efficient_frontier_3plus

November 7, 2023

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

0.1 Efficient Frontier with 3+ Assets

To map out the efficient frontier with 3+ assets, we need to solve the following minimization problem. Suppose we have N assets with expected return vector   and covariance matrix Σ.

For a desired level of target return $\bar{r}_p$, we must find portfolio weights $\mathbf{w}$ that minimize portfolio variance $\mathbf{w}'\Sigma\mathbf{w}$ subject to attaining the desired target return $\mathbf{w}' \ = \bar{r}_p$ and subject to the budget constraint $\mathbf{w}'\mathbf{1}_N = 1$.

In other words, we must solve the following Lagrangian:

$$\min_{\mathbf{w}} \left\{ \mathbf{w}'\Sigma\mathbf{w} + \lambda_1(\mathbf{w}' \ - \mu_p) + \lambda_2(\mathbf{w}'\mathbf{1}_N - 1) \right\}$$

Taking the derivative with respect to $\mathbf{w}$, we get

$$2\Sigma\mathbf{w} + \lambda_1 \ + \lambda_2\mathbf{1}_N = \mathbf{0}_N$$

Along with the constraints $\mathbf{w}' \ = \mu_p$ and $\mathbf{w}'\mathbf{1}_N = 1$, these three equations constitute an $N + 2$ system in unknowns $\mathbf{w}$, λ_1, and λ_2, which can be written in matrix form as follows:

$$\begin{bmatrix} 2\Sigma & & \mathbf{1}_N \\ ' & 0 & 0 \\ \mathbf{1}_N' & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_N \\ \mu_p \\ 1 \end{bmatrix}$$

Define a function to find the weights, expected returns, and volatilities for an asset space with expected returns , covariance matrix Σ, and desired return $\mathbf{r}$:

[2]:
```
def find_optimal_weights(r, mu, sigma):
    N = len(mu)
    A = np.zeros((N+2,N+2))
    A[:N,:N] = 2*sigma
    A[:N,N] = mu
    A[:N,N+1] = np.ones(N)
    A[N,:N] = mu
    A[N+1,:N] = np.ones(N)

    b = np.zeros((N+2,1))
    b[N] = r
    b[N+1] = 1

    sol = np.linalg.solve(A,b)
    return sol[:N]
```

Try for a specific set of stocks: MSFT, AAPL, UNH, GOOGL, TSLA

[3]:
```
url = "https://www.portfoliovisualizer.com/asset-correlations?
  ↪s=y&symbols=MSFT+AAPL+UNH+GOOGL+TSLA&startDate=09%2F01%2F2013&endDate=08%2F31%2F2023&timePe
df = pd.read_html(url)[0].iloc[:5,:].set_index('Ticker')
df
```

[3]:
```
                                    Name  MSFT  AAPL   UNH GOOGL  TSLA  \
Ticker
MSFT              Microsoft Corporation  1.00  0.58  0.24  0.63  0.35
AAPL                         Apple Inc.  0.58  1.00  0.28  0.45  0.46
UNH     UnitedHealth Group Incorporated  0.24  0.28  1.00  0.23  0.19
GOOGL                     Alphabet Inc.  0.63  0.45  0.23  1.00  0.33
TSLA                          Tesla Inc  0.35  0.46  0.19  0.33  1.00

       Annualized Return Daily Standard Deviation Monthly Standard Deviation  \
Ticker
MSFT              27.92%                    1.71%                      6.10%
AAPL              28.61%                    1.80%                      7.98%
UNH               22.71%                    1.61%                      5.73%
GOOGL             20.45%                    1.76%                      6.93%
TSLA              36.79%                    3.54%                     17.95%

       Annualized Standard Deviation
Ticker
MSFT                          21.13%
AAPL                          27.66%
UNH                           19.87%
GOOGL                         24.02%
TSLA                          62.17%
```

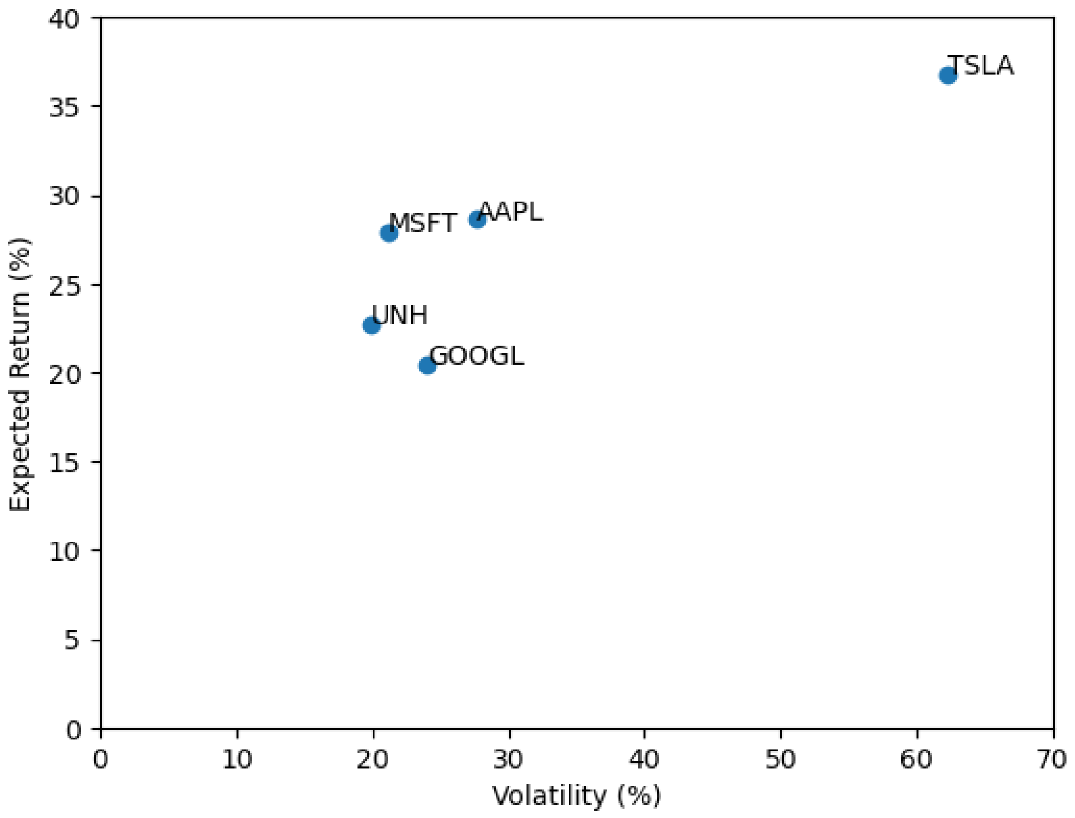
Extract correlations, average past returns, and standard deviations from the data:

[4]:
```
correl = df[['MSFT', 'AAPL', 'UNH', 'GOOGL', 'TSLA']].astype(float).values
meanrets = (df['Annualized Return'].str.strip('%').astype(float)/100).values
vols = (df['Annualized Standard Deviation'].str.strip('%').astype(float)/100).
  ↪values
print("Correlations:\n", correl)
print("Mean Returns:\n", meanrets)
print("Standard Deviations:\n", vols)
```

```
Correlations:
 [[1.   0.58 0.24 0.63 0.35]
 [0.58 1.   0.28 0.45 0.46]
 [0.24 0.28 1.   0.23 0.19]
 [0.63 0.45 0.23 1.   0.33]
 [0.35 0.46 0.19 0.33 1.  ]]
Mean Returns:
 [0.2792 0.2861 0.2271 0.2045 0.3679]
Standard Deviations:
 [0.2113 0.2766 0.1987 0.2402 0.6217]
```

Let's plot these returns in volatility-expected return space:

[5]:
```
fig=plt.figure()
plt.plot(vols*100, meanrets*100, 'o')
[plt.text(vols[i]*100, meanrets[i]*100, df.index[i]) for i in range(len(df))]
plt.xlim(0,70)
plt.ylim(0,40)
plt.xlabel('Volatility (%)')
plt.ylabel('Expected Return (%)')
plt.show()
```



Build covariance matrix:

[6]:
```
covar = correl * (vols.reshape((1,5)) * vols.reshape((5,1)))
```

Let's test out the function with a target return of 30%. In other words, we will find the portfolio of MSFT, AAPL, UNH, GOOGL, and TSLA that has the lowest volatility subject to having an expected return of 30%:

[7]:
```
w = find_optimal_weights(0.30, meanrets, covar)
print('Weights:\n', w)
print('Sum of Weights:', w.sum())
print('Expected Portfolio Return:', w.transpose() @ meanrets)
```

```
Weights:
 [[ 0.87042039]
 [ 0.15679966]
 [ 0.33871954]
 [-0.42732076]
 [ 0.06138117]]
Sum of Weights: 1.0
Expected Portfolio Return:  [0.3]
```

We see that such a portfolio goes short Google and long everything else. For an initial \$100 investment, it is telling us to borrow another \$42.73 by selling Google short and investing the combind \$142.73 as follows:

- $87.04 in Microsoft,
- $15.68 in Apple,
- $33.87 in United Healthcare, and
- $6.14 in Tesla.

What volatility are we expected to get with this portfolio?

[8]:
```
np.sqrt(w.transpose() @ covar @ w)[0,0]
```

[8]:
```
0.20715747407251123
```

To map out the efficient frontier, let's vary the expected return from 0% to 50% and find the portfolio (that is, its weights) with the lowest volatility for each expected return. We will define a vector of target returns and call the function for each target to get a vector of weights.

[9]:
```
portrets = np.linspace(0.1,0.5,1000)
weights = np.array([find_optimal_weights(r, meanrets, covar) for r in portrets])
```

Next, calculate the volatility for each set of weights:

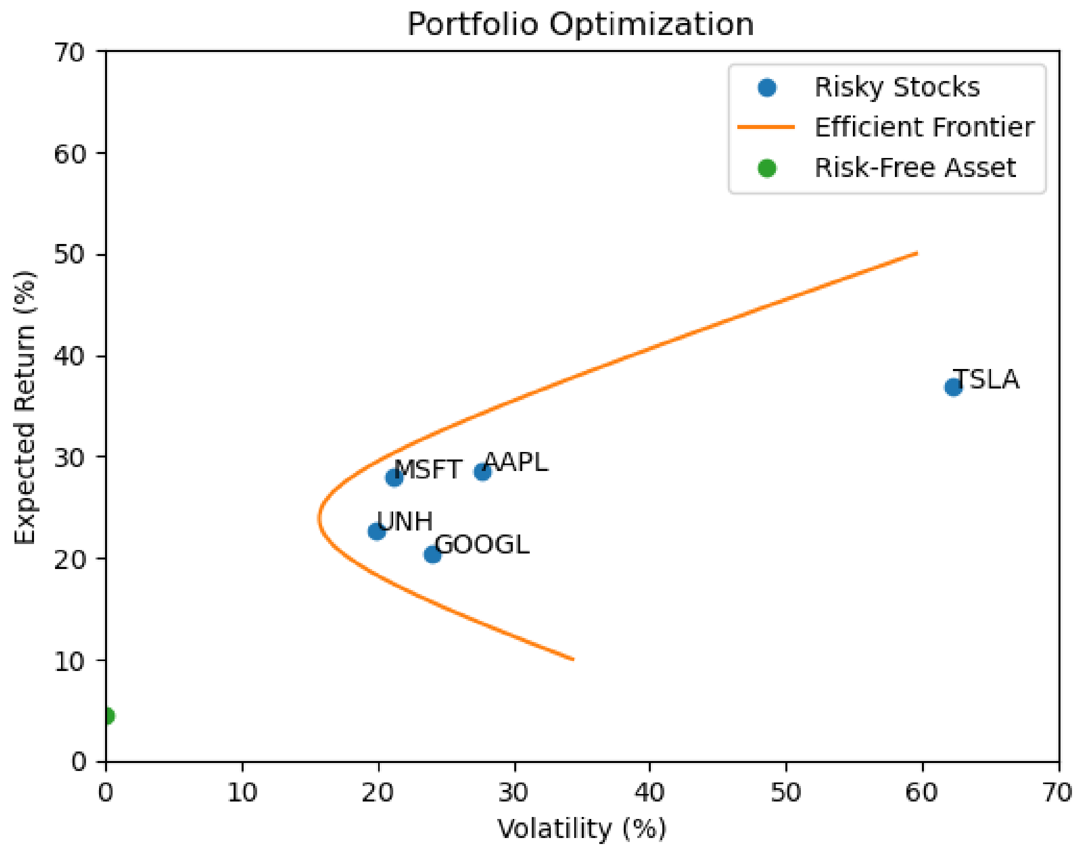
[10]:
```
def vol(w, Σ):
    return np.sqrt(w.transpose() @ Σ @ w)[0,0]
```

```
portvols = np.array([vol(w, covar) for w in weights])
```

Add the efficient frontier and the risk-free rate to our plot:

```
[11]: rf = 0.0458
```

```
[12]: fig=plt.figure()
plt.plot(vols*100, meanrets*100, 'o', label='Risky Stocks')
[plt.text(vols[i]*100, meanrets[i]*100, df.index[i]) for i in range(len(df))]
plt.plot(portvols*100, portrets*100, label='Efficient Frontier')
plt.plot(0, rf*100, 'o', label='Risk-Free Asset')
plt.xlim(0,70)
plt.ylim(0,70)
plt.xlabel('Volatility (%)')
plt.ylabel('Expected Return (%)')
plt.legend()
plt.title('Portfolio Optimization')
plt.show()
```



In principle, we can invest in any combination of the five stocks. However, any portfolio not on the orange line is *inefficient* – it has a higher volatility than necessary to achieve its expected return.

Of all the efficient portfolios, only one maximizes the Sharpe Ratio – the ratio of expected *excess* return per unit of volatility, or reward per unit of risk. The Sharpe Ratio is maximized at the *tangency portfolio*, which is the point where the efficient frontier is tangent to the capital allocation line.

We can find the tangency portfolio two ways. First, we can perform the same exercise we did in the two-asset example, where we calculated the slope of the efficient frontier at every point and found the portfolio where the efficient frontier slope is equal to the slope of the CAL through that portfolio, i.e. where the CAL is tangent to the frontier.

[13]:
```
# CAL Slopes
cal_slopes = (portrets - rf) / (portvols - 0)
cal_slopes=cal_slopes[1:]

# Efficient frontier slopes
frontier_slopes = np.diff(portrets) / np.diff(portvols)
```

[14]:
```
tangency_idx = abs(cal_slopes-frontier_slopes).argmin()
tan_sharpe=cal_slopes[tangency_idx]
tan_expret = portrets[tangency_idx]
tan_vol = portvols[tangency_idx]

print('Tangency Portfolio:\n', weights[tangency_idx,:])
print('Tangency Portfolio Expected Return:', tan_expret)
print('Tangency Portfolio Volatility:', tan_vol)
print('Tangency Portfolio Sharpe Ratio:', tan_sharpe)
```

```
Tangency Portfolio:
 [[ 0.55875795]
 [ 0.09882923]
 [ 0.43100169]
 [-0.09422316]
 [ 0.00563429]]
Tangency Portfolio Expected Return: 0.26496496496496497
Tangency Portfolio Volatility: 0.16756386907313675
Tangency Portfolio Sharpe Ratio: 1.3079521969726158
```

The tangency portfolio consists of a small short position in Google and long positions in the other four stocks. If we start out with a \$100, the tangency portfolio tells us to borrow another \$9.42 by selling Google short and investing the combind \$109.42 as follows:

- $55.88 in Microsoft,
- $9.88 in Apple,
- $43.10 in United Healthcare, and
- $0.56 in Tesla.

Notice how large our investment in UNH is, even though it had the lowest expected return of the four long stocks. This is because it has the lowest volatility of the five stocks, and it is relatively uncorrelated with the other (tech) firms.

Notice how much higher the tangency portfolio Sharpe Ratio is compared to the Sharpe Ratios of each individual asset:

[15]:
```
(meanrets - rf)/vols
```

[15]:
```
array([1.10459063, 0.86876356, 0.9124308 , 0.66069942, 0.51809554])
```

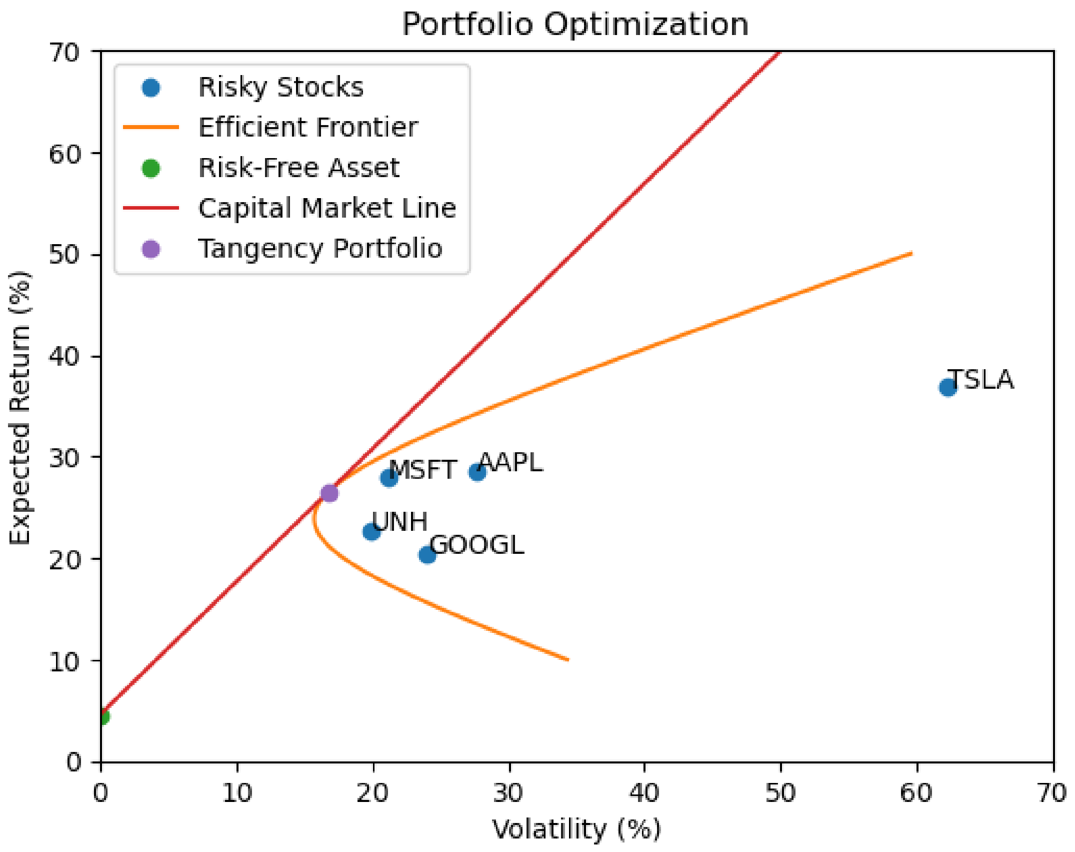
The tangency portfolio defines the mix of risky assets that all (mean-variance maximizing) investors should hold. But which exact portfolio they hold depends on their risk aversion. The more risk averse an investor is, the more they will hold the risk-free asset and the less they will hold the tangency portfolio, and vice versa. The set of all such portfolios is called the *capital market line* (CML) because it is the line that passes through the risk-free rate and the tangency portfolio. In other words, it has

- a slope equal to the Sharpe Ratio of the tangency portfolio, and
- a y-intercept equal to the risk-free rate.

Let's plot both the tangency portfolio and the CML on our graph:

[16]:
```
vol_max = np.max(portvols)
cml_max = rf + tan_sharpe * vol_max
```

[17]:
```
fig=plt.figure()
plt.plot(vols*100, meanrets*100, 'o', label='Risky Stocks')
[plt.text(vols[i]*100, meanrets[i]*100, df.index[i]) for i in range(len(df))]
plt.plot(portvols*100, portrets*100, label='Efficient Frontier')
plt.plot(0, rf*100, 'o', label='Risk-Free Asset')
plt.plot([0, vol_max*100], [rf*100, cml_max*100], label='Capital Market Line')
plt.plot(tan_vol*100, tan_expret*100, 'o', label='Tangency Portfolio')
plt.xlim(0,70)
plt.ylim(0,70)
plt.xlabel('Volatility (%)')
plt.ylabel('Expected Return (%)')
plt.legend()
plt.title('Portfolio Optimization')
plt.show()
```



0.2 [Optional] Second (Math) Way to Derive the Tangency Portfolio

The approach above worked well and was simple, but it required us to calculate the slope of the efficient frontier at every point. Can't we just solve for the tangency portfolio directly?

Sure: we can find the tangency portfolio by maximizing the Sharpe Ratio directly. The Sharpe Ratio is given by:

$$SR = \frac{\mu_p - r_f}{\sigma_p}$$

Plugging in for the expected return and volatility, we get the following maximization problem:

$$\max_{w} \frac{\mathbf{w}' \quad - r_f}{\sqrt{\mathbf{w}'\Sigma\mathbf{w}}}$$

subject to

$$\mathbf{w}'\mathbf{1}_N = 1$$

But this is a hard problem to solve! $\mathbf{w}$ shows up both in the numerator and the denominator, so the derivative is going to be quite messy. So let's try a different approach.

0.2.1 Portfolio Optimization with Mean-Variance Preferences

Instead, let us formulate a slightly different problem. Suppose we are an investor who dislikes variance γ times as much as they like expected return. In other words, for this investor to be willing to take on γ more risk in their portfolio, they need to be compensated with 1 more unit of expected return. This investor can invest in both risky assets and the risk-free asset, so she chooses N weights $\mathbf{w}$ and a weight w_f on the risk-free asset. The investor's problem is then:

$$\max_{\mathbf{w},w_f} \mathbf{w}' \ \ + w_f r_f - \gamma \mathbf{w}'\Sigma\mathbf{w}$$

subject to

$$\mathbf{w}'\mathbf{1}_N + w_f = 1$$

The Lagrangian is:

$$\mathcal{L}(w, w_f, \lambda) = \mathbf{w}' \ \ + w_f r_f - \gamma \mathbf{w}'\Sigma\mathbf{w} + \lambda(1 - \mathbf{w}'\mathbf{1}_N - w_f)$$

First-order conditions are:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{w}} = \ \ - 2\gamma\Sigma\mathbf{w} - \lambda\mathbf{1}_N = \mathbf{0}_N$$

$$\frac{\partial\mathcal{L}}{\partial w_f} = r_f - \lambda = 0$$

$$\frac{\partial\mathcal{L}}{\partial\lambda} = 1 - \mathbf{w}'\mathbf{1}_N - w_f = 0$$

The second equation tells us that the Lagrange multiplier is equal to the risk-free rate. Plugging in $\lambda = r_f$ into the first equation and rearranging, we get:

$$\ \ - r_f\mathbf{1}_N = 2\gamma\Sigma\mathbf{w}$$

Solving for $\mathbf{w}$ gives us:

$$\mathbf{w} = \frac{1}{2\gamma}\Sigma^{-1}(\ \ - r_f\mathbf{1}_N)$$

Our optimal portfolio depends on how much we dislike risk. If γ is high, we are strongly averse to risk and the values of $\mathbf{w}$ are close to 0, i.e. we invest mostly in the risk-free asset $w_f = 1 - \mathbf{w}\mathbf{1}_N$. If γ is small, we invest mostly in the risky assets, possibly even borrowing ($w_f < 0$) to invest more in the risky assets.

But importantly, optimal weights $\mathbf{w}$ are all proportional to γ. Varying γ scales all weights up or down by the same fraction – it does not change the composition of the portfolio. If one investor likes risk twice as much as another investor, she will buy twice as much of every asset.

0.2.2 Expected Return, Volatility, and Sharpe Ratio of the γ-Optimal Portfolio

What is the expected return and volatility of this portfolio? Plugging in $w_f = 1 - \mathbf{w}\mathbf{1}_N$, we get the following expected return:

$$\mathbf{w}' \ \ + w_f r_f = \mathbf{w}' \ \ + (1 - \mathbf{w}'1_N)r_f = r_f + \mathbf{w}'(\ \ - r_f\mathbf{1}_N)$$

Substituting in weights, we get:

$$r_f + \mathbf{w}'(\; - r_f \mathbf{1}_N) = r_f + \frac{1}{2\gamma}(\; - r_f \mathbf{1}_N)'\Sigma^{-1}(\; - r_f \mathbf{1}_N)$$

For the variance, we have:

$$\mathbf{w}'\Sigma\mathbf{w} = \frac{1}{2\gamma}(\; - r_f \mathbf{1}_N)'\Sigma^{-1}\Sigma\Sigma^{-1}\frac{1}{2\gamma}(\; - r_f \mathbf{1}_N)$$

Simplifying,

$$\mathbf{w}'\Sigma\mathbf{w} = \left(\frac{1}{2\gamma}\right)^2 (\; - r_f \mathbf{1}_N)'\Sigma^{-1}(\; - r_f \mathbf{1}_N)$$

The Sharpe Ratio is then:

$$SR = \frac{\mathbf{w}' \quad + w_f r_f - r_f}{\sqrt{\mathbf{w}'\Sigma\mathbf{w}}} = \frac{\mathbf{w}'(\; - r_f \mathbf{1}_N)}{\sqrt{\mathbf{w}'\Sigma\mathbf{w}}}$$

Plugging in the expected return and variance, we get:

$$SR = \frac{\frac{1}{2\gamma}(\; - r_f \mathbf{1}_N)'\Sigma^{-1}(\; - r_f \mathbf{1}_N)}{\frac{1}{2\gamma}\sqrt{(\; - r_f \mathbf{1}_N)'\Sigma^{-1}(\; - r_f \mathbf{1}_N)}} = \sqrt{(\; - r_f \mathbf{1}_N)'\Sigma^{-1}(\; - r_f \mathbf{1}_N)}$$

Notice that γ's canceled out! The sharpe ratio of the optimal portfolio does not depened on the investor's attitudes towards risk γ. While the choice of γ affects which specific portfolio they choose, every investor ends up with the same Sharpe Ratio.

Let's recap. We found that all investors choose the same mix of risky assets (the risky portfolio) regardless of their attitudes towards risk. Their risk aversion just affects how much of the risky portfolio they hold vs. how much of the risky asset they hold. We just derived the Capital Market Line! Not only that, we proved that investors ought to hold portfolios on that line. Let's plot the CML on the graph by varying γ:

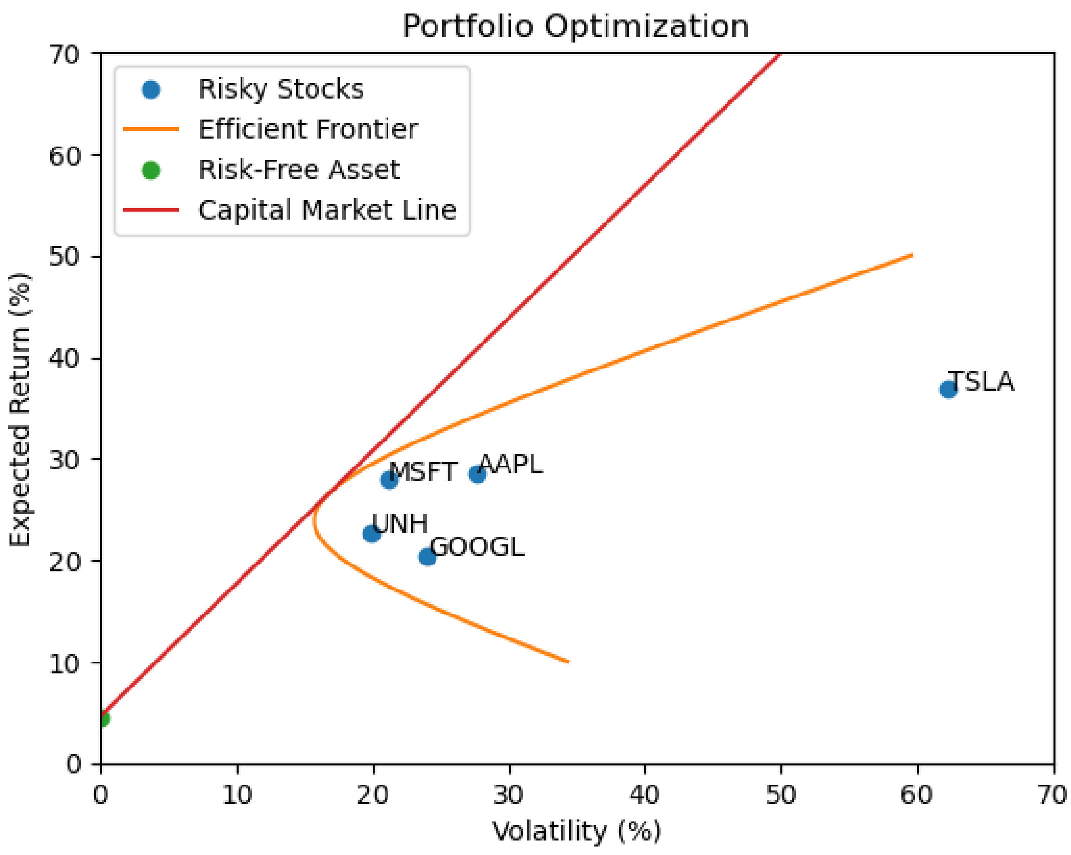
```
[18]: gammavec = np.linspace(0.001, 1000, 1000).reshape((1000,1))
      CML_weights = 1/(2*gammavec) * (np.linalg.inv(covar) @ (meanrets - rf))
```

Use the weights to calculate expected return and volatility:

```
[19]: CML_rets = rf + CML_weights @ (meanrets - rf)
      CML_vols = np.sqrt([w.transpose() @ covar @ w for w in CML_weights])
```

Plot:

```
[20]: fig=plt.figure()
      plt.plot(vols*100, meanrets*100, 'o', label='Risky Stocks')
      [plt.text(vols[i]*100, meanrets[i]*100, df.index[i]) for i in range(len(df))]
      plt.plot(portvols*100, portrets*100, label='Efficient Frontier')
      plt.plot(0, rf*100, 'o', label='Risk-Free Asset')
      plt.plot(CML_vols*100, CML_rets*100, label='Capital Market Line')
plt.xlim(0,70)
plt.ylim(0,70)
plt.xlabel('Volatility (%)')
plt.ylabel('Expected Return (%)')
plt.legend()
plt.title('Portfolio Optimization')
plt.show()
```



0.2.3 Tangency Portfolio

But what about the tangency portfolio? Isn't that what we're after?

Recall that the tangency portfolio is just a point on the CML where it is tangent to the efficient frontier, i.e. it is the one portfolio on the CML that is comprised entirely of risky assets. This is the portfolio chosen by an investor whose risk aversion γ_T is such that they choose a portfolio in which $w_f = 0$ or $\mathbf{w}'\mathbf{1}_N = 1$. What value is γ_T is that?

Recall that the optimal risky weights are given by:

$$\mathbf{w} = \frac{1}{2\gamma}\Sigma^{-1}(\quad - r_f\mathbf{1}_N)$$

Let's pre-multiply both sides by $\mathbf{1}'_N$:

$$1 = \mathbf{1}'_N\mathbf{w} = \frac{1}{2\gamma_T}\mathbf{1}'_N\Sigma^{-1}(\ -r_f\mathbf{1}_N)$$

where we impose the definition of the tangency portfolio $\mathbf{w}'\mathbf{1}_N = 1$ at $\gamma = \gamma_T$. Solving for γ_T:

$$\gamma_T = \frac{1}{2}\mathbf{1}'_N\Sigma^{-1}(\ -r_f\mathbf{1}_N)$$

Plugging this back into the expression for the optimal weights, we get the tangency portfolio weights:

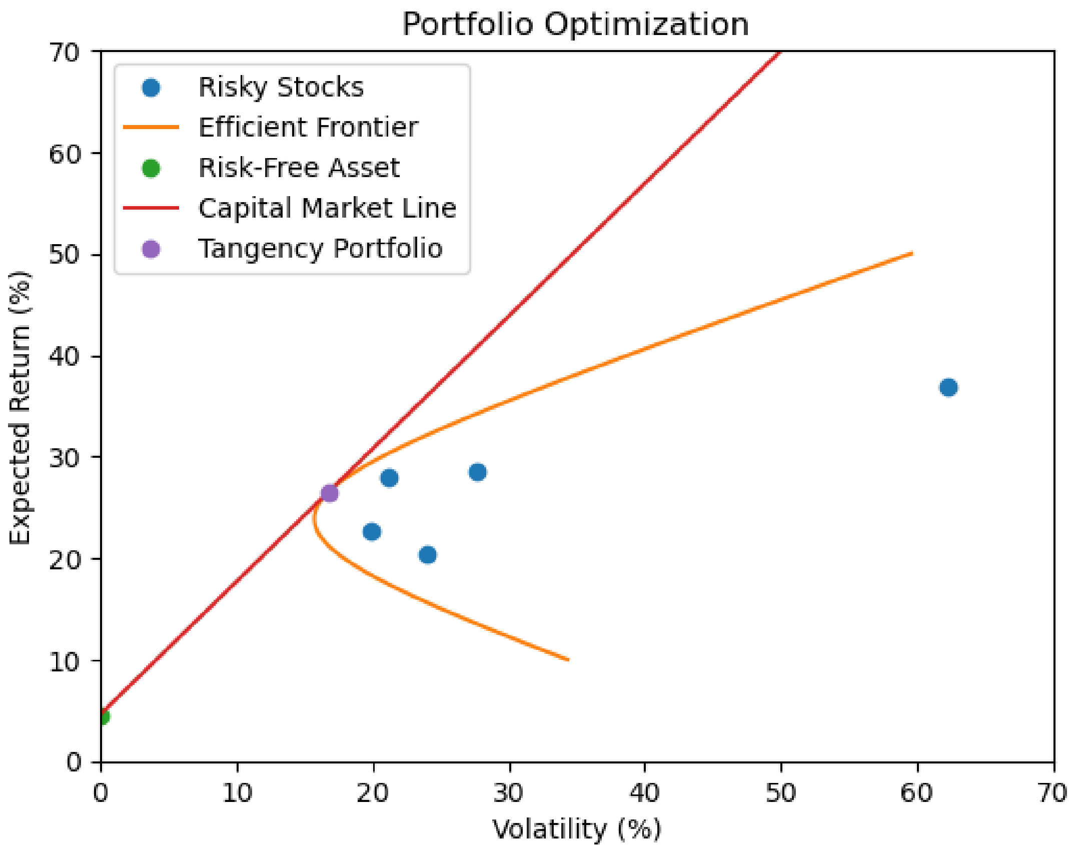
$$\mathbf{w}_T = \frac{\Sigma^{-1}(\ -r_f\mathbf{1}_N)}{\mathbf{1}'_N\Sigma^{-1}(\ -r_f\mathbf{1}_N)}$$

Let's calculate the tangency portfolio weights and plot its return and volatility on the graph:

[21]:
```python
exrets = meanrets - rf
covar_inv = np.linalg.inv(covar)
wT = covar_inv @ exrets / np.sum(covar_inv @ exrets)

rT = rf + wT @ exrets
volT = np.sqrt(wT.transpose() @ covar @ wT)
```

[22]:
```python
fig=plt.figure()
plt.plot(vols*100, meanrets*100, 'o', label='Risky Stocks')
plt.plot(portvols*100, portrets*100, label='Efficient Frontier')
plt.plot(0, rf*100, 'o', label='Risk-Free Asset')
plt.plot(CML_vols*100, CML_rets*100, label='Capital Market Line')
plt.plot(volT*100, rT*100, 'o', label='Tangency Portfolio')
plt.xlim(0,70)
plt.ylim(0,70)
plt.xlabel('Volatility (%)')
plt.ylabel('Expected Return (%)')
plt.legend()
plt.title('Portfolio Optimization')
plt.show()
```



[23]:
```
print('Tangency Portfolio:\n', wT)
print('Tangency Portfolio Expected Return:', rT)
print('Tangency Portfolio Volatility:', volT)
print('Tangency Portfolio Sharpe Ratio:', (rT-rf)/volT)
```

```
Tangency Portfolio:
 [ 0.56091689  0.09923081  0.43036243 -0.09653059  0.00602046]
Tangency Portfolio Expected Return: 0.26520765912633376
Tangency Portfolio Volatility: 0.16774867483993927
Tangency Portfolio Sharpe Ratio: 1.3079546490348488
```

Check that we got (approximately) the same solution as using the slope of the efficient frontier above. The second method may be more mathematically rigorous, but it's also quite a bit more involved!

[ ]:
```

```