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

Emil Blaignan, Robert Sellman, Sean Eizadi

2025-05-25

Contents

1 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.

2 Variable Selection

2.1 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
```

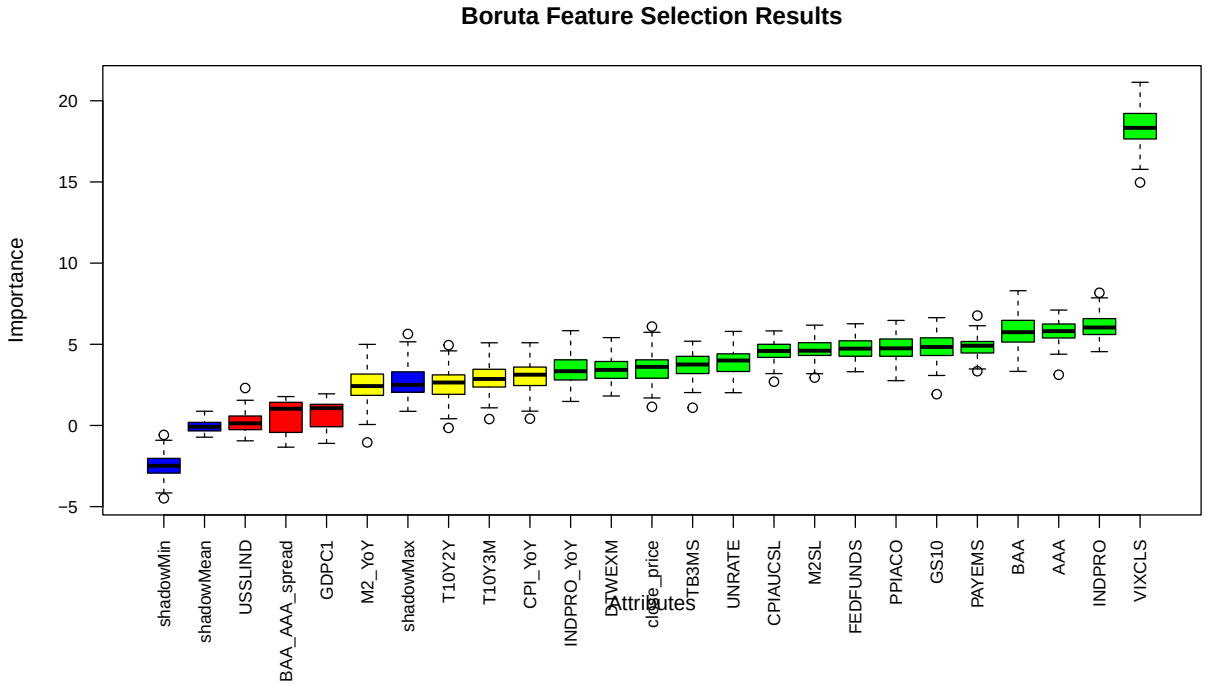
```
# Convert dates to proper format
if(is.numeric(cleaned_data$date)) {
  cleaned_data$date <- as.Date(cleaned_data$date, origin = "1970-01-01")
} else if(is.character(cleaned_data$date)) {
  cleaned_data$date <- as.Date(cleaned_data$date)
}

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 15.86356 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"
```

```r
# 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:
```

```r
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
```

2.1.1 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

```r
# 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:
```

```r
# 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)

2.2 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
```

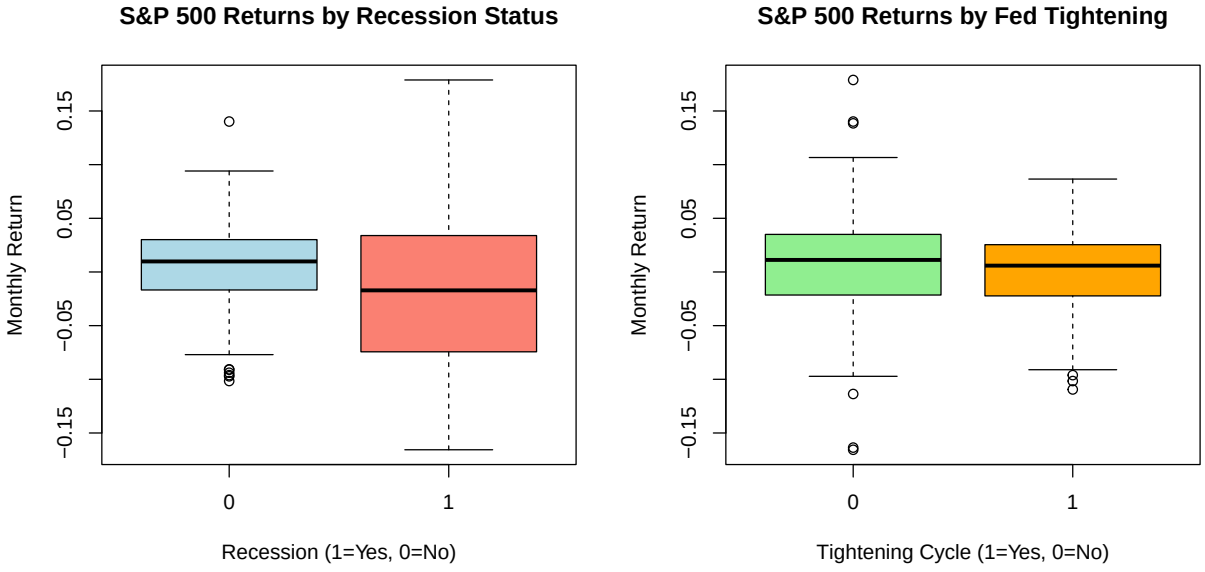
```
cat("\nFed Tightening Indicator T-Test Results:\n")
```

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

```r
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.

3 Descriptive Analysis

3.1 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.

3.2 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.

3.3 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")
```

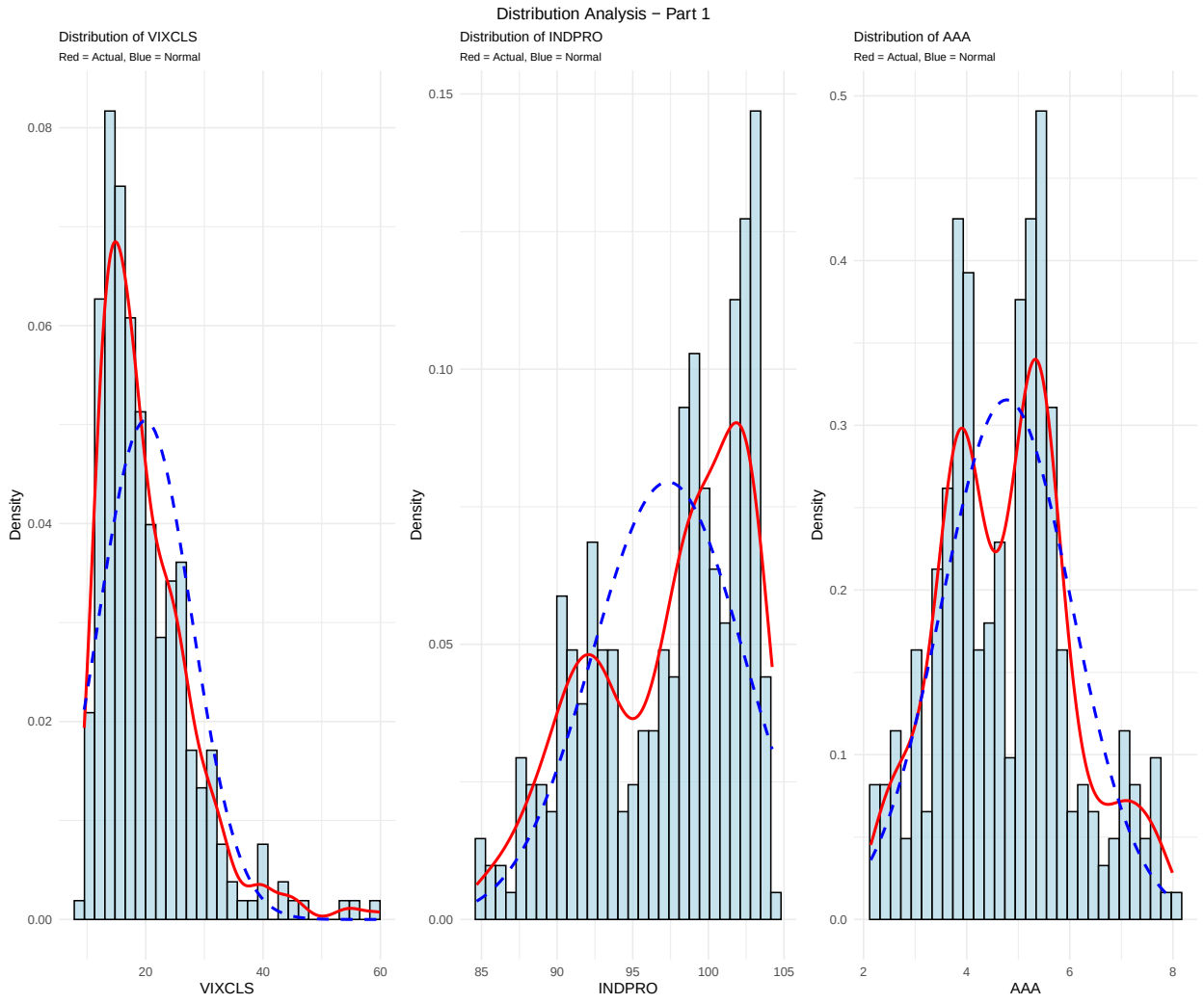
Table 1: Summary Statistics for Analysis Variables

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

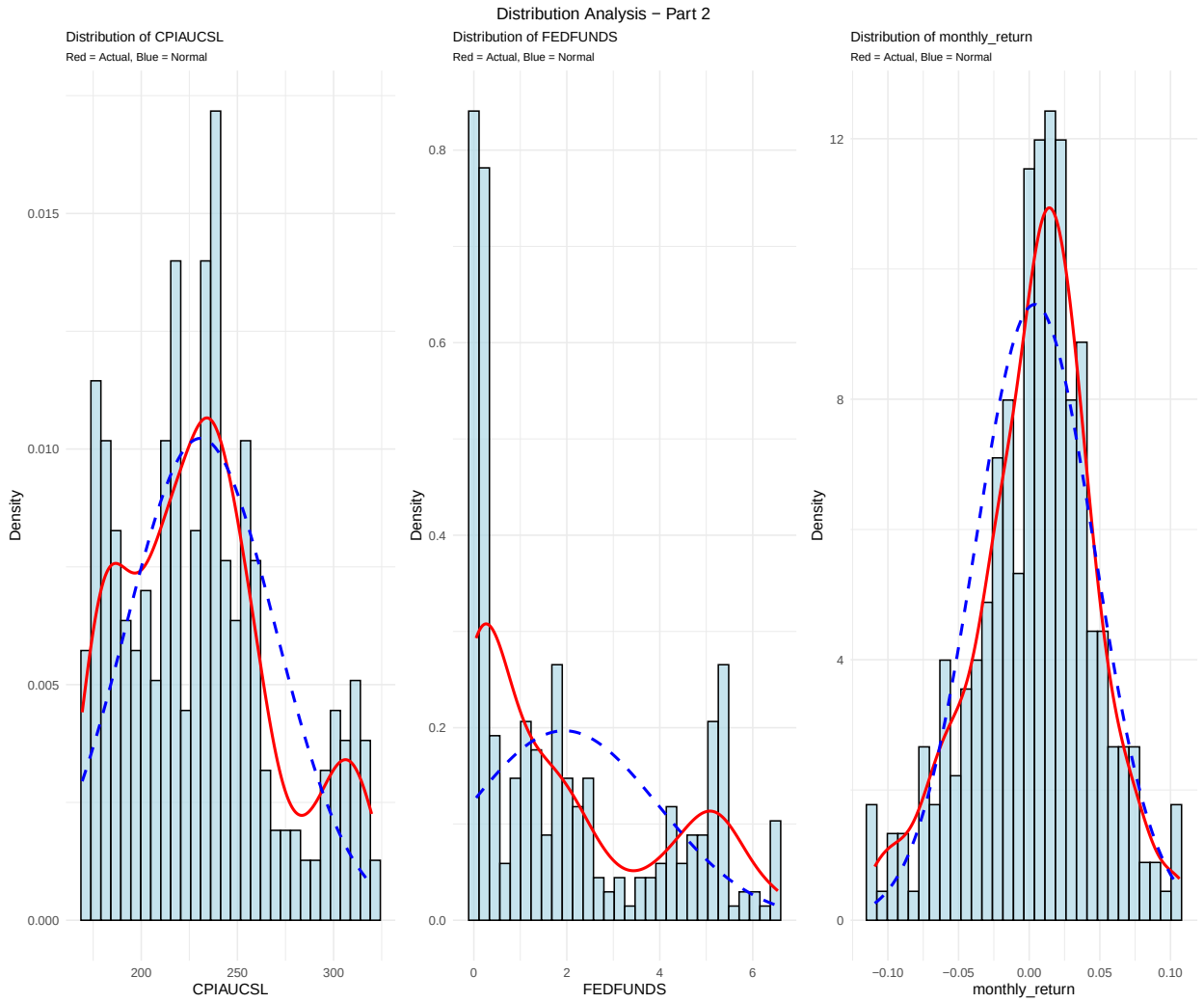
3.4 Distribution Plots with Fitted Normal Curves

```
# 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")
```



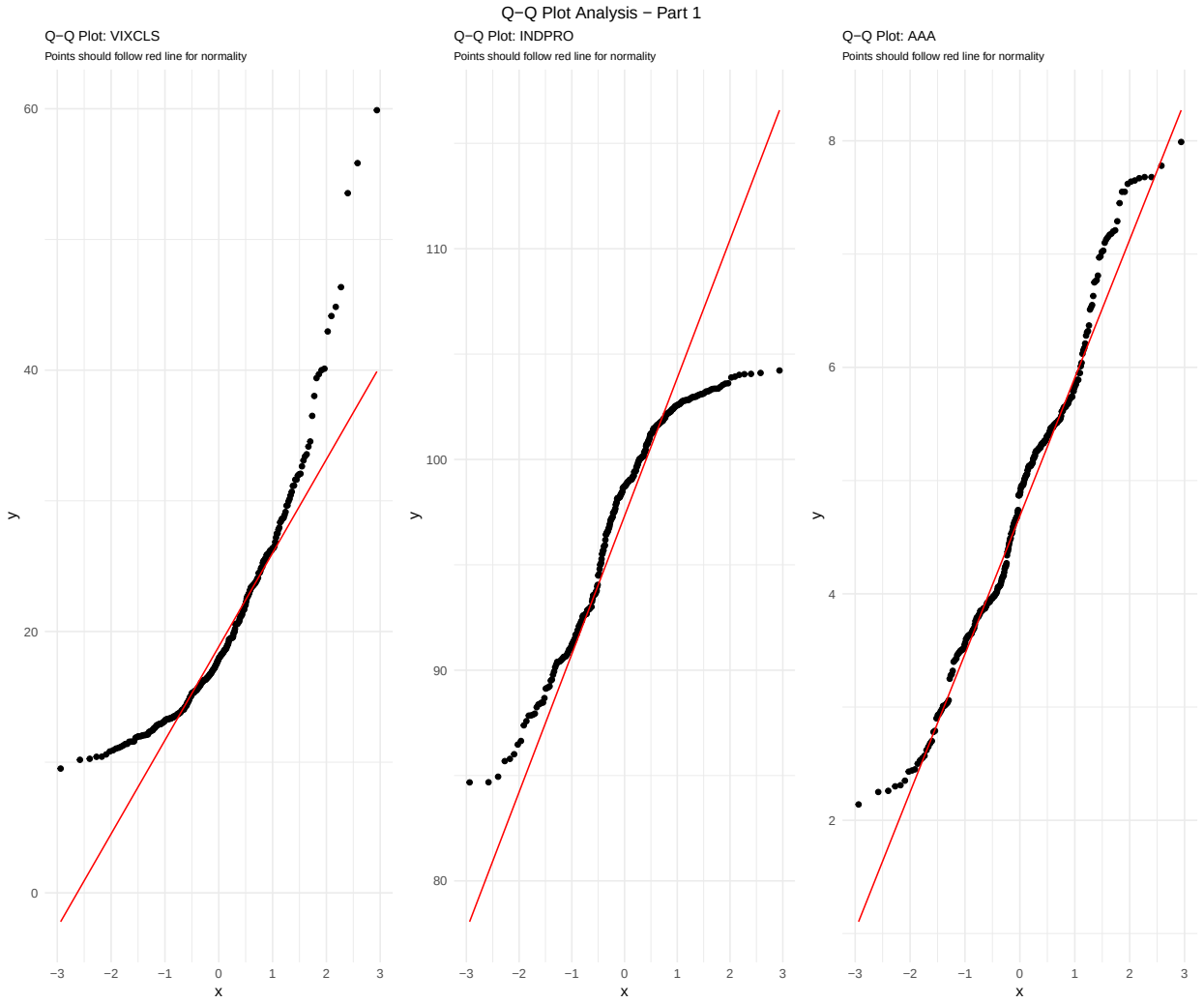

```
grid.arrange(grobs = hist_plots[4:6], ncol = 3, top = "Distribution Analysis - Part 2")
```



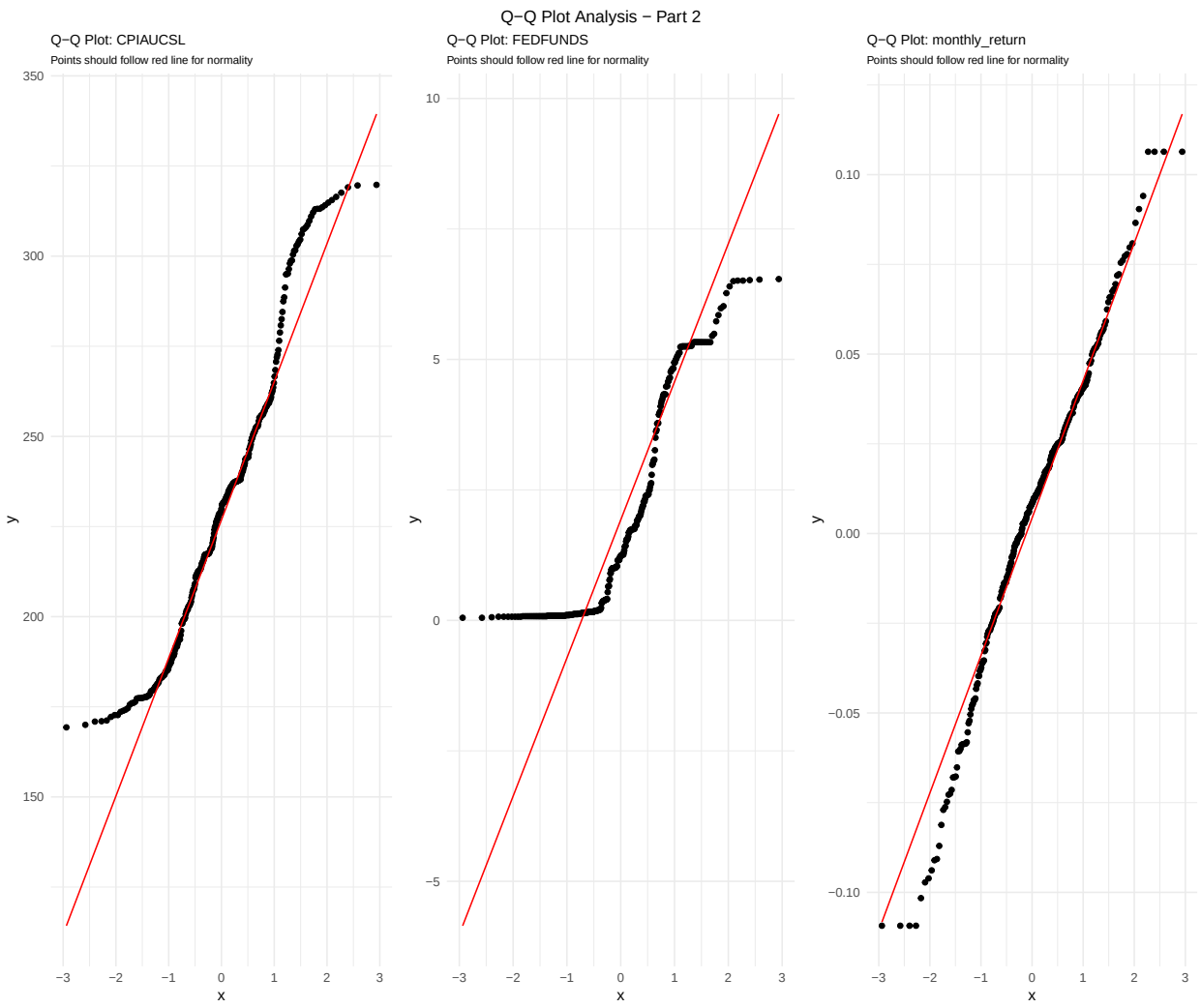
3.5 Normality Assessment via Q-Q Plots

```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")
```



3.6 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"
```

```r
# Test transformations for each candidate variable
if(length(transform_candidates) > 0) {

  # Function to test transformations for a single variable
  test_transformations <- function(var_name, data) {
    original_data <- data[!is.na(data)]
    orig_skew <- moments::skewness(original_data)

    # Initialize results
    results <- tibble(
      variable = var_name,
      transformation = "original",
      skewness = orig_skew,
      abs_skewness = abs(orig_skew)
    )

    # Test basic transformations
    if(min(original_data) > 0) {
      # Log transformation
      log_skew <- moments::skewness(log(original_data))
      results <- bind_rows(results, tibble(
        variable = var_name, transformation = "log",
        skewness = log_skew, abs_skewness = abs(log_skew)
      ))

      # Square root transformation
      sqrt_skew <- moments::skewness(sqrt(original_data))
      results <- bind_rows(results, tibble(
        variable = var_name, transformation = "sqrt",
        skewness = sqrt_skew, abs_skewness = abs(sqrt_skew)
      ))
    }

    # Yeo-Johnson transformation (works for all values)
    tryCatch({
      yj_lambda <- optimize(function(lambda) {
        transformed <- car::yjPower(original_data, lambda)
        if(any(is.na(transformed)) || any(is.infinite(transformed))) return(999)
        abs(moments::skewness(transformed))
      }, interval = c(-2, 2))$minimum

      yj_data <- car::yjPower(original_data, yj_lambda)
      if(!any(is.na(yj_data)) && !any(is.infinite(yj_data))) {
        yj_skew <- moments::skewness(yj_data)
        results <- bind_rows(results, tibble(
          variable = var_name,
          transformation = paste0("yeo_johnson_", round(yj_lambda, 3)),
          skewness = yj_skew, abs_skewness = abs(yj_skew)
        ))
      }
    }, error = function(e) {
      cat("Yeo-Johnson failed for", var_name, "\n")
    })
```

```
    return(results)
  }

  # Test transformations for all candidates
  all_results <- tibble()
  for(var in transform_candidates) {
    var_results <- test_transformations(var, analysis_data[[var]])
    all_results <- bind_rows(all_results, var_results)
  }

  # Get original skewness for each variable
  original_skewness <- all_results %>%
    filter(transformation == "original") %>%
    select(variable, original_skewness = skewness)

  # Get best transformation for each variable
  best_transformations <- all_results %>%
    group_by(variable) %>%
    slice_min(abs_skewness, n = 1) %>%
    ungroup() %>%
    select(variable, best_transformation = transformation, final_skewness = skewness)

  # Create summary table
  transformation_summary <- original_skewness %>%
    left_join(best_transformations, by = "variable") %>%
    mutate(improvement = abs(original_skewness) - abs(final_skewness)) %>%
    select(variable, original_skewness, best_transformation, final_skewness, improvement)

  kable(transformation_summary, digits = 3, caption = "Transformation Analysis Summary")
}
```

Table 2: Transformation Analysis Summary

variable	original_skewness	best_transformation	final_skewness	improvement
CPIAUCSL	0.503	yeo_johnson_-0.582	0.000	0.503
FEDFUNDS	0.800	yeo_johnson_-0.607	0.000	0.800
INDPRO	-0.584	yeo_johnson_2	-0.518	0.066
VIXCLS	1.721	yeo_johnson_-0.873	0.000	1.721

```
# Show detailed results for one variable as example
if(length(transform_candidates) > 0) {
  example_var <- transform_candidates[1]
  cat("\nDetailed transformation options for", example_var, ":\n")

  example_results <- all_results %>%
    filter(variable == example_var) %>%
    arrange(abs_skewness) %>%
    select(transformation, skewness, abs_skewness)

  kable(example_results, digits = 3)
}
```

```
##
## Detailed transformation options for CPIAUCSL :
```

transformation	skewness	abs_skewness
yeo_johnson_-0.582	0.000	0.000
log	0.180	0.180
sqrt	0.340	0.340
original	0.503	0.503

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.

3.7 Correlation Analysis

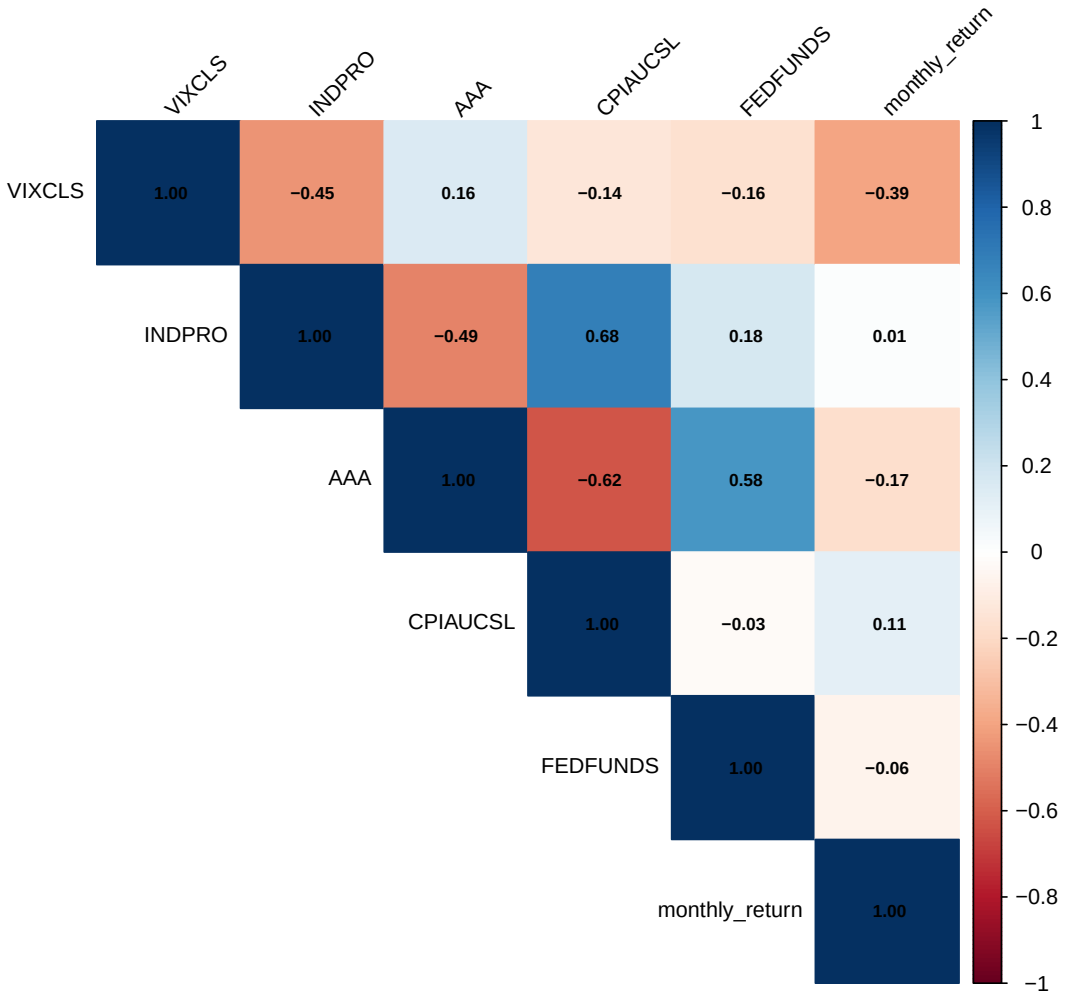
```r
# Calculate correlation matrix
cor_matrix <- analysis_data %>%
  select(all_of(numeric_vars)) %>%
  cor(use = "pairwise.complete.obs")

print("Correlation Matrix:")
```

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

```r
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
```

```r
# 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
```

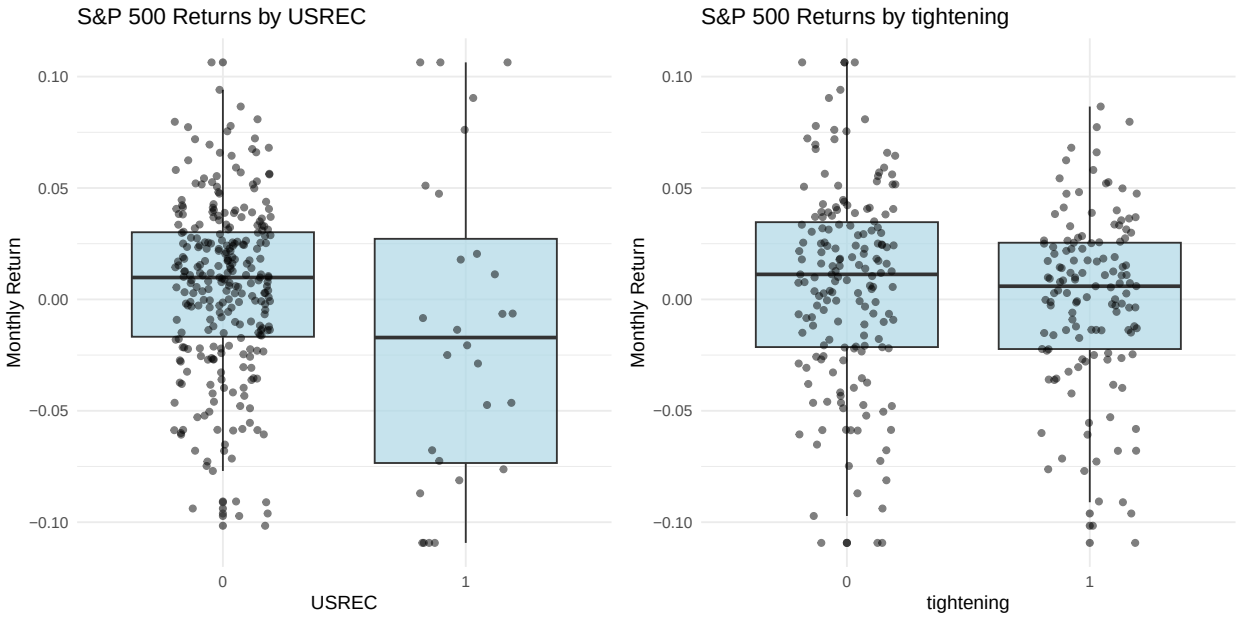
3.8 Factor Variable Analysis

```
# 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

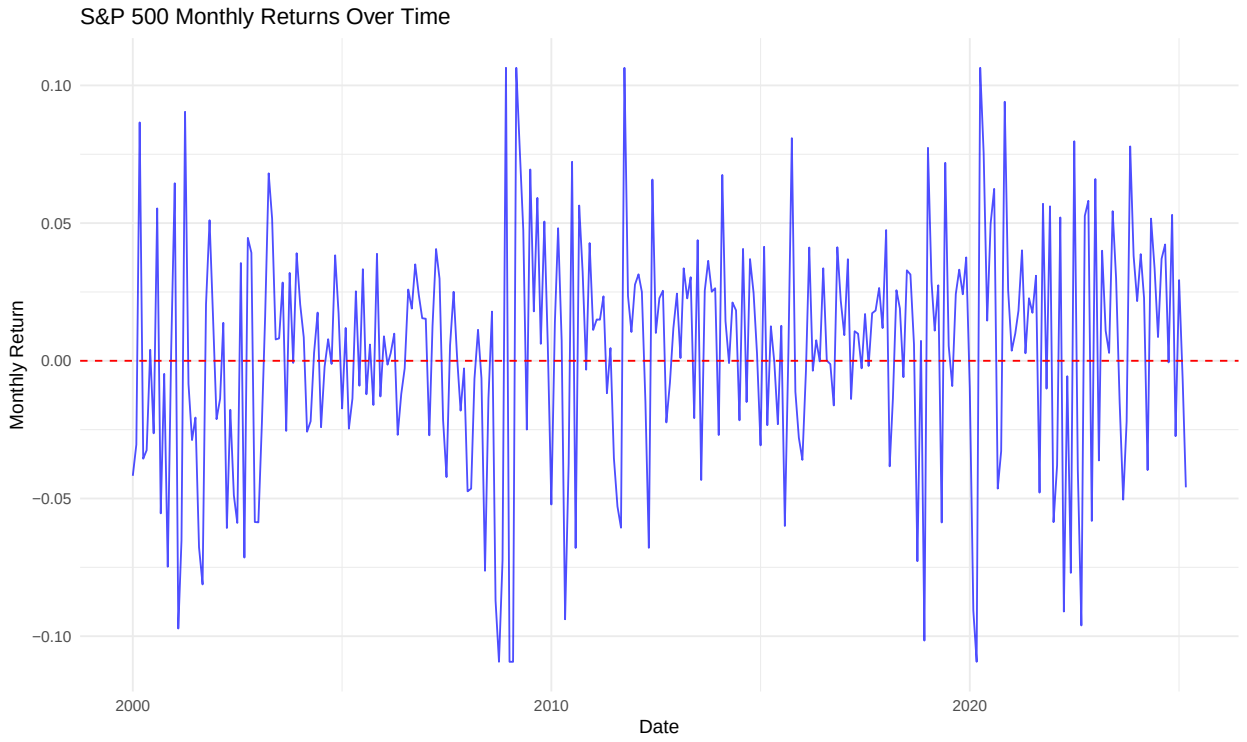
variable	value	count	percentage
USREC	0	275	90.76
USREC	1	28	9.24
tightening	0	170	56.11
tightening	1	133	43.89

```
# 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)
```

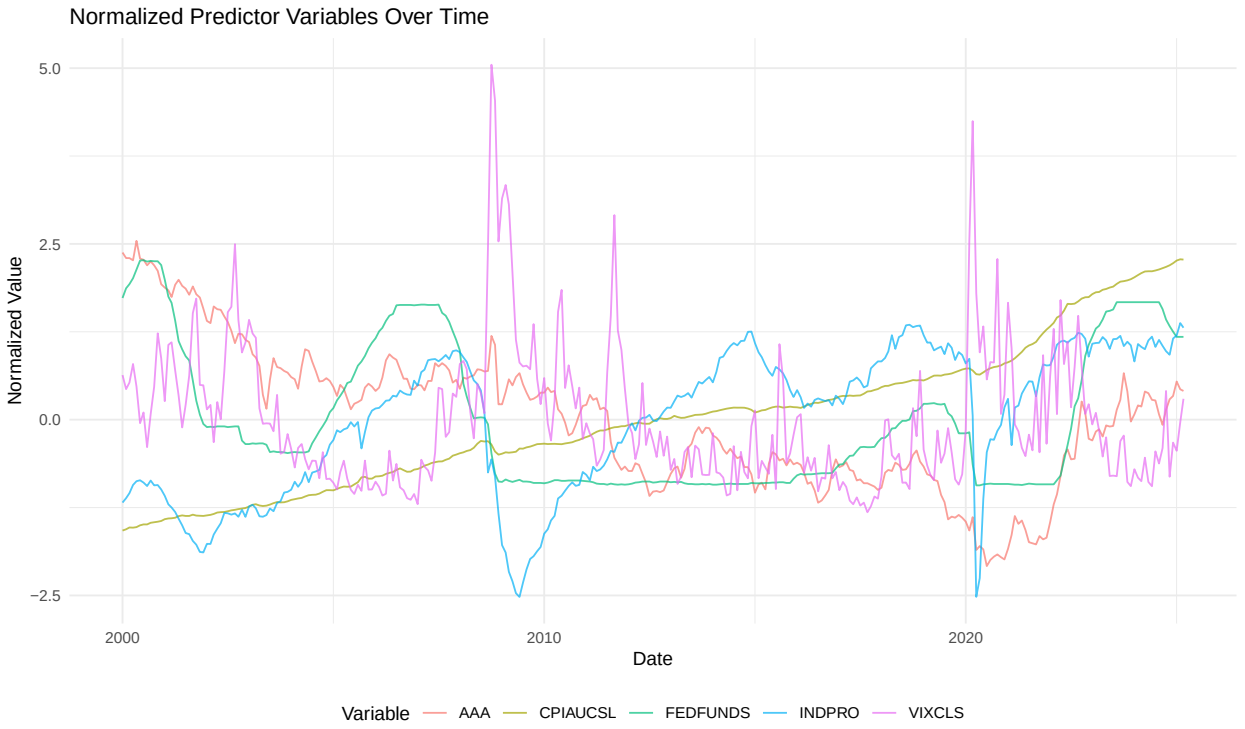


3.9 Time Series Visualization

```
# 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)
```



3.10 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.

4 Model Building (To be completed)

[Model building section will follow with competing regression models, diagnostics, and selection procedures as outlined in the project proposal]

5 Conclusion (To be completed)

[Final interpretation and economic insights]

6 References

Federal Reserve Economic Data (FRED). Federal Reserve Bank of St. Louis. https://fred.stlouisfed.org/

Yahoo Finance. S&P 500 Historical Data. https://finance.yahoo.com/

National Bureau of Economic Research (NBER). US Business Cycle Expansions and Contractions. https://www.nber.org/research/data/us-business-cycle-expansions-and-contractions