Mean-Variance Optimal Risky Portfolio Using Simulation

Minh Tran Bao Chau; Kabir Singh Kalsi; Saifuddin Ahmed Lnu

03-Nov-2024

Introduction

Portfolio optimization is an important task done by portfolio managers and of central importance to the field of quantitative finance. With the assumption that returns follow a multivariate normal distribution with known mean and variance, mean-variance optimization, is the optimal procedure to maximize the risk-adjusted return. Therefore we can calculate a simulation optimization model so as the optimal investment decision can be made. Computationally, the optimal portfolio is the one with the highest Sharpe ratio.

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
```

```
library(PerformanceAnalytics)
```

```
##
## Attaching package: 'PerformanceAnalytics'
```

```
## The following object is masked from 'package:graphics':
##
##     legend
```

```
library(ggplot2)
```

R function: myMeanVarPort

```
myMeanVarPort <- function(tickers, begin_date, end_date, risk_free_rate) {
  getSymbols(tickers, from = begin_date, to = end_date, auto.assign = TRUE)

  prices <- do.call(merge, lapply(tickers, function(ticker) Cl(get(ticker))))
  returns <- na.omit(Return.calculate(prices))

  stock_means <- colMeans(returns) * 252
  cov_matrix <- cov(returns) * 252

  N <- length(tickers)
  set.seed(12)
  num_portfolios <- 100 * N
  weights <- matrix(runif(num_portfolios * N), nrow = num_portfolios)
  weights <- weights / rowSums(weights)

  portfolio_returns <- weights %*% stock_means
  portfolio_risk <- sqrt(diag(weights %*% cov_matrix %*% t(weights)))
  sharpe_ratios <- (portfolio_returns - risk_free_rate) / portfolio_risk

  portfolios <- data.frame(weights, portfolio_returns, portfolio_risk, sharpe_ratios)
  colnames(portfolios) <- c(tickers, "Mean", "Sigma", "SR")

  list(stock_means = stock_means, cov_matrix = cov_matrix, portfolios = portfolios)
}
```

Plug in input for simulation

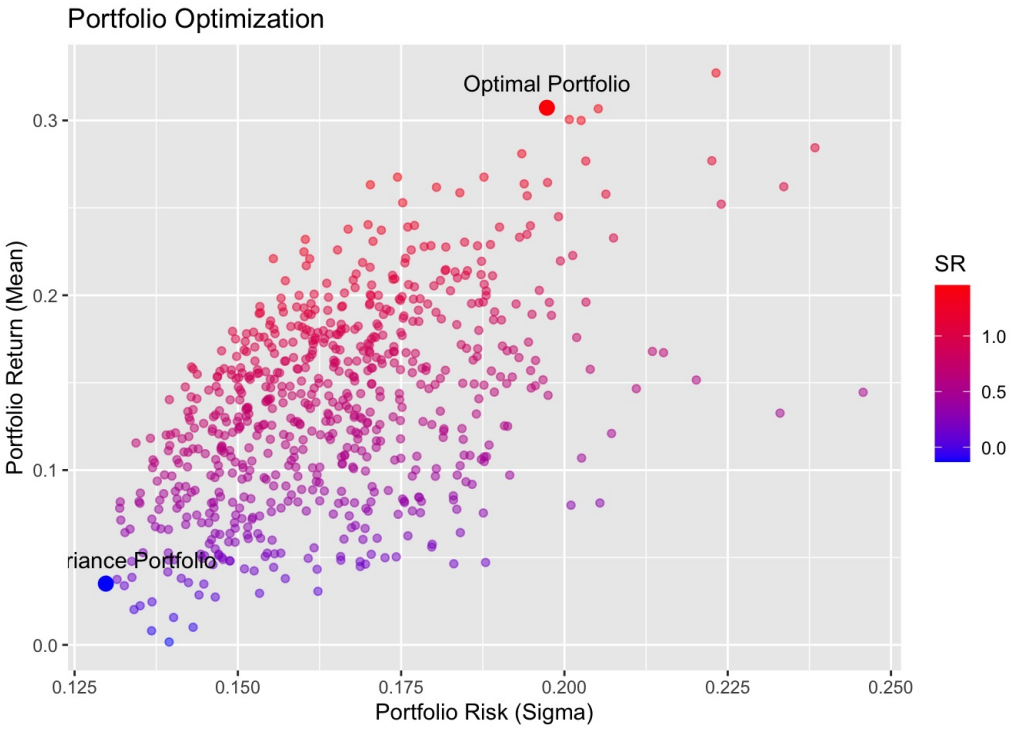
```
tickers <- c("GE", "XOM", "GBX", "SBUX", "PFE", "HMC", "NVDA")
begin_date <- "2014-01-01"
end_date <- "2017-12-31"
risk_free_rate <- 0.02

results <- myMeanVarPort(tickers, begin_date, end_date, risk_free_rate)
```

Plotting results and visualization

```
optimal_portfolio <- results$portfolios[which.max(results$portfolios$SR), ]
min_variance_portfolio <- results$portfolios[which.min(results$portfolios$Sigma), ]

ggplot(results$portfolios, aes(x = Sigma, y = Mean)) +
  geom_point(aes(color = SR), alpha = 0.5) +
  geom_point(data = optimal_portfolio, aes(x = Sigma, y = Mean), color = "red", size = 3) +
  geom_point(data = min_variance_portfolio, aes(x = Sigma, y = Mean), color = "blue", size = 3) +
  annotate("text", x = optimal_portfolio$Sigma, y = optimal_portfolio$Mean, label = "Optimal Portfolio", vjust =
-1) +
  annotate("text", x = min_variance_portfolio$Sigma, y = min_variance_portfolio$Mean, label = "Min Variance Portf
olio", vjust = -1) +
  labs(title = "Portfolio Optimization", x = "Portfolio Risk (Sigma)", y = "Portfolio Return (Mean)") +
  scale_color_gradient(low = "blue", high = "red")
```



Similar function using new yfR package

```
library(yfR)
rnd_tickers <- c("GE", "XOM", "GBX", "SBUX", "PFE", "HMC", "NVDA")

myMeanVarPort_yFR <- function(tickers, begin_date, end_date, risk_free_rate) {
  prices <- yf_get(tickers = rnd_tickers,  first_date = begin_date, last_date = end_date)
  returns <- na.omit(Return.calculate(prices))

  stock_means <- colMeans(returns) * 252
  cov_matrix <- cov(returns) * 252

  num_portfolios <- 100 * length(tickers)
  results <- matrix(NA, nrow = num_portfolios, ncol = 3 + length(tickers))
  colnames(results) <- c("Return", "Risk", "Sharpe", tickers)

  set.seed(12)
  for (i in 1:num_portfolios) {
    weights <- runif(length(tickers))
    weights <- weights / sum(weights)

    portfolio_return <- sum(weights * stock_means)
    portfolio_risk <- sqrt(t(weights) %*% cov_matrix %*% weights)
    sharpe_ratio <- (portfolio_return - risk_free_rate) / portfolio_risk

    results[i, ] <- c(portfolio_return, portfolio_risk, sharpe_ratio, weights)
  }

  results <- as.data.frame(results)
  optimal_portfolio <- results[which.max(results$Sharpe), ]
  min_variance_portfolio <- results[which.min(results$Risk), ]

  list(stock_means = stock_means, optimal_portfolio = optimal_portfolio, min_variance_portfolio = min_variance_po
rtfolio)
}

#portfolio_metrics_yFR <- myMeanVarPort_yFR(rnd_tickers, "2014-01-01", "2017-12-31", 0.02)
```

Pros

- Flexibility: Simulated portfolios allow assessment of multiple assest classes and investment strategies.
- Diverse Outcomes: By having a look of multiple simulations, we can analyze a lot of outcomes, assisting to recognize robust portfolio strategy.
- Visualization: Packages in R for visualizing results, therefore it is easier to intepret complex data to wide ranges of stakeholders.

Cons

- Data input: If data input is low quality or biased, it could lead to unreliable results.
- Assumptions: A lot of simulations depend on market behavior, if the assumption is flawed, the results can be misleading.

- Data intensity: If the simulation data is huge, it might be time-consuming when running the results.

Conclusion

Simulated portfolios can be a useful tool for optimization in R, providing insights into risk and return profiles. However, it’s noteworthy to be vigilant of their limitations and to validate models against real-world scenarios to ensure reliable outcomes. Balancing these pros and cons is essential for effective portfolio management.