Portfolio Construction and Risk Analysis

Rebecca Li

2025-02-11

This project explores portfolio optimization and risk assessment using historical stock data from technology and financial sectors. The analysis is divided into two main problems:

Portfolio Construction and Optimization:

Objective: Construct portfolios targeting annual expected returns of 5%, 7.5%, 10%, 12.5%, and 15%, using historical data from 2020-2022 and 2023-present.

Methodology: Utilized Markowitz Mean-Variance Optimization to determine portfolio weights. Calculated tangency and minimum variance portfolios, subject to no short selling. Evaluated performance using Sharpe ratios and risk-return characteristics.

Findings: Adding financial stocks to the portfolio improved diversification, leading to better risk-adjusted returns.
CAPM-Based Risk Analysis:

Objective: Assess the systematic risk (β) and abnormal returns (α) of selected stocks using the Capital Asset Pricing Model (CAPM).

Methodology: Estimated alpha and beta coefficients for each stock relative to SPY ETF and QQQ ETF as market proxies. Tested statistical significance of alpha values to determine whether stocks exhibit excess returns beyond market trends.

Findings: Most stocks' alpha values were insignificant, indicating that they closely follow market movements, reinforcing the efficiency of diversification strategies.

#Constructing Portfolios with Tech Stocks

stocks Included: Amazon (AMZN), Apple (AAPL), Cisco (CSCO), Meta (META), Microsoft (MSFT), Moderna (MRNA), Netflix (NFLX), Starbucks (SBUX), Tesla (TSLA)

Goal: Construct portfolios with expected returns of 5%, 7.5%, 10%, 12.5%, and 15% using historical data (2020-2022, 2023-present) while considering a risk-free asset (10-year treasury bond). No short selling is allowed.

```
library(quadprog)
library(pracma)
```

```
## Warning: 程辑包'pracma'是用R版本4.3.3 来建造的
```

```
library(dplyr)
```

```
##
## 载入程辑包：'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(lubridate)
```

```
##
## 载入程辑包：'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(readr)
```

```
folder_path <- "C:/Users/lenovo/Desktop/STAT 417/hw 2/Data Files/"

all_files <- list.files(folder_path)
filtered_files <- all_files[!grepl("-", all_files)]
filtered_files <- all_files[-1]
print(filtered_files)
```

```
##  [1] "Amazon.csv"    "AMEX (1).csv"  "AMEX (2).csv"  "AMEX.csv"
##  [5] "Apple.csv"     "BAC (1).csv"   "BAC (2).csv"   "BAC.csv"
##  [9] "C (1).csv"     "C (2).csv"     "C.csv"         "Cisco.csv"
## [13] "DIA.csv"       "GS (1).csv"    "GS (2).csv"    "GS.csv"
## [17] "Meta.csv"      "Microsoft.csv" "Moderna.csv"   "MS (1).csv"
## [21] "MS (2).csv"    "MS.csv"        "Netflix.csv"   "QQQ.csv"
## [25] "SPY.csv"       "Starbucks.csv" "Tesla.csv"     "WFC (1).csv"
## [29] "WFC (2).csv"   "WFC.csv"
```

```
datasets <- lapply(filtered_files, function(x) {
  curr_data <- read.table(paste0(folder_path, x), sep = ",", header = TRUE)
  curr_data$Date <- as.Date(curr_data$Date, "%m/%d/%Y")
  curr_data <- curr_data[order(curr_data$Date), ]
  curr_data$Close.Last <- as.numeric(gsub("\\$", "", curr_data$Close.Last))
  curr_data
})
```

```
stock_names <- sub("\\..*", "", filtered_files)

returns_list <- lapply(datasets, function(df) {
  df$Return <- c(NA, diff(log(df$Close.Last)))
  df <- df[!is.na(df$Return), ]
  df
})
names(returns_list) <- stock_names

get_period_returns <- function(start_date, end_date) {
  returns_matrix <- do.call(cbind, lapply(returns_list, function(df) {
    subset(df, Date >= start_date & Date <= end_date)$Return
  }))
  colnames(returns_matrix) <- stock_names
  return(returns_matrix)
}

returns_2020_2022 <- get_period_returns("2020-01-01", "2022-12-31")
returns_2023_present <- get_period_returns("2023-01-01", Sys.Date())
```

```
## Warning in (function (..., deparse.level = 1) : number of rows of result is not
## a multiple of vector length (arg 1)
```

```
annualize <- function(returns) {
  mu <- colMeans(returns, na.rm = TRUE) * 252
  sigma <- cov(returns, use = "complete.obs") * 252
  list(mu = mu, sigma = sigma)
}

data_2020_2022 <- annualize(returns_2020_2022)
data_2023_present <- annualize(returns_2023_present)
```

Extract Risk-Free Rate

```
treasury_data <- read.csv("C:/Users/lenovo/Desktop/STAT 417/hw 2/10-year-treasury-bond-rate-yie
ld-chart.csv", header=FALSE)

treasury_data$Yield <- treasury_data$V2 / 100

treasury_data <- treasury_data %>%
  mutate(Date = parse_date_time(V1, orders = c("mdy", "ymd", "dmy", "Y/m/d")))

treasury_data <- treasury_data[order(treasury_data$Date), ]

rf_2020_2022 <- mean(subset(treasury_data, Date >= "2020-01-01" & Date <= "2022-12-31")$Yield,
na.rm = TRUE)
rf_2023_present <- mean(subset(treasury_data, Date >= "2023-01-01")$Yield, na.rm = TRUE)

cat("Computed Risk-Free Rate (2020-2022):", rf_2020_2022, "\n")
```

```
## Computed Risk-Free Rate (2020-2022): 0.01778188
```

```
cat("Computed Risk-Free Rate (2023-present):", rf_2023_present, "\n")
```

```
## Computed Risk-Free Rate (2023-present): 0.0409957
```

```
target_returns <- c(0.05, 0.075, 0.10, 0.125, 0.15)

compute_portfolio <- function(mu, sigma, target_returns, risk_free_rate) {
  n <- length(mu)
  sigma <- sigma + diag(1e-6, n)
  feasible_target_returns <- target_returns[target_returns >= min(mu) & target_returns <= max(m
u)]

  if (length(feasible_target_returns) == 0) {
    stop("Error: No feasible target returns within expected return range!")
  }

  Amat <- cbind(rep(1, n), mu, diag(1, n))
  weights_matrix <- matrix(0, nrow = length(feasible_target_returns), ncol = n)
  sharpe_ratios <- numeric(length(feasible_target_returns))
  sdP <- numeric(length(feasible_target_returns))

  for (i in 1:length(feasible_target_returns)) {
    bvec <- c(1, feasible_target_returns[i], rep(0, n))

    result <- tryCatch({
      solve.QP(Dmat = 2 * sigma, dvec = rep(0, n), Amat = Amat, bvec = bvec, meq = 2)
    }, error = function(e) return(NULL))

    if (!is.null(result)) {
      weights_matrix[i, ] <- result$solution
      sdP[i] <- sqrt(result$value)
      sharpe_ratios[i] <- (feasible_target_returns[i] - risk_free_rate) / sdP[i]  # Use dynamic
risk-free rate
    } else {
      cat("Skipping target return:", feasible_target_returns[i], "due to infeasibility.\n")
    }
  }

  return(list(weights = weights_matrix, sharpe = sharpe_ratios, sdP = sdP))
}

cat("Risk-Free Rate (2023-present):", rf_2023_present, "\n")
```

```
## Risk-Free Rate (2023-present): 0.0409957
```

```
cat("Expected Return Range (2023-present):", range(data_2023_present$mu), "\n")
```

```
## Expected Return Range (2023-present): -0.6542265 0.8723972
```

```
portfolio_2020_2022 <- compute_portfolio(data_2020_2022$mu, data_2020_2022$sigma, target_returns, rf_2020_2022)
portfolio_2023_present <- compute_portfolio(data_2023_present$mu, data_2023_present$sigma, target_returns, rf_2023_present)
```

```
cat("\nPortfolio Weights for Expected Returns (2020-2022):\n")
```

```
##
## Portfolio Weights for Expected Returns (2020-2022):
```

```
print(portfolio_2020_2022$weights)
```

```
##               [,1]          [,2]          [,3]          [,4]          [,5]
## [1,]  9.803751e-02 4.371014e-14 1.626757e-13 -2.427804e-13 5.030928e-17
## [2,]  7.247692e-02 4.284545e-14 1.192869e-13 -3.170531e-13 4.803071e-17
## [3,]  4.557743e-02 4.198404e-14 8.612166e-14 -3.665285e-13 4.521697e-17
## [4,]  2.307597e-03 4.088768e-14 1.471501e-13 -1.501562e-13 3.867838e-17
## [5,] -3.353185e-18 3.974992e-14 6.653940e-14 -3.882698e-13 3.487024e-03
##               [,6]          [,7]          [,8]          [,9]          [,10]
## [1,]  2.320093e-18  7.099728e-15  1.015010e-14 -4.644125e-15  2.741936e-13
## [2,] -1.897403e-17  3.410685e-15  8.914034e-15 -6.827309e-15 -1.889982e-14
## [3,] -4.014959e-17 -3.534861e-16  7.671908e-15 -8.969039e-15 -3.113092e-13
## [4,] -6.096203e-17 -4.839883e-15  6.002523e-15 -1.107461e-14 -5.681545e-13
## [5,] -1.036939e-16 -1.245976e-14 -8.039948e-16 -1.038553e-14 -7.676987e-13
##              [,11]        [,12]     [,13]         [,14]         [,15]
## [1,] 1.093857e-16 1.076183e-01 0.7489972 -2.958763e-17 -2.091791e-14
## [2,] 1.198822e-16 6.886665e-02 0.7918815 -2.141658e-17 -1.767346e-14
## [3,] 1.308829e-16 2.989674e-02 0.8244636 -1.363285e-17 -1.434806e-14
## [4,] 1.470685e-16 1.734723e-18 0.7514575 -8.190830e-18 -1.068448e-14
## [5,] 1.496698e-16 1.867307e-17 0.8446201  3.975190e-18  3.259467e-15
##              [,16]         [,17]        [,18]      [,19]         [,20]
## [1,]  3.635607e-14 -2.003679e-17 1.554872e-16 0.02238367 -2.739520e-15
## [2,]  4.058891e-14 -2.828811e-17 1.475297e-16 0.05175410  4.459214e-15
## [3,]  4.468913e-14 -3.657233e-17 1.401594e-16 0.08079168  1.152220e-14
## [4,]  4.698738e-14 -4.209908e-17 1.410923e-16 0.10750332  1.692212e-14
## [5,] -7.486842e-15 -4.544886e-17 1.381969e-16 0.13042004  2.549043e-14
##              [,21]         [,22]         [,23]        [,24]         [,25]
## [1,] -1.506258e-14 -1.472794e-14  2.296338e-02 1.202903e-15 -8.430969e-16
## [2,] -9.844981e-15 -1.178327e-14  1.502084e-02 9.154023e-16 -7.589274e-16
## [3,] -7.198242e-15 -1.301921e-14  6.871135e-03 6.322424e-16  1.239940e-02
## [4,] -3.025916e-14 -5.617730e-14  0.000000e+00 3.760750e-16  1.387316e-01
## [5,] -1.630543e-14 -3.687166e-14 -1.682866e-19 1.832546e-16  0.000000e+00
##              [,26]         [,27]         [,28]         [,29]         [,30]
## [1,] -6.570711e-17 -3.791095e-17  1.020638e-18 -7.508975e-15 -1.140328e-14
## [2,] -7.523259e-17 -2.693499e-17  4.329365e-20 -1.058172e-14 -1.223435e-14
## [3,] -8.411332e-17 -1.601134e-17 -5.771471e-19 -1.363489e-14 -1.314703e-14
## [4,] -8.872254e-17 -6.142165e-18  1.080410e-18 -1.657430e-14 -1.539051e-14
## [5,] -1.132165e-16  2.147286e-02  1.116212e-17  1.815501e-14 -2.110062e-14
```

```
cat("\nPortfolio Weights for Expected Returns (2023-present):\n")
```

```
##
## Portfolio Weights for Expected Returns (2023-present):
```

```
print(portfolio_2023_present$weights)
```

```
##               [,1]          [,2]          [,3]          [,4]          [,5]
## [1,] -5.269125e-18 -5.929694e-14 -9.390281e-14 -3.787886e-15 -6.291311e-17
## [2,] -1.745025e-18 -7.953287e-14 -5.037792e-14 -3.855618e-15  0.000000e+00
## [3,]  1.779075e-18 -9.976880e-14 -6.853020e-15 -3.923350e-15  0.000000e+00
## [4,]  3.642307e-18 -1.861345e-13 -3.976415e-14 -3.759293e-15  3.461245e-02
## [5,] -1.042329e-17  3.730153e-14 -5.611523e-14  5.643807e-15  6.304872e-02
##               [,6]          [,7]          [,8]          [,9]         [,10]
## [1,]  0.000000e+00 -5.686779e-15 -1.083501e-14  5.896705e-15  2.811418e-17
## [2,]  1.422878e-15 -1.269751e-15 -5.282432e-15  4.788966e-15  7.648456e-15
## [3,]  6.215681e-16 -9.752384e-16 -4.702782e-15  4.229023e-15  3.999571e-15
## [4,]  1.522139e-15 -1.520773e-15 -4.778229e-15  3.368960e-15  9.691482e-16
## [5,] -5.922358e-14 -1.196369e-15  1.236996e-13 -2.484974e-17 -9.248230e-16
##              [,11]      [,12]     [,13]         [,14]        [,15]
## [1,] -1.780934e-14 0.06263971 0.7980455 -3.392607e-13 7.722589e-15
## [2,] -9.812105e-15 0.06345656 0.8424957 -2.120634e-13 6.739125e-15
## [3,] -8.870993e-15 0.06427340 0.8869458 -8.486607e-14 5.755661e-15
## [4,] -9.185416e-15 0.06268854 0.8894437  1.886834e-15 5.231641e-15
## [5,]  1.957035e-16 0.05550163 0.8129354 -3.346784e-15 6.334415e-15
##              [,16]         [,17]         [,18]         [,19]         [,20]
## [1,] -3.434631e-14 1.856136e-17  1.927693e-17  8.017665e-02  5.137408e-15
## [2,] -3.210126e-14 2.802721e-17 -7.182922e-17  5.033917e-02  2.607834e-15
## [3,] -2.985622e-14 1.827024e-17  1.525469e-18  2.050168e-02  7.826041e-17
## [4,] -2.298739e-14 1.475184e-17  0.000000e+00 -2.933370e-18 -3.394146e-15
## [5,] -3.144726e-15 8.413318e-03  1.565889e-17  2.925204e-18  1.727854e-15
##              [,21]         [,22]         [,23]         [,24]         [,25]
## [1,] -1.990596e-15 9.391321e-18 -5.776411e-17 -1.378401e-16  1.772192e-16
## [2,] -1.567554e-15 1.787824e-17  7.842701e-18 -8.493215e-17 -2.897762e-17
## [3,] -1.144511e-15 2.636516e-17  1.691241e-18 -3.202419e-17 -2.534371e-17
## [4,] -2.522922e-15 2.876642e-17  3.836010e-19  1.065356e-17 -1.159328e-17
## [5,] -1.841505e-15 1.307852e-18  1.273827e-02  5.067679e-17  4.736264e-02
##             [,26]         [,27]         [,28]         [,29]         [,30]
## [1,] 5.913813e-02 -3.806500e-17  7.835969e-15  1.415379e-16  3.710824e-17
## [2,] 4.370861e-02 -3.586637e-17  8.372169e-15 -1.439137e-14  0.000000e+00
## [3,] 2.827910e-02 -3.366774e-17 -4.810655e-19 -8.922005e-15  4.521986e-15
## [4,] 1.325535e-02 -3.205215e-17  6.827025e-19 -5.811024e-15  3.446057e-15
## [5,] 2.168404e-19 -3.609358e-17 -7.631900e-16 -1.914984e-19 -9.689445e-16
```

Report the weights for the tangency portfolio and the corresponding Sharpe ratio for each of the two periods considered.

```
compute_tangency_portfolio <- function(mu, sigma, rf) {
  n <- length(mu)
  sigma <- sigma + diag(1e-6, n)
  excess_returns <- mu - rf
  inv_sigma <- solve(sigma)
  ones <- rep(1, n)
  w_tangency <- inv_sigma %*% excess_returns / sum(ones %*% inv_sigma %*% excess_returns)
  tangency_return <- sum(w_tangency * mu)
  tangency_risk <- sqrt(t(w_tangency) %*% sigma %*% w_tangency)
  sharpe_ratio <- (tangency_return - rf) / tangency_risk
  return(list(weights = w_tangency, expected_return = tangency_return,
              risk = tangency_risk, sharpe = sharpe_ratio))
}

tangency_2020_2022 <- compute_tangency_portfolio(data_2020_2022$mu, data_2020_2022$sigma, rf_20
20_2022)
tangency_2023_present <- compute_tangency_portfolio(data_2023_present$mu, data_2023_present$sig
ma, rf_2023_present)



cat("\nTangency Portfolio Weights (2020-2022):\n")
```

```
##
## Tangency Portfolio Weights (2020-2022):
```

```
print(tangency_2020_2022$weights)
```

```
##                 [,1]
## Amazon     -2.2851900
## AMEX (1)    0.6154518
## AMEX (2)    0.6154518
## AMEX        0.6154518
## Apple       8.2188099
## BAC (1)     0.5578121
## BAC (2)     0.5578121
## BAC         0.5578121
## C (1)      -4.2496743
## C (2)      -4.2496743
## C          -4.2496743
## Cisco      -1.9716487
## DIA       -10.5011867
## GS (1)      2.6658731
## GS (2)      2.6658731
## GS          2.6658731
## Meta       -2.6604978
## Microsoft   9.0066049
## Moderna     1.9645373
## MS (1)      2.3821877
## MS (2)      2.3821877
## MS          2.3821877
## Netflix     0.5915457
## QQQ       -27.8497082
## SPY        19.6070674
## Starbucks   0.8778592
## Tesla       2.7113541
## WFC (1)    -0.8748329
## WFC (2)    -0.8748329
## WFC        -0.8748329
```

```
cat("\nExpected Return of Tangency Portfolio (2020-2022):", tangency_2020_2022$expected_return, "\n")
```

```
##
## Expected Return of Tangency Portfolio (2020-2022): 9.697168
```

```
cat("Risk (Standard Deviation) of Tangency Portfolio (2020-2022):", tangency_2020_2022$risk, "\n")
```

```
## Risk (Standard Deviation) of Tangency Portfolio (2020-2022): 4.278983
```

```
cat("Sharpe Ratio of Tangency Portfolio (2020-2022):", tangency_2020_2022$sharpe, "\n")
```

```
## Sharpe Ratio of Tangency Portfolio (2020-2022): 2.262076
```

```
cat("\nTangency Portfolio Weights (2023-present):\n")
```

```
##
## Tangency Portfolio Weights (2023-present):
```

```
print(tangency_2023_present$weights)
```

```
##                 [,1]
## Amazon     0.50636515
## AMEX (1)   0.49443537
## AMEX (2)   0.49443537
## AMEX       0.49443537
## Apple      1.90600642
## BAC (1)   -0.71169006
## BAC (2)   -0.71169006
## BAC       -0.71169006
## C (1)      0.43631877
## C (2)      0.43631877
## C          0.43631877
## Cisco      0.51156173
## DIA       -6.12824419
## GS (1)     0.70178229
## GS (2)     0.70178229
## GS         0.70178229
## Meta       2.15282742
## Microsoft -1.61713231
## Moderna   -1.42929888
## MS (1)    -0.05190060
## MS (2)    -0.05190060
## MS        -0.05190060
## Netflix    1.28355668
## QQQ        2.67203369
## SPY       -2.29294652
## Starbucks -0.55775244
## Tesla     -0.01175795
## WFC (1)    0.46598129
## WFC (2)    0.46598129
## WFC        0.46598129
```

```
cat("\nExpected Return of Tangency Portfolio (2023-present):", tangency_2023_present$expected_return, "\n")
```

```
##
## Expected Return of Tangency Portfolio (2023-present): 5.369009
```

```
cat("Risk (Standard Deviation) of Tangency Portfolio (2023-present):", tangency_2023_present$risk, "\n")
```

```
## Risk (Standard Deviation) of Tangency Portfolio (2023-present): 1.547302
```

```
cat("Sharpe Ratio of Tangency Portfolio (2023-present):", tangency_2023_present$sharpe, "\n")
```

```
## Sharpe Ratio of Tangency Portfolio (2023-present): 3.443421
```

Minimum Variance Portfolio and its Sharpe Ratio

```
compute_minimum_variance_portfolio <- function(mu, sigma, rf) {
  n <- length(mu)
  sigma <- sigma + diag(1e-6, n)
  inv_sigma <- solve(sigma)
  ones <- rep(1, n)
  w_mvp <- inv_sigma %*% ones / sum(ones %*% inv_sigma %*% ones)
  mvp_return <- sum(w_mvp * mu)
  mvp_risk <- sqrt(t(w_mvp) %*% sigma %*% w_mvp)
  sharpe_ratio <- (mvp_return - rf) / mvp_risk
  return(list(weights = w_mvp, expected_return = mvp_return,
              risk = mvp_risk, sharpe = sharpe_ratio))
}

# Define the actual risk-free rates
rf_2020_2022 <- 0.015  # Example: 1.5% for 2020-2022 (Replace with real data)
rf_2023_present <- 0.045  # Example: 4.5% for 2023-present (Replace with real data)

# Compute Minimum Variance Portfolio for both periods
mvp_2020_2022 <- compute_minimum_variance_portfolio(data_2020_2022$mu, data_2020_2022$sigma, rf
_2020_2022)
mvp_2023_present <- compute_minimum_variance_portfolio(data_2023_present$mu, data_2023_present
$sigma, rf_2023_present)

# 2020-2022
cat("\nMinimum Variance Portfolio Weights (2020-2022):\n")
```

```
##
## Minimum Variance Portfolio Weights (2020-2022):
```

```
print(mvp_2020_2022$weights)
```

```
##                   [,1]
## Amazon     0.115569274
## AMEX (1)  -0.072302549
## AMEX (2)  -0.072302549
## AMEX      -0.072302549
## Apple     -0.167359086
## BAC (1)   -0.005843004
## BAC (2)   -0.005843004
## BAC       -0.005843004
## C (1)     -0.053171129
## C (2)     -0.053171129
## C         -0.053171129
## Cisco      0.024884956
## DIA        0.743693151
## GS (1)     0.026402145
## GS (2)     0.026402145
## GS         0.026402145
## Meta      -0.020075735
## Microsoft -0.254099065
## Moderna    0.031920845
## MS (1)    -0.071940566
## MS (2)    -0.071940566
## MS        -0.071940566
## Netflix    0.021946352
## QQQ       -0.629446894
## SPY        1.770030994
## Starbucks -0.034251023
## Tesla     -0.011227229
## WFC (1)   -0.020340411
## WFC (2)   -0.020340411
## WFC       -0.020340411
```

```
cat("\nExpected Return of MVP (2020-2022):", mvp_2020_2022$expected_return, "\n")
```

```
##
## Expected Return of MVP (2020-2022): 0.03782996
```

```
cat("Risk (Standard Deviation) of MVP (2020-2022):", mvp_2020_2022$risk, "\n")
```

```
## Risk (Standard Deviation) of MVP (2020-2022): 0.1947391
```

```
cat("Sharpe Ratio of MVP (2020-2022):", mvp_2020_2022$sharpe, "\n")
```

```
## Sharpe Ratio of MVP (2020-2022): 0.1172336
```

```
#2023-present
cat("\nMinimum Variance Portfolio Weights (2023-present):\n")
```

```
##
## Minimum Variance Portfolio Weights (2023-present):
```

```
print(mvp_2023_present$weights)
```

```
##                   [,1]
## Amazon    -0.039651023
## AMEX (1)  -0.014336705
## AMEX (2)  -0.014336705
## AMEX      -0.014336705
## Apple      0.017179318
## BAC (1)   -0.022530565
## BAC (2)   -0.022530565
## BAC       -0.022530565
## C (1)     -0.010375923
## C (2)     -0.010375923
## C         -0.010375923
## Cisco      0.053110831
## DIA        0.529101591
## GS (1)    -0.029964939
## GS (2)    -0.029964939
## GS        -0.029964939
## Meta      -0.004689751
## Microsoft  0.049817502
## Moderna   -0.004383824
## MS (1)    -0.014054553
## MS (2)    -0.014054553
## MS        -0.014054553
## Netflix    0.002461878
## QQQ       -0.603102007
## SPY        1.239009283
## Starbucks  0.001091197
## Tesla     -0.020119168
## WFC (1)    0.017987410
## WFC (2)    0.017987410
## WFC        0.017987410
```

```
cat("\nExpected Return of MVP (2023-present):", mvp_2023_present$expected_return, "\n")
```

```
##
## Expected Return of MVP (2023-present): 0.06500778
```

```
cat("Risk (Standard Deviation) of MVP (2023-present):", mvp_2023_present$risk, "\n")
```

```
## Risk (Standard Deviation) of MVP (2023-present): 0.1038742
```

```
cat("Sharpe Ratio of MVP (2023-present):", mvp_2023_present$sharpe, "\n")
```

```
## Sharpe Ratio of MVP (2023-present): 0.1926156
```

Comments: 1. The Tangency Portfolio has higher expected returns, but comes with higher risk. 2. The Tangency Portfolio has the highest Sharpe Ratio, meaning it provides the best risk-adjusted return. 3. The expected return of both portfolios increased in the 2023-present period.

#Expanding the Portfolio to Include Financial Stocks Additional Stocks Included: American Express (AXP), Bank of America (BAC), Citigroup (C), Goldman Sachs (GS), Morgan Stanley (MS), Wells Fargo (WFC)

Goal: Compare portfolio structures when financial stocks are included.

How does the portfolio change when banking & financial stocks are included?

```
# List of selected stocks (Tech + Banking)
selected_stocks <- c("AMZN", "AAPL", "CSCO", "META", "MSFT", "MRNA", "NFLX", "SBUX", "TSLA","AX
P", "BAC", "C", "GS", "MS", "WFC")

folder_path <- "C:/Users/lenovo/Desktop/STAT 417/hw 2/Data Files/"

all_files <- list.files(folder_path, full.names = TRUE)

filtered_files <- all_files[sapply(selected_stocks, function(stock) any(grepl(stock, all_file
s)))]

cat("Filtered Files:\n")
```

```
## Filtered Files:
```

```
print(filtered_files)
```

```
##  [1] "C:/Users/lenovo/Desktop/STAT 417/hw 2/Data Files/C (2).csv"
##  [2] "C:/Users/lenovo/Desktop/STAT 417/hw 2/Data Files/C.csv"
##  [3] "C:/Users/lenovo/Desktop/STAT 417/hw 2/Data Files/Cisco.csv"
##  [4] "C:/Users/lenovo/Desktop/STAT 417/hw 2/Data Files/DIA.csv"
##  [5] "C:/Users/lenovo/Desktop/STAT 417/hw 2/Data Files/GS (1).csv"
##  [6] "C:/Users/lenovo/Desktop/STAT 417/hw 2/Data Files/SPY.csv"
##  [7] "C:/Users/lenovo/Desktop/STAT 417/hw 2/Data Files/Starbucks.csv"
##  [8] "C:/Users/lenovo/Desktop/STAT 417/hw 2/Data Files/Tesla.csv"
##  [9] "C:/Users/lenovo/Desktop/STAT 417/hw 2/Data Files/WFC (1).csv"
## [10] "C:/Users/lenovo/Desktop/STAT 417/hw 2/Data Files/WFC (2).csv"
```

```
if (length(filtered_files) == 0) {
  stop("Error: No matching stock files found. Check filenames or folder path!")
}

read_stock_data <- function(file) {
  tryCatch({
    curr_data <- read.csv(file, header = TRUE)
    if (!"Date" %in% names(curr_data) | !"Close.Last" %in% names(curr_data)) {
      stop(paste("Error: File", file, "does not have required columns (Date, Close.Last)"))
    }

    curr_data$Date <- as.Date(curr_data$Date, tryFormats = c("%m/%d/%Y", "%Y-%m-%d"))  # Standa
rdize date format
    curr_data <- curr_data[order(curr_data$Date), ]
    curr_data$Close.Last <- as.numeric(gsub("\\$", "", curr_data$Close.Last))  # Convert price
to numeric

    curr_data$Return <- c(NA, diff(log(curr_data$Close.Last)))
    curr_data <- curr_data[!is.na(curr_data$Return), ]  # Remove NA values

    return(curr_data)
  }, error = function(e) {
    cat("Error reading file:", file, "\n", e$message, "\n")
    return(NULL)
  })
}

datasets <- lapply(filtered_files, read_stock_data)
datasets <- datasets[!sapply(datasets, is.null)]

stock_names <- selected_stocks[1:length(datasets)]
names(datasets) <- stock_names

get_period_returns <- function(start_date, end_date) {
  returns_matrix <- do.call(cbind, lapply(datasets, function(df) {
    subset(df, Date >= start_date & Date <= end_date)$Return
  }))

  if (!is.null(returns_matrix)) {
    colnames(returns_matrix) <- names(datasets)
  }

  return(returns_matrix)
}

returns_2020_2022 <- get_period_returns("2020-01-01", "2022-12-31")
returns_2023_present <- get_period_returns("2023-01-01", Sys.Date())
```

```
## Warning in (function (..., deparse.level = 1) : number of rows of result is not
## a multiple of vector length (arg 3)
```

```
annualize <- function(returns) {
  mu <- colMeans(returns, na.rm = TRUE) * 252
  sigma <- cov(returns, use = "complete.obs") * 252
  list(mu = mu, sigma = sigma)
}

data_2020_2022 <- annualize(returns_2020_2022)
data_2023_present <- annualize(returns_2023_present)

cat("\nAnnualized Statistics Computed!\n")
```

```
##
## Annualized Statistics Computed!
```

```
cat("2020-2022 Expected Returns:\n", data_2020_2022$mu, "\n")
```

```
## 2020-2022 Expected Returns:
##  -0.18963 -0.18963 -0.002231529 0.05009162 0.1336876 0.05747632 0.0402369 0.4951417 -0.08821
771 -0.08821771
```

```
cat("2023-Present Expected Returns:\n", data_2023_present$mu, "\n")
```

```
## 2023-Present Expected Returns:
##  0.2838539 0.2838539 0.09969214 0.1207116 0.3008819 0.2239129 -0.002102645 0.6367633 0.31200
34 0.3120034
```

```
tangency_2020_2022_new <- compute_tangency_portfolio(data_2020_2022$mu, data_2020_2022$sigma, r
f_2020_2022)
tangency_2023_present_new <- compute_tangency_portfolio(data_2023_present$mu, data_2023_present
$sigma, rf_2023_present)

mvp_2020_2022_new <- compute_minimum_variance_portfolio(data_2020_2022$mu, data_2020_2022$sigm
a, rf_2020_2022)
mvp_2023_present_new <- compute_minimum_variance_portfolio(data_2023_present$mu, data_2023_pres
ent$sigma, rf_2023_present)
```

```
cat("\nNew Tangency Portfolio Weights (2020-2022):\n")
```

```
##
## New Tangency Portfolio Weights (2020-2022):
```

```
print(tangency_2020_2022_new$weights)
```

```
##            [,1]
## AMZN -1.8892976
## AAPL -1.8892976
## CSCO -0.4118513
## META  3.9601533
## MSFT  4.0721599
## MRNA -2.9285995
## NFLX -0.1612813
## SBUX  0.6925332
## TSLA -0.2222595
## AXP  -0.2222595
```

```
cat("\nExpected Return of New Tangency Portfolio (2020-2022):", tangency_2020_2022_new$expected_return, "\n")
```

```
##
## Expected Return of New Tangency Portfolio (2020-2022): 1.667524
```

```
cat("Risk (Standard Deviation) of New Tangency Portfolio (2020-2022):", tangency_2020_2022_new$risk, "\n")
```

```
## Risk (Standard Deviation) of New Tangency Portfolio (2020-2022): 1.040167
```

```
cat("Sharpe Ratio of New Tangency Portfolio (2020-2022):", tangency_2020_2022_new$sharpe, "\n")
```

```
## Sharpe Ratio of New Tangency Portfolio (2020-2022): 1.58871
```

```
cat("\nNew Tangency Portfolio Weights (2023-present):\n")
```

```
##
## New Tangency Portfolio Weights (2023-present):
```

```
print(tangency_2023_present_new$weights)
```

```
##              [,1]
## AMZN -0.001426797
## AAPL -0.001426797
## CSCO -0.148813642
## META -7.257032093
## MSFT  0.988947446
## MRNA  7.300920288
## NFLX -0.515808435
## SBUX  0.013222447
## TSLA  0.310708791
## AXP   0.310708791
```

```
cat("\nExpected Return of New Tangency Portfolio (2023-present):", tangency_2023_present_new$expected_return, "\n")
```

```
##
## Expected Return of New Tangency Portfolio (2023-present): 1.244062
```

```
cat("Risk (Standard Deviation) of New Tangency Portfolio (2023-present):", tangency_2023_present_new$risk, "\n")
```

```
## Risk (Standard Deviation) of New Tangency Portfolio (2023-present): 0.5710597
```

```
cat("Sharpe Ratio of New Tangency Portfolio (2023-present):", tangency_2023_present_new$sharpe, "\n")
```

```
## Sharpe Ratio of New Tangency Portfolio (2023-present): 2.099714
```

```
cat("\nNew Minimum Variance Portfolio Weights (2020-2022):\n")
```

```
##
## New Minimum Variance Portfolio Weights (2020-2022):
```

```
print(mvp_2020_2022_new$weights)
```

```
##              [,1]
## AMZN -0.124882598
## AAPL -0.124882598
## CSCO  0.006800063
## META  1.115777322
## MSFT -0.055203721
## MRNA  0.370235992
## NFLX -0.078767067
## SBUX -0.051138222
## TSLA -0.028969585
## AXP  -0.028969585
```

```
cat("\nExpected Return of New MVP (2020-2022):", mvp_2020_2022_new$expected_return, "\n")
```

```
##
## Expected Return of New MVP (2020-2022): 0.0937599
```

```
cat("Risk (Standard Deviation) of New MVP (2020-2022):", mvp_2020_2022_new$risk, "\n")
```

```
## Risk (Standard Deviation) of New MVP (2020-2022): 0.2270815
```

```
cat("Sharpe Ratio of New MVP (2020-2022):", mvp_2020_2022_new$sharpe, "\n")
```

```
## Sharpe Ratio of New MVP (2020-2022): 0.3468353
```

```
cat("\nNew Minimum Variance Portfolio Weights (2023-present):\n")
```

```
##
## New Minimum Variance Portfolio Weights (2023-present):
```

```
print(mvp_2023_present_new$weights)
```

```
##             [,1]
## AMZN -0.02137236
## AAPL -0.02137236
## CSCO  0.03588893
## META  0.99838932
## MSFT -0.14299413
## MRNA  0.16446198
## NFLX  0.00282240
## SBUX -0.03813414
## TSLA  0.01115517
## AXP   0.01115517
```

```
cat("\nExpected Return of New MVP (2023-present):", mvp_2023_present_new$expected_return, "\n")
```

```
##
## Expected Return of New MVP (2023-present): 0.08843514
```

```
cat("Risk (Standard Deviation) of New MVP (2023-present):", mvp_2023_present_new$risk, "\n")
```

```
## Risk (Standard Deviation) of New MVP (2023-present): 0.1086879
```

```
cat("Sharpe Ratio of New MVP (2023-present):", mvp_2023_present_new$sharpe, "\n")
```

```
## Sharpe Ratio of New MVP (2023-present): 0.3996317
```

Comments: Banking stocks added diversification, reducing portfolio volatility. Risk decreased. Technology Stock Weights Decreased. Banking Stocks Took Significant Portfolio Weight.

#CAPM Analysis - Alpha and Beta Calculation

Goal: Compute stock alphas and betas using the SPY ETF and QQQ ETF as market benchmarks.

Estimating $\alpha$and β for 9 Stocks Using SPY ETF as Market Portfolio.

```
compute_alpha_beta <- function(stock_returns, market_returns) {
  results <- data.frame(Stock = character(), Alpha = numeric(), Beta = numeric(), stringsAsFact
ors = FALSE)

  for (stock in colnames(stock_returns)) {
    stock_data <- stock_returns[, stock]
    market_data <- market_returns

    valid_data <- complete.cases(stock_data, market_data)

    if (sum(valid_data) > 10) {
      model <- lm(stock_data[valid_data] ~ market_data[valid_data])
      alpha <- coef(model)[1]  # Intercept (alpha)
      beta <- coef(model)[2]   # Slope (beta)

      results <- rbind(results, data.frame(Stock = stock, Alpha = alpha, Beta = beta))
    } else {
      cat("Skipping stock:", stock, "due to insufficient data.\n")
    }
  }

  return(results)
}

spy_data <- read.csv("C:/Users/lenovo/Desktop/STAT 417/hw 2/SPY.csv")
spy_data$Date <- as.Date(spy_data$Date, "%m/%d/%Y")
spy_data <- spy_data[order(spy_data$Date), ]
spy_data$Close.Last <- as.numeric(gsub("\\$", "", spy_data$Close.Last))
spy_data$Return <- c(NA, diff(log(spy_data$Close.Last)))

market_2020_2022 <- subset(spy_data, Date >= "2020-01-01" & Date <= "2022-12-31")$Return
market_2023_present <- subset(spy_data, Date >= "2023-01-01")$Return

market_2020_2022 <- market_2020_2022[1:nrow(returns_2020_2022)]
market_2023_present <- market_2023_present[1:nrow(returns_2023_present)]

alpha_beta_2020_2022 <- compute_alpha_beta(returns_2020_2022, market_2020_2022)
alpha_beta_2023_present <- compute_alpha_beta(returns_2023_present, market_2023_present)

cat("\nAlpha & Beta for 2020-2022:\n")
```

```
##
## Alpha & Beta for 2020-2022:
```

```
print(alpha_beta_2020_2022)
```

```
##              Stock         Alpha      Beta
## (Intercept)   AMZN -1.061267e-03 1.3537607
## (Intercept)1  AAPL -1.061267e-03 1.3537607
## (Intercept)2  CSCO -2.218049e-04 0.9336593
## (Intercept)3  META -2.092091e-05 0.9632434
## (Intercept)4  MSFT  2.645707e-04 1.1659718
## (Intercept)5  MRNA -9.463692e-20 1.0000000
## (Intercept)6  NFLX -9.358418e-05 1.1103723
## (Intercept)7  SBUX  1.623259e-03 1.4976668
## (Intercept)8  TSLA -6.343123e-04 1.2462348
## (Intercept)9   AXP -6.343123e-04 1.2462348
```

```
cat("\nAlpha & Beta for 2023-present:\n")
```

```
##
## Alpha & Beta for 2023-present:
```

```
print(alpha_beta_2023_present)
```

```
##              Stock         Alpha      Beta
## (Intercept)   AMZN -4.627049e-06 1.0995883
## (Intercept)1  AAPL -4.627049e-06 1.0995883
## (Intercept)2  CSCO -1.646767e-04 0.7023365
## (Intercept)3  META -1.601638e-04 0.7593723
## (Intercept)4  MSFT  5.592530e-05 1.1039043
## (Intercept)5  MRNA -3.078636e-19 1.0000000
## (Intercept)6  NFLX -8.180984e-04 0.8217975
## (Intercept)7  SBUX  4.011568e-04 2.3022548
## (Intercept)8  TSLA  2.393090e-04 0.9722218
## (Intercept)9   AXP  2.393090e-04 0.9722218
```

Test the Significance ofα

```
compute_alpha_beta_significance <- function(stock_returns, market_returns) {
  results <- data.frame(Stock = character(), Alpha = numeric(), Beta = numeric(), Alpha_pval =
numeric(), stringsAsFactors = FALSE)
  cat("\nChecking Data Dimensions:\n")
  cat("Stock Returns:", dim(stock_returns), "\n")
  cat("Market Returns:", length(market_returns), "\n")

  for (stock in colnames(stock_returns)) {
    stock_data <- stock_returns[, stock]
    market_data <- market_returns
    min_length <- min(length(stock_data), length(market_data))
    stock_data <- stock_data[1:min_length]
    market_data <- market_data[1:min_length]
    valid_data <- complete.cases(stock_data, market_data)

    if (sum(valid_data) > 10) {
      model <- tryCatch(
        lm(stock_data[valid_data] ~ market_data[valid_data]),
        error = function(e) return(NULL)
      )

      if (!is.null(model)) {
        alpha <- coef(model)[1]  # Intercept (alpha)
        beta <- coef(model)[2]   # Slope (beta)
        alpha_pval <- summary(model)$coefficients[1, 4]  # P-value for alpha

        results <- rbind(results, data.frame(Stock = stock, Alpha = alpha, Beta = beta, Alpha_p
val = alpha_pval))
      } else {
        cat("Skipping stock:", stock, "due to regression failure.\n")
      }
    } else {
      cat("Skipping stock:", stock, "due to insufficient valid data points.\n")
    }
  }

  return(results)
}

cat("\nChecking SPY ETF Data:\n")
```

```
##
## Checking SPY ETF Data:
```

```
print(head(spy_data))
```

```
##            Date Close.Last    Volume   Open   High    Low       Return
## 2515 2015-01-13   202.0800 214341100 204.12 205.48 200.51           NA
## 2514 2015-01-14   200.8600 192551700 199.65 201.10 198.57 -0.006055511
## 2513 2015-01-15   199.0199 175383200 201.63 202.01 198.88 -0.009203328
## 2512 2015-01-16   201.6300 211677200 198.77 201.82 198.55  0.013029515
## 2511 2015-01-20   202.0550 130756800 202.40 202.72 200.17  0.002105603
## 2510 2015-01-21   203.0800 122729500 201.50 203.66 200.94  0.005060053
```

```
market_2020_2022 <- subset(spy_data, Date >= "2020-01-01" & Date <= "2022-12-31")$Return
market_2023_present <- subset(spy_data, Date >= "2023-01-01")$Return

market_2020_2022 <- market_2020_2022[1:nrow(returns_2020_2022)]
market_2023_present <- market_2023_present[1:nrow(returns_2023_present)]


alpha_beta_significance_2020_2022 <- compute_alpha_beta_significance(returns_2020_2022, market_
2020_2022)
```

```
##
## Checking Data Dimensions:
## Stock Returns: 756 10
## Market Returns: 756
```

```
## Warning in summary.lm(model): essentially perfect fit: summary may be
## unreliable
```

```
alpha_beta_significance_2023_present <- compute_alpha_beta_significance(returns_2023_present, m
arket_2023_present)
```

```
##
## Checking Data Dimensions:
## Stock Returns: 522 10
## Market Returns: 522
```

```
## Warning in summary.lm(model): essentially perfect fit: summary may be
## unreliable
```

```
cat("\nAlpha, Beta, and Alpha Significance for 2020-2022:\n")
```

```
##
## Alpha, Beta, and Alpha Significance for 2020-2022:
```

```
print(alpha_beta_significance_2020_2022)
```

```
##              Stock         Alpha      Beta Alpha_pval
## (Intercept)   AMZN -1.061267e-03 1.3537607  0.1371037
## (Intercept)1  AAPL -1.061267e-03 1.3537607  0.1371037
## (Intercept)2  CSCO -2.218049e-04 0.9336593  0.6491785
## (Intercept)3  META -2.092091e-05 0.9632434  0.8928224
## (Intercept)4  MSFT  2.645707e-04 1.1659718  0.6230129
## (Intercept)5  MRNA -9.463692e-20 1.0000000  0.6327255
## (Intercept)6  NFLX -9.358418e-05 1.1103723  0.8560191
## (Intercept)7  SBUX  1.623259e-03 1.4976668  0.2502850
## (Intercept)8  TSLA -6.343123e-04 1.2462348  0.3916088
## (Intercept)9   AXP -6.343123e-04 1.2462348  0.3916088
```

```
cat("\nAlpha, Beta, and Alpha Significance for 2023-present:\n")
```

```
##
## Alpha, Beta, and Alpha Significance for 2023-present:
```

```
print(alpha_beta_significance_2023_present)
```

```
##              Stock         Alpha      Beta Alpha_pval
## (Intercept)   AMZN -4.627049e-06 1.0995883 0.99395648
## (Intercept)1  AAPL -4.627049e-06 1.0995883 0.99395648
## (Intercept)2  CSCO -1.646767e-04 0.7023365 0.73079230
## (Intercept)3  META -1.601638e-04 0.7593723 0.32841518
## (Intercept)4  MSFT  5.592530e-05 1.1039043 0.92206854
## (Intercept)5  MRNA -3.078636e-19 1.0000000 0.09126755
## (Intercept)6  NFLX -8.180984e-04 0.8217975 0.30219398
## (Intercept)7  SBUX  4.011568e-04 2.3022548 0.77727815
## (Intercept)8  TSLA  2.393090e-04 0.9722218 0.73404346
## (Intercept)9   AXP  2.393090e-04 0.9722218 0.73404346
```

```
cat("\nSignificant Alphas (2020-2022):\n")
```

```
##
## Significant Alphas (2020-2022):
```

```
print(subset(alpha_beta_significance_2020_2022, Alpha_pval < 0.05))
```

```
## [1] Stock      Alpha      Beta       Alpha_pval
## <0 行> (或0-长度的row.names)
```

```
cat("\nSignificant Alphas (2023-present):\n")
```

```
##
## Significant Alphas (2023-present):
```

```
print(subset(alpha_beta_significance_2023_present, Alpha_pval < 0.05))
```

```
## [1] Stock      Alpha      Beta       Alpha_pval
## <0 行> (或0-长度的row.names)
```

#If QQQ ETF is considered as the market portfolio.

```
compute_alpha_beta_significance <- function(stock_returns, market_returns) {
  results <- data.frame(Stock = character(), Alpha = numeric(), Beta = numeric(), Alpha_pval =
numeric(), stringsAsFactors = FALSE)

  # Print dimensions to check data alignment
  cat("\nChecking Data Dimensions:\n")
  cat("Stock Returns:", dim(stock_returns), "\n")
  cat("Market Returns:", length(market_returns), "\n")

  for (stock in colnames(stock_returns)) {
    stock_data <- stock_returns[, stock]
    market_data <- market_returns

    # Ensure stock and market data have the same length
    min_length <- min(length(stock_data), length(market_data))
    stock_data <- stock_data[1:min_length]
    market_data <- market_data[1:min_length]

    # Remove NA values
    valid_data <- complete.cases(stock_data, market_data)

    if (sum(valid_data) > 10) {  # Ensure sufficient data points
      model <- tryCatch(
        lm(stock_data[valid_data] ~ market_data[valid_data]),
        error = function(e) return(NULL)
      )

      if (!is.null(model)) {
        alpha <- coef(model)[1]  # Intercept (alpha)
        beta <- coef(model)[2]   # Slope (beta)
        alpha_pval <- summary(model)$coefficients[1, 4]  # P-value for alpha

        results <- rbind(results, data.frame(Stock = stock, Alpha = alpha, Beta = beta, Alpha_p
val = alpha_pval))
      } else {
        cat("Skipping stock:", stock, "due to regression failure.\n")
      }
    } else {
      cat("Skipping stock:", stock, "due to insufficient valid data points.\n")
    }
  }

  return(results)
}


qqq_data <- read.csv("C:/Users/lenovo/Desktop/STAT 417/hw 2/QQQ.csv")  # Update path if needed
qqq_data$Date <- as.Date(qqq_data$Date, "%m/%d/%Y")
qqq_data <- qqq_data[order(qqq_data$Date), ]
qqq_data$Close.Last <- as.numeric(gsub("\\$", "", qqq_data$Close.Last))
qqq_data$Return <- c(NA, diff(log(qqq_data$Close.Last)))  # Compute daily log returns


cat("\nChecking QQQ ETF Data:\n")
```

```
##
## Checking QQQ ETF Data:
```

```
print(head(qqq_data))
```

```
##            Date Close.Last   Volume   Open     High      Low       Return
## 2515 2015-01-13     101.52 56135260 102.54 103.6200 100.7000           NA
## 2514 2015-01-14     100.96 51233350 100.50 101.4829 100.0700 -0.005531425
## 2513 2015-01-15      99.65 50854600 101.43 101.5900  99.5250 -0.013060352
## 2512 2015-01-16     100.82 35196110  99.52 100.9500  99.3600  0.011672702
## 2511 2015-01-20     101.62 30848640 101.43 101.8800 100.2900  0.007903618
## 2510 2015-01-21     102.14 40022140 101.32 102.6190 100.9648  0.005104055
```

```
market_qqq_2020_2022 <- subset(qqq_data, Date >= "2020-01-01" & Date <= "2022-12-31")$Return
market_qqq_2023_present <- subset(qqq_data, Date >= "2023-01-01")$Return


market_qqq_2020_2022 <- market_qqq_2020_2022[1:nrow(returns_2020_2022)]
market_qqq_2023_present <- market_qqq_2023_present[1:nrow(returns_2023_present)]

alpha_beta_qqq_2020_2022 <- compute_alpha_beta_significance(returns_2020_2022, market_qqq_2020_
2022)
```

```
##
## Checking Data Dimensions:
## Stock Returns: 756 10
## Market Returns: 756
```

```
alpha_beta_qqq_2023_present <- compute_alpha_beta_significance(returns_2023_present, market_qqq
_2023_present)
```

```
##
## Checking Data Dimensions:
## Stock Returns: 522 10
## Market Returns: 522
```

```
cat("\nAlpha, Beta, and Alpha Significance for 2020-2022 (QQQ):\n")
```

```
##
## Alpha, Beta, and Alpha Significance for 2020-2022 (QQQ):
```

```
print(alpha_beta_qqq_2020_2022)
```

```
##              Stock         Alpha      Beta Alpha_pval
## (Intercept)   AMZN -1.008005e-03 0.8581590  0.2529547
## (Intercept)1  AAPL -1.008005e-03 0.8581590  0.2529547
## (Intercept)2  CSCO -2.219439e-04 0.7156949  0.6800155
## (Intercept)3  META -8.969312e-06 0.6977495  0.9780749
## (Intercept)4  MSFT  2.972021e-04 0.7835928  0.6601328
## (Intercept)5  MRNA -6.493925e-06 0.7878593  0.9752972
## (Intercept)6  NFLX -9.535037e-05 0.8565309  0.8691570
## (Intercept)7  SBUX  1.514990e-03 1.5109254  0.2419400
## (Intercept)8  TSLA -5.800371e-04 0.7723836  0.5135090
## (Intercept)9   AXP -5.800371e-04 0.7723836  0.5135090
```

```
cat("\nAlpha, Beta, and Alpha Significance for 2023-present (QQQ):\n")
```

```
##
## Alpha, Beta, and Alpha Significance for 2023-present (QQQ):
```

```
print(alpha_beta_qqq_2023_present)
```

```
##              Stock         Alpha      Beta Alpha_pval
## (Intercept)   AMZN  2.103367e-04 0.5427068  0.7561502
## (Intercept)1  AAPL  2.103367e-04 0.5427068  0.7561502
## (Intercept)2  CSCO -1.353896e-04 0.4318020  0.7853797
## (Intercept)3  META -7.214315e-05 0.4224372  0.7636603
## (Intercept)4  MSFT  2.576547e-04 0.5559363  0.6873464
## (Intercept)5  MRNA -2.530102e-05 0.6676315  0.8517313
## (Intercept)6  NFLX -7.753746e-04 0.4985813  0.3383825
## (Intercept)7  SBUX -1.058045e-05 1.8157501  0.9938331
## (Intercept)8  TSLA  4.854495e-04 0.4356336  0.5216963
## (Intercept)9   AXP  4.854495e-04 0.4356336  0.5216963
```

```
cat("\nSignificant Alphas (2020-2022) with QQQ:\n")
```

```
##
## Significant Alphas (2020-2022) with QQQ:
```

```
print(subset(alpha_beta_qqq_2020_2022, Alpha_pval < 0.05))
```

```
## [1] Stock      Alpha      Beta       Alpha_pval
## <0 行> (或0-长度的row.names)
```

```
cat("\nSignificant Alphas (2023-present) with QQQ:\n")
```

```
##
## Significant Alphas (2023-present) with QQQ:
```

```
print(subset(alpha_beta_qqq_2023_present, Alpha_pval < 0.05))
```

```
## [1] Stock      Alpha      Beta       Alpha_pval
## <0 行> (或0-长度的row.names)
```

Comments: Tech-heavy stocks (AMZN, AAPL, META, MSFT) have higher β values when using QQQ. Stocks Are More Volatile with QQQ Than SPY. The choice of market portfolio affects β values more than α.