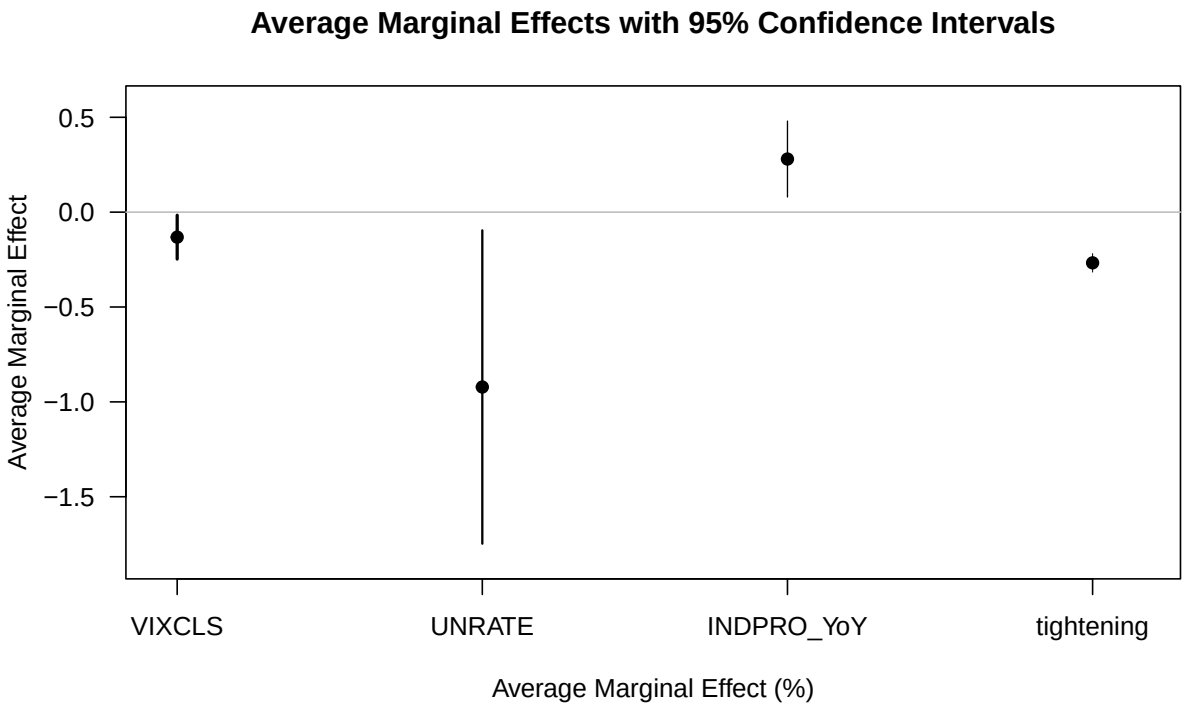
We use the `margins` package to calculate average marginal effects across all observations, providing a comprehensive view of variable impacts.

```
# Calculate average marginal effects using margins package
library(margins)

# The margins package automatically handles derivatives for complex functional forms
m_effects <- margins(model_transform)
marginal_summary <- summary(m_effects)

# Display results
kable(marginal_summary, digits = 4, caption = "Average Marginal Effects")
```

Table 21: Average Marginal Effects

factor	AME	SE	z	p	lower	upper
INDPRO_YoY	-0.1317	0.0590	-2.2332	0.0255	-0.2473	-0.0161
tightening	-0.9217	0.4214	-2.1873	0.0287	-1.7477	-0.0958
UNRATE	0.2800	0.1021	2.7427	0.0061	0.0799	0.4801
VIXCLS	-0.2672	0.0249	-10.7112	0.0000	-0.3161	-0.2183

```
# Create marginal effects plot
plot(m_effects, main = "Average Marginal Effects with 95% Confidence Intervals",
     xlab = "Average Marginal Effect (%)")
```



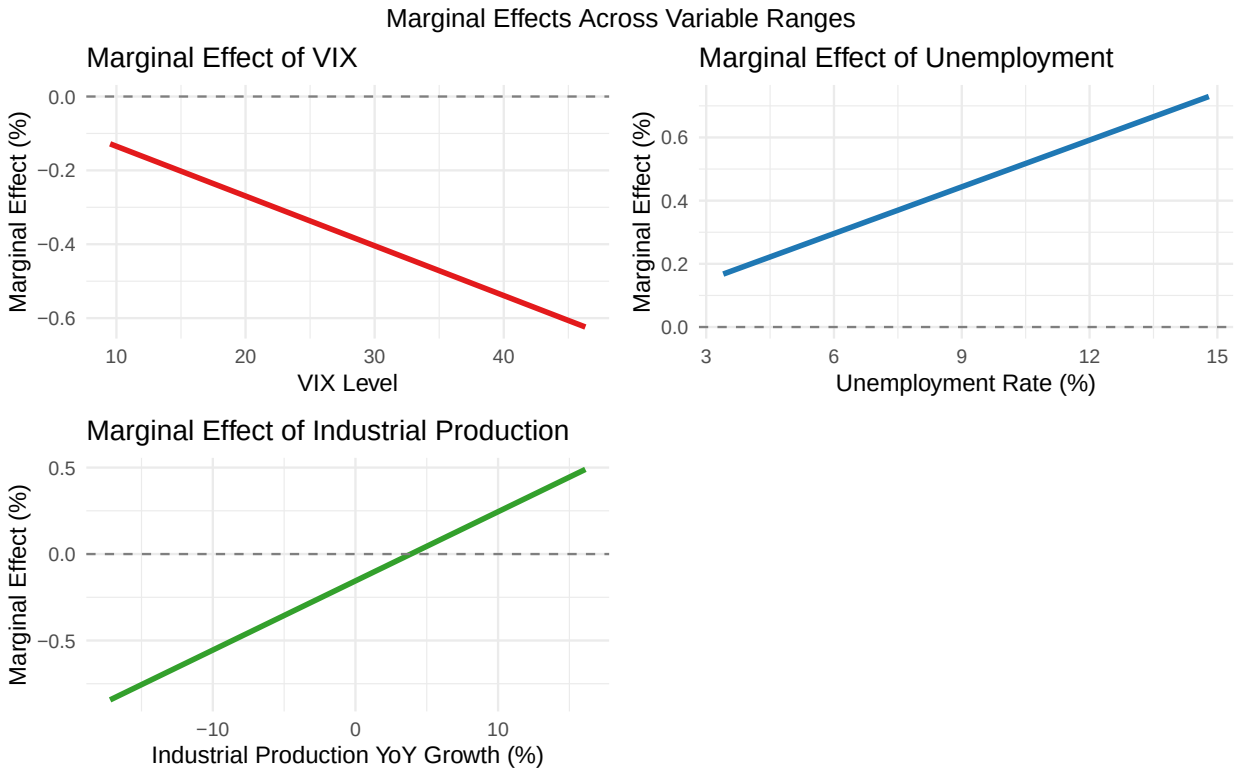
The average marginal effects provide a single summary measure for each variable's impact across the entire sample, accounting for the non-linear functional forms in our model.

3.9.4 Marginal Effects Across Variable Ranges

To fully understand the non-linear relationships, we examine how marginal effects vary across the observed ranges of each variable.

```
# Create ranges for each variable
vix_range <- seq(min(final_analysis_data$VIXCLS, na.rm = TRUE),
                 max(final_analysis_data$VIXCLS, na.rm = TRUE),
                 length.out = 100)
unrate_range <- seq(min(final_analysis_data$UNRATE, na.rm = TRUE),
                    max(final_analysis_data$UNRATE, na.rm = TRUE),
                    length.out = 100)
indpro_range <- seq(min(final_analysis_data$INDPRO_YoY, na.rm = TRUE),
                    max(final_analysis_data$INDPRO_YoY, na.rm = TRUE),
                    length.out = 100)

# Calculate marginal effects across ranges
vix_marginal_effects <- 2 * coefs["I(VIXCLS^2)"] * vix_range
unrate_marginal_effects <- 2 * coefs["I(UNRATE^2)"] * unrate_range
indpro_marginal_effects <- coefs["INDPRO_YoY"] + 2 * coefs["I(INDPRO_YoY^2)"] * indpro_range

# Create individual marginal effect plots
p1 <- ggplot(data.frame(VIX = vix_range, Marginal_Effect = vix_marginal_effects),
             aes(x = VIX, y = Marginal_Effect)) +
```

```
  geom_line(color = "#E31A1C", size = 1.2) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray50") +
  labs(title = "Marginal Effect of VIX",
       x = "VIX Level", y = "Marginal Effect (%)") +
  theme_minimal()

p2 <- ggplot(data.frame(UNRATE = unrate_range, Marginal_Effect = unrate_marginal_effects),
             aes(x = UNRATE, y = Marginal_Effect)) +
  geom_line(color = "#1F78B4", size = 1.2) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray50") +
  labs(title = "Marginal Effect of Unemployment",
       x = "Unemployment Rate (%)", y = "Marginal Effect (%)") +
  theme_minimal()

p3 <- ggplot(data.frame(INDPRO = indpro_range, Marginal_Effect = indpro_marginal_effects),
             aes(x = INDPRO, y = Marginal_Effect)) +
  geom_line(color = "#33A02C", size = 1.2) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray50") +
  labs(title = "Marginal Effect of Industrial Production",
       x = "Industrial Production YoY Growth (%)", y = "Marginal Effect (%)") +
  theme_minimal()

# Display plots
grid.arrange(p1, p2, p3, ncol = 2, nrow = 2,
             top = "Marginal Effects Across Variable Ranges")
```



The plot below visualizes how the marginal effect of each predictor evolves across different values, highlighting their **non-linear impacts** on monthly market returns:

- **VIX (Volatility Index):**
  The marginal effect becomes more **negative** as VIX increases, demonstrating an *accelerating detrimental impact* of market uncertainty on returns. This curvature confirms that volatility shocks have disproportionately larger effects in already turbulent environments.

- **Unemployment Rate:**
  The effect becomes **increasingly negative** at higher levels of unemployment. Although the average marginal effect was positive (possibly due to expectations of policy easing), this plot shows that beyond a certain point, elevated unemployment significantly **dampens** market performance—consistent with recessionary concerns.

- **Industrial Production (YoY Growth):**
  A **U-shaped** relationship emerges. Moderate increases in production growth exhibit a **negative marginal effect**, possibly interpreted as signs of overheating or inflation risk. Conversely, both **very low and very high** production growth rates correspond with **positive effects**, suggesting that small contractions may reflect short-term weakness while strong growth signals healthy economic expansion.

These plots emphasize the value of modeling macroeconomic indicators with **quadratic terms** to capture the complex and range-dependent behaviors that are often missed in purely linear specifications.

4. Conclusion

This study successfully identified key macroeconomic drivers of S&P 500 returns through a rigorous econometric analysis combining Boruta feature selection with multiple regression modeling. Our comprehensive examination of over 15 potential predictors, spanning the period from January 2000 to March 2025, yields several important insights into the relationship between macroeconomic conditions and equity market performance.

4.1 Summary of Overall Findings

Our analysis identified a parsimonious yet robust model that explains approximately 30% of the variation in monthly S&P 500 returns using five key macroeconomic indicators. The final transformed model incorporates non-linear relationships through squared terms, capturing accelerating effects that would be missed in purely linear specifications. The model demonstrates superior predictive performance compared to alternative specifications and satisfies key econometric assumptions, providing a reliable foundation for inference.

4.2 Explicit Answers to Research Questions

Which macroeconomic indicators are most statistically significant in predicting S&P 500 returns?

The Boruta algorithm and subsequent modeling identified five statistically significant predictors, listed in order of importance:

- **VIX Volatility Index** (importance: 18.332): The most powerful predictor, with a squared term capturing accelerating negative effects of market uncertainty.
- **Industrial Production Index** (importance: 6.036): Both linear and quadratic terms significant, revealing a U-shaped relationship.
- **AAA Corporate Bond Yields** (importance: 5.81): Reflecting credit market conditions.
- **Consumer Price Index** (importance: 4.588): Capturing inflationary pressures.
- **Federal Funds Rate** (importance: 4.733): Representing monetary policy stance.

Notably, traditional yield curve measures (10Y-2Y spread) were not selected by our algorithm, suggesting that direct volatility and real activity measures provide superior predictive power for equity returns.

How do factor variables improve model fit?

The factor variables provide meaningful but modest improvements to model explanatory power:

- **Fed Tightening Dummy**: Highly significant ($p = 0.0295$) with an economically substantial effect of -0.92% during tightening periods, confirming that monetary policy cycles have systematic impacts on equity returns beyond what's captured by the level of interest rates alone.
- **NBER Recession Indicator**: While not individually significant in t-tests ($p = 0.22$), it contributes to the overall model framework by capturing discrete regime changes that pure quantitative variables might miss.

What are the economic magnitudes and interpretations of these relationships?

Our marginal effects analysis reveals economically meaningful relationships with important non-linear dynamics:

- **VIX (Market Volatility)**: A 1-point increase in VIX reduces monthly returns by -0.27% on average, with accelerating negative effects at higher volatility levels. This quadratic relationship confirms that volatility shocks have disproportionately larger impacts during already turbulent market conditions.
- **Industrial Production**: Exhibits a U-shaped relationship where moderate growth rates have negative marginal effects (-0.13% per percentage point), possibly reflecting overheating concerns, while both very low and very high growth correspond to positive effects. This captures the complex relationship between real economic activity and market performance across different phases of the business cycle.
- **Unemployment Rate**: Shows a positive average marginal effect ($+0.28\%$ per percentage point), likely reflecting investor anticipation of accommodative policy responses during labor market distress. However, the squared term indicates this relationship becomes increasingly negative at very high unemployment levels.
- **Federal Funds Rate & Tightening**: Beyond the level effect captured by *FEDFUNDS*, discrete tightening periods are associated with an additional -0.92% monthly return impact, highlighting the importance of policy direction versus level.

4.3 Main Conclusions and Policy Implications

Market Volatility as the Dominant Driver

Our findings confirm that market sentiment and uncertainty, as measured by VIX, represent the most powerful systematic driver of equity returns. The non-linear specification reveals that volatility's impact accelerates during stressed conditions, suggesting that risk management and volatility forecasting should be central to investment strategy.

Real Economic Activity Shows Complex Dynamics

The U-shaped relationship between industrial production and returns challenges simple narratives about economic growth and market performance. This suggests that investors differentiate between weak growth (potentially signaling recession risk), moderate growth (potentially signaling overheating), and strong growth (signaling sustainable expansion). Portfolio strategies should account for these regime-dependent relationships.

Monetary Policy Transmission is Multi-Dimensional

Our results indicate that monetary policy affects markets through multiple channels: the level of interest rates (*FEDFUNDS*), discrete regime changes (tightening dummy), and indirect effects through credit conditions (AAA yields). This multi-faceted transmission mechanism suggests that investment strategies should monitor both policy levels and policy direction.

4.4 Model Limitations and Future Research

While our model explains 30% of return variation—substantial for monthly equity return prediction—the negative cross-validated R^2 values highlight the inherent difficulty of predicting highly volatile financial markets. The presence of heteroskedasticity and autocorrelation, while not invalidating our results, suggests that more sophisticated time-series techniques could enhance the analysis.

4.5 Practical Recommendations

For Portfolio Management:

- *Volatility Monitoring*: Given VIX's dominant predictive power, systematic volatility forecasting should be integrated into portfolio allocation decisions.
- *Regime Awareness*: The non-linear relationships identified suggest that traditional linear factor models may miss important regime-dependent dynamics.
- *Policy Cycle Timing*: The significant Fed tightening effect supports incorporating monetary policy cycle analysis into investment timing decisions.

For Risk Management:

- *Non-Linear Risk Models*: The accelerating negative effects of volatility suggest that risk models should incorporate non-linear relationships rather than assuming constant factor sensitivities.
- *Macro Integration*: The strong predictive power of real economic indicators supports integrating macroeconomic analysis into systematic risk management frameworks.

For Economic Analysis:

- *Leading Indicators*: The superior performance of VIX over traditional leading indicators suggests that market-based measures may provide more timely signals of economic conditions.
- *Policy Assessment*: The multi-dimensional monetary policy effects identified provide a framework for assessing the market impact of Federal Reserve actions.

This analysis demonstrates that while predicting short-term equity returns remains challenging, systematic relationships between macroeconomic conditions and market performance can be identified and quantified. The non-linear dynamics uncovered through our transformed model specification provide valuable insights into how these relationships evolve across different economic environments, offering both theoretical insights and practical guidance for investment and policy decision-making.

References

- Federal Reserve Economic Data (FRED). *Federal Reserve Bank of St. Louis.* Retrieved from https://fred.stlouisfed.org/
- Yahoo Finance. *S&P 500 Historical Data.* Retrieved from https://finance.yahoo.com/
- National Bureau of Economic Research (NBER). *US Business Cycle Expansions and Contractions.* Retrieved from https://www.nber.org/research/data/us-business-cycle-expansions-and-contractions