Untitled

Ivy Liu

2025-07-28

```r
# Portfolio Optimization: A Monte Carlo Study

# Prepared by Tom Miller, June 20, 2025

# Monte Carlo Study Showing Portfolio Opportunity Sets

# Generate above 700 random asset allocation sets: 4-by-1 vectors.
# For each set, the four portfolio asset weights must sum to 1.
# So, we generate three and set the fourth so the sum is 1.
# Weights can be negative (indicating a short position in the asset).
# Many sets of weights will be suboptimal.

# Assume multivariate normal distributions of returns
# with means, standard deviations, and correlations
# as specified in the problem setup.

# functions from 'Modern Applied Statistics in S' (Venables and Ripley 2002)
library(MASS)
library(ggplot2) # visualization of the simulation
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```r
# mean returns for portfolio assets A, B, C, and D
targetMeanVector <- c(0.1759, 0.2176, 0.6119, 0.2617)

# standard deviations of returns
targetSDVector <- c(0.2822, 0.3229, 0.4390, 0.2640)

# correlation matrix for the four portfolio assets
targetCorMatrix <- matrix(c(
  1.0000, 0.5540, 0.4868, 0.6141,
  0.5540, 1.0000, 0.5767, 0.6477,
  0.4868, 0.5767, 1.0000, 0.5949,
  0.6141, 0.6477, 0.5949, 1.0000),
  nrow = 4, ncol = 4)

# compute the covariance matrix
targetCovMatrix <- diag(targetSDVector) %*% targetCorMatrix %*% diag(targetSDVector)

# number of return sets to generate
sampleSize <- 700
```

```
# set seed so results are reproducible across executions
set.seed(1111)
# generate multivariate normal returns
returnsData <- mvrnorm(n = sampleSize,
                       mu = targetMeanVector,
                       Sigma = targetCovMatrix)
returnsDataFrame <- as.data.frame(returnsData)
names(returnsDataFrame) <- c("AAPL", "GOOG", "META", "MSFT")

# check statistics from the generated returns data
print(summary(returnsDataFrame))
```

```
##       AAPL               GOOG               META              MSFT
##  Min.   :-0.73804   Min.   :-0.74710   Min.   :-0.7406   Min.   :-0.58119
##  1st Qu.:-0.02278   1st Qu.: 0.00171   1st Qu.: 0.3420   1st Qu.: 0.07259
##  Median : 0.16967   Median : 0.21182   Median : 0.6300   Median : 0.26465
##  Mean   : 0.16513   Mean   : 0.21387   Mean   : 0.6231   Mean   : 0.25628
##  3rd Qu.: 0.35227   3rd Qu.: 0.44121   3rd Qu.: 0.9385   3rd Qu.: 0.43137
##  Max.   : 1.09814   Max.   : 1.23139   Max.   : 1.7923   Max.   : 1.30150
```

```
# check correlation matrix of the generated returns data
cat("\nTarget Correlation of Returns for Generated Data:\n")
```

```
##
## Target Correlation of Returns for Generated Data:
```

```
print(targetCorMatrix)
```

```
##        [,1]   [,2]   [,3]   [,4]
## [1,] 1.0000 0.5540 0.4868 0.6141
## [2,] 0.5540 1.0000 0.5767 0.6477
## [3,] 0.4868 0.5767 1.0000 0.5949
## [4,] 0.6141 0.6477 0.5949 1.0000
```

```
cat("\nActual Correlation of Returns in Generated Data:\n")
```

```
##
## Actual Correlation of Returns in Generated Data:
```

```
print(cor(returnsDataFrame))
```

```
##           AAPL      GOOG      META      MSFT
## AAPL 1.0000000 0.5494631 0.4979419 0.6255798
## GOOG 0.5494631 1.0000000 0.5762297 0.6435262
## META 0.4979419 0.5762297 1.0000000 0.6076079
## MSFT 0.6255798 0.6435262 0.6076079 1.0000000
```

```r
# compute covariance matrix for the sample data
# to be used later in portfolio calculations
cat("\nTarget Covariance of Returns for Generated Data:\n")
```

```
##
## Target Covariance of Returns for Generated Data:
```

```r
print(targetCovMatrix)
```

```
##            [,1]       [,2]       [,3]       [,4]
## [1,] 0.07963684 0.05048180 0.06030761 0.04575094
## [2,] 0.05048180 0.10426441 0.08174901 0.05521358
## [3,] 0.06030761 0.08174901 0.19272100 0.06894653
## [4,] 0.04575094 0.05521358 0.06894653 0.06969600
```

```r
cat("\nActual Covariance of Returns in Generated Data:\n")
```

```
##
## Actual Covariance of Returns in Generated Data:
```

```r
dataCovMatrix <- cov(returnsDataFrame)
print(dataCovMatrix)
```

```
##            AAPL       GOOG       META       MSFT
## AAPL 0.07618120 0.04782908 0.06058624 0.04688190
## GOOG 0.04782908 0.09946246 0.08011184 0.05510543
## META 0.06058624 0.08011184 0.19433135 0.07272662
## MSFT 0.04688190 0.05510543 0.07272662 0.07372201
```

```r
# Generate a random sets of weights using
# a uniform distribution from -1 to 1 if
# shortsOK is TRUE. Otherwise use a uniform
# distribution from 0 to 1 with scaling to
# ensure that the weights sum to 1.
makeWeights <- function(shortsOK) {
    if (shortsOK) {
        threeWeights <- runif(3, min = -1, max = 1)
        fourthWeight <- 1 - sum(threeWeights) # ensures sum of 1
        return(c(threeWeights,fourthWeight))
    }
    if (!shortsOK) {
        initialWeights <- runif(4, min = 0, max = 1)
        return(initialWeights/sum(initialWeights)) # ensures sum of 1
    }
}

# generate sets of portfolio weights in shortsOK state
set.seed(9999) # set seed so results are reproducible
weightsMatrix <- matrix(NA, nrow = sampleSize, ncol = 4)
for (iset in 1:sampleSize)
```

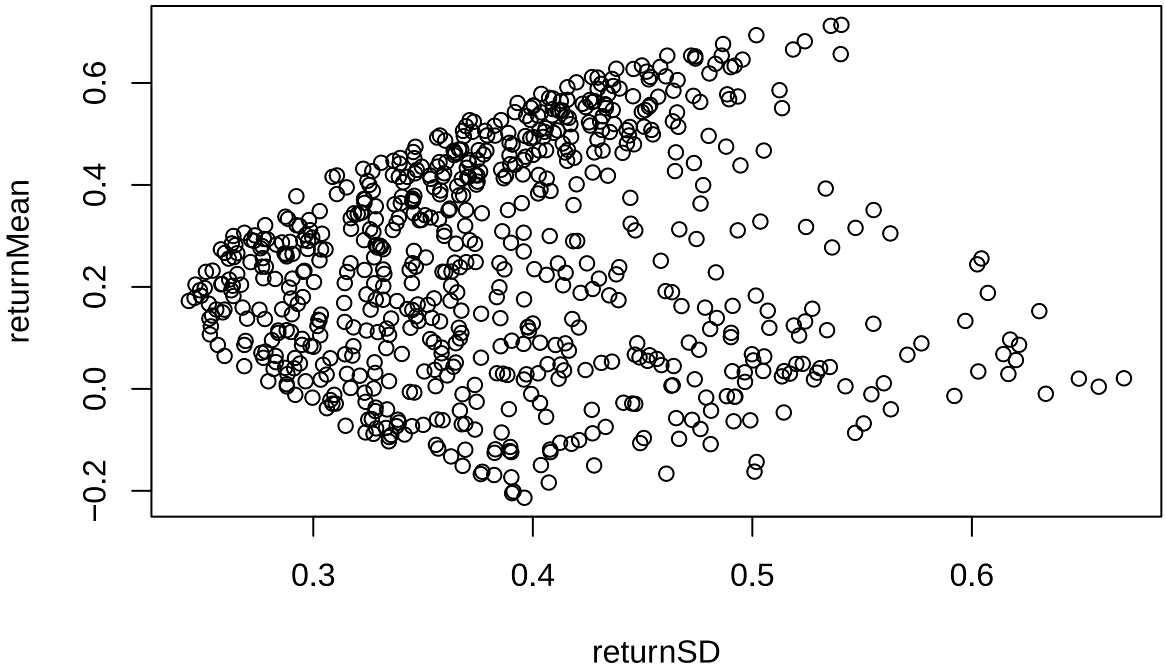
```
    weightsMatrix[iset,] <- makeWeights(shortsOK = TRUE)

# Compute the portfolio return for each set of weights
# for each set of weights applied to each set of returns,
# storing calculations in a new data frame for the portfolios.
portfolioResults = NULL
for (iset in 1:sampleSize) {
    Positions <- 1 # has shorts is 1
    w1 <- weightsMatrix[iset,1]
    w2 <- weightsMatrix[iset,2]
    w3 <- weightsMatrix[iset,3]
    w4 <- weightsMatrix[iset,4]
    if (w1 > 0 && w2 > 0 && w3 > 0 && w4 > 0) Positions <- 2 # no shorts

    returnVector <- weightsMatrix[iset,] %*% t(returnsData)
    returnMean <- mean(returnVector)
    returnSD <- as.numeric(sqrt(t(weightsMatrix[iset,]) %*% dataCovMatrix %*% weightsMatrix[iset,]))
    thisPortfolioResult <- data.frame(w1, w2, w3, w4, Positions, returnMean, returnSD)
    portfolioResults <- rbind(portfolioResults, thisPortfolioResult)
}

# check portfolio results
cat("\nSummary of portfolio results with short positions allowed:\n")
```

```
##
## Summary of portfolio results with short positions allowed:
```

```
print(summary(portfolioResults))
```

```
##        w1                  w2                  w3                  w4
##  Min.   :-0.99885   Min.   :-0.99972   Min.   :-0.99973   Min.   :-1.4131
##  1st Qu.:-0.53759   1st Qu.:-0.50020   1st Qu.:-0.54154   1st Qu.: 0.2482
##  Median :-0.01290   Median : 0.02385   Median :-0.05586   Median : 0.9678
##  Mean   :-0.01094   Mean   : 0.01550   Mean   :-0.03202   Mean   : 1.0275
##  3rd Qu.: 0.47658   3rd Qu.: 0.51116   3rd Qu.: 0.46668   3rd Qu.: 1.7545
##  Max.   : 0.99934   Max.   : 0.99482   Max.   : 0.99975   Max.   : 3.6639
##    Positions      returnMean          returnSD
##  Min.   :1.00   Min.   :-0.21361   Min.   :0.2433
##  1st Qu.:1.00   1st Qu.: 0.06426   1st Qu.:0.3277
##  Median :1.00   Median : 0.23668   Median :0.3762
##  Mean   :1.02   Mean   : 0.24487   Mean   :0.3868
##  3rd Qu.:1.00   3rd Qu.: 0.42767   3rd Qu.:0.4362
##  Max.   :2.00   Max.   : 0.71392   Max.   :0.6693
```

```
with(portfolioResults, plot(returnSD, returnMean))
```



```
shortsOKResults <- portfolioResults
shortsOKResults$ShortsOK = rep("Shorts OK", times = sampleSize)

# generate sets of portfolio weights with no shorts allowed
set.seed(9999) # set seed so results are reproducible
weightsMatrix <- matrix(NA, nrow = sampleSize, ncol = 4)
for (iset in 1:sampleSize)
    weightsMatrix[iset,] <- makeWeights(shortsOK = FALSE)

# Compute the portfolio return for each set of weights
# for each set of weights applied to each set of returns,
# storing calculations in a new data frame for the portfolios.
portfolioResults = NULL
for (iset in 1:sampleSize) {
    Positions <- 1 # has shorts is 1
    w1 <- weightsMatrix[iset,1]
    w2 <- weightsMatrix[iset,2]
    w3 <- weightsMatrix[iset,3]
    w4 <- weightsMatrix[iset,4]
    if (w1 > 0 && w2 > 0 && w3 > 0 && w4 > 0) Positions <- 2 # no shorts
    returnVector <- weightsMatrix[iset,] %*% t(returnsData)
    returnMean <- mean(returnVector)
    returnSD <- as.numeric(sqrt(t(weightsMatrix[iset,]) %*% dataCovMatrix %*% weightsMatrix[iset,]))
    thisPortfolioResult <- data.frame(w1, w2, w3, w4, Positions, returnMean, returnSD)
    portfolioResults <- rbind(portfolioResults, thisPortfolioResult)
}
```

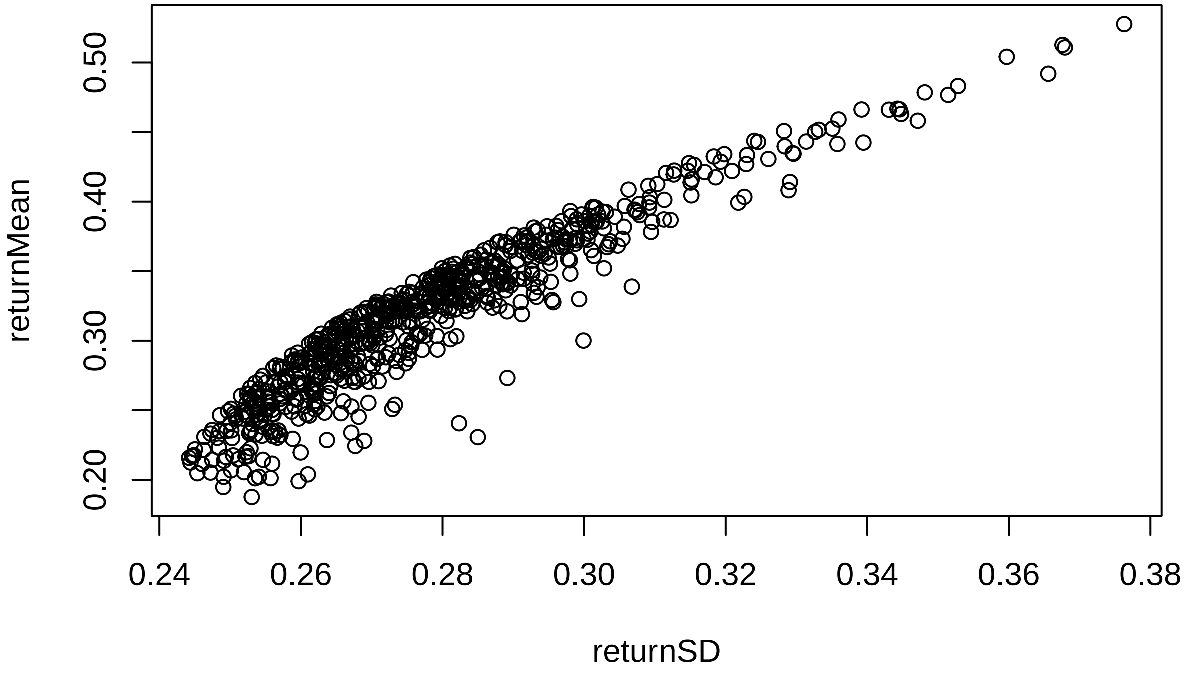
```
# check portfolio results
cat("\nSummary of portfolio results with long positions only:\n")
```

```
##
## Summary of portfolio results with long positions only:
```

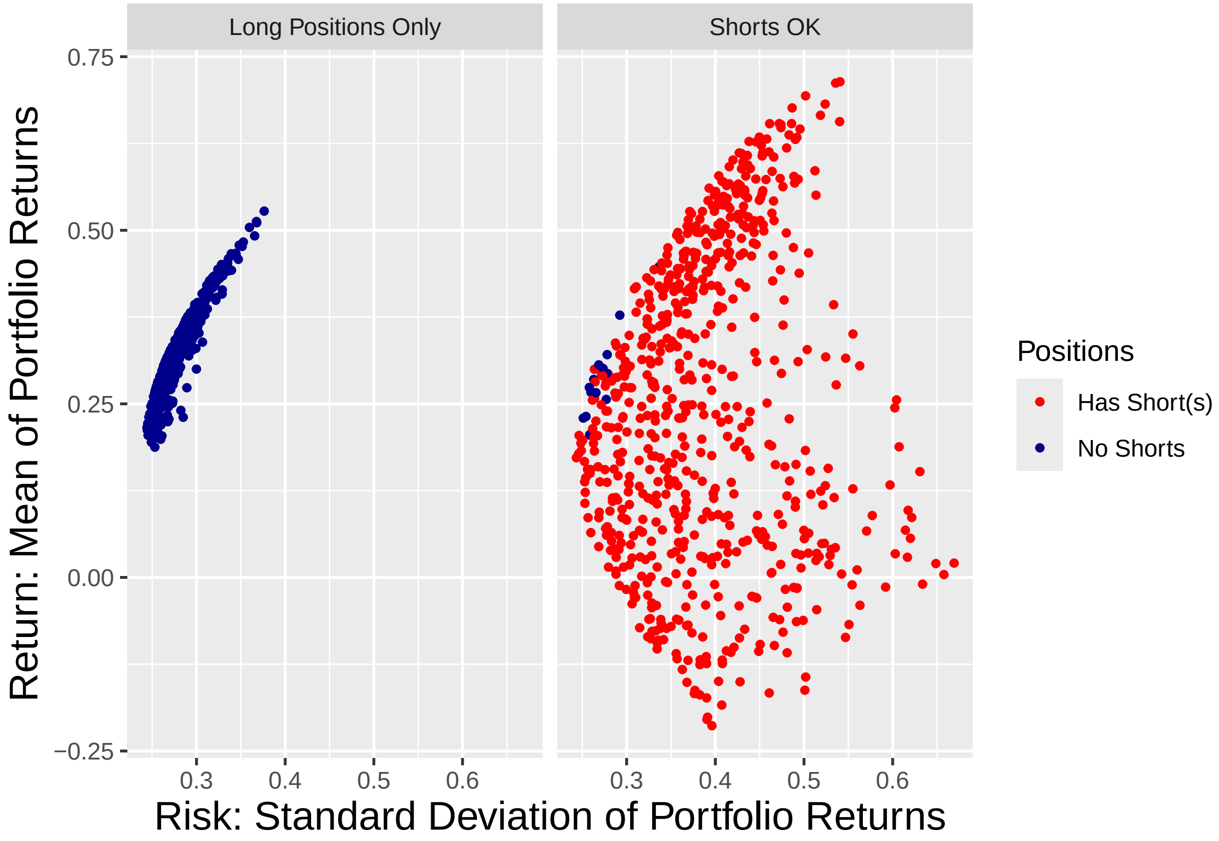
```
print(summary(portfolioResults))
```

```
##        w1                  w2                  w3                 w4
##  Min.   :0.0000709   Min.   :0.0006501   Min.   :0.001087   Min.   :0.0000689
##  1st Qu.:0.1314990   1st Qu.:0.1367747   1st Qu.:0.160793   1st Qu.:0.1339661
##  Median :0.2466478   Median :0.2540729   Median :0.258966   Median :0.2333626
##  Mean   :0.2495896   Mean   :0.2467636   Mean   :0.260156   Mean   :0.2434911
##  3rd Qu.:0.3444534   3rd Qu.:0.3356580   3rd Qu.:0.341754   3rd Qu.:0.3381186
##  Max.   :0.7673518   Max.   :0.7484792   Max.   :0.751567   Max.   :0.8926828
##    Positions   returnMean        returnSD
##  Min.   :2   Min.   :0.1876   Min.   :0.2442
##  1st Qu.:2   1st Qu.:0.2749   1st Qu.:0.2625
##  Median :2   Median :0.3211   Median :0.2749
##  Mean   :2   Mean   :0.3185   Mean   :0.2783
##  3rd Qu.:2   3rd Qu.:0.3519   3rd Qu.:0.2889
##  Max.   :2   Max.   :0.5276   Max.   :0.3763
```

```
with(portfolioResults, plot(returnSD, returnMean)) # preliminary plot on console
```



```
noShortsResults <- portfolioResults
noShortsResults$ShortsOK = rep("Long Positions Only", times = sampleSize)

# merge the two data frames
plottingFrame = rbind(shortsOKResults,noShortsResults)
plottingFrame$Positions <- factor(plottingFrame$Positions,
    labels = c("Has Short(s)", "No Shorts"))

plottingFrame$ShortsOK = factor(plottingFrame$ShortsOK)

# complete plot for both conditions: Shorts Allowed and No Shorts
facetPlot <- ggplot(plottingFrame, aes(x=returnSD,y=returnMean,
    colour=Positions)) +
    geom_point(size = 1) +
    xlab("Risk: Standard Deviation of Portfolio Returns") +
    ylab("Return: Mean of Portfolio Returns") +
    scale_color_manual(values = c("red","darkblue")) +
    theme(axis.title = element_text(size=15)) +
    facet_wrap( ~ ShortsOK, ncol = 2)

print(facetPlot)
```



```
# export pdf plot for inclusion in LaTeX document
pdf(file = "451-portfolio-optimization-monte-carlo-figure.pdf", width = 11, height = 8.5)
print(facetPlot)
```

```r
dev.off()
```

```
## pdf
##   2
```

```r
cat("\nRun Complete\n")
```

```
##
## Run Complete
```