Pearl Fund: Portfolio Anlaysis & Optmization

Navein Suresh & Suryaveer Singh

2024-03-19

Overview

In this project, our primary goal is to enhance the risk-adjusted returns of a carefully curated portfolio of stocks. The focus of our analysis is on a equity/fixed income portfolio, nicknamed — "Pearl Fund." This portfolio comprises Stocks & ETFs from a diverse array of industries and asset classes, namely:

1. Chevron Corporation (NYSE: CVX)
2. Costco Wholesale Corporation (NASDAQ: COST)
3. Ford Motor Company (NYSE: F)
4. HP Inc. (NYSE: HPQ)
5. Johnson & Johnson (NYSE: JNJ)
6. JPMorgan Chase & Co. (NYSE: JPM)
7. Microsoft Corporation (NASDAQ: MSFT)
8. PepsiCo, Inc. (NASDAQ: PEP)
9. Procter & Gamble Company (NYSE: PG)
10. Raytheon Technologies Corporation (NYSE: RTX)
11. iShares 20+ Year Treasury Bond ETF (NASDAQ: TLT)

Our approach involves the utilization of sophisticated financial models, which will be thoroughly explained later in this documentation. Furthermore, we leverage the power of various statistical and financial packages. We use th R programming language as our preferred tool for seamless portfolio optimization. Through a comprehensive analysis of historical data, we aim to evaluate and compare the performance of our portfolio against two benchmark index funds—S&P 500 and NASDAQ.

Tip

Detailed Background Overview: Pearl Fund

Once you're familiarized with the code, feel free to explore and add your own curated collections of assets to see how they perform! To this make sure to enter the *exchange traded ticker symbols* to the *stock_tickers* vector in the second cell below. You must ensure that the assets you choose were traded on an exchange from a minimum of 2004 onward.

Before proceeding, ensure that the following libraries are installed and imported for seamless execution of the project:

```r
suppressWarnings(suppressPackageStartupMessages({
  library(tidyverse)
  library(ggplot2)
  library(plotly)
  library(knitr)
  library(tidyquant)
  library(corrplot)
  library(quantmod)
  library(gganimate)
  library(PerformanceAnalytics)
  library(PortfolioAnalytics)
  library(DEoptim)
  library(ROI)
  library(tseries)
  library(TTR)
  library(fPortfolio)
  require(ROI.plugin.glpk)
  require(ROI.plugin.quadprog)
}))
```

Equity Analysis

Our analysis spans from January 1, 2004, until the latest market day close, providing a comprehensive view of various economic scenarios, including critical events such as the global financial crisis of 2008 and the SARS COVID-19 epidemic in 2020. We use the FRED economic data set to retrieve the most recent 10-year Treasury yields for our statistical testing we will run later.

```r
stock_tickers <- c("TLT", "MSFT", "HPQ", "COST", "F", "PEP", "JPM", "RTX", "PG", "CVX", "J

# Treasury Yields (FRED)
ten_year_treasury_yield_table = arrange(tq_get("DGS10", get = "economic.data"), desc(date)
latest_ten_year_treasury_yield = slice(ten_year_treasury_yield_table, 1)$price
```

```r
cat("Latest 10-year Treasury Yield: ",latest_ten_year_treasury_yield,"%")
```

```
Latest 10-year Treasury Yield:  4.34 %
```

Monthly Returns (Table View)

We proceed to collect our monthly stock returns using the tidyquant functions. We split this data set into two sets, one set ranging from 2004 to 5 years prior to the current date and another one from the previous 5 years. After some data pre-processing, we display the 5 most recent monthly returns for each of our stocks for the entire timeline below.

```r
stock_returns_monthly_full_raw <- tq_get(stock_tickers, get  = "stock.prices", from = "200

# Transformed Monthly Stock Returns (long format)
stock_returns_monthly_full_long = stock_returns_monthly_full_raw %>% group_by(symbol) %>%

# Transformed Monthly Stock Returns (wide format)
stock_returns_monthly_full_wide = pivot_wider(stock_returns_monthly_full_long, names_from=

# Displays the most recent 5 monthly stock returns
stock_returns_monthly_full_wide_decreasing =  stock_returns_monthly_full_wide[order(stock_
as.data.frame(head(stock_returns_monthly_full_wide_decreasing, 5))
```

```
        date         TLT         MSFT         HPQ        COST           F
1 2024-03-18 -0.01304786  0.008896607  0.06636740 -0.01660197 -0.02090026
2 2024-02-29 -0.02252203  0.042318377 -0.01323577  0.07210393  0.09007387
3 2024-01-31 -0.02245145  0.057281150 -0.04586244  0.05272086 -0.03855610
4 2023-12-29  0.08685769 -0.007574393  0.03496809  0.13893188  0.18810908
5 2023-11-30  0.09923865  0.122945401  0.11431823  0.07491826  0.05230767
           PEP        JPM          RTX          PG          CVX         JNJ
1  0.035804998 0.03547246  0.042266098  0.01428215  0.022367011 -0.02862815
2 -0.011441607 0.06710253 -0.009434506  0.01145477  0.042316332  0.02331857
3 -0.007713055 0.03136482  0.082956981  0.07910756 -0.011598336  0.01378068
4  0.009210248 0.08982576  0.032646025 -0.04546644  0.038718657  0.01344889
5  0.038542747 0.12239323  0.008485235  0.02326199 -0.004286768  0.05095416
```

The data set below shows the last 5 monthly returns from 2004 to 5 years prior to today's date.

```
# Raw Stock Monthly Returns
stock_returns_monthly_raw <- tq_get(stock_tickers, get  = "stock.prices", from = "2004-01-

# Transformed Monthly Stock Returns (long format)
stock_returns_monthly_long = stock_returns_monthly_raw %>% group_by(symbol) %>% tq_transmu

# Transformed Monthly Stock Returns (wide format)
stock_returns_monthly_wide = pivot_wider(stock_returns_monthly_long, names_from=symbol, va

# Displays the most recent 5 monthly stock returns
stock_returns_monthly_wide_decreasing =  stock_returns_monthly_wide[order(stock_returns_mo
as.data.frame(head(stock_returns_monthly_wide_decreasing, 5))
```

```
        date          TLT        MSFT          HPQ        COST            F
1 2019-03-29  0.055716791  0.05275390 -0.006859361  0.10697616  0.001140035
2 2019-02-28 -0.013763727  0.07735731 -0.104403278  0.02190189 -0.003409059
3 2019-01-31  0.003785446  0.02815820  0.076735060  0.05360571  0.170367147
4 2018-12-31  0.058534830 -0.08404739 -0.104077708 -0.11920618 -0.187034777
5 2018-11-30  0.017872047  0.04268381 -0.047224674  0.01399537 -0.014659908
          PEP          JPM         RTX          PG         CVX         JNJ
1  0.05975444 -0.029992563  0.02562265  0.05580916  0.03010534  0.02305355
2  0.03468290  0.008309405  0.07069356  0.02156130  0.05348490  0.03356045
3  0.01982274  0.068843582  0.10884704  0.05780042  0.05386544  0.03122814
4 -0.08678666 -0.122043245 -0.12606703 -0.02740471 -0.08533722 -0.12151120
5  0.08506857  0.019904631 -0.01345479  0.06574222  0.07570461  0.05604294
```

We repeat the above process for the 2nd batch of monthly stock returns which lasts for the last 5 years.

```
stock_returns_monthly_test_raw <- tq_get(stock_tickers, get  = "stock.prices", from = Sys.

stock_returns_monthly_test_long = stock_returns_monthly_test_raw %>% group_by(symbol) %>%

stock_returns_monthly_test_wide = pivot_wider(stock_returns_monthly_test_long, names_from=

stock_returns_monthly_test_wide_decreasing =  stock_returns_monthly_test_wide[order(stock_
as.data.frame(head(stock_returns_monthly_test_wide_decreasing, 5))
```

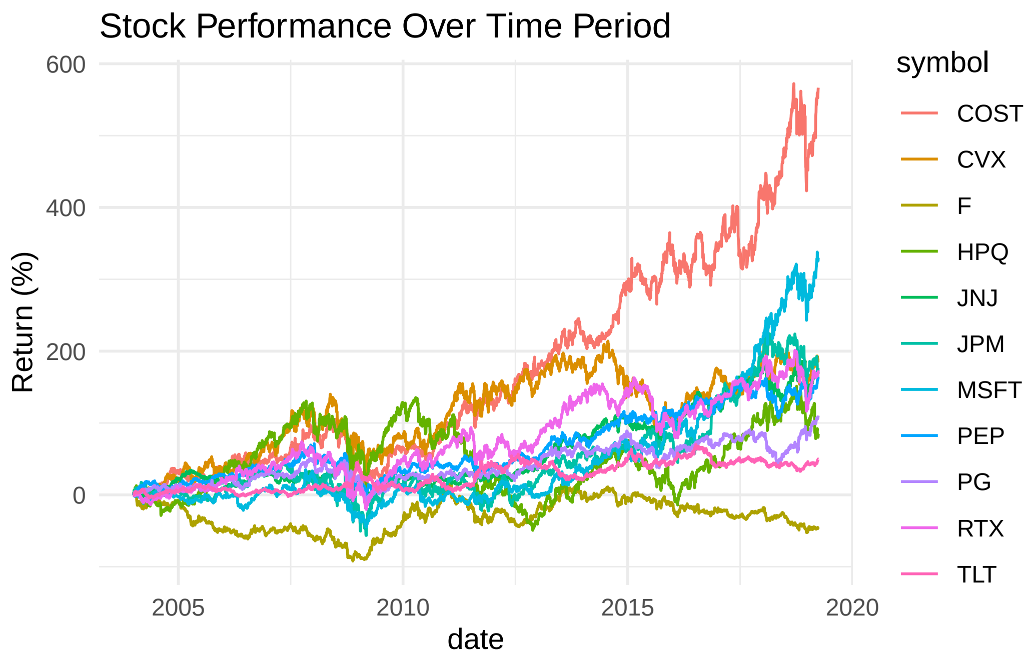
```
        date         TLT        MSFT        HPQ        COST           F
1 2024-03-18 -0.01304786  0.008896607  0.06636740 -0.01660197 -0.02090026
```

```
2 2024-02-29 -0.02252211  0.042318377 -0.01323577  0.07210393  0.09007387
3 2024-01-31 -0.02245129  0.057281150 -0.04586244  0.05272086 -0.03855610
4 2023-12-29  0.08685742 -0.007574393  0.03496809  0.13893188  0.18810908
5 2023-11-30  0.09923874  0.122945401  0.11431823  0.07491826  0.05230767
           PEP        JPM          RTX          PG          CVX         JNJ
1  0.035804998 0.03547246  0.042266098  0.01428215  0.022367011 -0.02862815
2 -0.011441607 0.06710253 -0.009434506  0.01145477  0.042316332  0.02331857
3 -0.007713055 0.03136482  0.082956981  0.07910756 -0.011598336  0.01378068
4  0.009210248 0.08982576  0.032646025 -0.04546644  0.038718657  0.01344889
5  0.038542747 0.12239323  0.008485331  0.02326199 -0.004286768  0.05095416
```

Stock Growth & Monthly Returns (Graph View)

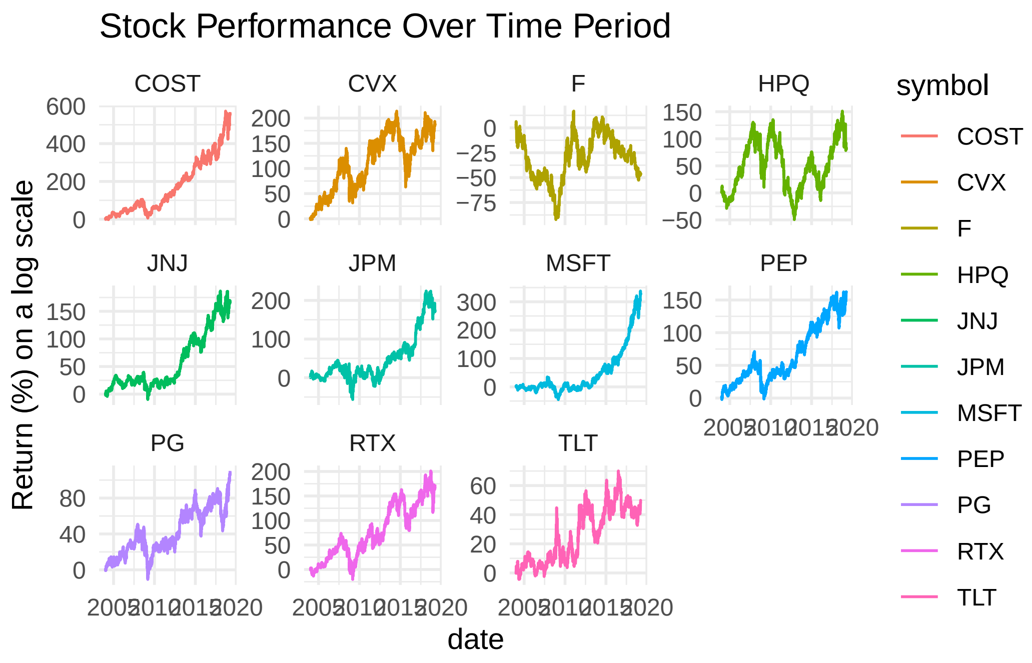
Displayed below is a graphical representation of the total returns (%) for our chosen stocks covering the chosen time period. To facilitate accurate time series analysis, the data has been standardized.

```r
stock_returns_monthly_raw1 <- stock_returns_monthly_raw %>%
  group_by(symbol) %>%
  mutate(return = 100 * (close - first(close)) / first(close)) %>%
  ggplot(aes(date, return, color = symbol)) +
  geom_line() +
  labs(title = "Stock Performance Over Time Period",
       y = "Return (%)") +
  theme_minimal()
stock_returns_monthly_raw1
```



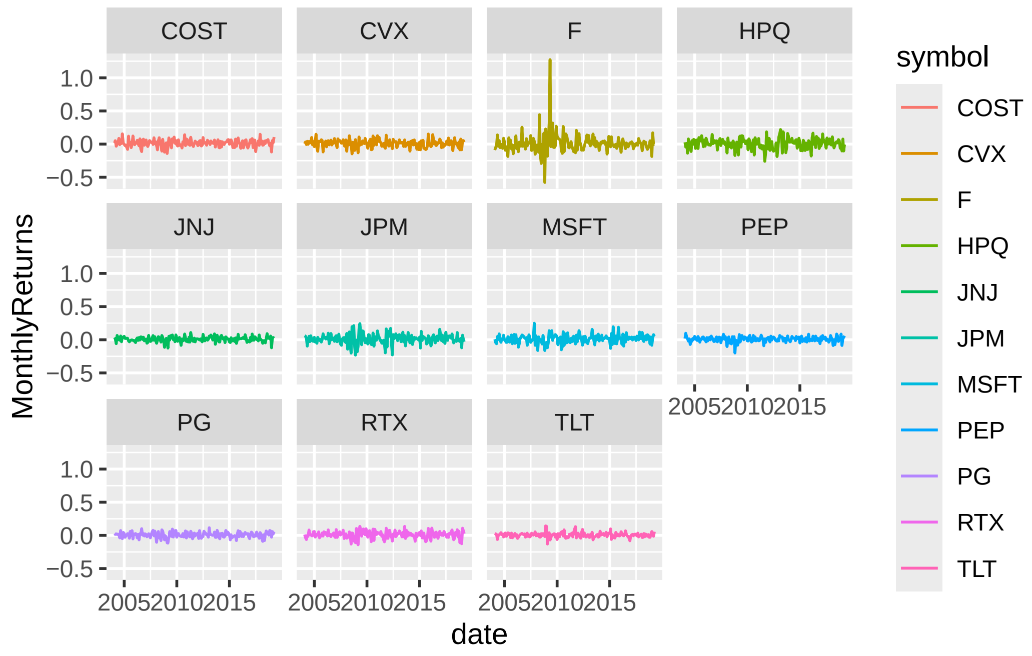
The individual graphs below show the total return (to date) for each asset on their own scales.

```
stock_returns_monthly_raw %>%
  group_by(symbol) %>%
  mutate(return = 100 * (close - first(close)) / first(close)) %>%
  ggplot(aes(date, return, color = symbol)) +
  geom_line() +
  labs(title = "Stock Performance Over Time Period",
       y = "Return (%) on a log scale") +
  theme_minimal() +
  facet_wrap(~symbol, scales = "free_y")
```



Furthermore, within the same temporal framework, we present the monthly returns for our selected stocks. The graphical representation highlights the varying levels of volatility, with more erratic and spread-out patterns evident in stocks characterized by higher volatility.

```
stock_returns_monthly_long %>% ggplot(aes(date, `MonthlyReturns`, color=symbol)) +
  geom_line() +
  facet_wrap(~ symbol)
```



Statistical Metrics

We calculate the expected returns (based off historical returns), standard deviations, variance, and total count for each of our assets.

```
Stock_Statistics = stock_returns_monthly_long %>% group_by(symbol)  %>% summarise(Historic
Stock_Statistics
```

```
# A tibble: 11 x 5
   symbol Historic_Expected_Returns Standard_Deviation Variance Count
   <chr>                      <dbl>              <dbl>    <dbl> <int>
 1 COST                     0.0135              0.0527  0.00278   183
 2 CVX                      0.0103              0.0565  0.00319   183
 3 F                        0.00766             0.143   0.0204    183
 4 HPQ                      0.00819             0.0809  0.00654   183
 5 JNJ                      0.00858             0.0393  0.00154   183
 6 JPM                      0.0107              0.0764  0.00584   183
 7 MSFT                     0.0124              0.0649  0.00422   183
 8 PEP                      0.00838             0.0405  0.00164   183
 9 PG                       0.00731             0.0422  0.00178   183
10 RTX                      0.00876             0.0535  0.00286   183
11 TLT                      0.00581             0.0369  0.00136   183
```

```
animation1 = ggplot(Stock_Statistics, aes(x = Standard_Deviation, y = Historic_Expected_Re
geom_point(size = 5) +
theme_bw() + ggtitle(" Monthly Historic Risk-Return Tradeoff") +
xlab("Volatility") + ylab("Expected Returns")

ggplotly(animation1)
```

Calculation of Excess Returns

Moving forward, our analysis involves the computation of excess returns. This is achieved by subtracting our anticipated monthly return value from each historical monthly return value, resulting in an array of excess returns. This practice is widely adopted in risk-adjusted portfolio analysis to refine the assessment of portfolio performance.

Using our split data set, we calculate the historical expected mean from the first group and our excess returns using the monthly returns from the last 5 years with respect to the calculated expected returns to ensure the excess returns don't contain future data priced in.

```
excess_returns_table <- stock_returns_monthly_test_long %>% left_join(Stock_Statistics, by

excess_returns_table <- excess_returns_table %>% select(`symbol`, `date`, `Excess_Returns`

# Displays the most recent 5 monthly excess stock returns
excess_returns_table_decreasing =  excess_returns_table[order(excess_returns_table$date, d
head(excess_returns_table_decreasing,length(stock_tickers))
```

```
# A tibble: 11 x 3
# Groups:   symbol [11]
   symbol date       Excess_Returns
   <chr>  <date>              <dbl>
 1 TLT    2024-03-18       -0.0189
 2 MSFT   2024-03-18       -0.00350
 3 HPQ    2024-03-18        0.0582
 4 COST   2024-03-18       -0.0301
 5 F      2024-03-18       -0.0286
 6 PEP    2024-03-18        0.0274
 7 JPM    2024-03-18        0.0248
 8 RTX    2024-03-18        0.0335
 9 PG     2024-03-18        0.00697
10 CVX    2024-03-18        0.0121
11 JNJ    2024-03-18       -0.0372
```

We compute two matrices essential for our optimization process: the variance-covariance matrix and the correlation matrix. Each value within these matrices is derived from the following respective equations:

Note

Calculation of the correlation matrix necessitates the input of the variance-covariance matrix

$$Var - CoVar : \Sigma = \frac{X^T X}{n-1}$$

$$Corr : \frac{\Sigma}{\sigma^T \sigma}$$

```
excess_returns_wider =  excess_returns_table |> pivot_wider(names_from = symbol, values_fr

M = as.matrix(excess_returns_wider[ ,2:12])

# Variance-Covariance Matrix
cat("Variance-Covariance Matrix")
```

```
Variance-Covariance Matrix
```

```
cat("\n")
```

```
round(var(M), 4)
```

```
         TLT   MSFT    HPQ   COST      F    PEP     JPM     RTX     PG     CVX
TLT   0.0022 0.0011 0.0002 0.0013 0.0006 0.0003 -0.0003 -0.0005 0.0001 -0.0008
MSFT  0.0011 0.0040 0.0026 0.0021 0.0032 0.0013  0.0018  0.0016 0.0012  0.0015
HPQ   0.0002 0.0026 0.0084 0.0016 0.0045 0.0013  0.0039  0.0031 0.0011  0.0035
COST  0.0013 0.0021 0.0016 0.0041 0.0037 0.0013  0.0014  0.0010 0.0012  0.0014
F     0.0006 0.0032 0.0045 0.0037 0.0171 0.0021  0.0064  0.0037 0.0011  0.0067
PEP   0.0003 0.0013 0.0013 0.0013 0.0021 0.0021  0.0014  0.0019 0.0016  0.0020
JPM  -0.0003 0.0018 0.0039 0.0014 0.0064 0.0014  0.0065  0.0042 0.0010  0.0050
RTX  -0.0005 0.0016 0.0031 0.0010 0.0037 0.0019  0.0042  0.0074 0.0017  0.0048
PG    0.0001 0.0012 0.0011 0.0012 0.0011 0.0016  0.0010  0.0017 0.0026  0.0010
CVX  -0.0008 0.0015 0.0035 0.0014 0.0067 0.0020  0.0050  0.0048 0.0010  0.0093
JNJ  -0.0001 0.0012 0.0012 0.0009 0.0015 0.0014  0.0016  0.0017 0.0013  0.0021
```

```
         JNJ
TLT  -0.0001
MSFT  0.0012
HPQ   0.0012
COST  0.0009
F     0.0015
PEP   0.0014
JPM   0.0016
RTX   0.0017
PG    0.0013
CVX   0.0021
JNJ   0.0024
```

```
cat("\n")
```

```
cor_mat = cor(M)

# Correlation Matrix with percentage
cat("Correlation Matrix as percentages")
```

```
Correlation Matrix as percentages
```

```
cat("\n")
```

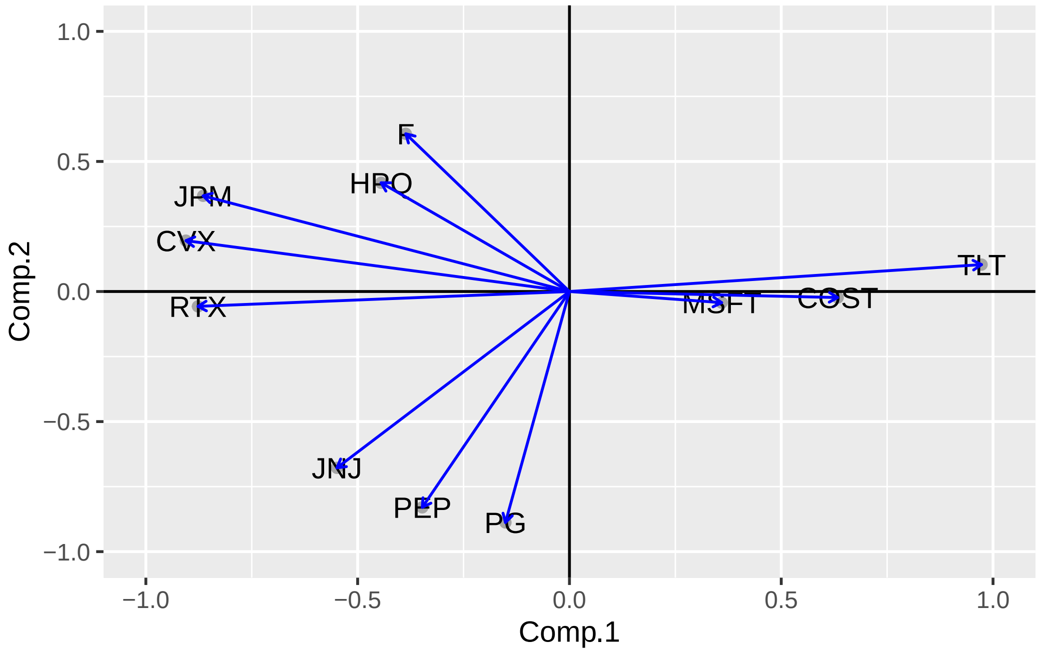
```
round(cor_mat * 100, 3)
```

```
         TLT    MSFT     HPQ    COST       F     PEP     JPM     RTX      PG
TLT  100.000  35.842   5.817  44.883  10.673  15.342  -7.346 -12.845   5.357
MSFT  35.842 100.000  45.846  51.514  39.485  44.252  35.511  29.832  37.324
HPQ    5.817  45.846 100.000  27.962  37.762  32.054  52.444  38.962  24.309
COST  44.883  51.514  27.962 100.000  44.728  44.217  26.540  17.660  35.222
F     10.673  39.485  37.762  44.728 100.000  35.504  61.046  32.734  16.498
PEP   15.342  44.252  32.054  44.217  35.504 100.000  38.843  47.241  67.462
JPM   -7.346  35.511  52.444  26.540  61.046  38.843 100.000  60.866  25.247
RTX  -12.845  29.832  38.962  17.660  32.734  47.241  60.866 100.000  37.344
PG     5.357  37.324  24.309  35.222  16.498  67.462  25.247  37.344 100.000
CVX  -17.916  24.219  39.735  22.026  53.042  44.369  63.913  57.958  19.748
JNJ   -5.135  38.110  25.718  27.999  23.067  63.610  41.401  39.104  49.764
         CVX     JNJ
```

```
TLT   -17.916  -5.135
MSFT   24.219  38.110
HPQ    39.735  25.718
COST   22.026  27.999
F      53.042  23.067
PEP    44.369  63.610
JPM    63.913  41.401
RTX    57.958  39.104
PG     19.748  49.764
CVX   100.000  44.973
JNJ    44.973 100.000
```

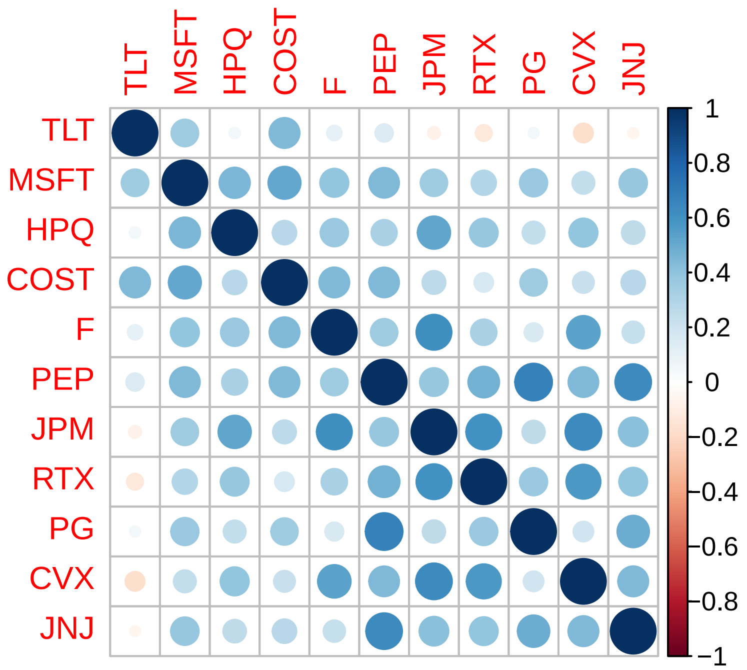
We run a *Principal Component Analysis* calculation to display the groupings and correlations among the various assets in the pearl fund. The graph below displays the groups of closely related or correlated associations with certain stocks. The horizontal axis (Comp.1) and the vertical axis (Comp.2) represent the first two principal components derived from the PCA. These are the two dimensions that capture the most variance in the data set. From an investment standpoint, this plot might help in understanding which stocks move together (correlated), and potentially, it might help in identifying which stocks contribute most to the variance in the portfolio. This could be used for risk management and diversification; for instance, once might choose to invest in stocks that are less correlated with each other according to the PCA results. For example, the biplot shows that **TLT**, which is an ETF that tracks long-term treasury bonds, has a very different loading compared to other assets, indicating that its price movements are likely less correlated with those of the stocks.

```r
pca = princomp(cor_mat, cor = TRUE)
pca_df = as.data.frame(cor(cor_mat, pca$scores))

suppressWarnings(suppressPackageStartupMessages({pca_df |>
  ggplot(aes(x = Comp.1, y = Comp.2)) +
  geom_point(alpha = 0.3) +
  geom_text(label = rownames(pca_df)) +
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = 0) +
  scale_x_continuous(limits = c(-1,1)) +
  scale_y_continuous(limits = c(-1,1)) +

  # Adding arrows from origin to points
  geom_segment(aes(x = 0, y = 0,
                   xend = Comp.1, yend = Comp.2),
               arrow = arrow(length = unit(0.05, "inches")),
               color = "blue",
               data = subset(pca_df, rownames(pca_df) %in% c("TLT", "MSFT", "HPQ", "COST",
               size = 0.5,
               lineend = "round")}))
```



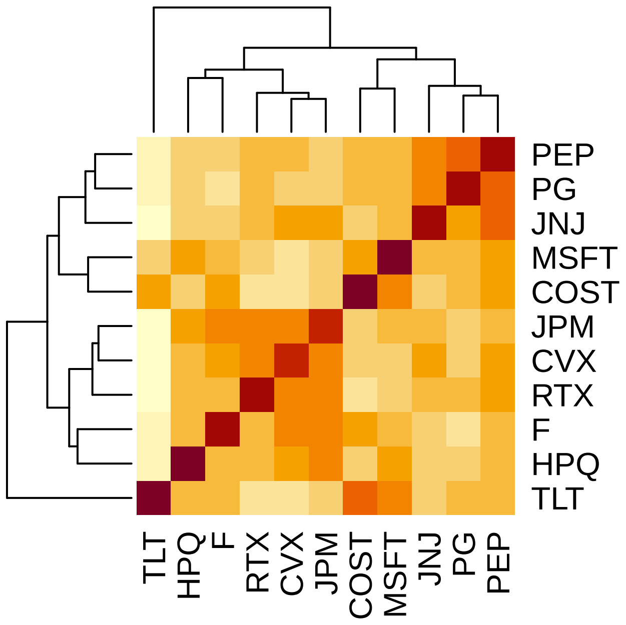
The following visuals will help aid the user with further understanding the correlations between the certain assets and larger groupings. Below we include a correlogram & a heat-map. The correlogram displays the correlations of various stock pairs on a grid to display the direction (blue shows + and red shows -) and the intensity (increasing size/shade of dot shows a value closer to a magnitude of 1). A white box shows a correlation of 0, ruling out a linear association.

```
corrplot(cor_mat, method = "circle")
```



The dendrogram below shows another display of correlation pairings as well as clustering of certain groups of assets to show similarities/relatedness.

```
heatmap(cor_mat)
```



Benchmark Performance

Upon acquiring the historical monthly stock returns for our designated time frame, our focus shifts to the evaluation of the two ETFs of the comparison benchmark index funds—SPDR S&P 500 ETF Trust (NYSE: SPY) and NASDAQ COMPOSITE (NYSE: QQQ). We display the 5 most recent SPY & NASDAQ Monthly returns available.

S&P 500 (Monthly % Returns):

```
# Raw S&P 500 Monthly Returns
spy_returns_monthly_raw <- tq_get("SPY", get = "stock.prices", from = "2004-01-01", to = "

# Transformed S&P 500 Monthly Returns
spy_returns_monthly = tq_transmute(spy_returns_monthly_raw, select = adjusted, mutate_fun

# Transforms data into percentages
spy_returns_monthly_percentages = data.frame("date" = spy_returns_monthly$date, "SPYPercen

# Displays the most recent 5 monthly SPY returns (%)
spy_returns_monthly_percentages_decreasing =  spy_returns_monthly_percentages[order(spy_re
```

```r
head(spy_returns_monthly_percentages_decreasing[, c("date", "SPYPercentReturns")], 5)
```

```
          date SPYPercentReturns
240 2023-12-29          4.565537
239 2023-11-30          9.134381
238 2023-10-31         -2.170868
237 2023-09-29         -4.743442
236 2023-08-31         -1.625202
```

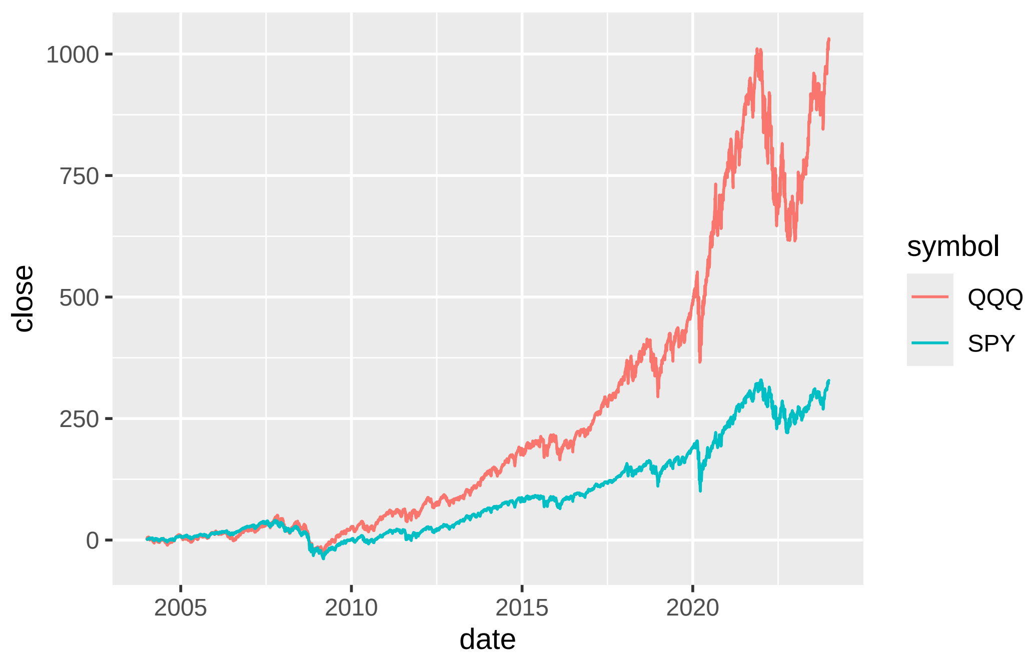
NASDAQ (Monthly % Returns):

```r
# Raw NASDAQ Monthly Returns
nasdaq_returns_monthly_raw <- tq_get("QQQ", get = "stock.prices", from = "2004-01-01", to

# Transformed NASDAQ Monthly Returns
nasdaq_returns_monthly = tq_transmute(nasdaq_returns_monthly_raw, select = adjusted, mutat

# Transforms data into percentages
nasdaq_returns_monthly_percentages = data.frame("date" = nasdaq_returns_monthly$date, "NAS

# Displays the most recent 5 monthly NASDAQ returns (%)
nasdaq_returns_monthly_percentages_decreasing =  nasdaq_returns_monthly_percentages[order(

head(nasdaq_returns_monthly_percentages_decreasing[, c("date", "NASDAQPercentReturns")], 5
```

```
          date NASDAQPercentReturns
240 2023-12-29             5.586963
239 2023-11-30            10.818816
238 2023-10-31            -2.065471
237 2023-09-29            -5.079875
236 2023-08-31            -1.483004
```

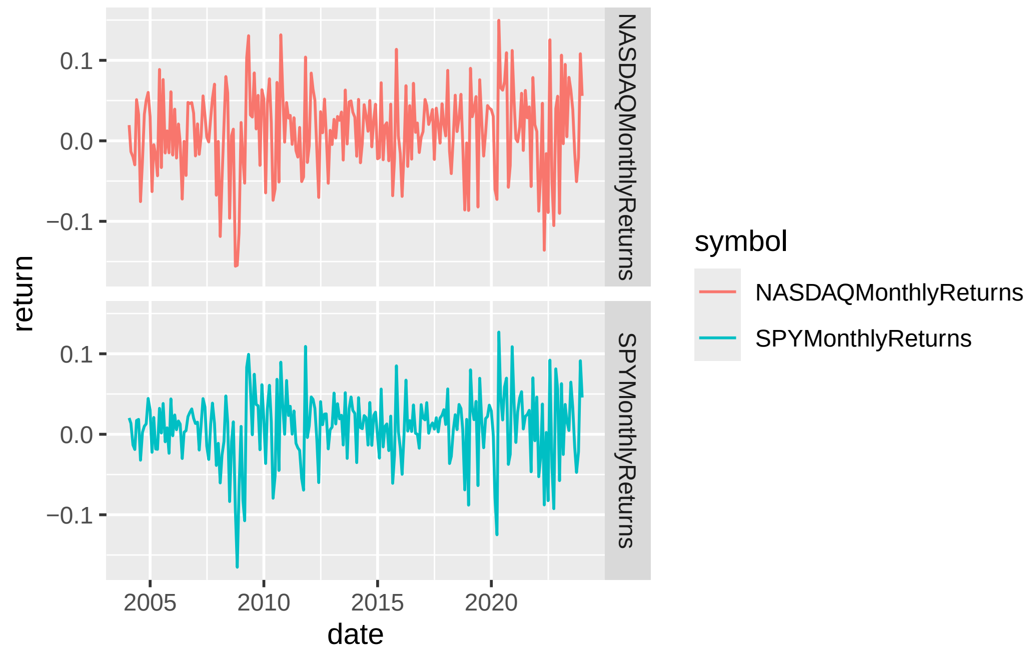
The line graph below illustrates the price trends over our time period for the benchmark returns — SPY and QQQ. As before, we standardize the index prices to facilitate a more effective comparison.

```r
# Combining benchmark raw dataframes
benchmark_returns_raw = rbind(spy_returns_monthly_raw, nasdaq_returns_monthly_raw)

benchmark_returns_raw %>% group_by(symbol) %>% mutate(close = 100*(close - first(close))/f
```



Now, we proceed to consolidate the returns of the two baseline index funds into a unified table. Additionally, a graph is presented to visually depict the combined returns of the two benchmarks within a single frame.

```
benchmark_combined_table_of_returns = left_join(spy_returns_monthly, nasdaq_returns_monthl

benchmark_long = benchmark_combined_table_of_returns |>
  pivot_longer(cols = 2:3, values_to = "return", names_to = "symbol") |>
  arrange(symbol)

benchmark_long |>
  ggplot(aes(x = date, y = return, color = symbol)) +
  geom_line() +
  facet_grid(symbol ~ .)
```



Portfolio Optimization

The groundwork laid by our analysis and statistical metrics sets the stage for the upcoming portfolio optimization process. Drawing inspiration from the tenets of Modern Portfolio Theory (MPT), our objective is to optimize the portfolio of assets. Employing a mean-variance framework analysis, we seek to maximize expected returns for a specified level of risk tolerance, represented by standard deviation. We will also be utilizing other constraints and objectives to customize what we aim with our financial returns. Our approach involves leveraging concepts such as the efficient set mathematics, efficient frontier model, and other relevant methodologies.

Equally Weighted Portfolio Initialization

```
init_weights = rep((1/length(stock_tickers)), length(stock_tickers))
assets = setNames(init_weights, stock_tickers)
assets
```

```
       TLT       MSFT        HPQ       COST          F        PEP        JPM
0.09090909 0.09090909 0.09090909 0.09090909 0.09090909 0.09090909 0.09090909
```

```
       RTX         PG        CVX        JNJ 
0.09090909 0.09090909 0.09090909 0.09090909 
```

Next, using our optimization libraries we create a portfolio object where we will input our asset, initialized weights, portfolio constraints, and our objectives. Our desired outcome will be a vector of weights which optimizes for our objectives under our desired constraints.

Comments on Constraints

1. Weight Sum -> sets a range of values for the total weight of out portfolio assets
2. Leverage -> sets a range of values to depict additional borrowing
3. Box -> sets a range of values for which the assets can be weighted at
4. Return -> sets a target return rate for our portfolio to achieve

Comments on Objectives

1. Return -> aims to create a portfolio to maximize the return
2. Risk Aversion Parameter -> sets the level of risk aversion for the optimization. A higher risk aversion value indicates a greater emphasis on minimizing risk in the portfolio.
3. Risk -> aims to create a portfolio which minimizes the risk (StdDev, var, etc..)

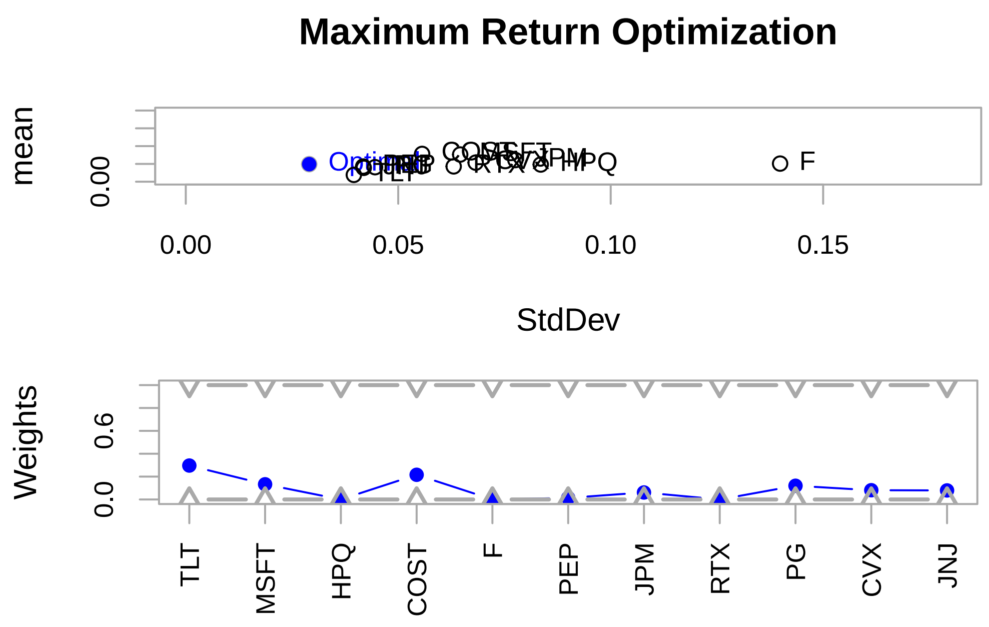
```r
pearlfund = portfolio.spec(assets = assets)
print(pearlfund)
```

```
**************************************************
PortfolioAnalytics Portfolio Specification 
**************************************************

Call:
portfolio.spec(assets = assets)

Number of assets: 11 
Asset Names
 [1] "TLT"  "MSFT" "HPQ"  "COST" "F"    "PEP"  "JPM"  "RTX"  "PG"   "CVX" 
More than 10 assets, only printing the first 10
```

```r
# CONSTRAINTS

pearlfund = add.constraint(portfolio=pearlfund, type="weight_sum", min_sum=1, max_sum=1)
# pearlfund <- add.constraint(portfolio=pearlfund, type="leverage", min_sum=0.99, max_sum=
pearlfund <- add.constraint(portfolio=pearlfund, type="box", min=0.00, max=1)
# pearlfund = add.constraint(portfolio=pearlfund, type="return", return_target=0.0150)
# OBJECTIVES

pearlfund = add.objective(portfolio=pearlfund, type='return', name='mean')
pearlfund = add.objective(portfolio = pearlfund, type='risk', name="var", risk_aversion=7)
# pearlfund = add.objective(pearlfund, type = "risk", name = "StdDev")

# OPTIMIZE.PORTFOLIO

opt_maxret <- optimize.portfolio(R=stock_returns_monthly_full_wide, portfolio=pearlfund, o

# OPTIMAL WEIGHTS

print(opt_maxret)
```

```
***********************************
PortfolioAnalytics Optimization
***********************************

Call:
optimize.portfolio(R = stock_returns_monthly_full_wide, portfolio = pearlfund,
    optimize_method = "ROI", trace = TRUE)

Optimal Weights:
   TLT   MSFT    HPQ   COST      F    PEP    JPM    RTX     PG    CVX    JNJ
0.2966 0.1328 0.0000 0.2159 0.0000 0.0137 0.0612 0.0000 0.1199 0.0808 0.0791

Objective Measure:
    mean
0.009853


 StdDev
0.02905
```

```
# VISUALIZATIONS

plot(opt_maxret, risk.col="StdDev", return.col="mean", main="Maximum Return Optimization",
```



Our visual above displays a line graph of the associated weights of our assets and a scatter plot comparing the mean vs. StdDev of the individual portfolio compared to the mean vs. StdDev of the portfolio as an aggregate whole.

The following code extracts the weights from our portfolio object and stores it as a vector of weights.

```
optimized_list_of_weights = opt_maxret[1]$weights
optimized_vector_of_weights = unlist(optimized_list_of_weights)
optimized_vector_of_weights
```

```
          TLT          MSFT           HPQ          COST             F
 2.965681e-01  1.328091e-01 -4.072160e-19  2.158554e-01  2.809125e-18
          PEP           JPM           RTX            PG           CVX
 1.372503e-02  6.119163e-02 -4.589240e-19  1.199263e-01  8.083995e-02
          JNJ
 7.908454e-02
```

Optimized Portfolio Visualizations

Fund Total Growth Return

The visual below displays how much you would have right now if you had invested $10,000 at the start of the time period till today.

> *Please note that this does not account for tax consequences, which could reduce your final capital depending on tax policy and brackets

```r
# Data Retstructuring

# Pearl Fund

aggregate_portfolio_monthly_returns_growth <- stock_returns_monthly_long %>%
    tq_portfolio(assets_col = symbol,
                 returns_col= `MonthlyReturns`,
                 weights = optimized_vector_of_weights,
                 col_rename    = "investment.growth",
                 wealth.index = TRUE) %>%
  mutate(investment.growth = investment.growth * 10000)
```

```
Warning in check_weights(weights, assets_col, map, x): Sum of weights must be 1.
```

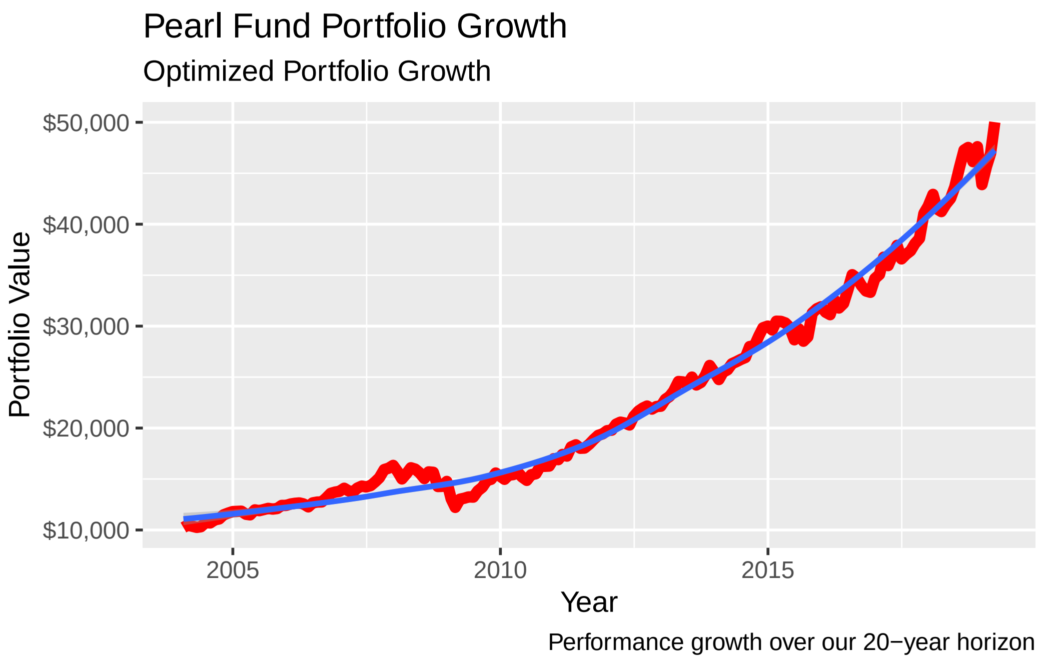
```r
# Pearl Fund - Equally Weighted

equally_weighted_portfolio_monthly_returns_growth <- stock_returns_monthly_long %>%
    tq_portfolio(assets_col = symbol,
                 returns_col= `MonthlyReturns`,
                 weights = init_weights,
                 col_rename    = "investment.growth",
                 wealth.index = TRUE) %>%
  mutate(investment.growth = investment.growth * 10000)
```

```
Warning in check_weights(weights, assets_col, map, x): Sum of weights must be 1.
```

```r
# SPY Portfolio

vals1 <- rep("SPY", 240)
spy_returns_monthly_ticker = spy_returns_monthly
```

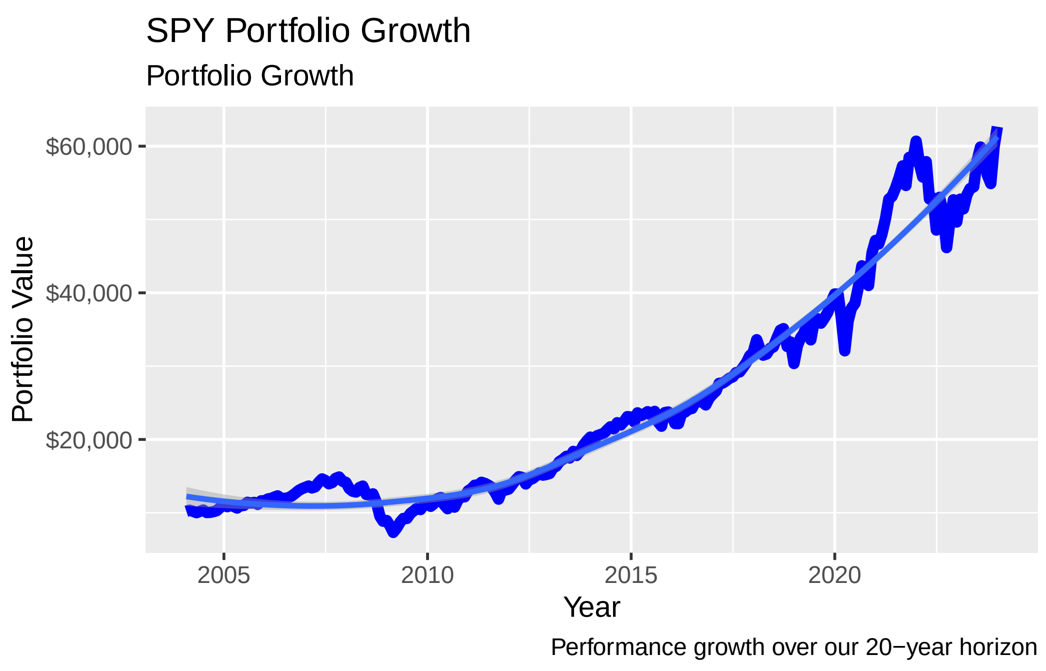
```r
spy_returns_monthly_ticker$symbol <- vals1

spy_returns_monthly_ticker_growth <- spy_returns_monthly_ticker %>%
    tq_portfolio(assets_col = symbol,
                 returns_col= `SPYMonthlyReturns`,
                 weights = c(1),
                 col_rename   = "investment.growth",
                 wealth.index = TRUE) %>%
  mutate(investment.growth = investment.growth * 10000)

# NASDAQ Portfolio

vals2 = rep("NASDAQ", 240)
nasdaq_returns_monthly_ticker = nasdaq_returns_monthly
nasdaq_returns_monthly_ticker$symbol <- vals2

nasdaq_returns_monthly_ticker_growth <- nasdaq_returns_monthly_ticker %>%
    tq_portfolio(assets_col = symbol,
                 returns_col= `NASDAQMonthlyReturns`,
                 weights = c(1),
                 col_rename   = "investment.growth",
                 wealth.index = TRUE) %>%
  mutate(investment.growth = investment.growth * 10000)

# Pearl Fund Portfolio Growth
ggplot(aggregate_portfolio_monthly_returns_growth, aes(x = date, y = investment.growth)) +
  geom_line(linewidth = 2, color = "red") +
  geom_smooth(method = "loess") +
  labs(title = "Pearl Fund Portfolio Growth",
       subtitle = "Optimized Portfolio Growth",
       caption = "Performance growth over our 20-year horizon",
       x = "Year", y = "Portfolio Value") +
  scale_y_continuous(labels = scales::dollar)
```

```
`geom_smooth()` using formula = 'y ~ x'
```



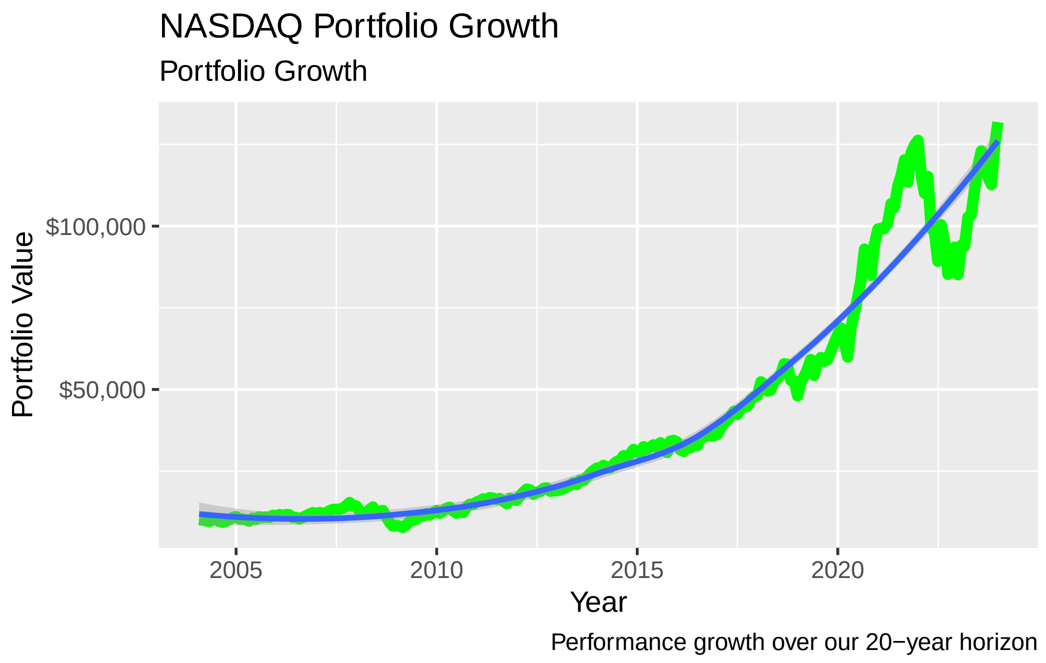
```r
# SPY Portfolio Growth
ggplot(spy_returns_monthly_ticker_growth, aes(x = date, y = investment.growth)) +
  geom_line(linewidth = 2, color = "blue") +
  geom_smooth(method = "loess") +
  labs(title = "SPY Portfolio Growth",
       subtitle = "Portfolio Growth",
       caption = "Performance growth over our 20-year horizon",
       x = "Year", y = "Portfolio Value") +
  scale_y_continuous(labels = scales::dollar)
```

```
`geom_smooth()` using formula = 'y ~ x'
```



```
# NASDAQ Portfolio Growth
ggplot(nasdaq_returns_monthly_ticker_growth, aes(x = date, y = investment.growth)) +
  geom_line(linewidth = 2, color = "green") +
  geom_smooth(method = "loess") +
  labs(title = "NASDAQ Portfolio Growth",
       subtitle = "Portfolio Growth",
       caption = "Performance growth over our 20-year horizon",
       x = "Year", y = "Portfolio Value") +
  scale_y_continuous(labels = scales::dollar)
```

```
`geom_smooth()` using formula = 'y ~ x'
```



Fund Monthly Returns

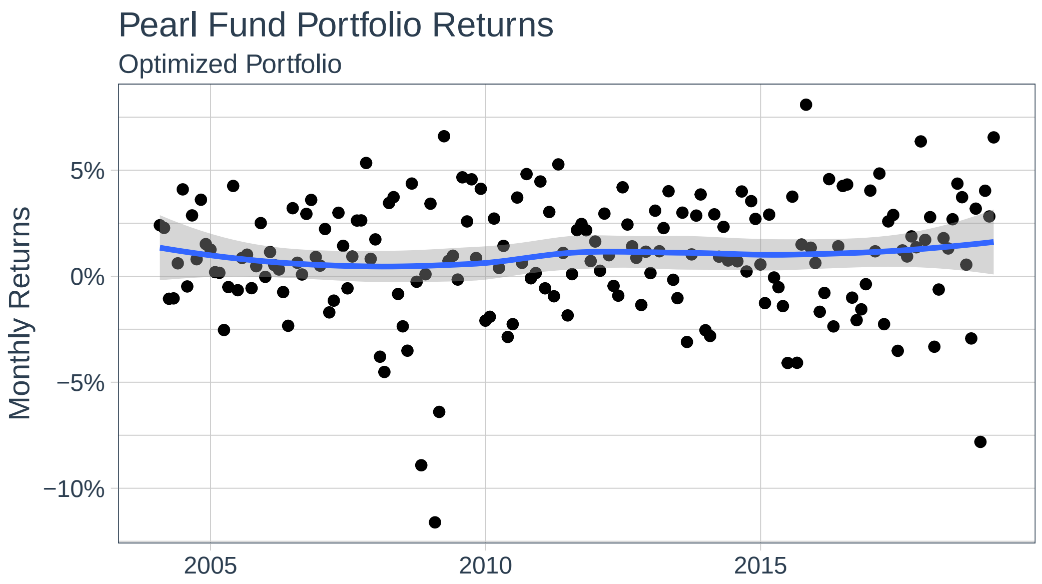
The following visual displays a scatter plot of the monthly returns throughout the time period of interest. A trend line (using "loess") is fitted onto the graph to display the variation and direction at different regions of the scatter plot.

```
aggregate_portfolio_monthly_returns = stock_returns_monthly_long %>% tq_portfolio(assets_c
```

```
Warning in check_weights(weights, assets_col, map, x): Sum of weights must be 1.
```

```
aggregate_portfolio_monthly_returns %>% ggplot(aes(x = date, y = `AggregateMonthlyReturns`
  geom_smooth(method = "loess") +
  theme_tq() + scale_color_tq() + scale_y_continuous(labels = scales::percent)
```

```
`geom_smooth()` using formula = 'y ~ x'
```



Shows an above−zero trend indicating long−term positive returns

Annualized Returns

To get a better appreciation for the fund's return, we proceed to annualize the the monthly returns.

```
portfolio_monthly_mean = unname(unlist(opt_maxret[2]$objective_measures$mean))

# Convert to annualized % return
annualized_return <- paste0(100*((1 + portfolio_monthly_mean)^12 - 1), "%")
names(annualized_return) = "Annualized Return"

cat("Annualized Return:", annualized_return)
```

```
Annualized Return: 12.4864655712676%
```

Comparative Statistical Analysis

To objectively test our portfolio, we will be calculating several financial metrics in relation to both our benchmarks — SPY & QQQ.

Valuable tq_performance tests... *table.Stats, table.CAPM, table.AnnualizedReturns, table.Correlation, table.DownsideRisk, table.DownsideRiskRatio, table.HigherMoments, table.InformationRatio, table.Variability, VaR, SharpeRatio*

To begin, it is important to combine the benchmark index monthly returns with our portfolio monthly returns.

```
# Portfolio + S&P500 Table
spy_combined_table_of_returns = left_join(aggregate_portfolio_monthly_returns, spy_returns

# Portfolio + NASDAQ Table
nasdaq_combined_table_of_returns = left_join(aggregate_portfolio_monthly_returns, nasdaq_r

# Displays the most recent 5 monthly returns for the tables above
spy_combined_table_of_returns_decreasing =  spy_combined_table_of_returns[order(spy_combin
head(spy_combined_table_of_returns_decreasing, 5)
```

```
# A tibble: 5 x 3
  date       AggregateMonthlyReturns SPYMonthlyReturns
  <date>                       <dbl>             <dbl>
1 2019-03-29                  0.0655            0.0181
2 2019-02-28                  0.0282            0.0324
3 2019-01-31                  0.0403            0.0801
4 2018-12-31                 -0.0781           -0.0880
5 2018-11-30                  0.0319            0.0185
```

```
nasdaq_combined_table_of_returns_decreasing =  nasdaq_combined_table_of_returns[order(nasd
head(nasdaq_combined_table_of_returns_decreasing, 5)
```

```
# A tibble: 5 x 3
  date       AggregateMonthlyReturns NASDAQMonthlyReturns
  <date>                       <dbl>                <dbl>
1 2019-03-29                  0.0655              0.0392
2 2019-02-28                  0.0282              0.0299
3 2019-01-31                  0.0403              0.0901
4 2018-12-31                 -0.0781             -0.0866
5 2018-11-30                  0.0319             -0.00265
```

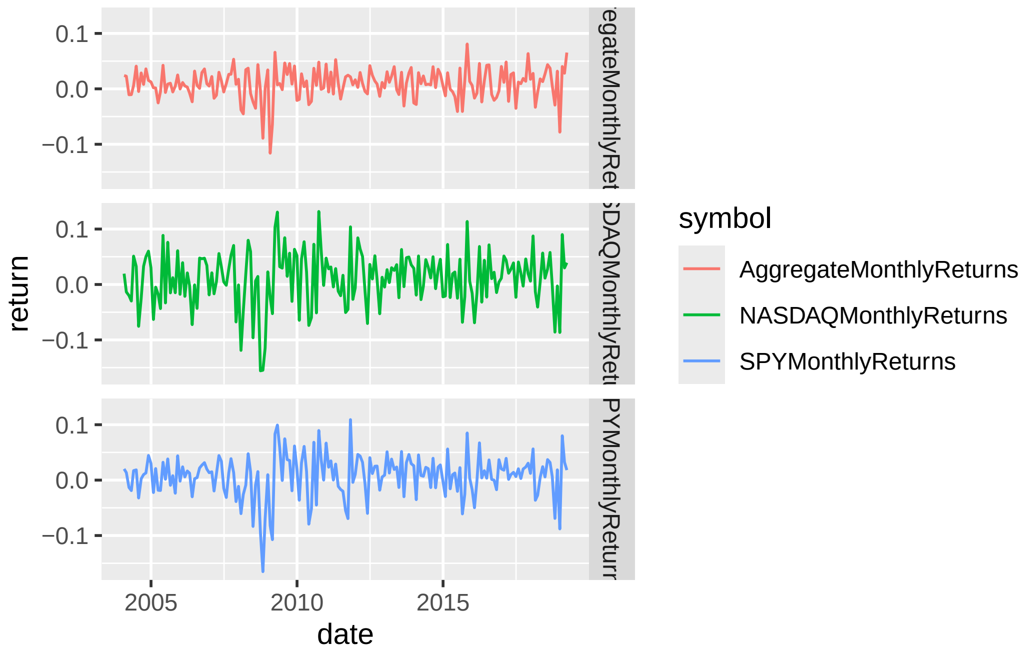
The following code combines all the funds of interest that intend to be comparing into a single table (wide format) once again with the 5 most recent returns displayed.

```r
# Combined Table (3 Funds)
three_fund_combined_table_of_returns = left_join(spy_combined_table_of_returns, nasdaq_ret

three_fund_combined_table_of_returns_decreasing <- arrange(three_fund_combined_table_of_re

head(three_fund_combined_table_of_returns_decreasing, 5)
```

```
# A tibble: 5 x 4
  date       AggregateMonthlyReturns SPYMonthlyReturns NASDAQMonthlyReturns
  <date>                       <dbl>             <dbl>                <dbl>
1 2019-03-29                  0.0655            0.0181              0.0392
2 2019-02-28                  0.0282            0.0324              0.0299
3 2019-01-31                  0.0403            0.0801              0.0901
4 2018-12-31                 -0.0781           -0.0880             -0.0866
5 2018-11-30                  0.0319            0.0185             -0.00265
```

This is a great display of a side by side view of the monthly returns for our three funds of comparison for our time period of interest. The following visuals are a great measure of the overall volatility of our portfolio compared to the two benchmarks. Graphs that display more “ups” and “downs” with monthly returns indicate larger deviations from the mean. Given similar annualized returns and expected long term value, a risk-averse investor should prefer the portfolios/funds which minimize the volatility/risk for the given time period.

```r
three_fund_long = three_fund_combined_table_of_returns |>
  pivot_longer(cols = 2:4, values_to = "return", names_to = "symbol") |>
  arrange(symbol)

three_fund_long |>
  ggplot(aes(x = date, y = return, color = symbol)) +
  geom_line() +
  facet_grid(symbol ~ .)
```



The tq_performance function in tidyquant allows us to produce statistical metrics for financial analysis for our Peal Fund against each of the benchmark index funds.

```
# Pearl Fund vs. SPY
statisticsA = spy_combined_table_of_returns %>% tq_performance(Ra = `AggregateMonthlyRetur

# Pearl Fund vs. NASDAQ
statisticsB = nasdaq_combined_table_of_returns %>% tq_performance(Ra = `AggregateMonthlyRe

as.data.frame(statisticsA)
```

```
  ActivePremium  Alpha AnnualizedAlpha  Beta  Beta-  Beta+ Correlation
1        0.0268 0.0055          0.0675 0.498 0.5017 0.4153      0.7008
  Correlationp-value InformationRatio R-squared TrackingError TreynorRatio
1                  0           0.2783    0.4912        0.0963       0.2235
```

```
as.data.frame(statisticsB)
```

```
  ActivePremium  Alpha AnnualizedAlpha   Beta Beta- Beta+ Correlation
1       -0.0084 0.0055          0.0681 0.3477 0.318 0.288      0.6162
  Correlationp-value InformationRatio R-squared TrackingError TreynorRatio
```

1	0	-0.0623	0.3797	0.1342	0.3202

Mean-Variance Analysis (Efficient Frontier)

The efficient frontier is a concept in financial portfolio theory that represents the set of optimal portfolios that offer the highest expected return for a given level of risk, or the lowest risk for a given level of expected return. It is useful for investors and portfolio managers as it helps identify the optimal balance between risk and return, enabling them to construct portfolios that maximize potential gains while minimizing exposure to volatility.

To apply this framework to our portfolio, we first need to modify our monthly stock returns table so that the date column is removed with once again the 5 most recent returns displayed.

```
stock_returns_monthly_wide_dateless = stock_returns_monthly_wide[, -1]

stock_returns_monthly_wide_decreasing <- arrange(stock_returns_monthly_wide, desc(date))

head(stock_returns_monthly_wide_decreasing[, -1], 5)
```

```
# A tibble: 5 x 11
      TLT    MSFT      HPQ    COST        F     PEP      JPM     RTX      PG
    <dbl>   <dbl>    <dbl>   <dbl>    <dbl>   <dbl>    <dbl>   <dbl>   <dbl>
1  0.0557   0.0528 -0.00686  0.107   0.00114  0.0598 -0.0300   0.0256  0.0558
2 -0.0138   0.0774 -0.104    0.0219 -0.00341  0.0347  0.00831  0.0707  0.0216
3  0.00379  0.0282  0.0767   0.0536  0.170    0.0198  0.0688   0.109   0.0578
4  0.0585  -0.0840 -0.104   -0.119  -0.187   -0.0868 -0.122   -0.126  -0.0274
5  0.0179   0.0427 -0.0472   0.0140 -0.0147   0.0851  0.0199  -0.0135  0.0657
# i 2 more variables: CVX <dbl>, JNJ <dbl>
```

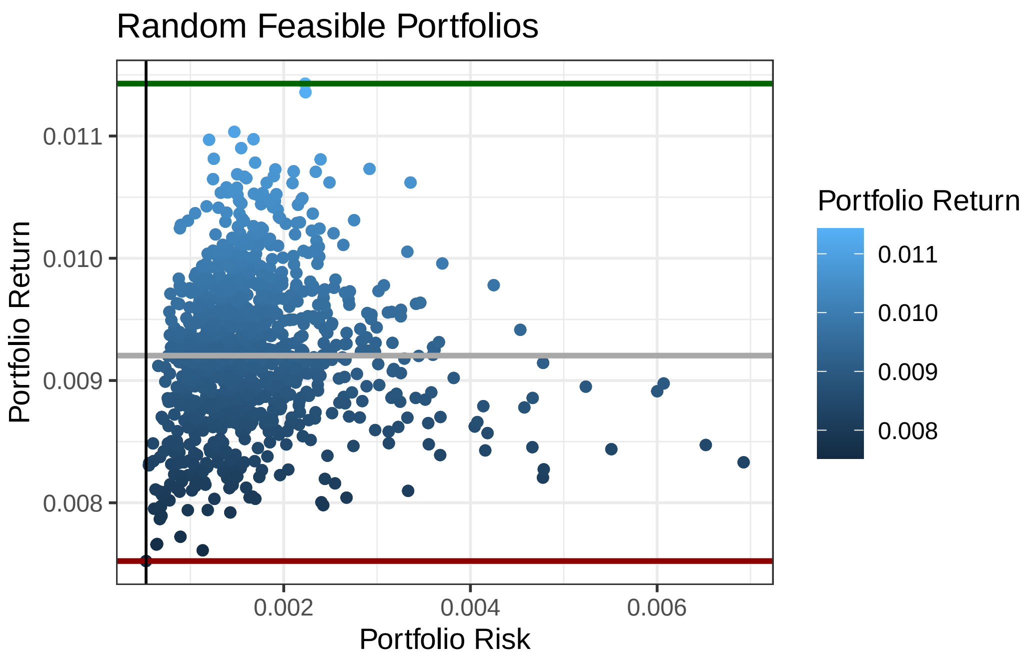
The next step is to simulate, using a loop, a series of random weights which we then apply certain matrices to. The result is a data frame which stores the list of returns and risk values for portfolios of different weights.

```
np1 = 1000
ret2 = stock_returns_monthly_wide_dateless  #excluding dates
mu1 = colMeans(ret2)  #mean returns
na1 = ncol(ret2)  #number of assets
varc1 = cov(ret2)
```

```r
riskp1 = NULL  #vector to store risk
retp1 = NULL  #vector to store returns
# using loops here (not aiming for efficiency but demonstration)

for (i in 1:np1) {
    w = diff(c(0, sort(runif(na1 - 1)), 1))  # random weights
    r1 = t(w) %*% mu1  #matrix multiplication
    sd1 = t(w) %*% varc1 %*% w
    retp1 = rbind(retp1, r1)
    riskp1 = rbind(riskp1, sd1)
}

# create a data frame of risk and return
d_p1 = data.frame(Ret = retp1, Risk = riskp1)
```

We use ggplot to and to visualize the data accordingly and as expected we achieve the curve as derived from the Modern Portfolio Framework.

```r
p1 = ggplot(d_p1, aes(Risk, Ret, colour = Ret))
# scatter plot
p1 = p1 + geom_point()
# scatter plot with density and identified port risk return (highest
# lowest returns and min risk)
p1 + geom_point() + geom_hline(yintercept = c(max(d_p1$Ret), median(d_p1$Ret),
    min(d_p1$Ret)), colour = c("darkgreen", "darkgray", "darkred"), size = 1) +
    geom_vline(xintercept = d_p1[(d_p1$Risk == min(d_p1$Risk)), ][, 2]) +
    labs(colour = "Portfolio Return", x = "Portfolio Risk", y = "Portfolio Return",
        title = "Random Feasible Portfolios") + theme_bw()
```

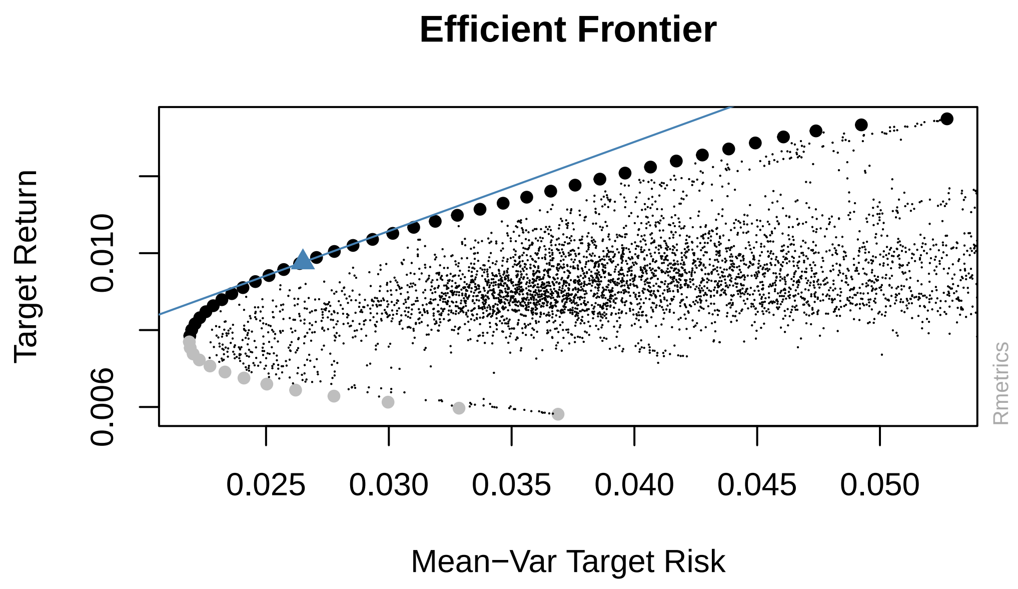


Here is another display of our plot using a different R frontier calculation package. To choose what you'd like to plot exactly in plot(), enter values from below as a vector:

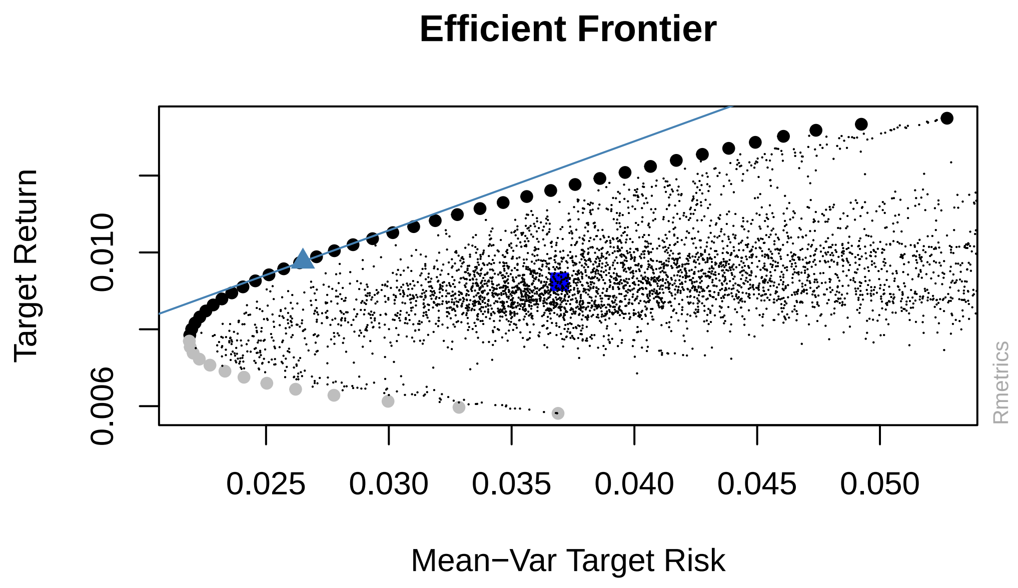
Syntax for choosing vector in plot function below:

1. Efficient Frontier
2. Global Minimum Variance Portfolio
3. Tangent (optimal portfolio)
4. Risk/Return of each asset
5. Equal Weights Portfolio
6. Two Asset Frontier
7. Monte Carlo
8. Sharpe Ratio

```
return.matrix = as.timeSeries(stock_returns_monthly_wide[, -1])

efficient.frontier = portfolioFrontier(return.matrix, `setRiskFreeRate<-`(portfolioSpec(),

plot(efficient.frontier, c(1, 3, 7))
```



```
plot(efficient.frontier, c(1,3,5,7))
```

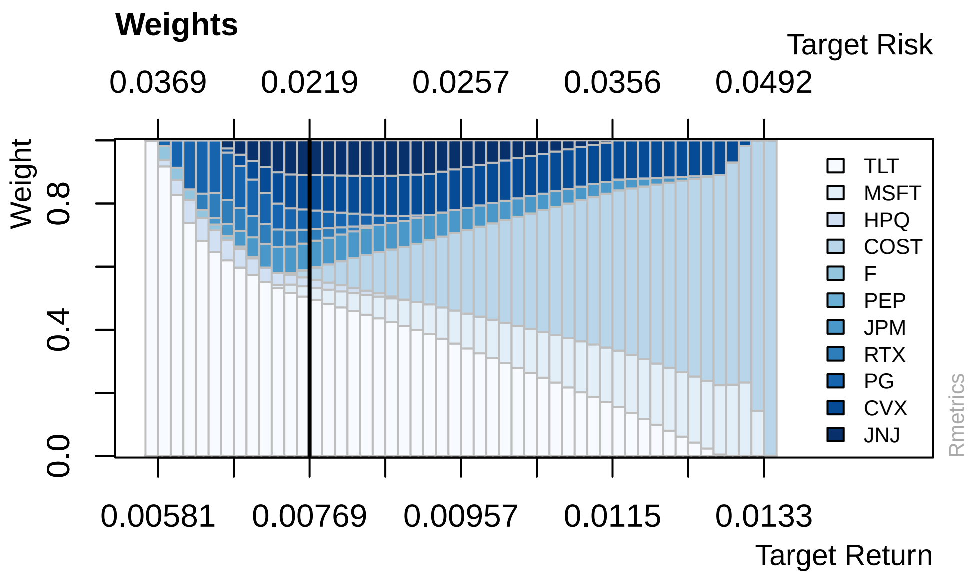


Optimal portfolios for a given target risk are displayed by the dark colored dots on the border or *frontier* of the Monte-Carlo simulated data points. The capital allocation line (blue line) which intersects the efficient frontier at a point of tangency indicating the *optimal portfolio* for an investor at the current risk-free rate (10 year yield on government bond).

The following visual is another very useful metric to understand portfolio weightings based on chosen return/risk levels.

> **Interpretation**
>
> *Pick a vertical line/column at a specified target return and target risk and you can find out the allocations of stocks in your portfolio (through the lengths of the appropriate colored bars) to achieve those chosen metrics.*

```
weightsPlot(efficient.frontier)
```



Here our goal is to produce a different type of frontier visual. Instead of plotting the traditional measures of risk (standard deviation or variance) against returns, we turn to a different measure of risk — *expected tail loss (ETL).* ETL is a another measure of risk which could be a better use in portfolio optimization in certain situations.

> ETL, also known as conditional value at risk (CVaR), is a risk measure that quantifies the expected loss in the tail of a distribution of possible outcomes beyond

a certain confidence level. ETL focuses specifically on the extreme or tail events, giving an indication of the potential loss in those extreme scenarios. The preference for ETL over variance in some cases arises because ETL puts more emphasis on extreme events, which can be crucial in risk assessment. Variance might not adequately capture the potential impact of rare but severe events, whereas ETL provides a clearer picture of the expected loss in the tail of the distribution.

The following code generates a series of ETL values for each of our portfolio weights at a chosen confidence level (95%).

```r
confidence_level <- 0.95
np1 <- length(retp1)
cvar_values <- numeric(np1)
returns <- rnorm(1000, mean(retp1), mean(riskp1))
etl_values <- numeric(np1)

for (i in 1:np1) {
  random_sample <- sample(returns, length(returns), replace = TRUE)
  sorted_returns <- sort(random_sample, decreasing = TRUE)
  tail_index <- floor((1 - confidence_level) * length(returns))
  tail_returns <- sorted_returns[1:tail_index]
  etl_values[i] <- mean(tail_returns)
}
```

We add these ETL values to our data frame of Returns and Risk for our various portfolios.

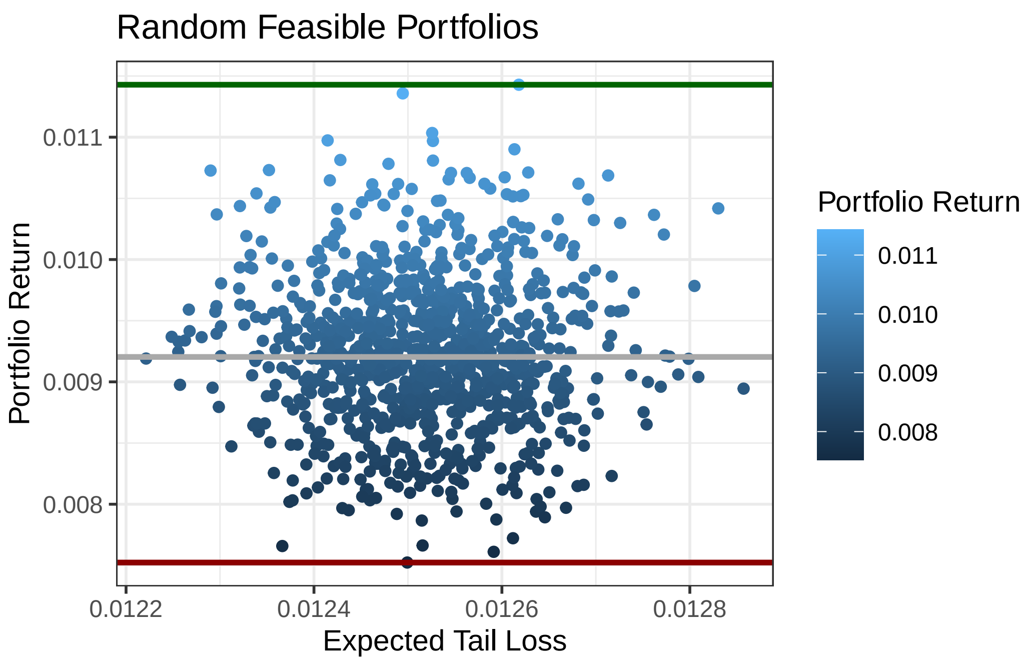
```r
# Add ETL to this table
d_p1 = mutate(d_p1, ETL = etl_values)
head(d_p1)
```

```
          Ret         Risk        ETL
1 0.008711368 0.0009864918 0.01259789
2 0.009627390 0.0034172455 0.01253147
3 0.010037148 0.0011846237 0.01233256
4 0.008696910 0.0020853853 0.01241849
5 0.008726054 0.0010508863 0.01252290
6 0.008599119 0.0015875734 0.01257106
```

Finally, we plot these ETL values against our return values for our portfolios and are able to visualize the results. By plotting ETL vs. Expected Return, you can assess whether the expected returns of the portfolio adequately compensates for the potential magnitude of extreme losses.

```r
p1 <- ggplot(d_p1, aes(ETL, Ret, colour = Ret)) +
  geom_point() +
  geom_hline(yintercept = c(max(d_p1$Ret), median(d_p1$Ret), min(d_p1$Ret)),
             colour = c("darkgreen", "darkgray", "darkred"), size = 1) +
  geom_vline(xintercept = d_p1[(d_p1$Risk == min(d_p1$Risk)), ][, 2]) +
  labs(colour = "Portfolio Return",
       x = "Expected Tail Loss",
       y = "Portfolio Return",
       title = "Random Feasible Portfolios") +
  theme_bw()

p1 + xlim(min(d_p1$ETL), max(d_p1$ETL)) + ylim(min(d_p1$Ret), max(d_p1$Ret))
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_vline()`).
```

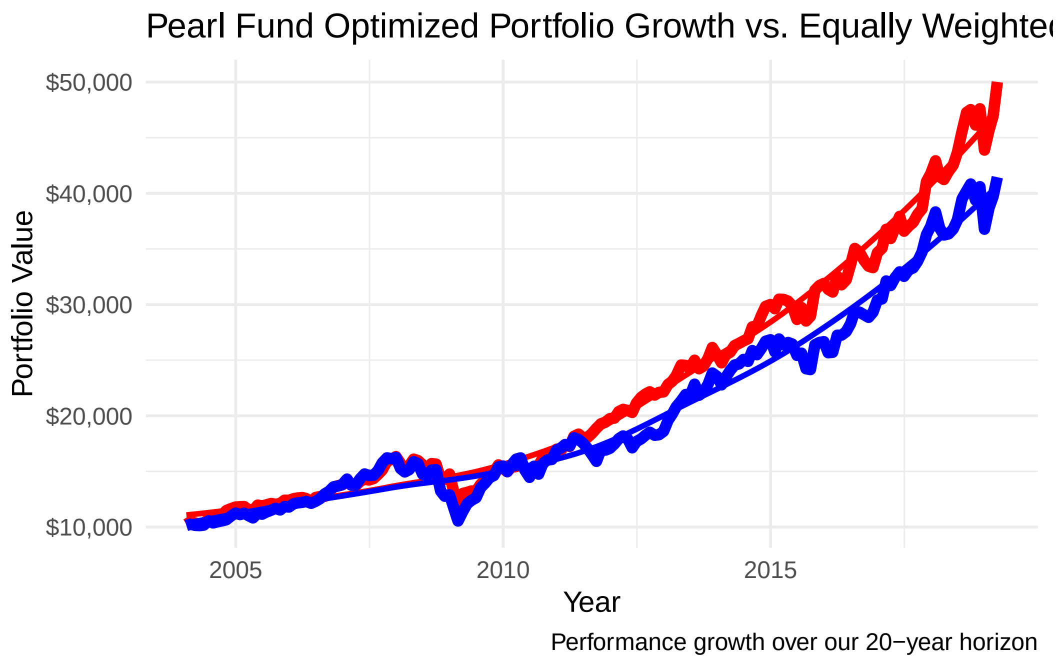


Conclusion

To conclude, we plot the Optimized Pearl Fund Portfolio (in red) versus the Equally Weighted Pearl Fund Portfolio (in blue). As evident in the graph below, the Optimized Portfolio generates higher returns than the Equally Weighted Portfolio, while rising more steadily.

```r
ggplot() +
  geom_line(data = aggregate_portfolio_monthly_returns_growth, aes(x = date, y = investmen
            linewidth = 2, color = "red") +
  geom_smooth(data = aggregate_portfolio_monthly_returns_growth, aes(x = date, y = investm
              method = "loess", color = "red", se = FALSE) +
  geom_line(data = equally_weighted_portfolio_monthly_returns_growth, aes(x = date, y = in
            linewidth = 2, color = "blue") +
  geom_smooth(data = equally_weighted_portfolio_monthly_returns_growth, aes(x = date, y =
              method = "loess", color = "blue", se = FALSE) +
  labs(title = "Pearl Fund Optimized Portfolio Growth vs. Equally Weighted Portfolio",
       subtitle = ,
       caption = "Performance growth over our 20-year horizon",
       x = "Year", y = "Portfolio Value") +
  scale_y_continuous(labels = scales::dollar) +
 theme_minimal()
```

```
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
```


Pearl Fund Optimized Portfolio Growth vs. Equally Weighte
Portfolio Value
$50,000
$40,000
$30,000
$20,000
$10,000
2005
2010
2015
Year
Performance growth over our 20−year horizon