MPT and Optimal Portfolio Allocation in ETFs

Summer 2023 personal mini-project

Yu-An(Aaron) Chen

2023-08-06

Contents

Introduction

In the realm of investment management, constructing a well-balanced portfolio that maximizes returns while minimizing risks remains a formidable challenge. To address this, Modern Portfolio Theory (MPT) has emerged as a prominent and empirically grounded framework. This implementation aims to leverage the principles of MPT to create a data-driven investment approach that emphasizes diversification, risk management, and evidence-based decision-making. By focusing on long-term objectives and employing statistical analysis, this implementation seeks to identify optimal portfolio combinations that lie on the efficient frontier, providing investors with superior risk-adjusted returns. Through the implementation of MPT, we endeavor to offer insights that can enhance the rational allocation of assets, instill confidence in investment decisions, and potentially pave the way for more efficient wealth accumulation strategies.

Implementation Details

This implementation optimizes the following function:

$$\min_{\mathbf{w}} \mathbf{w}^T \Sigma \mathbf{w}$$

Subject to:

$$A\mathbf{w} \geq b$$

Where

1. $\mathbf{w}$ is the vector of asset allocation weights
2. Σ is the weighted covariance matrix between different assets
3. A is the matrix representing the inequality constraints
4. b is the vector of constraint bounds.

In other words, this implementation optimizes a portfolio to maximize expected returns and minimize risk. It determines expected returns and risk of each asset based on a weighted measure of its past performance, volatility, and correlation with other assets.

Important notes:

1. In constructing the weighted covariance matrix between different assets, we used a linear function $f(date) = 1 + dateDiff(date - startDate)/totalDays$ to determine the weighting of movements between assets on each trading day. This ensures that more recent correlations between assets are taken more into account.
2. A similar approach was used to determine annual returns. Each annual return is weighted according to the number of years since the start of a user-defined record tracking start date, with weighting ranging from 0.75 to 1.25, scaling linearly. This amplifies the importance of more recent returns, and more closely reflects the current state of the market.
3. This implementation does NOT allow for short selling. That is, allocations cannot go below zero for any asset. This is so users can enjoy a beginner-friendly allocation that still largely achieves its purpose.
4. In the case where there is no possible allocation for the desired return, this implementation will simply return nothing. A lottery ticket may be better in those cases :)

Candidate Assets

In determine which pool of assets to choose from, we landed on a selection of all ETFs with at least $100 million in AUM and expense ratio less than 0.6%. The dataset we used can be found here: https://infolific.com/money-management/exchange-traded-funds/list-of-etfs/

Here are the reasons for this rationale:

1. We only looked at ETFs since they are more diversified, so projections on expected returns based purely on historical returns should have less error
2. Only ETFs with > $100M AUM are considered for liquidity concerns
3. Only ETFs with < 0.6% expense ratios are considered so lost returns don't compound significantly in the long run

With these constraints, there are 363 ETFs in the pool of candidate assets. They cover essentially all major investing styles, strategies, and asset classes. This is meant to largely represent the investable portion of the market, but it's open to change and iterative updates.

Code Implementation

```
#Gets info from quantmod API
getInfo = function(tickers, years){

  # call API
  keep = c()
  for (symbol in tickers){
    res = tryCatch(getSymbols(symbol, type = "xts"),
                   warning =  function(x){return("no")},
                   error = function(x){return("no")})
    if (res != "no"){
      keep = c(keep, symbol)
    }
  }
  tickers = keep
  n = length(tickers)

  # get prices for this ticker for the last "years" years
  getSymbols(tickers[1], type = "xts")
  tmp =  tail(get(tickers[1])[,1],252*years)

  # get percent changes over this period
  changes = diff(log(tmp[,1]))[-1,]

  # initialize price / daily change dataframes
  portfolio = data.frame(diff = tmp)
  p_dailyChange = data.frame(changes)
  badNames = c()

  # populate dataframes
  for (name in tickers[-1]){
    getSymbols(name, type = "xts")
    tmp = na.approx(tail(get(name)[,1],252*years))
    if (length(tmp[,1]) < 252*years){
      n = n - 1
      badNames = c(badNames, name)
      next
    }
    changes = diff(log(tmp[,1]))[-1,]
    portfolio = data.frame(portfolio, tmp)
    p_dailyChange = data.frame(p_dailyChange, changes)
  }

  remainingTickers = tickers[!tickers %in% unique(badNames)]
  names(portfolio) = remainingTickers
  names(p_dailyChange) = remainingTickers

  # get weighted covariance matrix of daily returns between tickers, where
  # most recent entry is twice as important as the earliest record.
  covWeights = 1 + c(1:nrow(p_dailyChange))/nrow(p_dailyChange)
  covmatrix = matrix(c(cov.wt(p_dailyChange, wt = covWeights))[[1]],
```

```
                    nrow=n, ncol=n)
  dimnames(covmatrix) = list(remainingTickers, remainingTickers)

  logReturn = function(rowNum){
      prices = portfolio[seq(1,252*years, length.out = years), rowNum]
      dates =  as.Date(rownames(portfolio)[seq(1,252*years,
                                               length.out = years)])
      zoo_data <- zoo(xts(prices, order.by = dates), order.by = dates)
      xts_data <- as.xts(zoo_data)

      log_returns <- na.omit(Return.calculate(xts_data, method = "log"))

      weights = seq(0.75, 1.25, length.out = years - 1)
      weightedReturn = mean((exp(log_returns) - 1) * weights)
      return(weightedReturn)
  }

  # annual geometric returns for each stock
  avgReturns = round(unlist(lapply(c(1:n),logReturn)), 3)
  names(avgReturns) = remainingTickers

  return(list(avgReturns, covmatrix, remainingTickers))
}
```

API Access & Feature Engineering Functions

```
getAllocHelper = function(desiredReturn, estReturns, covmatrix,
                          tickers, principle){
  # set up constrained optimization problem
  n = length(estReturns)
  covmatrix[is.na(covmatrix) | is.infinite(covmatrix)] <- 1
  Dmat = covmatrix
  dvec = rep(0,n)
  Amat = cbind(matrix(estReturns, nrow = n),
               matrix(rep(1, n), nrow = n), diag(n))
  bvec = matrix(c(desiredReturn, 1, rep(0,n)), ncol = 1)

  # first "meq" constraints are equality constraints
  # 1. expected return = desiredReturn
  # 2. sum of weights = 1
  # 3. all weights >= 0
  solution <- tryCatch(list("pass!", solve.QP(Dmat, dvec, Amat, bvec, meq = 2)),
                       error = function(error){
                              noSol = data.frame(matrix(rep(0, 3*length(tickers)),
                                                        ncol = 3))
                              rownames(noSol) = tickers
                              names(noSol) = c("strategy", "allocation", "amount")
                              return(list("error!", noSol))
                                })
  if (solution[[1]] == "error!"){
    return(solution[[2]])
```

```
  }
  solution = solution[[2]]

  # extract weighting and fund allocation
  strategy = round(solution$solution, digits = 3)
  allocation = abs(strategy)/sqrt(sum((strategy)^2))
  allocation =  allocation/sum(allocation)

  overallStrat = data.frame(t(rbind(strategy, allocation)))
  rownames(overallStrat) = tickers

  overallStrat = round(overallStrat, 3)

  overallStrat$amount = overallStrat$allocation * principle
  return(overallStrat)
}

getAlloc = function(years, principle, myEst, symbols,
                    baseReturn, desiredReturn, stepSize, actual){
  c(avgReturns, covmatrix, tickers) %<-% getInfo(symbols, years)
  returns = data.frame(Actual = avgReturns, myGuess = myEst)
  if(actual == 1){
    estReturns = avgReturns
  }
  else{
    estReturns = myEst
  }

  # dataframe of allocations for all tickers across all
  # desired return preferences
  targetWeights = data.frame(t(matrix(unlist(
                    mclapply(seq(baseReturn,desiredReturn,stepSize),
                             function(x){
                               return(getAllocHelper(x, estReturns,
                                                     covmatrix, tickers,
                                                     principle)$amount)
                             })), nrow = length(tickers))
                              )
                             )
  names(targetWeights) = tickers

  # additional metrics: variance & achieved estimated return
  portfolio_sd <- sqrt(252)* apply(targetWeights, 1,
                                   function(row){
                                     return(sqrt(t(row/principle)%*%
                                                   covmatrix%*%row/principle))})
  portfolio_sd = round(portfolio_sd, 4)

  return <- round(apply(targetWeights, 1,
                                function(row){
                                  return((row%*%estReturns)/principle)}), 2)

  targetWeightsClean = targetWeights[,apply(targetWeights, 2, function(x){
```

```
                                        return(max(x) > principle * 0.01)})]
  output = data.frame(Est_Return = return,
                      Risk = portfolio_sd,
                      targetWeightsClean[,rev(order(apply(targetWeightsClean,
                                                          2, which.max)))])

  zeros_cols <- colSums(output) == 0
  zeros_rows <- rowSums(output) == 0

  output = data.frame(subset(output, select = !zeros_cols))[!zeros_rows,]

  rownames(output) = NULL

  return(output)
}
```

Quadratic Solver & Wrapper Functions

```
etfs = read.csv("etfList.csv")

etfs = subset(etfs, Structure == "ETF")
etfs$Market.Cap..millions. = as.numeric(gsub("(\\$|,)","",
                                             etfs$Market.Cap..millions.))

etfs$Expense.Ratio = as.numeric(gsub("(\\%|,)","",
                                             etfs$Expense.Ratio))

etfs = subset(etfs, Market.Cap..millions. > 100)
etfs = subset(etfs, Expense.Ratio < 0.6)

# Liquidated ETF (edge case)
etfs = subset(etfs, Symbol != "ELV")

tickers = c(etfs$Symbol)
```

Filtering Tickers

Executing Program

Now we will, following Modern Portfolio Theory, determine the optimal portfolios across expected returns from 0% to 30%. As an example, we will designate the data collection look-back period to be 10 years and a total of $10000USD to allocate to the assets.

```
optimalAlloc = getAlloc(years = 10, principle = 10000,
                        myEst = NA, symbols = tickers,
                        baseReturn = 0, desiredReturn = 0.5,
                        stepSize = 0.02, actual = 1)
```

Est_Return	Risk	SMH	SLX	XLK	IHI	IHF	SHV	GSY	BIL
0.00	0.0051	0	0	0	0	0	4580	750	4530
0.02	0.0205	450	90	200	30	60	6910	0	2260
0.04	0.0409	910	180	410	60	110	8330	0	0
0.06	0.0614	1370	270	610	100	160	7490	0	0
0.08	0.0816	1830	350	810	130	210	6670	0	0
0.10	0.1020	2290	440	1010	160	260	5840	0	0
0.12	0.1225	2760	520	1210	190	310	5010	0	0
0.14	0.1430	3210	610	1410	230	370	4170	0	0
0.16	0.1634	3670	690	1620	260	420	3340	0	0
0.18	0.1839	4130	780	1820	290	470	2510	0	0
0.20	0.2047	4600	870	2020	320	520	1680	0	0
0.22	0.2248	5050	950	2220	360	570	850	0	0
0.24	0.2456	5520	1040	2420	390	620	10	0	0
0.26	0.2669	7170	1070	1760	0	0	0	0	0
0.28	0.2922	10000	0	0	0	0	0	0	0

```
kable_styling(kable(optimalAlloc),bootstrap_options = "striped") %>%
  scroll_box(width = "800px", height = "650px")
```

From the output, a few things stand out. First, we notice very heavy allocations to SHV and BIL when our target returns are low. This makes sense as these are treasury ETFs and are considered to be essentially risk-free assets. We also notice that as the target returns increases, allocations in more growth-oriented ETFs start to pick up, namely SMH, XLK, and IHI. This also makes sense as these names have a higher historical average returns over the last 10 years compared to other low-risk assets. We also see the portfolio risk steadily increasing as the target returns increase, which is to be expected – it's the price investors pay for higher expected returns. Note here that allocations with less than 1% weight are not displayed for simplicity.

Visualizations

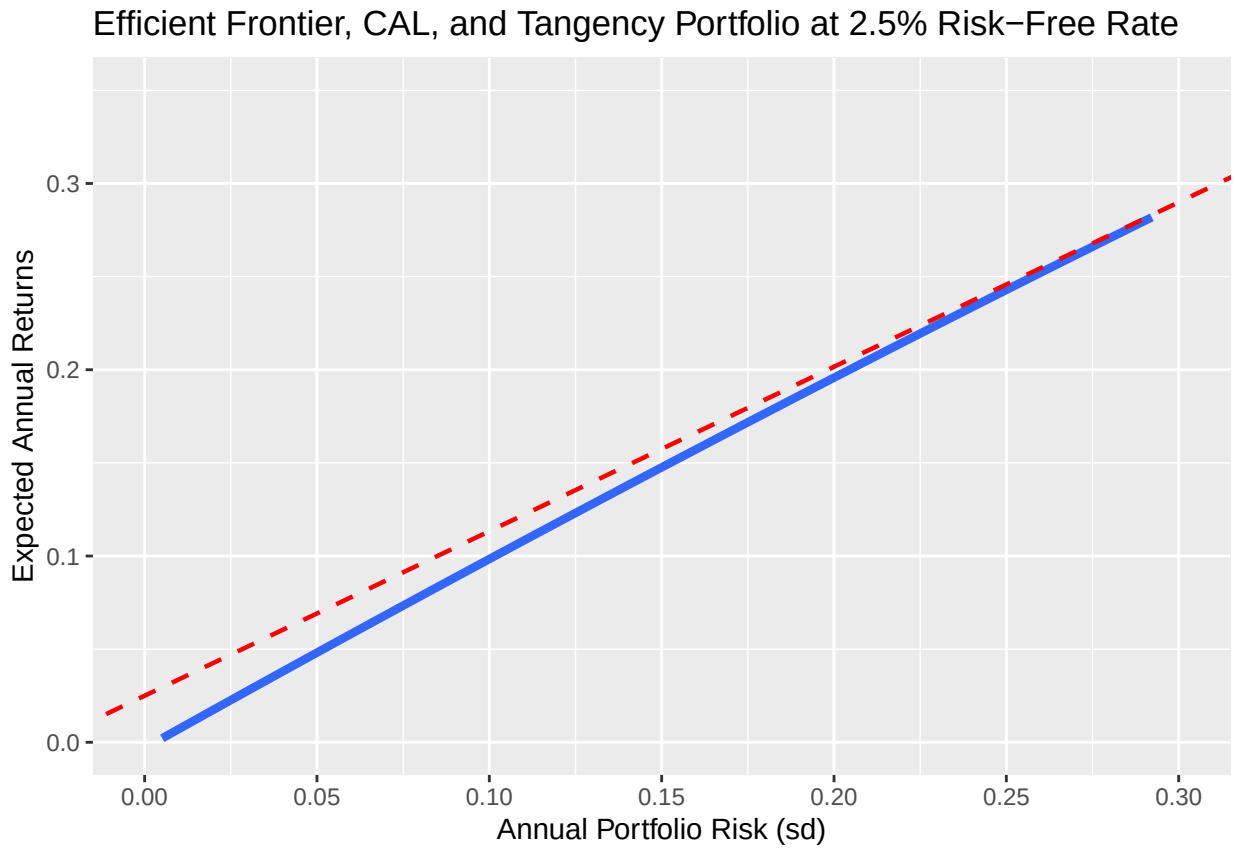
Efficient Frontier, Capital Allocation Line, and Tangency Portfolio

Now with the optimal allocations, we can visualize the relationship between expected annual returns and portfolio variance and find the allocation with the highest Sharpe Ratio given a 2.5% risk free rate (rough average of the 10-year treasury yield over the last 10 years).

```
frontier = data.frame(return = optimalAlloc$Est_Return,
                      risk = optimalAlloc$Risk,
                      risk2 = optimalAlloc$Risk^2)

model = lm(return ~ risk + risk2, data = frontier)

result <- function(a,b,c){
  if(delta(a,b,c) > 0){ # first case D>0
        x_1 = (-b+sqrt(delta(a,b,c)))/(2*a)
        x_2 = (-b-sqrt(delta(a,b,c)))/(2*a)
        result = c(x_1,x_2)
  }
  else if(delta(a,b,c) == 0){ # second case D=0
        x = -b/(2*a)
```

```
  }
  else {"There are no real roots."} # third case D<0
}

# Constructing delta
delta<-function(a,b,c){
      b^2-4*a*c
}

riskFreeRate = 0.025

bestVar = result(model$coefficients[3], 0,
                 riskFreeRate - model$coefficients[1])

bestEstReturns = predict(model, newdata =
                       list(risk = max(bestVar[2]),
                            risk2 = max(bestVar[2])^2))

ggplot(frontier, aes(x = risk, y = model$fitted.values)) +
  geom_smooth(method = "loess", formula = "y~x", se = FALSE, linewidth = 1.5) +
  geom_abline(intercept = riskFreeRate,
              slope = 2*model$coefficients[3]*max(bestVar) +
                      model$coefficients[2],
              color="red", linetype = "dashed", linewidth=0.8, alpha = 8) +
  geom_point(aes(x = max(bestVar), y = bestEstReturns), size = 2) +
  labs(title =
      "Efficient Frontier, CAL, and Tangency Portfolio at 2.5% Risk-Free Rate",
       y = "Expected Annual Returns", x = "Annual Portfolio Risk (sd)") +
  ylim(0,0.35) +
  scale_x_continuous(n.breaks = 7, limits = c(0,0.3))
```

```
## Warning: Removed 15 rows containing missing values (`geom_point()`).
```



From the plot above, we observe that the efficient frontier curve (blue) demonstrates a positive correlation between daily portfolio risk and expected returns, which is to be expected – the higher the expected return, the higher the portfolio risk. Any point on this line represents an optimal possible allocation of assets given an investors risk tolerance.

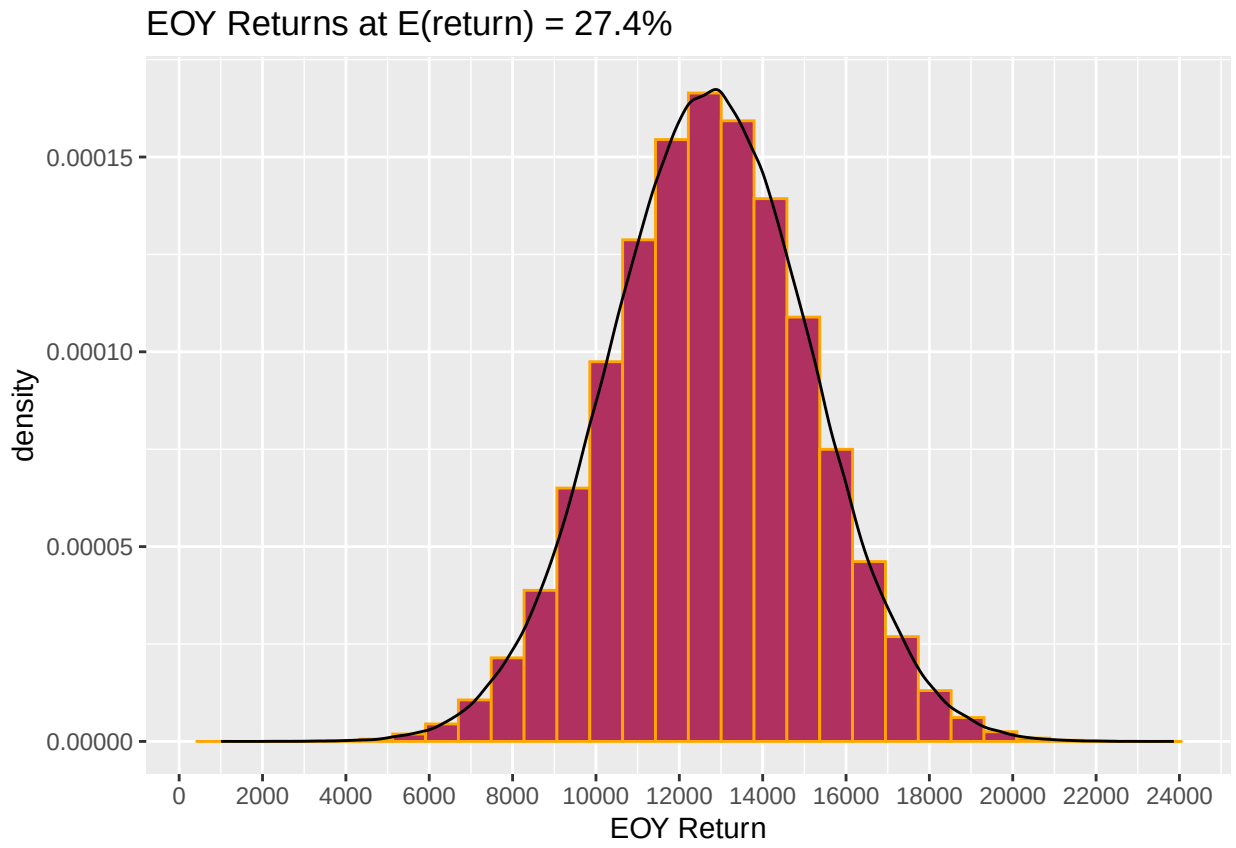
The red dashed line shows the capital allocation line and the point tangent to the efficient frontier is the allocation with the highest Sharpe Ratio. In this case, the expected return of the tangency portfolio is 27.4% with a portfolio risk of 23.9%. In the next section we will take the allocation of the tangency portfolio and simulate returns with its risk reward profile.

EOY Return Simluations

To simulate this, we will assume that annual returns follow a normal distribution with mean = principle * 1.274 and has the previously stated standard deviation of 23.9%. We will draw 500000 samples from this distribution to observe the distribution of possible outcomes from following this variance minimizing allocation strategy.

```
principle = 10000

endReturn = rnorm(500000, mean = principle*1.274, sd = principle*0.239)

ggplot(data.frame(endReturn), aes(x = endReturn)) +
  geom_histogram(aes(y = after_stat(density)),
                 fill = "maroon", color = "orange", bins = 30) +
  scale_x_continuous(n.breaks = 15) +
```

```
  geom_density() +
  labs(x = "EOY Return", title = "EOY Returns at E(return) = 27.4%")
```




```
lossProb = length(which(endReturn < principle*1.025))/length(endReturn)

#SPY Avg gain over last 10 years = 13.09%
gainProb = length(which(endReturn > 11309))/length(endReturn)
```

As shown, this risk-reward portfolio profile can lead to returns ranging from around -20% to +75% in the middle 95% range. This demonstrates that even when achieving minimum variance, having a portfolio with expected annual return standard deviation of 24.3% can still lead to significant negative returns. In fact, according to this model, there is a ~14.8% chance in expectation that a hypothetical investor with this portfolio does not beat the long-term risk-free rate of 2.5%. However, this risk is compensated by a ~72.5% probability that this allocation will beat the S&P500 index in returns in a year. Overall, this plot provides a theoretical lower bound visualization on the variance of EOY returns of a portfolio given previously stated parameters and risk tolerance – it serves as a useful guideline approximation on real-world potential returns.

Use Case & Limitations

There are plenty of use cases for these optimal allocation given an investor's risk tolerance, namely:

1. Serving as a guideline for rebalancing one's portfolio
2. Testing optimal allocations with certain assets pre-established

3. Managing risk/rewards given simulated return scenarios

However, there are a few key assumptions that one should keep in mind about utilizing MPT in this context, namely:

1. Expected returns of each ETF are calculated from weighted past performance
2. Associations between ETFs over time remain the same
3. Slight bias towards ETFs that have performed exceptionally well in recent years

To address these issues, this implementation has introduced a new parameter, "myEst", where users can define their estimated returns for each asset independent of past performance. Note that changing this parameter will not alter the covariance matrix, but will likely alter the resulting allocation weights for different assets.

Even with this in mind, it's important to note the optimal allocations should on be used as a reference instead of pure truth. It doesn't account for real world events nor circumstances and any output should therefore be interpreted with a large grain of salt.

Conclusion

In this implementation, we provided a weighted covariance and geometric return approach to the MPT optimization problem. We were able to show the optimal allocations given the ETFs as candidate assets. We also showed the risk reward profile of an hypothetical investor holding the tangency portfolio with a 27.4% expected annual return. In terms of next steps, we hope to incorporate analyst estimates on the stocks as a factor in estimating expected returns to obtain a more comprehensive view on an asset's outlook.

Special thanks to Professor O. at Rice University on Coursera for motivating this mini-project!