Asset Management Sector Analysis

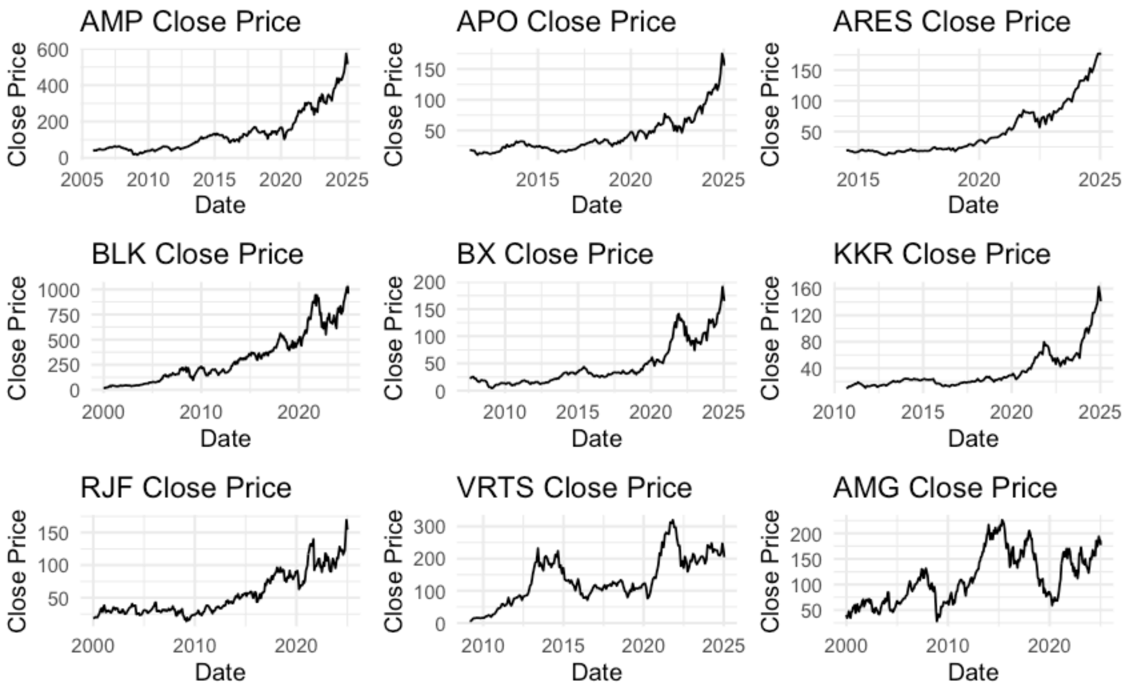
Code

Hide

```
library(ggplot2)
library(gridExtra)
library(forecast)
library(zoo)
library(tseries)
library(reshape2)
library(corrplot)
library(dplyr)
library(purrr)
library(PortfolioAnalytics)
library(PerformanceAnalytics)
library(ROI)
library(ROI.plugin.glpk)
library(ROI.plugin.quadprog)
library(xts)

AMG_Close <- AMG_monthly_data$Close
AMP_Close <- AMP_monthly_data$Close
APO_Close <- APO_monthly_data$Close
ARES_Close <- ARES_monthly_data$Close
BLK_Close <- BLK_monthly_data$Close
BX_Close <- BX_monthly_data$Close
KKR_Close <- KKR_monthly_data$Close
RJF_Close <- RJF_monthly_data$Close
VRTS_Close <- VRTS_monthly_data$Close
```

Hide

```
library(ggplot2)
library(gridExtra)

# Function to create a plot for each stock using Date and Close price
create_stock_plot <- function(data, title) {
  ggplot(data, aes(x = as.Date(Date), y = as.numeric(Close))) +
    geom_line() +
    ggtitle(title) +
    xlab("Date") + ylab("Close Price") +
    theme_minimal()
}

# Creating plots for each stock with correct Date mapping
p1 <- create_stock_plot(AMP_monthly_data, "AMP Close Price")
p2 <- create_stock_plot(APO_monthly_data, "APO Close Price")
p3 <- create_stock_plot(ARES_monthly_data, "ARES Close Price")
p4 <- create_stock_plot(BLK_monthly_data, "BLK Close Price")
p5 <- create_stock_plot(BX_monthly_data, "BX Close Price")
p6 <- create_stock_plot(KKR_monthly_data, "KKR Close Price")
p7 <- create_stock_plot(RJF_monthly_data, "RJF Close Price")
p8 <- create_stock_plot(VRTS_monthly_data, "VRTS Close Price")
p9 <- create_stock_plot(AMG_monthly_data, "AMG Close Price")

# Arrange the plots in a 3x3 grid
grid.arrange(p1, p2, p3,
             p4, p5, p6,
             p7, p8, p9,
             ncol = 3)
```



Moving Averages

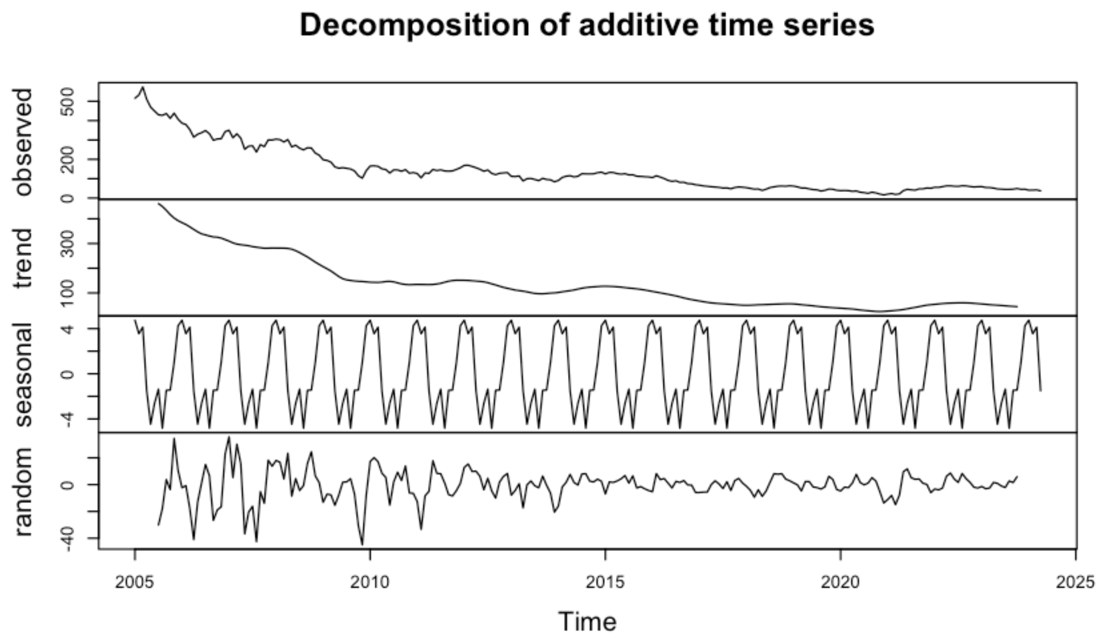
Hide

```
# Calculate 12-month moving average
AMP_monthly_data$MA_12 <- rollmean(AMP_monthly_data$Close, 12, fill = NA, align = "right")

# Plot without NA warning
ggplot(na.omit(AMP_monthly_data), aes(x = as.Date(Date))) +
  geom_line(aes(y = Close), color = "blue") +
  geom_line(aes(y = MA_12), color = "red", linetype = "dashed") +
  ggtitle("AMP Close Price with 12-Month Moving Average") +
  xlab("Date") + ylab("Price") +
  theme_minimal()
```

Time Series Decomposition

Hide

```
# Convert to time series object
AMP_ts <- ts(AMP_monthly_data$Close, start = c(2005, 1), frequency = 12)

# Decompose the time series
decomp <- decompose(AMP_ts)
plot(decomp)
```



Stationary Test (ADF test)

Hide

```
# Perform the Augmented Dickey-Fuller test
adf.test(AMP_ts, alternative = "stationary")
```

```

    Augmented Dickey-Fuller Test

data:  AMP_ts
Dickey-Fuller = -3.206, Lag order = 6, p-value = 0.08775
alternative hypothesis: stationary
```

ARIMA Model for Forecasting

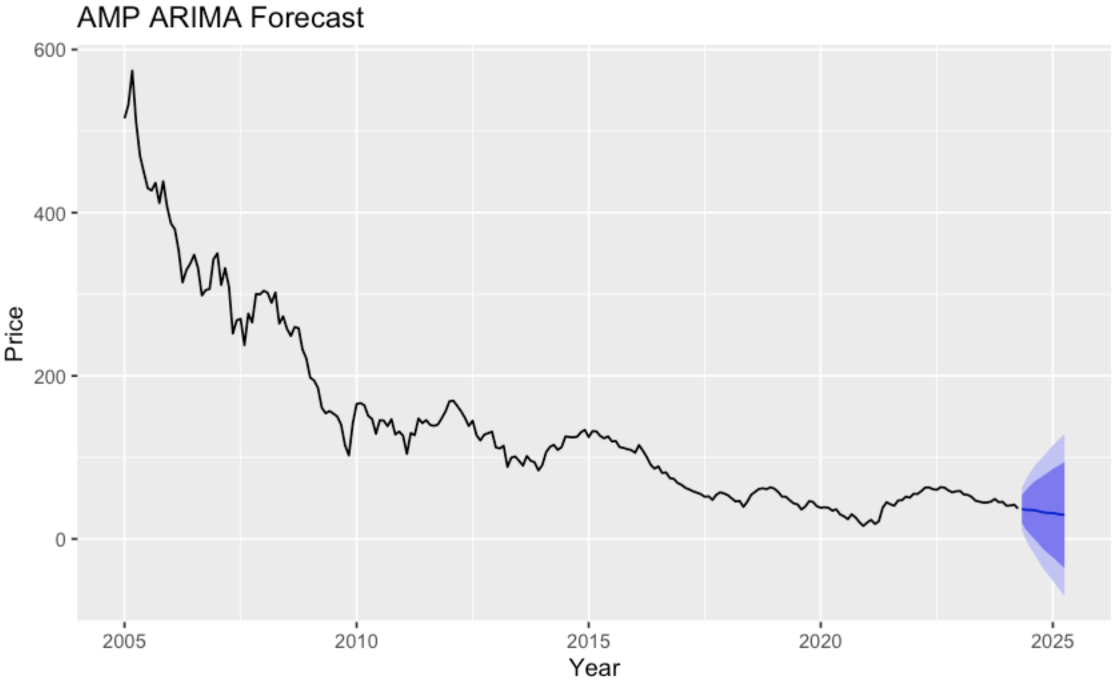
Hide

```
# Fit ARIMA model
fit_arima <- auto.arima(AMP_ts)
summary(fit_arima)
```

```
Series: AMP_ts
ARIMA(1,2,1)(1,0,1)[12]

Coefficients:
          ar1      ma1     sar1    sma1
      -0.0554  -0.9732  -0.7597  0.6480
s.e.   0.0694   0.0160   0.1961  0.2211

sigma^2 = 177.8:  log likelihood = -921.97
AIC=1853.95   AICc=1854.22   BIC=1871.14

Training set error measures:
                   ME    RMSE      MAE       MPE     MAPE     MASE        ACF1
Training set 1.087567 13.1598 8.273568 0.8329682 6.677914 0.244433 0.009017769
```

Hide

```
# Forecast for the next 12 months
forecast_arima <- forecast(fit_arima, h = 12)

# Plot the forecast
autoplot(forecast_arima) +
  ggtitle("AMP ARIMA Forecast") +
  xlab("Year") + ylab("Price")
```



1. The forecast shows a **downward trend** with **increasing uncertainty** (shaded region).
2. The **confidence interval widens**, suggesting the model is less certain about future prices.
3. There might be **over-differencing** due to the second-order differencing (`I(2)`).

Hide

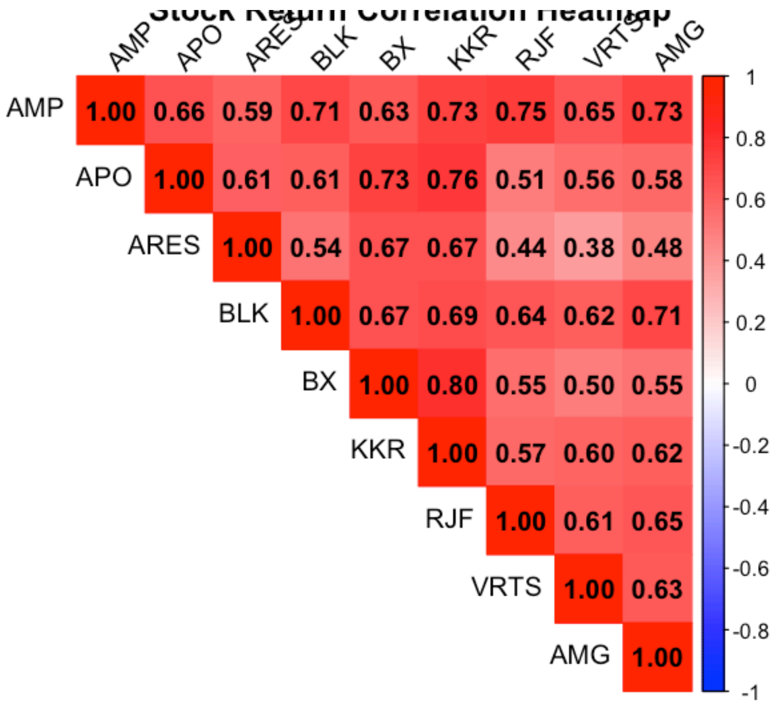
```
fit_sarima <- auto.arima(AMP_ts, seasonal = TRUE)
summary(fit_sarima)
```

```
Series: AMP_ts
ARIMA(1,2,1)(1,0,1)[12]

Coefficients:
          ar1      ma1     sar1    sma1
      -0.0554  -0.9732  -0.7597  0.6480
s.e.   0.0694   0.0160   0.1961  0.2211

sigma^2 = 177.8:  log likelihood = -921.97
AIC=1853.95   AICc=1854.22   BIC=1871.14

Training set error measures:
                   ME    RMSE      MAE       MPE     MAPE     MASE        ACF1
Training set 1.087567 13.1598 8.273568 0.8329682 6.677914 0.244433 0.009017769
```

Correlation Analysis

Hide

```
# 1. Calculate log returns with Date alignment
calculate_returns <- function(data) {
  data %>%
    mutate(Returns = c(NA, diff(log(Close)))) %>%
    select(Date, Returns)
}

# Apply the function to each stock
AMP_returns  <- calculate_returns(AMP_monthly_data)
APO_returns  <- calculate_returns(APO_monthly_data)
ARES_returns <- calculate_returns(ARES_monthly_data)
BLK_returns  <- calculate_returns(BLK_monthly_data)
BX_returns   <- calculate_returns(BX_monthly_data)
KKR_returns  <- calculate_returns(KKR_monthly_data)
RJF_returns  <- calculate_returns(RJF_monthly_data)
VRTS_returns <- calculate_returns(VRTS_monthly_data)
AMG_returns  <- calculate_returns(AMG_monthly_data)

# 2. Merge returns by Date (inner join keeps common dates)
returns_df <- reduce(list(AMP_returns, APO_returns, ARES_returns, BLK_returns,
                          BX_returns, KKR_returns, RJF_returns, VRTS_returns, AMG_returns),
                     full_join, by = "Date")

# 3. Rename columns for clarity
colnames(returns_df) <- c("Date", "AMP", "APO", "ARES", "BLK", "BX", "KKR", "RJF", "VRTS", "AMG")

# 4. Remove missing values (if any)
returns_df <- na.omit(returns_df)

# 5. Compute the correlation matrix
cor_matrix <- cor(returns_df[,-1], use = "complete.obs")

# 6. Plot the correlation heatmap
corrplot(cor_matrix, method = "color", type = "upper",
         col = colorRampPalette(c("blue", "white", "red"))(200),
         addCoef.col = "black", tl.col = "black", tl.srt = 45,
         title = "Stock Return Correlation Heatmap")
```



Portfolio Optimization

Hide

```
# 1. Calculate Mean Returns and Covariance Matrix
mean_returns <- colMeans(returns_df[,-1], na.rm = TRUE)
cov_matrix <- cov(returns_df[,-1], use = "complete.obs")

# 2. Initialize Variables
num_assets <- ncol(returns_df[,-1])
num_portfolios <- 10000
risk_free_rate <- 0.01 / 12  # Monthly risk-free rate

# 3. Prepare Storage for Results
results <- matrix(nrow = num_portfolios, ncol = num_assets + 3)
colnames(results) <- c(paste0("w_", colnames(returns_df[,-1])), "Return", "Risk", "Sharpe")

# 4. Monte Carlo Simulation for Random Portfolios
set.seed(123)
for (i in 1:num_portfolios) {
  # Random Weights
  weights <- runif(num_assets)
  weights <- weights / sum(weights)  # Normalize to sum to 1

  # Portfolio Return
  port_return <- sum(weights * mean_returns)

  # Portfolio Risk (Standard Deviation)
  port_risk <- sqrt(t(weights) %*% cov_matrix %*% weights)

  # Sharpe Ratio
  sharpe_ratio <- (port_return - risk_free_rate) / port_risk

  # Store Results
  results[i, ] <- c(weights, port_return, port_risk, sharpe_ratio)
}

# 5. Find the Portfolio with the Maximum Sharpe Ratio
max_sharpe_index <- which.max(results[, "Sharpe"])
optimal_portfolio <- results[max_sharpe_index, ]

# 6. Print Optimal Weights and Performance Metrics
optimal_weights <- optimal_portfolio[1:num_assets]
names(optimal_weights) <- colnames(returns_df[,-1])

cat("Optimal Weights:\n")
```

```
Optimal Weights:
```

Hide

```
print(round(optimal_weights, 4))
```

```
   AMP    APO   ARES    BLK     BX    KKR    RJF   VRTS    AMG
0.0028 0.0803 0.0088 0.1758 0.0005 0.0127 0.0765 0.2687 0.3738
```

Hide

```
cat("\nExpected Portfolio Return:", round(optimal_portfolio["Return"], 4), "\n")
```

```

Expected Portfolio Return: -0.0031
```

Hide

```
cat("Expected Portfolio Risk (Std Dev):", round(optimal_portfolio["Risk"], 4), "\n")
```

```
Expected Portfolio Risk (Std Dev): 0.0786
```

Hide

```
cat("Sharpe Ratio:", round(optimal_portfolio["Sharpe"], 4), "\n")
```

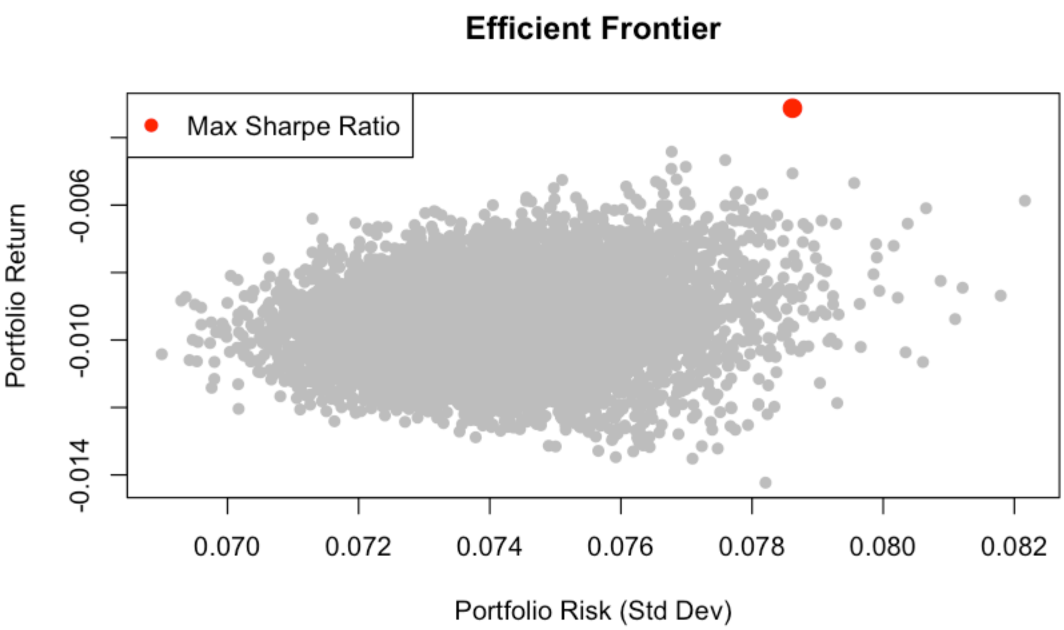
```
Sharpe Ratio: -0.0504
```

Hide

```
# 7. Plot Risk vs Return
plot(results[, "Risk"], results[, "Return"], col = "grey", pch = 16,
     xlab = "Portfolio Risk (Std Dev)", ylab = "Portfolio Return", main = "Efficient Frontier")
points(optimal_portfolio["Risk"], optimal_portfolio["Return"], col = "red", pch = 19, cex = 1.5)
```

Hide

```
legend("topleft", legend = "Max Sharpe Ratio", col = "red", pch = 19)
```



- The **Efficient Frontier** plot shows the distribution of 10,000 randomly generated portfolios.
- The **red dot** indicates the portfolio with the **maximum Sharpe Ratio**.
- The negative **Sharpe Ratio** suggests **poor risk-adjusted returns**, meaning the expected return is **negative** relative to its risk.

Allow Short Selling for Portfolio

Hide

```
# 1. Mean Returns and Covariance Matrix
mean_returns <- colMeans(returns_df[,-1], na.rm = TRUE)
cov_matrix <- cov(returns_df[,-1], use = "complete.obs")

# 2. Initialize Variables
num_assets <- ncol(returns_df[,-1])
num_portfolios <- 10000
risk_free_rate <- 0.01 / 12  # Monthly risk-free rate

# 3. Storage for Results
results <- matrix(nrow = num_portfolios, ncol = num_assets + 3)
colnames(results) <- c(paste0("w_", colnames(returns_df[,-1])), "Return", "Risk", "Sharpe")

# 4. Monte Carlo Simulation with Short Selling
set.seed(123)
for (i in 1:num_portfolios) {
  # Random Weights with Short Selling (allow negative weights)
  weights <- runif(num_assets, min = -1, max = 1)
  weights <- weights / sum(abs(weights))  # Normalize to sum to 1

  # Portfolio Return
  port_return <- sum(weights * mean_returns)

  # Portfolio Risk (Std Dev)
  port_risk <- sqrt(t(weights) %*% cov_matrix %*% weights)

  # Sharpe Ratio
  sharpe_ratio <- (port_return - risk_free_rate) / port_risk

  # Store Results
  results[i, ] <- c(weights, port_return, port_risk, sharpe_ratio)
}

# 5. Find the Portfolio with the Maximum Sharpe Ratio
max_sharpe_index <- which.max(results[, "Sharpe"])
optimal_portfolio <- results[max_sharpe_index, ]

# 6. Print Optimal Weights and Performance
optimal_weights <- optimal_portfolio[1:num_assets]
names(optimal_weights) <- colnames(returns_df[,-1])

cat("Optimal Weights with Short Selling:\n")
```

```
Optimal Weights with Short Selling:
```

Hide

```
print(round(optimal_weights, 4))
```

```
    AMP     APO    ARES     BLK      BX     KKR     RJF    VRTS     AMG 
-0.2213 -0.1508 -0.1842 -0.0511  0.0075  0.0589 -0.0400  0.0789  0.2073 
```

Hide

```
cat("\nExpected Portfolio Return:", round(optimal_portfolio["Return"], 4), "\n")
```

```

Expected Portfolio Return: 0.0079
```

Hide

```
cat("Expected Portfolio Risk (Std Dev):", round(optimal_portfolio["Risk"], 4), "\n")
```

```
Expected Portfolio Risk (Std Dev): 0.0293
```

Hide

```
cat("Sharpe Ratio:", round(optimal_portfolio["Sharpe"], 4), "\n")
```

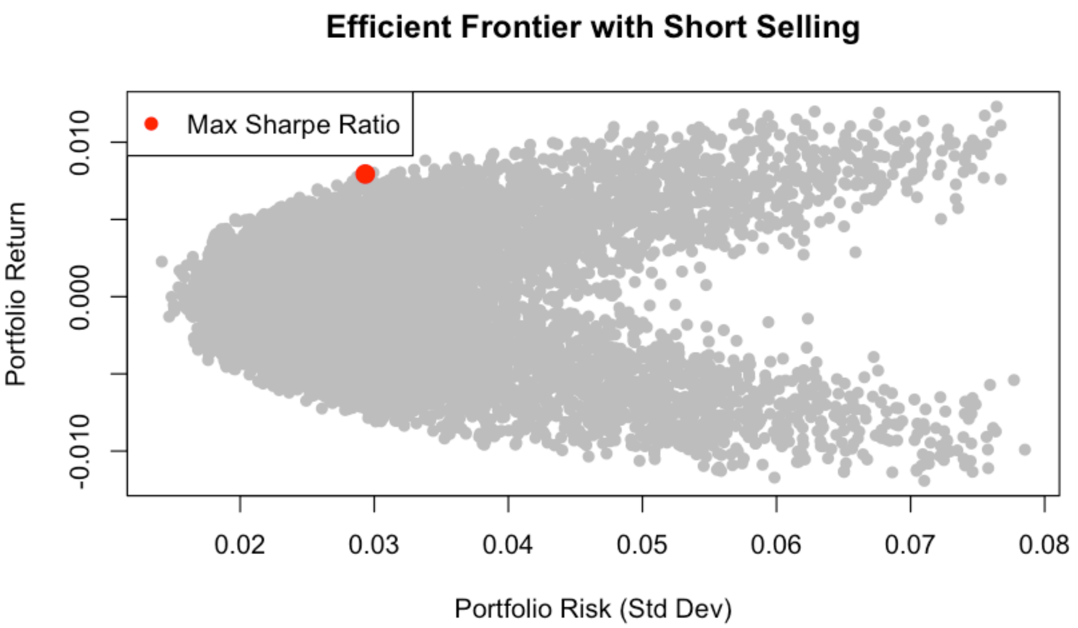
```
Sharpe Ratio: 0.2423
```

Hide

```
# 7. Plot Efficient Frontier
plot(results[, "Risk"], results[, "Return"], col = "grey", pch = 16,
     xlab = "Portfolio Risk (Std Dev)", ylab = "Portfolio Return", main = "Efficient Frontier with Short Sellin
g")
points(optimal_portfolio["Risk"], optimal_portfolio["Return"], col = "red", pch = 19, cex = 1.5)
```

Hide

```
legend("topleft", legend = "Max Sharpe Ratio", col = "red", pch = 19)
```



- **Expected Portfolio Return: 0.0079**
- **Expected Portfolio Risk (Std Dev): 0.0293**
- **Sharpe Ratio: 0.2423**
- The **Sharpe Ratio** improved from negative to **0.2423**, indicating **better risk-adjusted returns**
- **Short selling** was applied to **AMP, APO, ARES, RJF,** and **BLK**, helping hedge against downside risks.
- **Long positions** in **BX, KKR, VRTS,** and **AMG** focus on assets with stronger growth potential.
- The **red dot** represents the portfolio with the **maximum Sharpe Ratio**.

- The portfolio is positioned at **low risk** and **moderate return**.

Final Recommendation

1. Implement the Optimized Portfolio: Adopt the suggested asset weights, including **short positions**, to capitalize on market trends and hedge against downside risks.
2. **Diversify Further:**
   Consider adding assets from **different sectors** to reduce exposure to sector-specific risks.
3. **Periodic Rebalancing:**
   Regularly **rebalance** the portfolio to maintain the optimal risk-return balance, especially during market volatility.
4. **Monitor Macroeconomic Trends:**
   Stay alert to economic factors that could impact asset performance and adjust the portfolio accordingly.