Final Project

Adarsh (AP2459) || Atharwa (AP2467) || Rethyam (RG795) || Riley (RGB234)

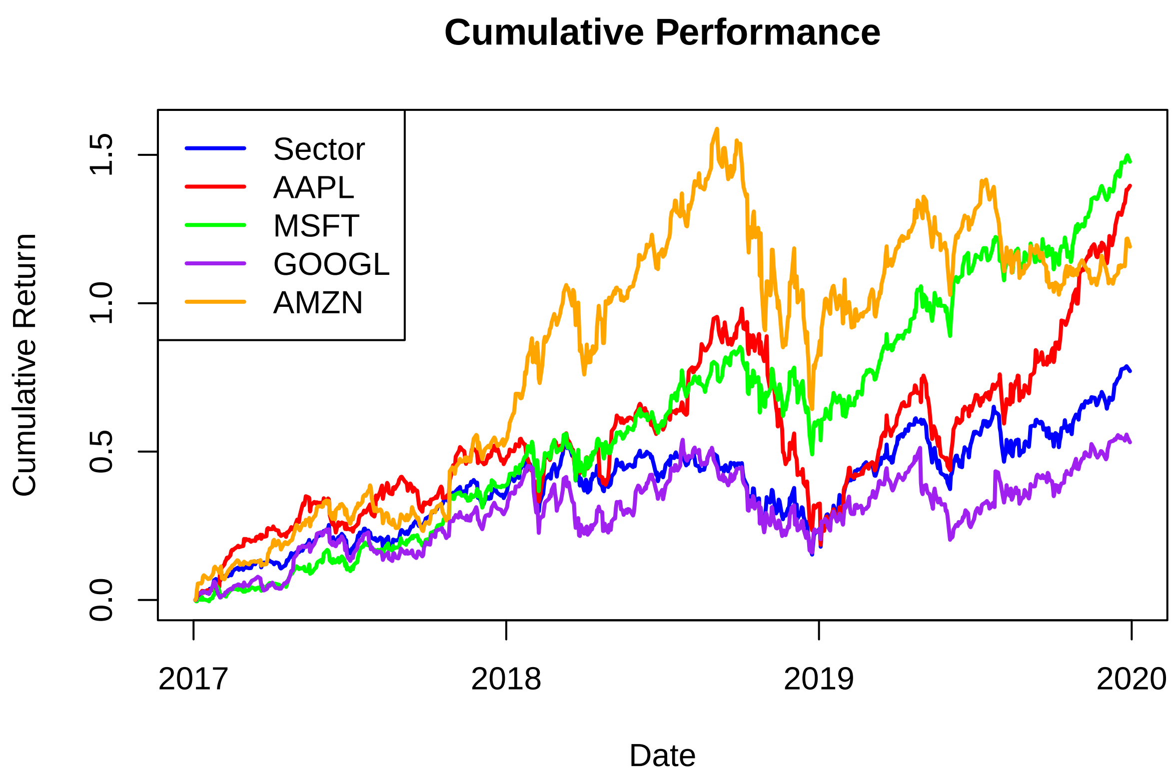
```r
# Load required libraries
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

## Loading required package: TTR

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

Project Question 1

```r
# Define the ticker for the Nasdaq Technology Sector and top companies
sector_ticker <- "^NDXT"  # Nasdaq Technology Sector Index
company_tickers <- c("AAPL", "MSFT", "GOOGL", "AMZN")
# Set the date range for analysis
start_date <- as.Date("2017-01-01")
end_date <- as.Date("2019-12-31")
# Function to get adjusted closing prices
get_adj_close <- function(ticker) {
  data <- getSymbols(ticker, src = "yahoo", from = start_date,
                     to = end_date, auto.assign = FALSE)
  Ad(data)
}
# Fetch data for the sector and companies
sector_prices <- get_adj_close(sector_ticker)
company_prices <- lapply(company_tickers, get_adj_close)
# Calculate daily log returns
sector_returns <- dailyReturn(sector_prices, type = "log")
company_returns <- lapply(company_prices, dailyReturn, type = "log")
```

```r
# Calculate cumulative returns with proper indexing
sector_cum_returns <- cumprod(1 + sector_returns) - 1
sector_cum_returns <- xts(sector_cum_returns, order.by = index(sector_returns))
company_cum_returns <- lapply(company_returns, function(x) {
  cum_returns <- cumprod(1 + x) - 1
  xts(cum_returns, order.by = index(x))
})
# Find the range of cumulative returns for setting plot boundaries
all_cum_returns <- c(sector_cum_returns, do.call(c, company_cum_returns))
ylim_range <- range(all_cum_returns, na.rm = TRUE)
# Plot cumulative returns with adjusted boundaries
plot(index(sector_cum_returns), sector_cum_returns, type = "l",
     col = "blue", lwd = 2,
     xlab = "Date", ylab = "Cumulative Return",
     main = "Cumulative Performance",
     ylim = ylim_range)  # Set y-axis limits
lines(index(company_cum_returns[[1]]), company_cum_returns[[1]],
      col = "red", lwd = 2)
lines(index(company_cum_returns[[2]]), company_cum_returns[[2]],
      col = "green", lwd = 2)
lines(index(company_cum_returns[[3]]), company_cum_returns[[3]],
      col = "purple", lwd = 2)
lines(index(company_cum_returns[[4]]), company_cum_returns[[4]],
      col = "orange", lwd = 2)
# Add a legend
legend("topleft", legend = c("Sector", company_tickers),
       col = c("blue", "red", "green", "purple", "orange"), lwd = 2)
```




```r
# Define the ticker for the Nasdaq Technology Sector and top companies
sector_ticker <- "^NDXT"  # Nasdaq Technology Sector Index
company_tickers <- c("AAPL", "MSFT", "GOOGL", "AMZN")
# Set the date range for analysis
start_date <- as.Date("2017-01-01")
end_date <- as.Date("2019-12-31")
# Function to get adjusted closing prices
get_adj_close <- function(ticker) {
  data <- getSymbols(ticker, src = "yahoo", from = start_date,
                     to = end_date, auto.assign = FALSE)
  Ad(data)
}
# Fetch data for the sector and companies
sector_prices <- get_adj_close(sector_ticker)
company_prices <- lapply(company_tickers, get_adj_close)
# Calculate daily log returns
sector_returns <- dailyReturn(sector_prices, type = "log")
company_returns <- lapply(company_prices, dailyReturn, type = "log")
# Combine sector and company returns into one data frame
all_returns <- data.frame(
  Date = index(sector_returns),
  Sector = coredata(sector_returns),
  do.call(cbind, lapply(company_returns, coredata))
)
colnames(all_returns) <- c("Date", "Sector", company_tickers)
# Print the first five and last five daily returns
```

```r
cat("First 5 Daily Returns:\n")
```

```
## First 5 Daily Returns:
```

```r
print(head(all_returns, 10))
```

```
##          Date       Sector         AAPL         MSFT         GOOGL         AMZN
## 1  2017-01-03  0.000000000  0.000000000  0.000000000  0.0000000000  0.000000000
## 2  2017-01-04  0.003385940 -0.001119632 -0.004484326 -0.0002970741  0.004646473
## 3  2017-01-05  0.002048053  0.005072550  0.000000000  0.0064783611  0.030269616
## 4  2017-01-06  0.007546725  0.011086624  0.008630247  0.0148822721  0.019715958
## 5  2017-01-09  0.006147943  0.009117607 -0.003187839  0.0023843885  0.001167704
## 6  2017-01-10  0.003953822  0.001007947 -0.000319236 -0.0014154876 -0.001280810
## 7  2017-01-11  0.004917092  0.005358879  0.009061390  0.0046501249  0.003912478
## 8  2017-01-12 -0.004292239 -0.004184200 -0.009220912 -0.0003976727  0.018132006
## 9  2017-01-13  0.005860983 -0.001762536  0.001436526  0.0016982684  0.004292412
## 10 2017-01-17 -0.006972432  0.008032324 -0.002715367 -0.0041967863 -0.009121890
```

```r
cat("\nLast 5 Daily Returns:\n")
```

```
##
## Last 5 Daily Returns:
```

```r
print(tail(all_returns, 10))
```

```
##           Date        Sector          AAPL          MSFT         GOOGL
## 744 2019-12-16  0.0111668799  0.0169728186  0.0064503626  0.0102159579
## 745 2019-12-17  0.0009559914  0.0019636322 -0.0054157561 -0.0042789617
## 746 2019-12-18  0.0021843551 -0.0023920419 -0.0020706978 -0.0022020338
## 747 2019-12-19  0.0078349586  0.0010005366  0.0086431011  0.0033452726
## 748 2019-12-20  0.0069721929 -0.0020737165  0.0108584422 -0.0038558130
## 749 2019-12-23  0.0014749507  0.0161866287  0.0000000000 -0.0004365031
## 750 2019-12-24  0.0005941190  0.0009503235 -0.0001905499 -0.0046012510
## 751 2019-12-26  0.0037284713  0.0196458728  0.0081632447  0.0133292209
## 752 2019-12-27 -0.0022347370 -0.0003795566  0.0018261825 -0.0057633867
## 753 2019-12-30 -0.0070355110  0.0059178054 -0.0086560364 -0.0110827265
##              AMZN
## 744  0.0046854248
## 745  0.0120510860
## 746 -0.0037093855
## 747  0.0046136852
## 748 -0.0032301703
## 749  0.0036318473
## 750 -0.0021160007
## 751  0.0435062436
## 752  0.0005509959
## 753 -0.0123283321
```

Project Question 2

```r
# Define a function to calculate summary statistics by year
calculate_yearly_stats <- function(returns, dates) {
  yearly_stats <- aggregate(returns,
                            by = list(Year = format(dates, "%Y")),
                            FUN = function(x) c(Mean = mean(x, na.rm = TRUE),
                                                SD = sd(x, na.rm = TRUE)))
  # Separate Mean and SD into separate columns
  yearly_stats <- do.call(data.frame, yearly_stats)
  colnames(yearly_stats)[-1] <- c("Mean", "SD")  # Rename columns
  return(yearly_stats)
}
# Calculate statistics for the sector
sector_yearly_stats <- calculate_yearly_stats(as.numeric(sector_returns),
                                              index(sector_returns))
sector_yearly_stats$Ticker <- sector_ticker
# Calculate statistics for each company
company_yearly_stats <- lapply(company_returns, function(x) {
  calculate_yearly_stats(as.numeric(x), index(x))
})
for (i in 1:length(company_tickers)) {
  company_yearly_stats[[i]]$Ticker <- company_tickers[i]
}
# Combine sector and company statistics
all_yearly_stats <- do.call(rbind, c(list(sector_yearly_stats),
                                     company_yearly_stats))
# Generate separate tables for each year
years <- unique(all_yearly_stats$Year)
tables_by_year <- lapply(years, function(year) {
  subset_stats <- all_yearly_stats[all_yearly_stats$Year == year,
                                   c("Ticker", "Mean", "SD")]
  subset_stats$Mean <-
    sprintf("%.3f (%.3f%%)",
            subset_stats$Mean, as.numeric(subset_stats$Mean) * 100)
  subset_stats$SD <- sprintf("%.3f", subset_stats$SD)
  colnames(subset_stats) <- c("Ticker", paste0("Mean_", year),
                              paste0("SD_", year))
  return(subset_stats)
})
# Print tables for each year
for (i in seq_along(years)) {
  cat(sprintf("Summary Statistics for %s\n", years[i]))
  print(tables_by_year[[i]])
  cat("\n")
}
```

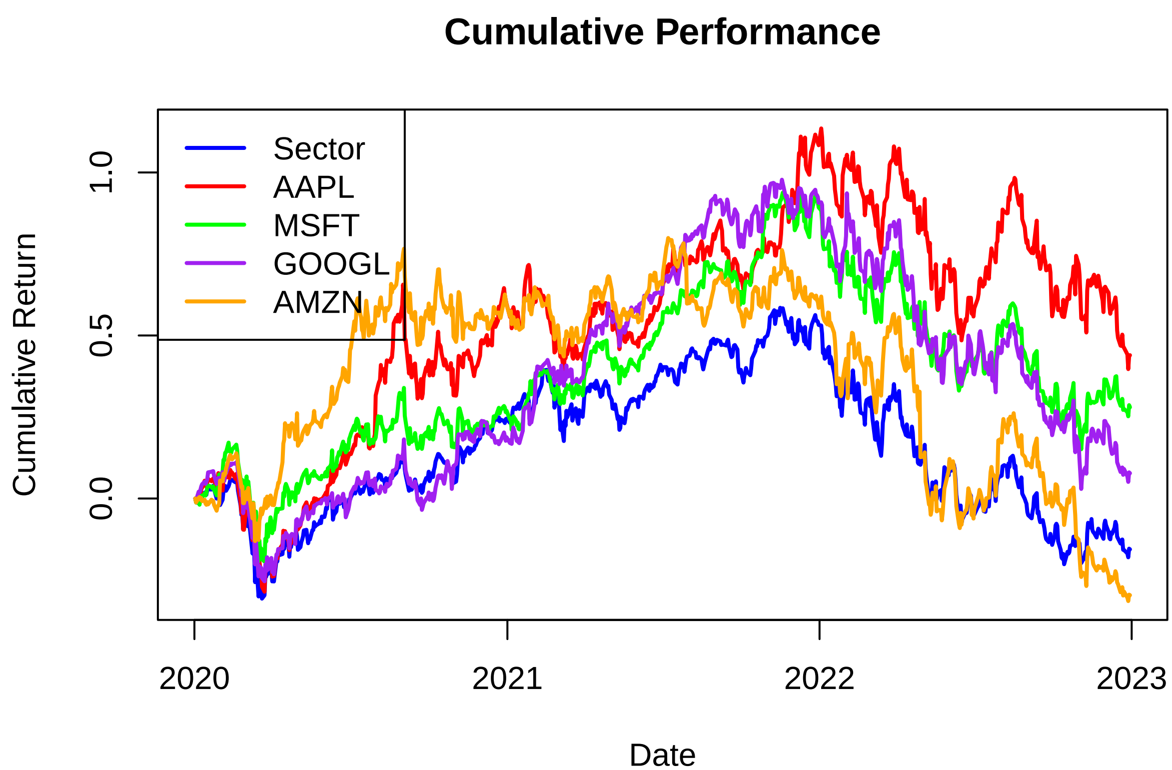
```
## Summary Statistics for 2017
##    Ticker      Mean_2017 SD_2017
## 1   ^NDXT 0.001 (0.121%)   0.008
## 4    AAPL 0.002 (0.156%)   0.011
## 7    MSFT 0.001 (0.133%)   0.009
## 10  GOOGL 0.001 (0.106%)   0.010
## 13   AMZN 0.002 (0.175%)   0.013
##
```

```
## Summary Statistics for 2018
##    Ticker        Mean_2018 SD_2018
## 2   ^NDXT -0.000 (-0.022%)   0.015
## 5    AAPL -0.000 (-0.022%)   0.018
## 8    MSFT   0.001 (0.075%)   0.018
## 11  GOOGL -0.000 (-0.003%)   0.018
## 14   AMZN   0.001 (0.100%)   0.023
##
## Summary Statistics for 2019
##    Ticker      Mean_2019 SD_2019
## 3   ^NDXT 0.002 (0.153%)   0.013
## 6    AAPL 0.003 (0.251%)   0.017
## 9    MSFT 0.002 (0.181%)   0.013
## 12  GOOGL 0.001 (0.099%)   0.015
## 15   AMZN 0.001 (0.082%)   0.014
```

Project Question 1 and 2 Extra for 2020-2022

```r
# Define the ticker for the Nasdaq Technology Sector and top companies
tech_index_ticker <- "^NDXT"  # Nasdaq Technology Sector Index
tech_company_tickers <- c("AAPL", "MSFT", "GOOGL", "AMZN")
# Set the date range for analysis
analysis_start_date <- as.Date("2020-01-01")
analysis_end_date <- as.Date("2022-12-31")
# Function to get adjusted closing prices
fetch_adj_close <- function(symbol) {
  price_data <- getSymbols(symbol, src = "yahoo", from = analysis_start_date,
                           to = analysis_end_date, auto.assign = FALSE)
  Ad(price_data)
}
# Fetch data for the sector and companies
tech_index_prices <- fetch_adj_close(tech_index_ticker)
tech_company_prices <- lapply(tech_company_tickers, fetch_adj_close)
# Calculate daily log returns
tech_index_returns <- dailyReturn(tech_index_prices, type = "log")
tech_company_returns <- lapply(tech_company_prices,
                               dailyReturn, type = "log")
# Calculate cumulative returns with proper indexing
tech_index_cum_returns <- cumprod(1 + tech_index_returns) - 1
tech_index_cum_returns <- xts(tech_index_cum_returns,
                              order.by = index(tech_index_returns))
tech_company_cum_returns <- lapply(tech_company_returns, function(y) {
  cumulative_returns <- cumprod(1 + y) - 1
  xts(cumulative_returns, order.by = index(y))
})
# Find the range of cumulative returns for setting plot boundaries
combined_cum_returns <- c(tech_index_cum_returns,
                          do.call(c, tech_company_cum_returns))
y_axis_limits <- range(combined_cum_returns, na.rm = TRUE)
# Plot cumulative returns with adjusted boundaries
plot(index(tech_index_cum_returns), tech_index_cum_returns, type = "l",
     col = "blue", lwd = 2,
```

```
     xlab = "Date", ylab = "Cumulative Return",
     main = "Cumulative Performance",
     ylim = y_axis_limits)  # Set y-axis limits
lines(index(tech_company_cum_returns[[1]]), tech_company_cum_returns[[1]],
      col = "red", lwd = 2)
lines(index(tech_company_cum_returns[[2]]), tech_company_cum_returns[[2]],
      col = "green", lwd = 2)
lines(index(tech_company_cum_returns[[3]]), tech_company_cum_returns[[3]],
      col = "purple", lwd = 2)
lines(index(tech_company_cum_returns[[4]]), tech_company_cum_returns[[4]],
      col = "orange", lwd = 2)
# Add a legend
legend("topleft", legend = c("Sector", tech_company_tickers),
       col = c("blue", "red", "green", "purple", "orange"), lwd = 2)
```




```
# Define a function to calculate summary statistics by year
compute_annual_stats <- function
(returns, dates) {
  annual_stats <- aggregate(returns,
                            by = list(Year = format(dates, "%Y")),
                            FUN = function(x) c(Mean = mean(x, na.rm = TRUE),
                                                SD = sd(x, na.rm = TRUE)))
  # Separate Mean and SD into separate columns
  annual_stats <- do.call(data.frame, annual_stats)
  colnames(annual_stats)[-1] <- c("Mean", "SD")  # Rename columns
```

```r
  return(annual_stats)
}
# Calculate statistics for the tech index
tech_index_annual_stats <- compute_annual_stats(as.numeric(tech_index_returns),
                                               index(tech_index_returns))
tech_index_annual_stats$Ticker <- tech_index_ticker
# Calculate statistics for each company
tech_company_annual_stats <- lapply(tech_company_returns, function(x) {
  compute_annual_stats(as.numeric(x), index(x))
})
for (i in 1:length(tech_company_tickers)) {
  tech_company_annual_stats[[i]]$Ticker <- tech_company_tickers[i]
}
# Combine tech index and company statistics
all_annual_stats <- do.call(rbind, c(list(tech_index_annual_stats),
                                     tech_company_annual_stats))
# Generate separate tables for each year
analysis_years <- unique(all_annual_stats$Year)
yearly_tables <- lapply(analysis_years, function(year) {
  filtered_stats <- all_annual_stats[all_annual_stats$Year == year,
                                     c("Ticker", "Mean", "SD")]
  filtered_stats$Mean <-
    sprintf("%.3f (%.3f%%)",
            filtered_stats$Mean, as.numeric(filtered_stats$Mean) * 100)
  filtered_stats$SD <- sprintf("%.3f", filtered_stats$SD)
  colnames(filtered_stats) <- c("Ticker", paste0("Mean_", year),
                                paste0("SD_", year))
  return(filtered_stats)
})
# Print tables for each year
for (j in seq_along(analysis_years)) {
  cat(sprintf("Summary Statistics for %s\n", analysis_years[j]))
  print(yearly_tables[[j]])
  cat("\n")
}
```

```
## Summary Statistics for 2020
##    Ticker      Mean_2020 SD_2020
## 1   ^NDXT 0.001 (0.119%)   0.025
## 4    AAPL 0.002 (0.228%)   0.029
## 7    MSFT 0.001 (0.133%)   0.028
## 10  GOOGL 0.001 (0.098%)   0.024
## 13   AMZN 0.002 (0.213%)   0.024
##
## Summary Statistics for 2021
##    Ticker      Mean_2021 SD_2021
## 2   ^NDXT 0.001 (0.095%)   0.016
## 5    AAPL 0.001 (0.118%)   0.016
## 8    MSFT 0.002 (0.167%)   0.013
## 11  GOOGL 0.002 (0.199%)   0.015
## 14   AMZN 0.000 (0.009%)   0.015
##
## Summary Statistics for 2022
```

```
##    Ticker        Mean_2022 SD_2022
## 3   ^NDXT -0.002 (-0.203%)   0.026
## 6    AAPL -0.001 (-0.122%)   0.022
## 9    MSFT -0.001 (-0.131%)   0.022
## 12  GOOGL -0.002 (-0.198%)   0.024
## 15   AMZN -0.003 (-0.273%)   0.032
```

Project Question 3

```r
# Define the annual risk-free rate
annual_risk_free_rate <- 0.02  # 2%
daily_risk_free_rate <- annual_risk_free_rate / 252  # Convert to daily rate
# Combine company returns into a matrix for portfolio optimization
company_returns_matrix <- do.call(merge, company_returns)
colnames(company_returns_matrix) <- company_tickers
# Define a function for portfolio optimization
portfolio_optimization <- function(returns_matrix) {
  # Calculate the covariance matrix
  cov_matrix <- cov(returns_matrix, use = "complete.obs")
  # Invert the covariance matrix
  inv_cov_matrix <- solve(cov_matrix)
  # Calculate weights for the minimum variance portfolio
  weights <-
    inv_cov_matrix %*% rep(1, ncol(returns_matrix)) / sum(inv_cov_matrix)
  return(as.numeric(weights))
}
# Calculate portfolio weights
portfolio_weights <- portfolio_optimization(company_returns_matrix)
names(portfolio_weights) <- company_tickers
# Calculate portfolio returns
portfolio_returns <-
  as.matrix(company_returns_matrix) %*% portfolio_weights
portfolio_returns <-
  xts(portfolio_returns, order.by = index(company_returns_matrix))
# Calculate daily summary statistics for the portfolio
portfolio_mean <- mean(portfolio_returns, na.rm = TRUE)  # Daily mean return
portfolio_mean_pct <- portfolio_mean * 100  # Convert to percentage
portfolio_sd <- sd(portfolio_returns, na.rm = TRUE)  # Daily standard deviation
# Calculate annualized Sharpe ratio
portfolio_annualized_sharpe <-
  ((portfolio_mean - daily_risk_free_rate) * 252) / (portfolio_sd * sqrt(252))
# Print Portfolio Weights
print("==== Portfolio Weights (Minimum Variance Portfolio) ====")
```

```
## [1] "==== Portfolio Weights (Minimum Variance Portfolio) ===="
```

```r
print(round(portfolio_weights, 3))
```

```
##   AAPL   MSFT  GOOGL   AMZN
##  0.260  0.502  0.308 -0.071
```

```r
# Print Portfolio and Sector Summary
sector_mean <- mean(sector_returns, na.rm = TRUE)
sector_mean_pct <- sector_mean * 100  # Convert to percentage
sector_sd <- sd(sector_returns, na.rm = TRUE)
sector_annualized_sharpe <-
  ((sector_mean - daily_risk_free_rate) * 252) / (sector_sd * sqrt(252))
print("\n==== Portfolio and Sector Return Summary ====")
```

```
## [1] "\n==== Portfolio and Sector Return Summary ===="
```

```r
print(paste("| Mean (%):", round(portfolio_mean_pct, 3), "%",
            "| Sigma (Daily):", round(portfolio_sd, 3),
            "| Annualized Sharpe (2% rf):",
            round(portfolio_annualized_sharpe, 3)))
```

```
## [1] "| Mean (%): 0.111 % | Sigma (Daily): 0.013 | Annualized Sharpe (2% rf): 1.29"
```

```r
print(paste("| Mean (%):", round(sector_mean_pct, 3), "%",
            "| Sigma (Daily):", round(sector_sd, 3),
            "| Annualized Sharpe (2% rf):",
            round(sector_annualized_sharpe, 3)))
```

```
## [1] "| Mean (%): 0.084 % | Sigma (Daily): 0.013 | Annualized Sharpe (2% rf): 0.956"
```

```r
# Ensure only numeric data is used for individual asset statistics
numeric_returns_matrix <-
  company_returns_matrix[, sapply(company_returns_matrix, is.numeric)]
# Print Individual Asset Summaries
print("\n==== Individual Asset Summaries ====")
```

```
## [1] "\n==== Individual Asset Summaries ===="
```

```r
for (i in 1:ncol(numeric_returns_matrix)) {
  ticker <- company_tickers[i]
  mean_return <-
    mean(numeric_returns_matrix[, i], na.rm = TRUE)  # Daily mean return
  mean_return_pct <- mean_return * 100  # Convert to percentage
  sd_return <-
    sd(numeric_returns_matrix[, i], na.rm = TRUE)  # Daily standard deviation
  annualized_sharpe <-
    ((mean_return - daily_risk_free_rate) * 252) / (sd_return * sqrt(252))

  print(paste("Asset:", ticker))
  print(paste("  Daily Mean Return (Decimal):", round(mean_return, 5)))
  print(paste("  Daily Mean Return (%):", round(mean_return_pct, 3), "%"))
  print(paste("  Daily Standard Deviation:", round(sd_return, 5)))
  print(paste("  Annualized Sharpe Ratio (2% rf):",
              round(annualized_sharpe, 3)))
  cat("\n")  # Add a blank line after each asset's statistics
}
```

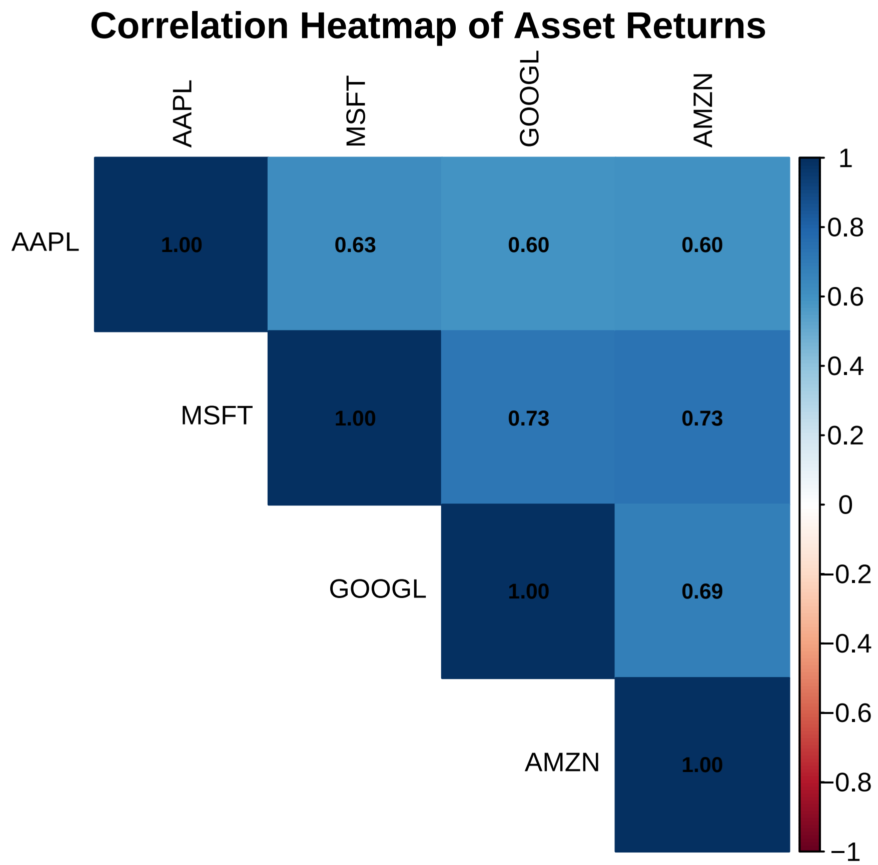
```
## [1] "Asset: AAPL"
## [1] "  Daily Mean Return (Decimal): 0.00128"
## [1] "  Daily Mean Return (%): 0.128 %"
## [1] "  Daily Standard Deviation: 0.01558"
## [1] "  Annualized Sharpe Ratio (2% rf): 1.227"
##
## [1] "Asset: MSFT"
## [1] "  Daily Mean Return (Decimal): 0.0013"
## [1] "  Daily Mean Return (%): 0.13 %"
## [1] "  Daily Standard Deviation: 0.01364"
## [1] "  Annualized Sharpe Ratio (2% rf): 1.419"
##
## [1] "Asset: GOOGL"
## [1] "  Daily Mean Return (Decimal): 0.00067"
## [1] "  Daily Mean Return (%): 0.067 %"
## [1] "  Daily Standard Deviation: 0.0145"
## [1] "  Annualized Sharpe Ratio (2% rf): 0.648"
##
## [1] "Asset: AMZN"
## [1] "  Daily Mean Return (Decimal): 0.00119"
## [1] "  Daily Mean Return (%): 0.119 %"
## [1] "  Daily Standard Deviation: 0.01725"
## [1] "  Annualized Sharpe Ratio (2% rf): 1.022"
```

```r
# Load the corrplot library
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.4.2
```

```
## corrplot 0.95 loaded
```

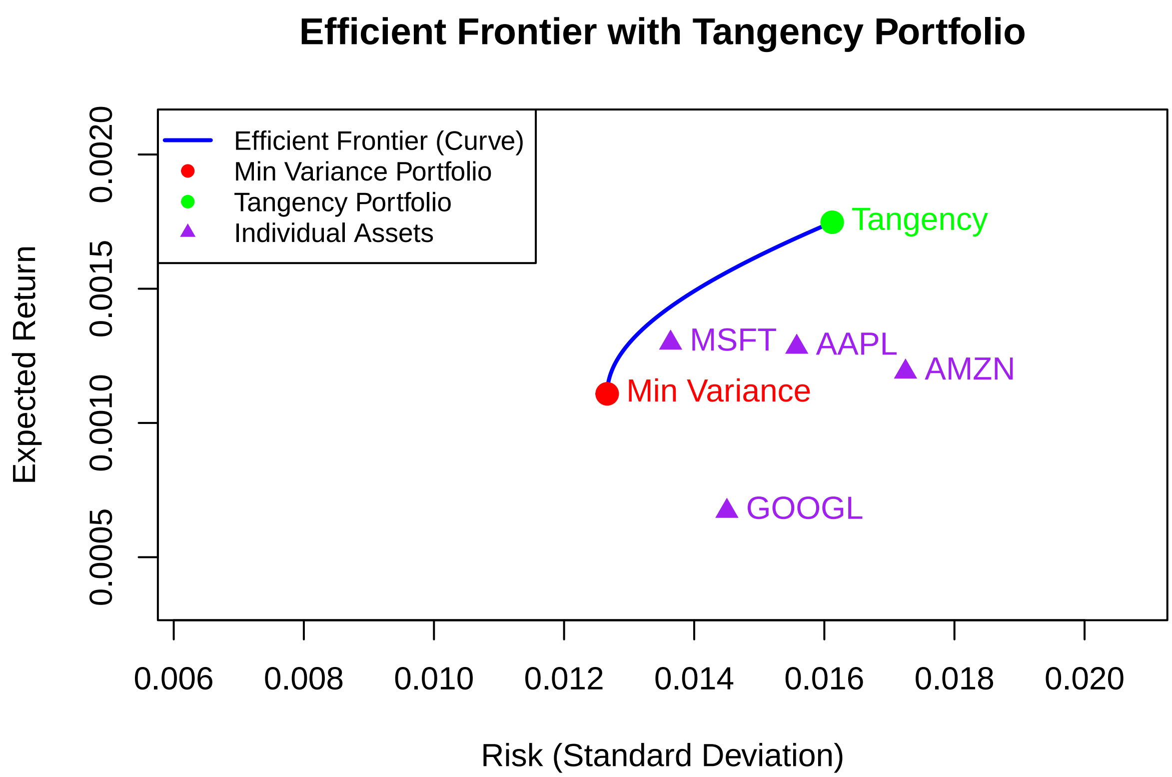
```r
# Calculate the correlation matrix
correlation_matrix <- cor(company_returns_matrix, use = "complete.obs")
# Plot the heatmap
corrplot(correlation_matrix, method = "color",
         type = "upper", tl.cex = 0.8, tl.col = "black",
         addCoef.col = "black", number.cex = 0.7,
         title = "Correlation Heatmap of Asset Returns", mar = c(0, 0, 1, 0))
# Add a color legend
color.legend <- legend("bottomright", legend = c("Low", "High"),
                        fill = c("blue", "red"),
                        title = "Correlation", cex = 0.8)
```




```r
# Function to calculate portfolio return and risk
portfolio_stats <- function(weights, returns_matrix) {
  # Calculate portfolio return
  expected_return <- sum(weights * colMeans(returns_matrix, na.rm = TRUE))
  # Calculate portfolio risk
  cov_matrix <- cov(returns_matrix, use = "complete.obs")
  portfolio_variance <- t(weights) %*% cov_matrix %*% weights
  portfolio_risk <- sqrt(portfolio_variance)
  return(c(ExpectedReturn = expected_return, Risk = portfolio_risk))
}
# Function to calculate the tangency portfolio weights
tangency_portfolio <- function(returns_matrix, risk_free_rate) {
  excess_returns <- colMeans(returns_matrix, na.rm = TRUE) - risk_free_rate
  cov_matrix <- cov(returns_matrix, use = "complete.obs")
  inv_cov_matrix <- solve(cov_matrix)
  weights <-
    inv_cov_matrix %*% excess_returns / sum(inv_cov_matrix %*% excess_returns)
  return(as.numeric(weights))
}
# Tangency Portfolio with Risk-Free Rate
risk_free_rate <- 0.02/252  # Convert annual risk-free rate to daily
tangency_weights <- tangency_portfolio(company_returns_matrix, risk_free_rate)
names(tangency_weights) <-
  colnames(company_returns_matrix)  # Assign asset names to weights
# Calculate tangency portfolio statistics
tangency_stats <- portfolio_stats(tangency_weights, company_returns_matrix)
```

```r
# Generate weights to extract the efficient frontier curve
set.seed(42)  # Ensure reproducibility
n_points <- 100  # Number of points on the efficient frontier
efficient_weights <-
  matrix(NA, nrow = n_points, ncol = ncol(company_returns_matrix))
# Generate weights linearly from minimum variance to maximum return
returns_mean <- colMeans(company_returns_matrix, na.rm = TRUE)
for (i in 1:n_points) {
  alpha <- (i - 1) / (n_points - 1)  # Weight factor (0 to 1)
  efficient_weights[i, ] <-
    (1 - alpha) * portfolio_weights + alpha * tangency_weights
}
# Normalize weights to ensure they sum to 1
efficient_weights <- efficient_weights / rowSums(efficient_weights)
# Calculate efficient frontier statistics
efficient_frontier <-
  t(apply(efficient_weights, 1, portfolio_stats,
          returns_matrix = company_returns_matrix))
# Individual assets (mean and standard deviation)
individual_assets <- data.frame(
  Risk = apply(company_returns_matrix, 2, sd, na.rm = TRUE),
  ExpectedReturn = colMeans(company_returns_matrix, na.rm = TRUE),
  Asset = colnames(company_returns_matrix)
)
# Adjust x and y limits to include tangency portfolio and individual assets
x_limits <-
  range(c(efficient_frontier[, "Risk"], tangency_stats["Risk"],
          individual_assets$Risk)) * c(0.5, 1.2)
y_limits <- range(c(efficient_frontier[, "ExpectedReturn"],
                    tangency_stats["ExpectedReturn"],
                    individual_assets$ExpectedReturn)) * c(0.5, 1.2)
# Plot the Efficient Frontier Curve
plot(efficient_frontier[, "Risk"], efficient_frontier[, "ExpectedReturn"],
     type = "l", col = "blue", lwd = 2,
     xlab = "Risk (Standard Deviation)",
     ylab = "Expected Return",
     main = "Efficient Frontier with Tangency Portfolio",
     xlim = x_limits, ylim = y_limits)
# Highlight the Minimum Variance Portfolio
min_var_stats <- portfolio_stats(portfolio_weights, company_returns_matrix)
points(min_var_stats["Risk"], min_var_stats["ExpectedReturn"],
       col = "red", pch = 19, cex = 1.5)
text(min_var_stats["Risk"], min_var_stats["ExpectedReturn"],
     labels = "Min Variance", pos = 4, col = "red")
# Highlight the Tangency Portfolio
points(tangency_stats["Risk"], tangency_stats["ExpectedReturn"],
       col = "green", pch = 19, cex = 1.5)
text(tangency_stats["Risk"], tangency_stats["ExpectedReturn"],
     labels = "Tangency", pos = 4, col = "green")
# Plot Individual Assets
points(individual_assets$Risk, individual_assets$ExpectedReturn,
       col = "purple", pch = 17, cex = 1.2)
text(individual_assets$Risk, individual_assets$ExpectedReturn,
```

```r
     labels = individual_assets$Asset, pos = 4, col = "purple")
# Add a legend
legend("topleft", legend = c("Efficient Frontier (Curve)",
                             "Min Variance Portfolio", "Tangency Portfolio",
                             "Individual Assets"),
       col = c("blue", "red", "green", "purple"), lwd = c(2, NA, NA, NA),
       pch = c(NA, 19, 19, 17), cex = 0.8)
```




```r
# Print Tangency Portfolio Weights
print("==== Tangency Portfolio Weights ====")
```

```
## [1] "==== Tangency Portfolio Weights ===="
```

```r
print(round(tangency_weights, 4))
```

```
##    AAPL    MSFT   GOOGL    AMZN
##  0.4993  1.1906 -0.7378  0.0479
```

```r
# Tangency Portfolio Summary Statistics (Daily)
tangency_mean <- tangency_stats["ExpectedReturn"]  # Daily mean return
tangency_mean_pct <- tangency_mean * 100  # Convert to percentage
tangency_sd <- tangency_stats["Risk"]  # Daily standard deviation
# Annualized Sharpe Ratio
```

```r
tangency_annualized_sharpe <-
  ((tangency_mean - daily_risk_free_rate) * 252) / (tangency_sd * sqrt(252))
# Print the Tangency Portfolio Daily Statistics and Annualized Sharpe
print("\n==== Tangency Portfolio Daily Statistics ====")
```

```
## [1] "\n==== Tangency Portfolio Daily Statistics ===="
```

```r
print(paste("Daily Mean Return (%):", round(tangency_mean_pct, 3), "%"))
```

```
## [1] "Daily Mean Return (%): 0.175 %"
```

```r
print(paste("Daily Standard Deviation (%):", round(tangency_sd * 100, 3), "%"))
```

```
## [1] "Daily Standard Deviation (%): 1.612 %"
```

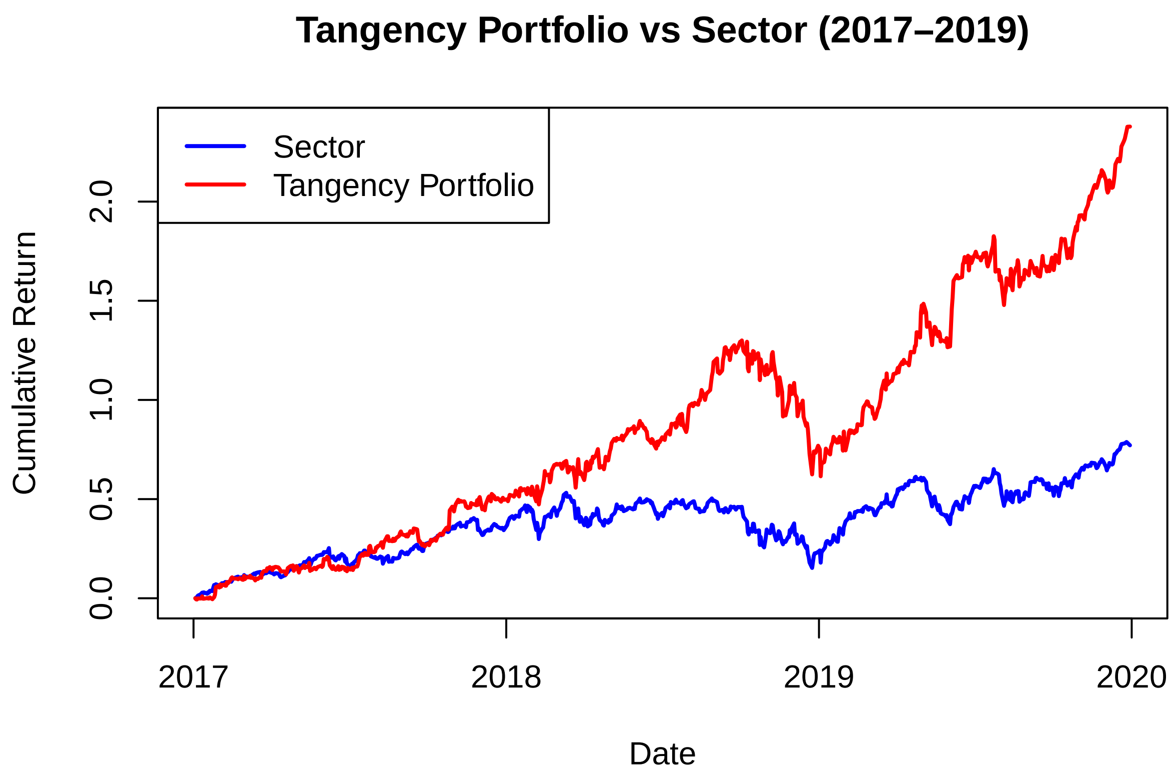
```r
print(paste("Annualized Sharpe Ratio (2% rf):",
            round(tangency_annualized_sharpe, 3)))
```

```
## [1] "Annualized Sharpe Ratio (2% rf): 1.643"
```

Project Question 4

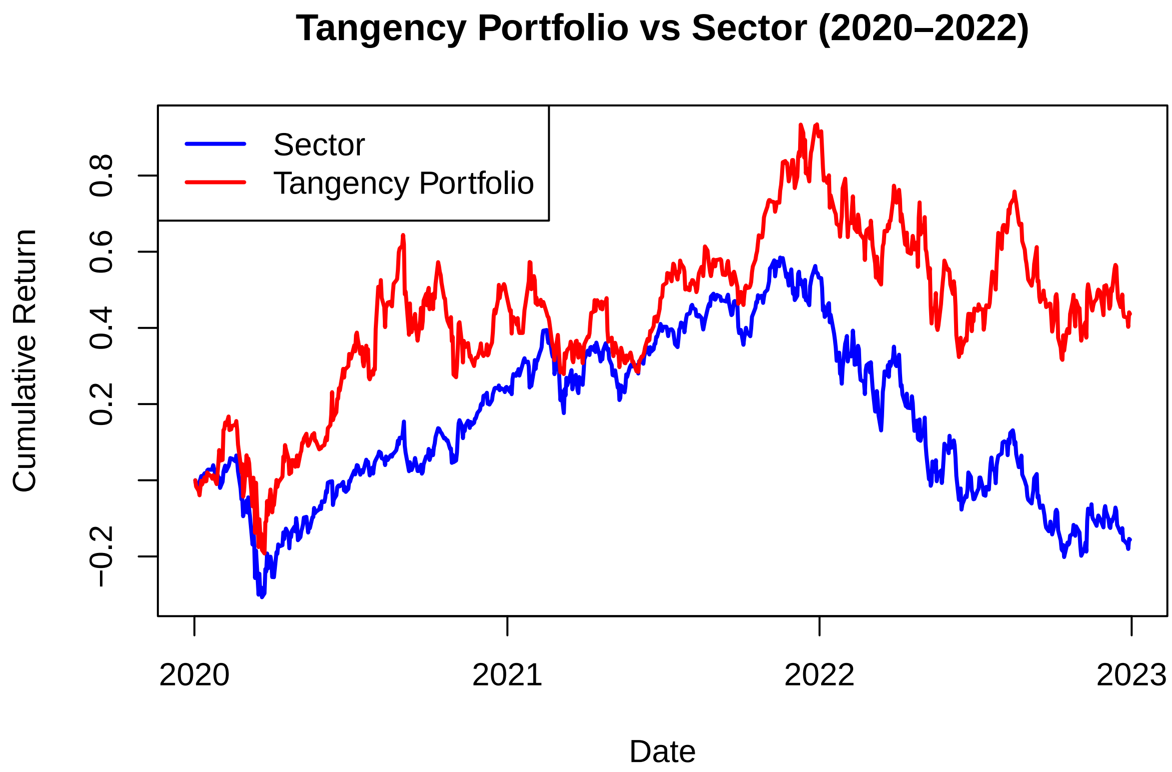
```r
# Ensure the tangency_weights and company_returns_matrix are defined
# Tangency Portfolio Weights (already calculated earlier)
tangency_weights <- tangency_portfolio(company_returns_matrix, risk_free_rate)
names(tangency_weights) <- colnames(company_returns_matrix)
# Calculate Tangency Portfolio Returns (2017-2019)
tangency_returns <- as.matrix(company_returns_matrix) %*% tangency_weights
tangency_returns <-
  xts(tangency_returns, order.by = index(company_returns_matrix))
# Function to calculate cumulative returns
calculate_cumulative_returns <- function(returns) {
  cum_returns <- cumprod(1 + returns) - 1
  xts(cum_returns, order.by = index(returns))
}
# Function to calculate annualized Sharpe ratio
calculate_sharpe_ratio <- function(returns, risk_free_rate_annual) {
  daily_risk_free_rate <- (1 + risk_free_rate_annual)^(1/252) - 1
  excess_returns <- returns - daily_risk_free_rate
  sharpe_ratio <-
    mean(excess_returns, na.rm = TRUE) / sd(excess_returns,
                                            na.rm = TRUE) * sqrt(252)
  return(sharpe_ratio)
}
# Calculate cumulative returns for the Tangency Portfolio and sector
tangency_cum_returns <- calculate_cumulative_returns(tangency_returns)
sector_cum_returns <- calculate_cumulative_returns(sector_returns)
# Calculate Sharpe ratios for 2017-2019
sharpe_tangency_2017_2019 <- calculate_sharpe_ratio(tangency_returns, 0.02)
```

```
sharpe_sector_2017_2019 <- calculate_sharpe_ratio(sector_returns, 0.02)
# Plot cumulative returns for 2017-2019
ylim_range <- range(tangency_cum_returns, sector_cum_returns, na.rm = TRUE)
plot(index(sector_cum_returns), sector_cum_returns,
     type = "l", col = "blue", lwd = 2,
     xlab = "Date", ylab = "Cumulative Return",
     main = "Tangency Portfolio vs Sector (2017-2019)",
     ylim = ylim_range)
lines(index(tangency_cum_returns), tangency_cum_returns,
      col = "red", lwd = 2)
legend("topleft", legend = c("Sector", "Tangency Portfolio"),
       col = c("blue", "red"), lwd = 2)
```




```
# Fetch data for 2020-2022
start_date <- as.Date("2020-01-01")
end_date <- as.Date("2022-12-31")
# Fetch data for sector and companies (2020-2022)
sector_prices_2020 <- getSymbols(sector_ticker,
                                 src = "yahoo", from = start_date,
                                 to = end_date, auto.assign = FALSE)
company_prices_2020 <- lapply(company_tickers, function(ticker) {
  getSymbols(ticker, src = "yahoo", from = start_date, to = end_date,
             auto.assign = FALSE)
})
# Calculate daily log returns for sector and companies (2020-2022)
```

```r
sector_returns_2020 <- dailyReturn(Ad(sector_prices_2020), type = "log")
company_returns_2020 <- lapply(company_prices_2020, function(prices) {
  dailyReturn(Ad(prices), type = "log")
})
# Combine company returns for 2020-2022
company_returns_matrix_2020 <- do.call(merge, company_returns_2020)
colnames(company_returns_matrix_2020) <- company_tickers
# Calculate Tangency Portfolio returns for 2020-2022 using the same weights
tangency_returns_2020 <-
  as.matrix(company_returns_matrix_2020) %*% tangency_weights
tangency_returns_2020 <- xts(tangency_returns_2020,
                             order.by = index(company_returns_matrix_2020))
# Calculate cumulative returns for Tangency Portfolio and sector (2020-2022)
tangency_cum_returns_2020 <-
  calculate_cumulative_returns(tangency_returns_2020)
sector_cum_returns_2020 <- calculate_cumulative_returns(sector_returns_2020)
# Calculate Sharpe ratios for 2020-2022
sharpe_tangency_2020_2022 <-
  calculate_sharpe_ratio(tangency_returns_2020, 0.02)
sharpe_sector_2020_2022 <- calculate_sharpe_ratio(sector_returns_2020, 0.02)
# Plot cumulative returns for 2020-2022
ylim_range_2020 <- range(tangency_cum_returns_2020,
                         sector_cum_returns_2020, na.rm = TRUE)
plot(index(sector_cum_returns_2020),
     sector_cum_returns_2020, type = "l", col = "blue", lwd = 2,
     xlab = "Date",
     ylab = "Cumulative Return",
     main = "Tangency Portfolio vs Sector (2020-2022)",
     ylim = ylim_range_2020)
lines(index(tangency_cum_returns_2020),
      tangency_cum_returns_2020, col = "red", lwd = 2)
legend("topleft", legend = c("Sector", "Tangency Portfolio"),
       col = c("blue", "red"), lwd = 2)
```




```r
# Print Sharpe ratios
print(paste("Sharpe Ratio (Tangency Portfolio, 2017-2019):",
            round(sharpe_tangency_2017_2019, 3)))
```

```
## [1] "Sharpe Ratio (Tangency Portfolio, 2017-2019): 1.644"
```

```r
print(paste("Sharpe Ratio (Sector, 2017-2019):",
            round(sharpe_sector_2017_2019, 3)))
```

```
## [1] "Sharpe Ratio (Sector, 2017-2019): 0.957"
```

```r
print(paste("Sharpe Ratio (Tangency Portfolio, 2020-2022):",
            round(sharpe_tangency_2020_2022, 3)))
```

```
## [1] "Sharpe Ratio (Tangency Portfolio, 2020-2022): 0.453"
```

```r
print(paste("Sharpe Ratio (Sector, 2020-2022):",
            round(sharpe_sector_2020_2022, 3)))
```

```
## [1] "Sharpe Ratio (Sector, 2020-2022): -0.026"
```

Project Question 5

```r
# Define start and end dates for the analysis
start_date <- as.Date("2017-01-01")
end_date <- as.Date("2019-12-31")
# Define S&P 500 ticker and fetch data for the sample period (2017-2019)
sp500_ticker <- "^GSPC"  # S&P 500 Index
sp500_prices <- getSymbols(sp500_ticker,
                           src = "yahoo", from = start_date,
                           to = end_date, auto.assign = FALSE)
sp500_returns <- dailyReturn(Ad(sp500_prices), type = "log")
# Fetch sector and company data for the same period
sector_prices <- getSymbols(sector_ticker, src = "yahoo",
                            from = start_date,
                            to = end_date, auto.assign = FALSE)
sector_returns <- dailyReturn(Ad(sector_prices), type = "log")
company_prices <- lapply(company_tickers, function(ticker) {
  getSymbols(ticker, src = "yahoo", from = start_date,
             to = end_date, auto.assign = FALSE)
})
company_returns <- lapply(company_prices, function(prices) {
  dailyReturn(Ad(prices), type = "log")
})
company_returns_matrix <- do.call(merge, company_returns)
colnames(company_returns_matrix) <- company_tickers
# Calculate portfolio returns using Tangency Weights
tangency_weights <- tangency_portfolio(company_returns_matrix, risk_free_rate)
portfolio_returns <- as.matrix(company_returns_matrix) %*% tangency_weights
portfolio_returns <- xts(portfolio_returns,
                         order.by = index(company_returns_matrix))
# Align all returns: S&P 500, sector, portfolio, and company returns
aligned_returns <- merge.xts(sp500_returns,
                             sector_returns, portfolio_returns,
                             company_returns_matrix)
aligned_returns <- na.omit(aligned_returns)  # Remove rows with NA values
colnames(aligned_returns) <- c("SP500", "Sector", "Portfolio", company_tickers)
# Debug: Check aligned data
if (nrow(aligned_returns) == 0) {
  stop("No valid data available after merging and NA removal.")
}
# CAPM Function with hypothesis testing
calculate_capm_with_pvalues <- function(dependent, market) {
  if (all(is.na(dependent)) || all(is.na(market)) ||
      length(dependent) == 0 || length(market) == 0) {
    return(c(alpha = NA, beta = NA, p_alpha = NA, p_beta = NA))
  }
  model <- lm(dependent ~ market)
  alpha <- coef(model)[1]
  beta <- coef(model)[2]
  # Extract p-values for alpha and beta
  p_values <- summary(model)$coefficients[, 4]
  p_alpha <- p_values[1]
  p_beta <- p_values[2]
  return(c(alpha = alpha, beta = beta, p_alpha = p_alpha, p_beta = p_beta))
}
```

```r
# Create a data frame for CAPM results
capm_results <- data.frame(Asset = c("Sector", "Portfolio", company_tickers),
                           Alpha = NA, Beta = NA,
                           P_Alpha = NA, P_Beta = NA)
# Define risk-free rate
annual_risk_free_rate <- 0.02  # 2% annual
daily_risk_free_rate <- annual_risk_free_rate / 252  # Convert to daily
# Adjust SP500 returns to excess returns
sp500_excess_returns <- aligned_returns[, "SP500"] - daily_risk_free_rate
# Calculate CAPM for the sector
sector_excess_returns <- aligned_returns[, "Sector"] - daily_risk_free_rate
capm_sector <- calculate_capm_with_pvalues(sector_excess_returns,
                                           sp500_excess_returns)
capm_results[1, 2:5] <- capm_sector
# Calculate CAPM for the portfolio
portfolio_excess_returns <-
  aligned_returns[, "Portfolio"] - daily_risk_free_rate
capm_portfolio <- calculate_capm_with_pvalues(portfolio_excess_returns,
                                              sp500_excess_returns)
capm_results[2, 2:5] <- capm_portfolio
# Calculate CAPM for each company
for (i in seq_along(company_tickers)) {
  company_excess_returns <-
    aligned_returns[, company_tickers[i]] - daily_risk_free_rate
  capm_company <-
    calculate_capm_with_pvalues(company_excess_returns, sp500_excess_returns)
  capm_results[i + 2, 2:5] <- capm_company
}
# Round results to 3 decimal places
capm_results$Alpha <- round(capm_results$Alpha, 3)
capm_results$Beta <- round(capm_results$Beta, 3)
capm_results$P_Alpha <- round(capm_results$P_Alpha, 3)
capm_results$P_Beta <- round(capm_results$P_Beta, 3)
# Print the CAPM results with hypothesis testing
print("CAPM Results with Hypothesis Testing (2017-2019):")
```

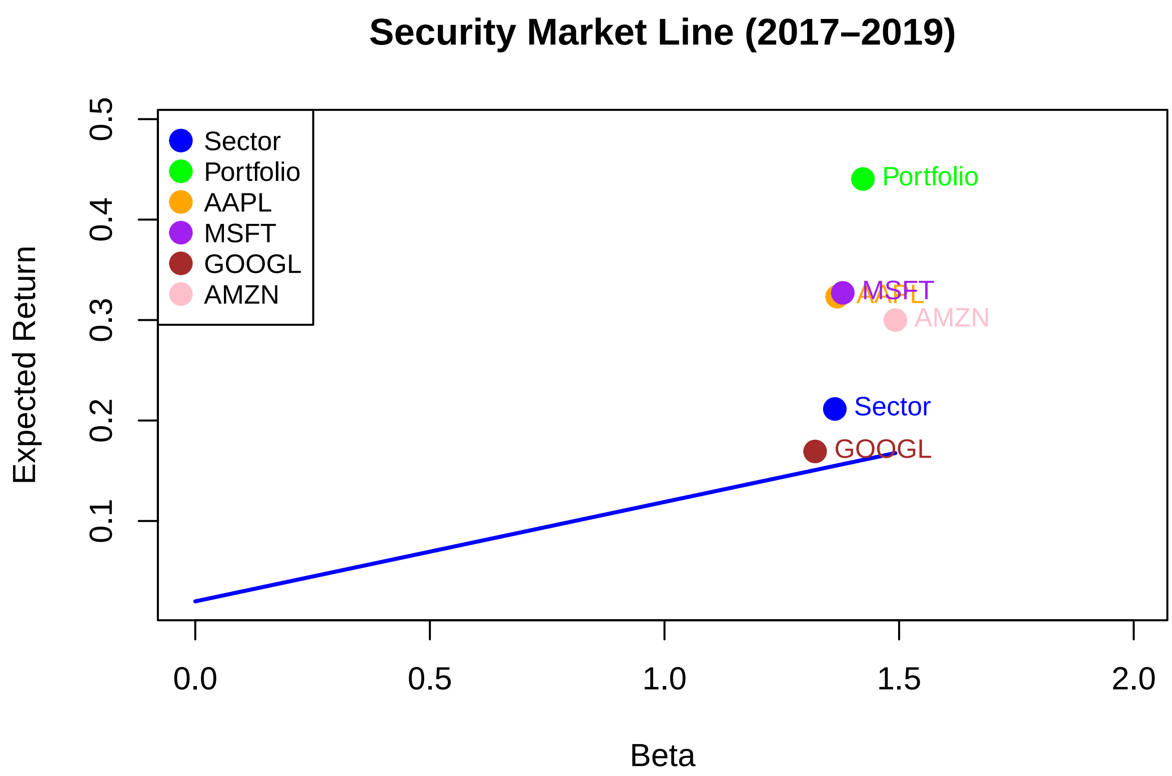
```
## [1] "CAPM Results with Hypothesis Testing (2017-2019):"
```

```r
print(capm_results)
```

```
##       Asset Alpha  Beta P_Alpha P_Beta
## 1    Sector 0.000 1.363   0.317      0
## 2 Portfolio 0.001 1.423   0.007      0
## 3      AAPL 0.001 1.368   0.097      0
## 4      MSFT 0.001 1.380   0.018      0
## 5     GOOGL 0.000 1.321   0.838      0
## 6      AMZN 0.001 1.492   0.244      0
```

```r
# Calculate Expected and Actual Returns
market_risk_premium <- mean(sp500_excess_returns, na.rm = TRUE) * 252
capm_results$Expected_Return <-
  annual_risk_free_rate + capm_results$Beta * market_risk_premium
```

```r
# Calculate Actual Returns (annualized)
actual_returns <- colMeans(aligned_returns, na.rm = TRUE) * 252  # Annualize
capm_results$Actual_Return <-
  c(actual_returns["Sector"], actual_returns["Portfolio"],
    actual_returns[company_tickers])
# Define X and Y limits dynamically
x_lim <- c(0, max(capm_results$Beta, na.rm = TRUE) + 0.5)
y_lim <- c(annual_risk_free_rate, max(capm_results$Expected_Return,
                                      capm_results$Actual_Return,
                                      na.rm = TRUE) + 0.05)
# Plot the Security Market Line (SML)
plot(
  x = c(0, max(capm_results$Beta, na.rm = TRUE)),  # SML from Beta = 0
  y = c(annual_risk_free_rate,
        max(capm_results$Expected_Return, na.rm = TRUE)),
  type = "l", col = "blue", lwd = 2,
  xlab = "Beta", ylab = "Expected Return",
  main = "Security Market Line (2017-2019)",
  xlim = x_lim, ylim = y_lim
)
# Add individual points for assets
asset_colors <- c("blue", "green", "orange", "purple", "brown", "pink")
for (i in 1:nrow(capm_results)) {
  points(capm_results$Beta[i], capm_results$Actual_Return[i],
         col = asset_colors[i], pch = 19, cex = 1.5)
}
# Annotate the points with asset names
for (i in 1:nrow(capm_results)) {
  text(
    capm_results$Beta[i], capm_results$Actual_Return[i],
    labels = capm_results$Asset[i], pos = 4, cex = 0.8, col = asset_colors[i]
  )
}
# Add a legend to the plot
legend(
  "topleft", legend = capm_results$Asset,
  col = asset_colors, pch = 19, cex = 0.8, pt.cex = 1.5
)
```



Project Question 6

```
# Define the specific date range for CAPM (2020-2022)
capm_start_date_2020 <- as.Date("2020-01-01")
capm_end_date_2022 <- as.Date("2022-12-31")
# Fetch S&P 500 (^GSPC) data for CAPM analysis
sp500_prices_2020 <- getSymbols("^GSPC", src = "yahoo",
                                from = capm_start_date_2020,
                                to = capm_end_date_2022, auto.assign = FALSE)
sp500_returns_2020 <- dailyReturn(Ad(sp500_prices_2020), type = "log")
# Fetch sector data for the same period
sector_prices_2020 <- getSymbols(sector_ticker, src = "yahoo",
                                 from = capm_start_date_2020,
                                 to = capm_end_date_2022, auto.assign = FALSE)
sector_returns_2020 <- dailyReturn(Ad(sector_prices_2020), type = "log")
# Fetch company data for the same period
company_prices_2020 <- lapply(company_tickers, function(ticker) {
  getSymbols(ticker, src = "yahoo",
             from = capm_start_date_2020,
             to = capm_end_date_2022, auto.assign = FALSE)
})
company_returns_2020 <- lapply(company_prices_2020, function(prices) {
  dailyReturn(Ad(prices), type = "log")
})
```

```r
# Combine company returns into a matrix
company_returns_matrix_2020 <- do.call(merge, company_returns_2020)
colnames(company_returns_matrix_2020) <- company_tickers
# Calculate Tangency Portfolio returns for 2020-2022 using the same weights
tangency_weights <-
  tangency_portfolio(company_returns_matrix, daily_risk_free_rate)
portfolio_returns_2020 <-
  as.matrix(company_returns_matrix_2020) %*% tangency_weights
portfolio_returns_2020 <-
  xts(portfolio_returns_2020, order.by = index(company_returns_matrix_2020))

# Align all returns: S&P 500, sector, portfolio, and company returns
aligned_returns_2020 <- merge.xts(sp500_returns_2020,
                                  sector_returns_2020,
                                  portfolio_returns_2020,
                                  company_returns_matrix_2020)
aligned_returns_2020 <- na.omit(aligned_returns_2020)  # Remove rows with NA values
colnames(aligned_returns_2020) <-
  c("SP500", "Sector", "Portfolio", company_tickers)
# Define risk-free rate
annual_risk_free_rate <- 0.02  # 2% annual
daily_risk_free_rate <- annual_risk_free_rate / 252  # Convert to daily
# Adjust S&P 500 returns to excess returns
sp500_excess_returns_2020 <-
  aligned_returns_2020[, "SP500"] - daily_risk_free_rate
# CAPM Function with hypothesis testing
calculate_capm_with_pvalues <- function(dependent, market) {
  if (all(is.na(dependent)) || all(is.na(market)) ||
      length(dependent) == 0 || length(market) == 0) {
    return(c(alpha = NA, beta = NA, p_alpha = NA, p_beta = NA))
  }
  model <- lm(dependent ~ market)
  alpha <- coef(model)[1]
  beta <- coef(model)[2]
  # Extract p-values for alpha and beta
  p_values <- summary(model)$coefficients[, 4]
  p_alpha <- p_values[1]
  p_beta <- p_values[2]
  return(c(alpha = round(alpha, 3), beta = round(beta, 3),
           p_alpha = round(p_alpha, 3), p_beta = round(p_beta, 3)))
}
# Create a data frame for CAPM results (2020-2022)
capm_results_2020 <-
  data.frame(Asset = c("Sector", "Portfolio", company_tickers),
                               Alpha = NA, Beta = NA,
                               P_Alpha = NA, P_Beta = NA)
# Calculate CAPM for the sector
sector_excess_returns_2020 <-
  aligned_returns_2020[, "Sector"] - daily_risk_free_rate
capm_sector_2020 <-
  calculate_capm_with_pvalues(sector_excess_returns_2020,
                              sp500_excess_returns_2020)
capm_results_2020[1, 2:5] <- capm_sector_2020
```

```
# Calculate CAPM for the portfolio
portfolio_excess_returns_2020 <-
  aligned_returns_2020[, "Portfolio"] - daily_risk_free_rate
capm_portfolio_2020 <-
  calculate_capm_with_pvalues(portfolio_excess_returns_2020,
                              sp500_excess_returns_2020)
capm_results_2020[2, 2:5] <- capm_portfolio_2020
# Calculate CAPM for each company
for (i in seq_along(company_tickers)) {
  company_excess_returns_2020 <-
    aligned_returns_2020[, company_tickers[i]] - daily_risk_free_rate
  capm_company_2020 <-
    calculate_capm_with_pvalues(company_excess_returns_2020,
                                sp500_excess_returns_2020)
  capm_results_2020[i + 2, 2:5] <- capm_company_2020
}
# Print the CAPM results with hypothesis testing
print("CAPM Results with Hypothesis Testing (2020-2022):")
```

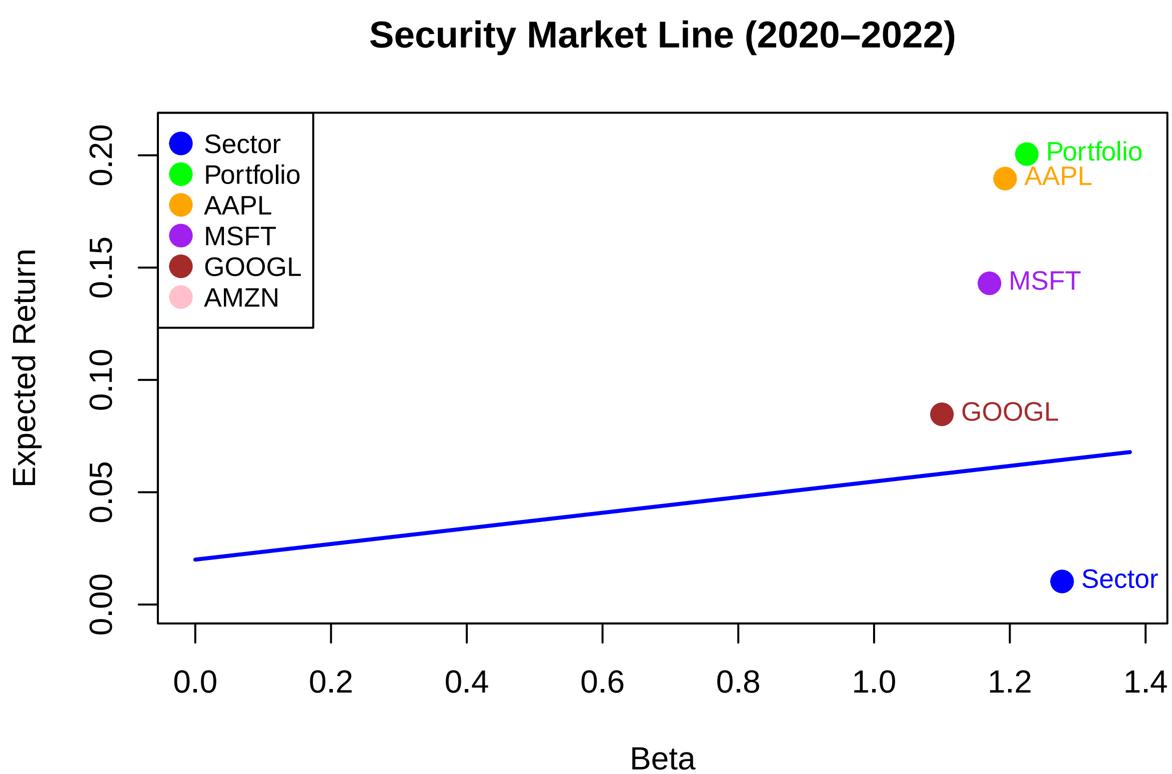
```
## [1] "CAPM Results with Hypothesis Testing (2020-2022):"
```

```
print(capm_results_2020)
```

```
##       Asset Alpha  Beta P_Alpha P_Beta
## 1    Sector 0.000 1.277   0.564      0
## 2 Portfolio 0.001 1.225   0.334      0
## 3      AAPL 0.001 1.193   0.286      0
## 4      MSFT 0.000 1.170   0.422      0
## 5     GOOGL 0.000 1.100   0.819      0
## 6      AMZN 0.000 1.001   0.576      0
```

```
# Calculate the market risk premium (annualized)
market_risk_premium <- mean(sp500_excess_returns_2020, na.rm = TRUE) * 252
# Calculate Expected Returns using CAPM formula
capm_results_2020$Expected_Return <-
  annual_risk_free_rate + capm_results_2020$Beta * market_risk_premium
# Calculate Actual Returns (annualized)
actual_returns <- colMeans(aligned_returns_2020, na.rm = TRUE) * 252
capm_results_2020$Actual_Return <-
  c(actual_returns["Sector"], actual_returns["Portfolio"],
    actual_returns[company_tickers])
# Define X and Y axis limits dynamically with added buffer for zoom-out
x_lim <- c(0, max(capm_results_2020$Beta, na.rm = TRUE) + 0.1)
y_lim <- c(0, max(capm_results_2020$Expected_Return,
                  capm_results_2020$Actual_Return, na.rm = TRUE) + 0.01)
# Plot the Security Market Line (SML) with zoomed-out view
plot(
  x = c(0, max(capm_results_2020$Beta, na.rm = TRUE) + 0.1),  # Extend X-axis
  y = c(annual_risk_free_rate,
        annual_risk_free_rate + (max(capm_results_2020$Beta,
                                     na.rm = TRUE) + 0.1)*market_risk_premium),
  type = "l", col = "blue", lwd = 2,
```

```
  xlab = "Beta", ylab = "Expected Return",
  main = "Security Market Line (2020-2022)",
  xlim = x_lim, ylim = y_lim
)
# Add individual points for assets
for (i in 1:nrow(capm_results_2020)) {
  points(capm_results_2020$Beta[i],
         capm_results_2020$Actual_Return[i],
         col = asset_colors[i], pch = 19, cex = 1.5)
}
# Annotate the points with asset names
for (i in 1:nrow(capm_results_2020)) {
  text(
    capm_results_2020$Beta[i],
    capm_results_2020$Actual_Return[i],
    labels = capm_results_2020$Asset[i],
    pos = 4, cex = 0.8, col = asset_colors[i]
  )
}
# Add a legend to the plot
legend(
  "topleft", legend = capm_results_2020$Asset,
  col = asset_colors, pch = 19, cex = 0.8, pt.cex = 1.5
)
```



Project Question 7

```r
# Define a function to calculate VaR and ES
calculate_var_es <- function(returns, confidence_level = 0.05) {
  # Value at Risk (VaR): quantile of returns
  var <- quantile(returns, probs = confidence_level, na.rm = TRUE)
  # Expected Shortfall (ES): mean of returns below the VaR
  es <- mean(returns[returns <= var], na.rm = TRUE)
  c(VaR = var, ES = es)
}
# Initialize the results data frame
var_es_results <- data.frame(
  Asset = c("Sector", "Tangency Portfolio", company_tickers),
  VaR = NA,
  ES = NA
)
# Sector VaR and ES
var_es_results[1, 2:3] <- calculate_var_es(as.numeric(sector_returns),
                                           confidence_level = 0.05)
# Tangency Portfolio VaR and ES
var_es_results[2, 2:3] <- calculate_var_es(as.numeric(portfolio_returns),
                                           confidence_level = 0.05)
# Company VaR and ES
for (i in seq_along(company_tickers)) {
  var_es_results[i + 2, 2:3] <-
    calculate_var_es(as.numeric(company_returns[[i]]), confidence_level = 0.05)
}
# Round the results to 3 decimals
var_es_results$VaR <- round(var_es_results$VaR, 3)
var_es_results$ES <- round(var_es_results$ES, 3)
# Print the results
print("VaR and ES Results (2017-2019):")
```

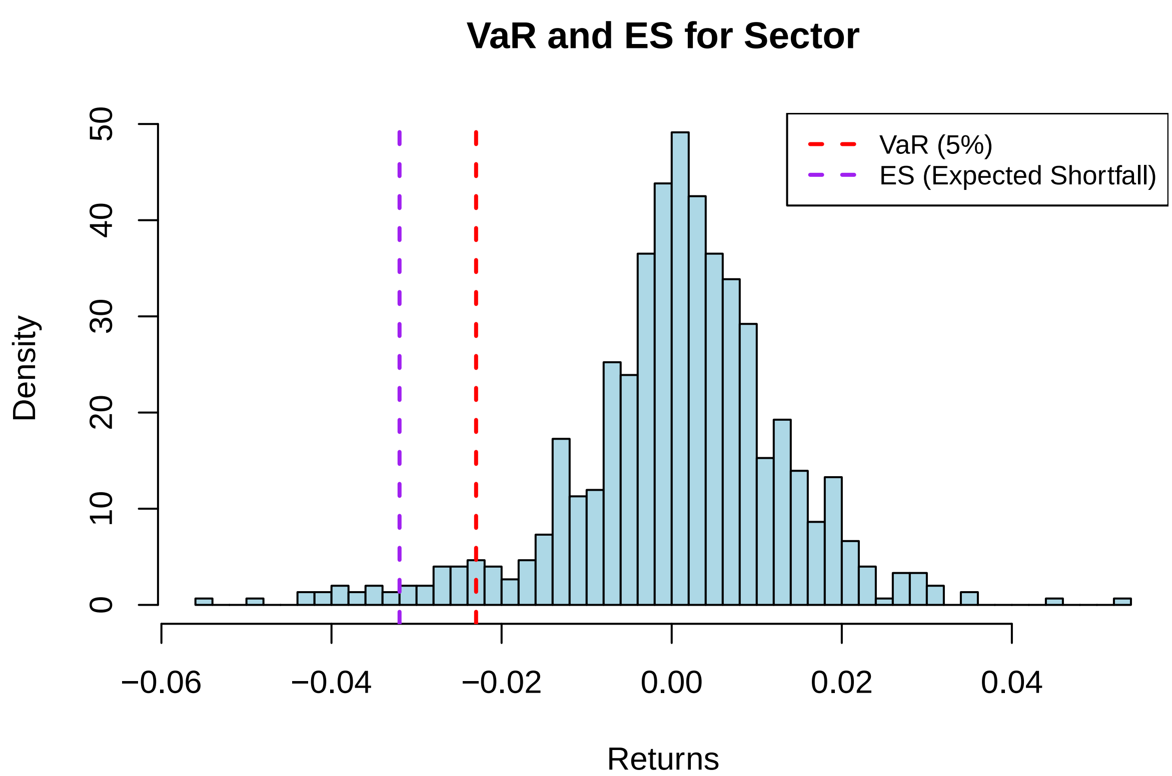
```
## [1] "VaR and ES Results (2017-2019):"
```

```r
print(var_es_results)
```

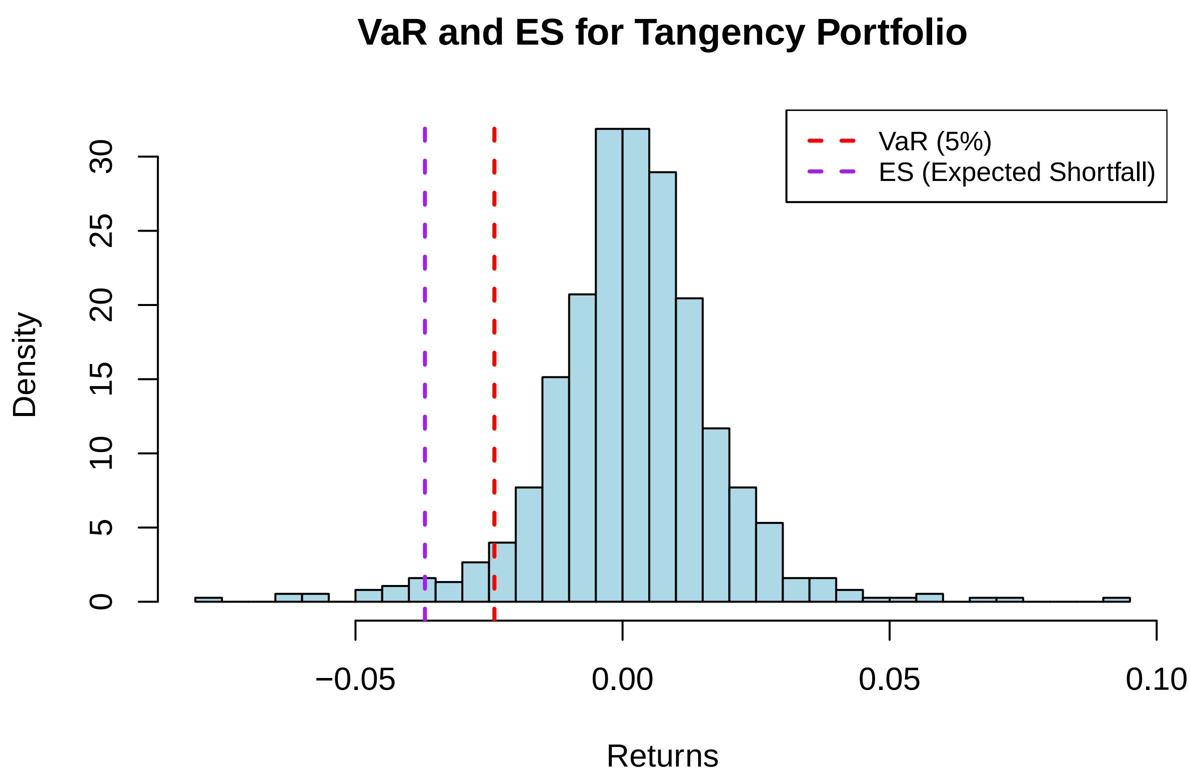
```
##                Asset    VaR     ES
## 1             Sector -0.023 -0.032
## 2 Tangency Portfolio -0.024 -0.037
## 3               AAPL -0.024 -0.037
## 4               MSFT -0.021 -0.033
## 5              GOOGL -0.025 -0.036
## 6               AMZN -0.027 -0.042
```

```r
# Define a function to plot the quantiles
plot_var_es <- function(returns, var, es, asset_name) {
  hist(returns, breaks = 50, col = "lightblue", probability = TRUE,
       main = paste("VaR and ES for", asset_name),
       xlab = "Returns", ylab = "Density")
  # Add vertical lines for VaR and ES
```

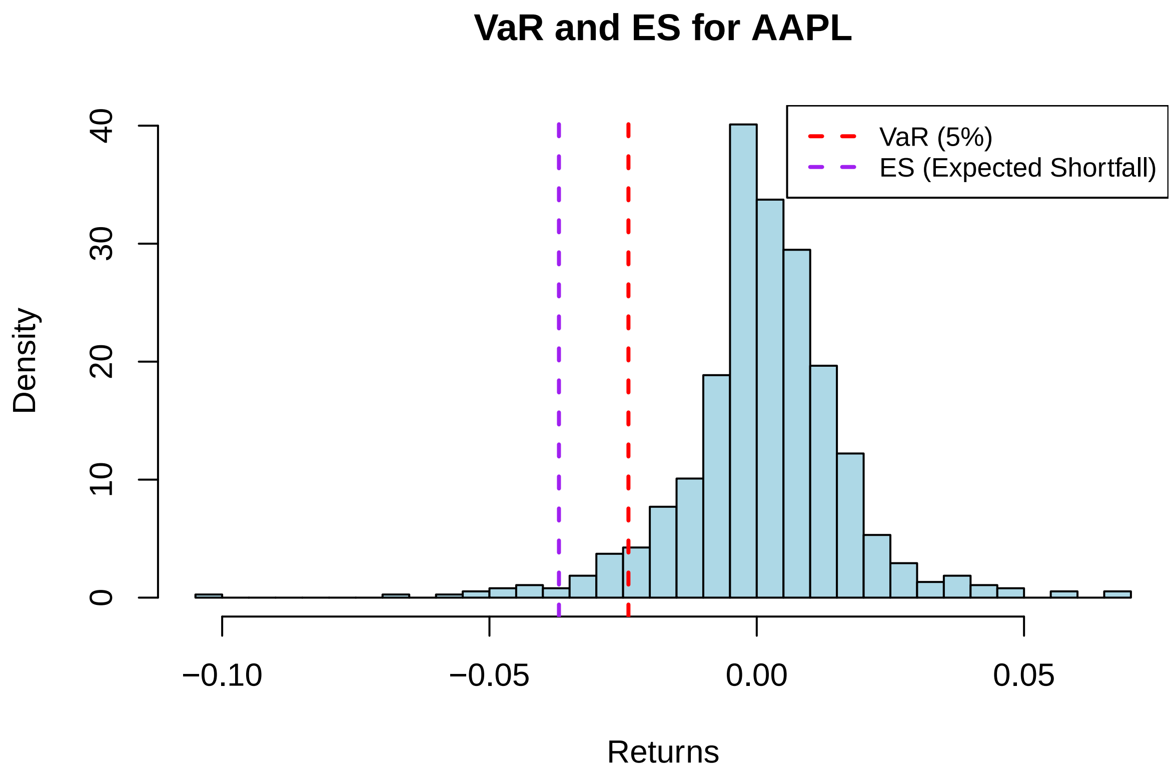
```r
  abline(v = var, col = "red", lwd = 2, lty = 2)  # VaR line
  abline(v = es, col = "purple", lwd = 2, lty = 2)  # ES line
  # Add a legend
  legend("topright", legend = c("VaR (5%)", "ES (Expected Shortfall)"),
         col = c("red", "purple"), lty = 2, lwd = 2, cex = 0.8)
}
# Plot for Sector
sector_var <- var_es_results$VaR[1]
sector_es <- var_es_results$ES[1]
plot_var_es(as.numeric(sector_returns), sector_var, sector_es, "Sector")
```

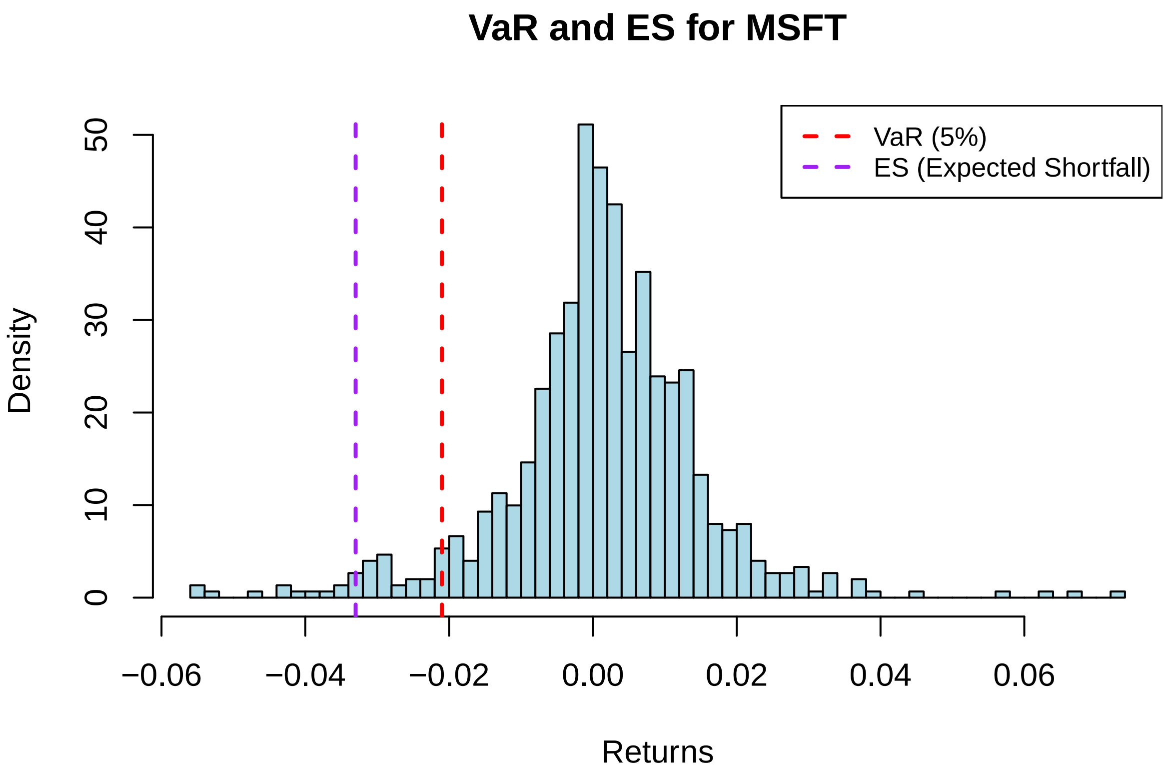


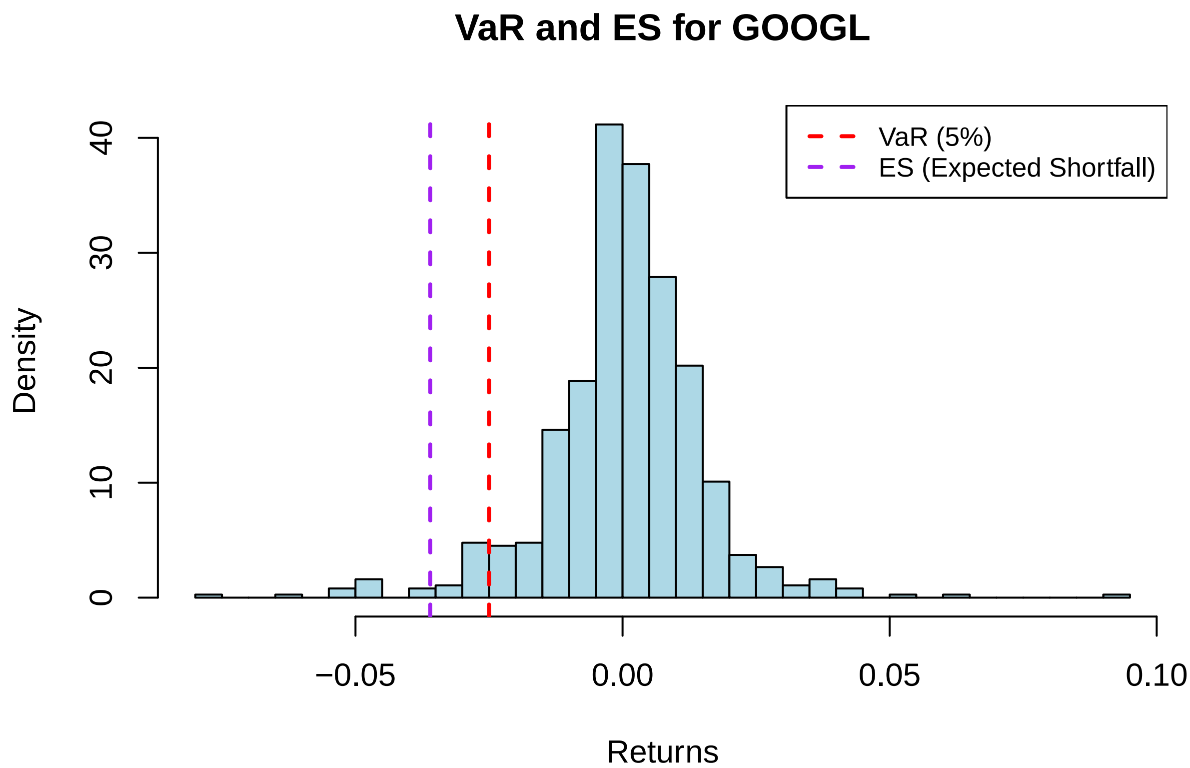

```r
# Plot for Tangency Portfolio
portfolio_var <- var_es_results$VaR[2]
portfolio_es <- var_es_results$ES[2]
plot_var_es(as.numeric(portfolio_returns),
            portfolio_var, portfolio_es, "Tangency Portfolio")
```

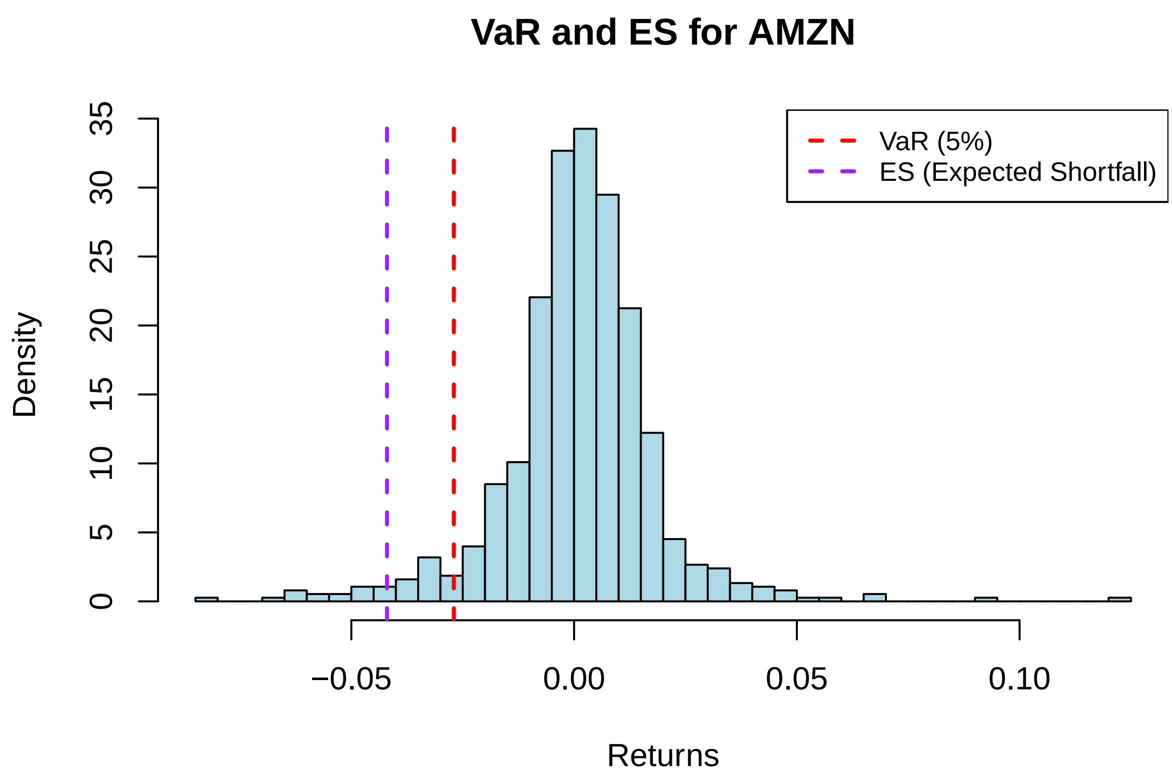



```r
# Plots for Individual Companies
for (i in seq_along(company_tickers)) {
  company_var <- var_es_results$VaR[i + 2]
  company_es <- var_es_results$ES[i + 2]
  company_name <- company_tickers[i]
  # Plot for each company
  plot_var_es(as.numeric(company_returns[[i]]),
              company_var, company_es, company_name)
}
```


VaR and ES for AAPL
VaR (5%)
ES (Expected Shortfall)
Density
0
10
20
30
40
−0.10
−0.05
0.00
0.05
Returns


VaR and ES for MSFT
VaR (5%)
ES (Expected Shortfall)
Density
0
10
20
30
40
50
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06
Returns


VaR and ES for GOOGL
VaR (5%)
ES (Expected Shortfall)
Density
0
10
20
30
40
−0.05
0.00
0.05
0.10
Returns



Project Question 8

```
# Define a function to calculate VaR and ES (from Part 7)
calculate_var_es <- function(returns, confidence_level = 0.05) {
  # Value at Risk (VaR): quantile of returns
  var <- quantile(returns, probs = confidence_level, na.rm = TRUE)
  # Expected Shortfall (ES): mean of returns below the VaR
  es <- mean(returns[returns <= var], na.rm = TRUE)
  c(VaR = var, ES = es)
}
# Fetch sector and company returns for 2020-2022
sector_returns_2020 <-
  dailyReturn(Ad(getSymbols(sector_ticker, src = "yahoo",
                            from = capm_start_date_2020,
                            to = capm_end_date_2022,
                            auto.assign = FALSE)), type = "log")
company_returns_2020 <- lapply(company_tickers, function(ticker) {
  dailyReturn(Ad(getSymbols(ticker,
                            src = "yahoo", from = capm_start_date_2020,
                            to = capm_end_date_2022, auto.assign = FALSE)),
              type = "log")
})
# Calculate portfolio returns for 2020-2022 using the same weights
company_returns_matrix_2020 <- do.call(merge.xts, company_returns_2020)
```

```r
portfolio_returns_2020 <-
  xts(as.matrix(company_returns_matrix_2020) %*% portfolio_weights,
      order.by = index(company_returns_matrix_2020))
# Calculate VaR and ES for sector, portfolio, and companies (2020-2022)
var_es_results_2020 <- data.frame(
  Asset = c("Sector", "Portfolio", company_tickers),
  VaR = NA,
  ES = NA
)
# Sector VaR and ES
var_es_results_2020[1, 2:3] <-
  calculate_var_es(as.numeric(sector_returns_2020), confidence_level = 0.05)
# Portfolio VaR and ES
var_es_results_2020[2, 2:3] <-
  calculate_var_es(as.numeric(portfolio_returns_2020), confidence_level = 0.05)
# Company VaR and ES
for (i in seq_along(company_tickers)) {
  var_es_results_2020[i + 2, 2:3] <-
    calculate_var_es(as.numeric(company_returns_2020[[i]]),
                     confidence_level = 0.05)
}
# Round the results to the 3rd decimal
var_es_results_2020$VaR <- round(var_es_results_2020$VaR, 3)
var_es_results_2020$ES <- round(var_es_results_2020$ES, 3)
# Print the results
print("VaR and ES Results (2020-2022):")
```

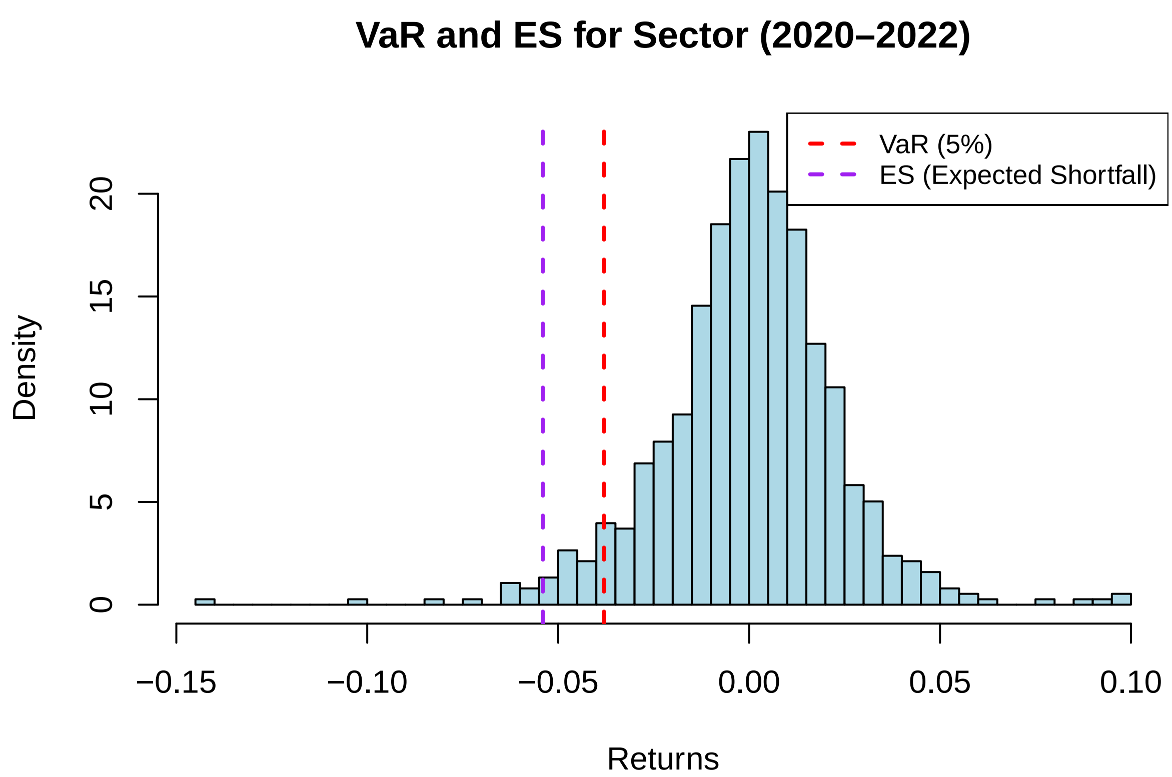
```
## [1] "VaR and ES Results (2020-2022):"
```

```r
print(var_es_results_2020)
```

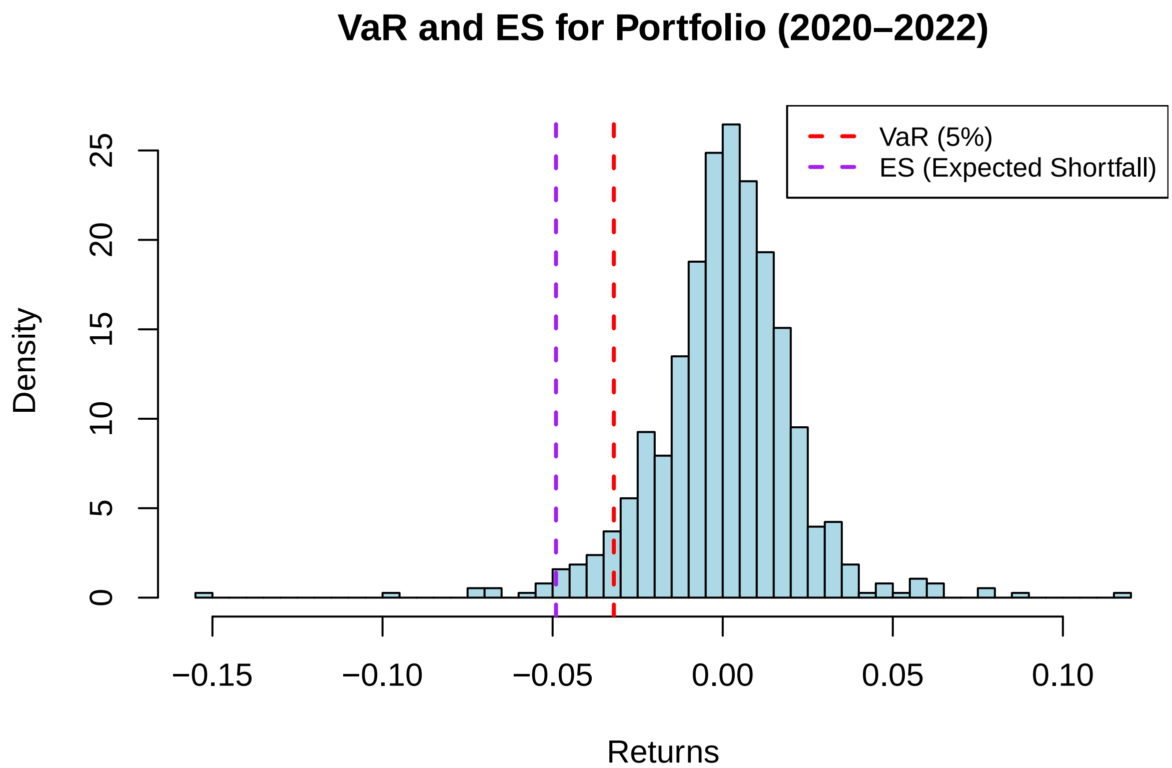
```
##       Asset    VaR     ES
## 1    Sector -0.038 -0.054
## 2 Portfolio -0.032 -0.049
## 3      AAPL -0.035 -0.052
## 4      MSFT -0.034 -0.051
## 5     GOOGL -0.036 -0.051
## 6      AMZN -0.038 -0.058
```

```r
# Define a function to plot the quantiles with positive VaR and ES
plot_var_es <- function(returns, var, es, asset_name) {
  hist(returns, breaks = 50, col = "lightblue", probability = TRUE,
       main = paste("VaR and ES for", asset_name, "(2020-2022)"),
       xlab = "Returns", ylab = "Density")
  # Add vertical lines for VaR and ES
  abline(v = var, col = "red", lwd = 2, lty = 2)  # VaR line
  abline(v = es, col = "purple", lwd = 2, lty = 2)  # ES line
  # Add a legend
  legend("topright", legend = c("VaR (5%)", "ES (Expected Shortfall)"),
         col = c("red", "purple"), lty = 2, lwd = 2, cex = 0.8)
}
# Plot for Sector
```

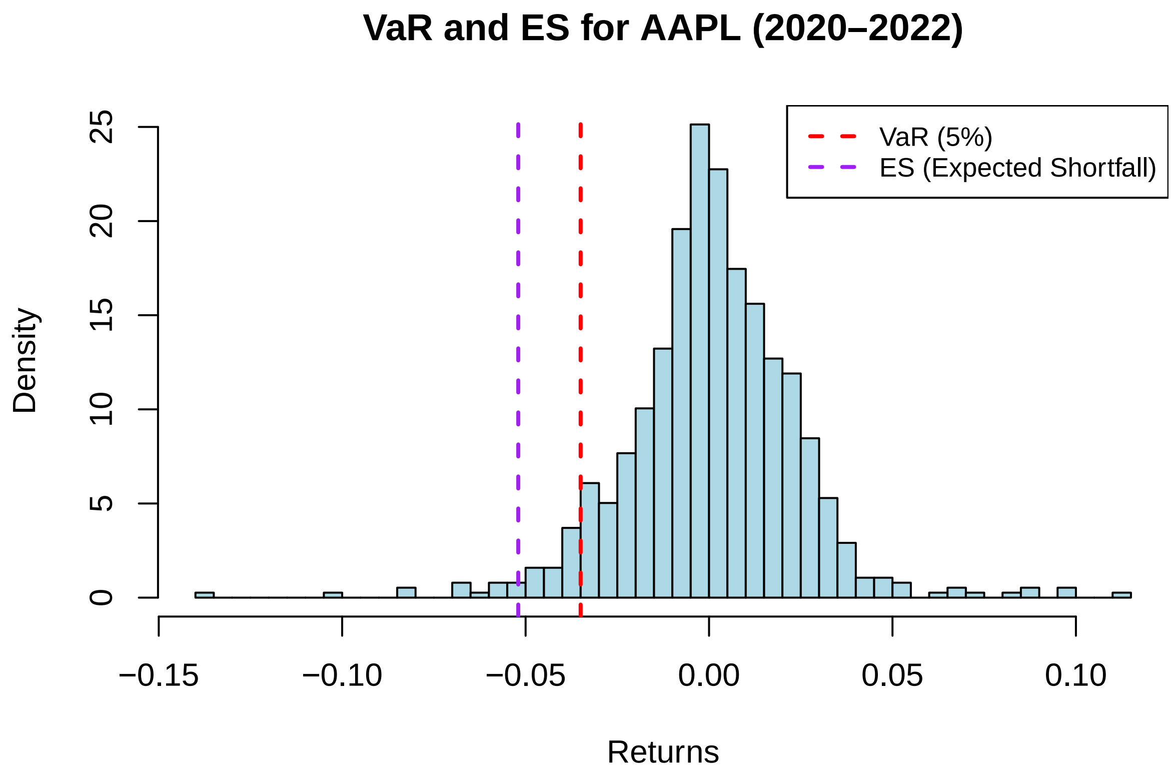
```r
sector_var_2020 <- var_es_results_2020$VaR[1]
sector_es_2020 <- var_es_results_2020$ES[1]
plot_var_es(as.numeric(sector_returns_2020),
            sector_var_2020, sector_es_2020, "Sector")
```



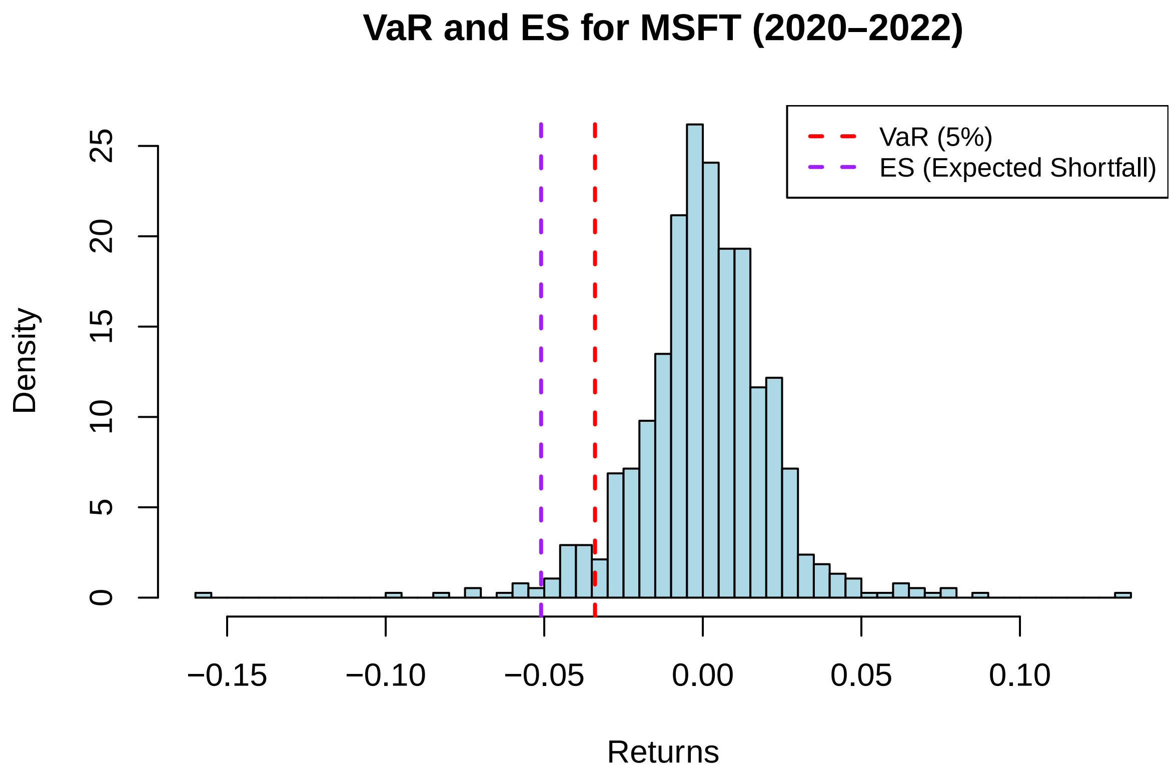

```r
# Plot for Portfolio
portfolio_var_2020 <- var_es_results_2020$VaR[2]
portfolio_es_2020 <- var_es_results_2020$ES[2]
plot_var_es(as.numeric(portfolio_returns_2020),
            portfolio_var_2020, portfolio_es_2020, "Portfolio")
```

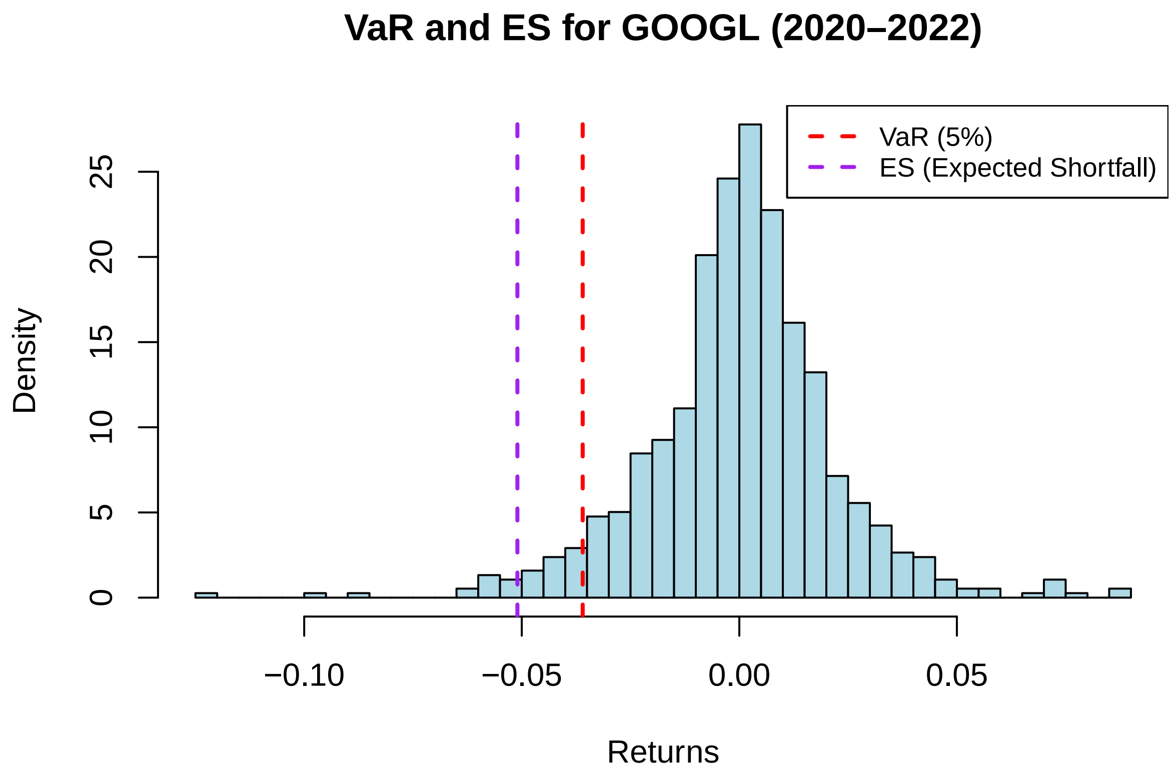


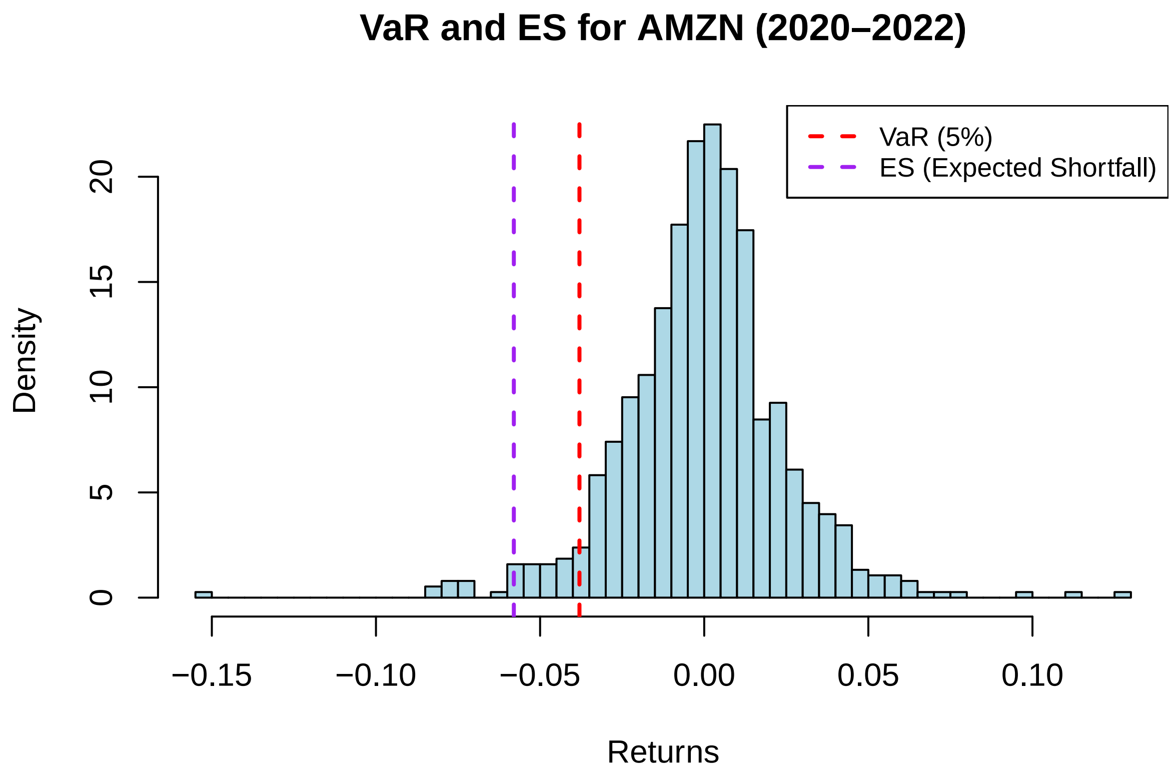

```r
# Plots for Individual Companies
for (i in seq_along(company_tickers)) {
  company_var_2020 <- var_es_results_2020$VaR[i + 2]
  company_es_2020 <- var_es_results_2020$ES[i + 2]
  company_name <- company_tickers[i]
  # Plot for each company
  plot_var_es(as.numeric(company_returns_2020[[i]]),
              company_var_2020, company_es_2020, company_name)
}
```



VaR and ES for AAPL (2020–2022)
VaR (5%)
ES (Expected Shortfall)
Density
Returns
0
5
10
15
20
25
−0.15
−0.10
−0.05
0.00
0.05
0.10


VaR and ES for MSFT (2020–2022)
VaR (5%)
ES (Expected Shortfall)
Density
Returns
0
5
10
15
20
25
−0.15
−0.10
−0.05
0.00
0.05
0.10


VaR and ES for GOOGL (2020–2022)
VaR (5%)
ES (Expected Shortfall)
Density
0
5
10
15
20
25
−0.10
−0.05
0.00
0.05
Returns


VaR and ES for AMZN (2020–2022)
VaR (5%)
ES (Expected Shortfall)
Density
Returns
0
5
10
15
20
−0.15
−0.10
−0.05
0.00
0.05
0.10