INTRO TO PORTFOLIO RISK MANAGEMENT IN PYTHON

Portfolio Composition

Dakota Wixom
Quantitative Analyst | QuantCourse.com

Calculating Portfolio Returns

PORTFOLIO RETURN FORMULA:

$$R_p = R_{a_1} w_{a_1} + R_{a_2} w_{a_2} + \ldots + R_{a_n} w_{a_1}$$

- R_p: Portfolio return
- R_{a_n}: Return for asset n
- w_{a_n}: Weight for asset n

Calculating Portfolio Returns in Python

Assuming StockReturns is a pandas DataFrame of stock returns, you can calculate the portfolio return for a set of portfolio weights as follows:

```
In [1]: import numpy as np
In [2]: portfolio_weights = np.array([0.25, 0.35, 0.10, 0.20, 0.10])
In [3]: port_ret = StockReturns.mul(portfolio_weights, axis=1).sum(axis=1)
In [4]: port_ret
Out [4]: Date
         2017-01-03    0.008082
         2017-01-04    0.000161
         2017-01-05    0.003448
         ...
In [5]: StockReturns["Portfolio"] = port_ret
```

Equally Weighted Portfolios in Python

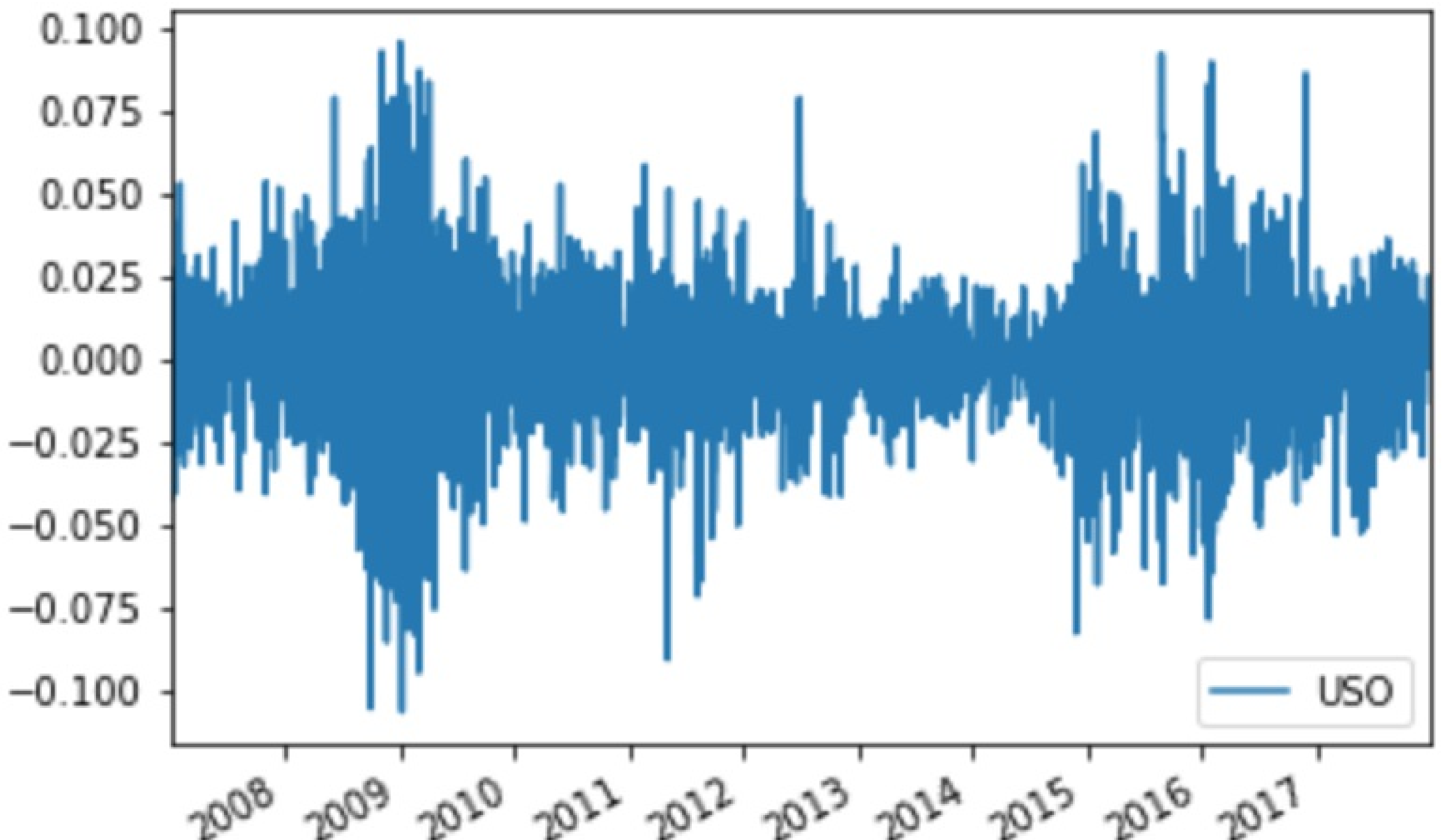
Assuming StockReturns is a pandas DataFrame of stock returns, you can calculate the portfolio return for an **equally weighted** portfolio as follows:

```
In [1]: import numpy as np
In [2]: numstocks = 5
In [3]: portfolio_weights_ew = np.repeat(1/numstocks, numstocks)
In [4]: StockReturns.iloc[:,0:numstocks].mul(portfolio_weights_ew, axis=1).sum(ax
Out [4]: Date
         2017-01-03    0.008082
         2017-01-04    0.000161
         2017-01-05    0.003448
         ...
```

Plotting Portfolio Returns in Python

To plot the daily returns in Python:

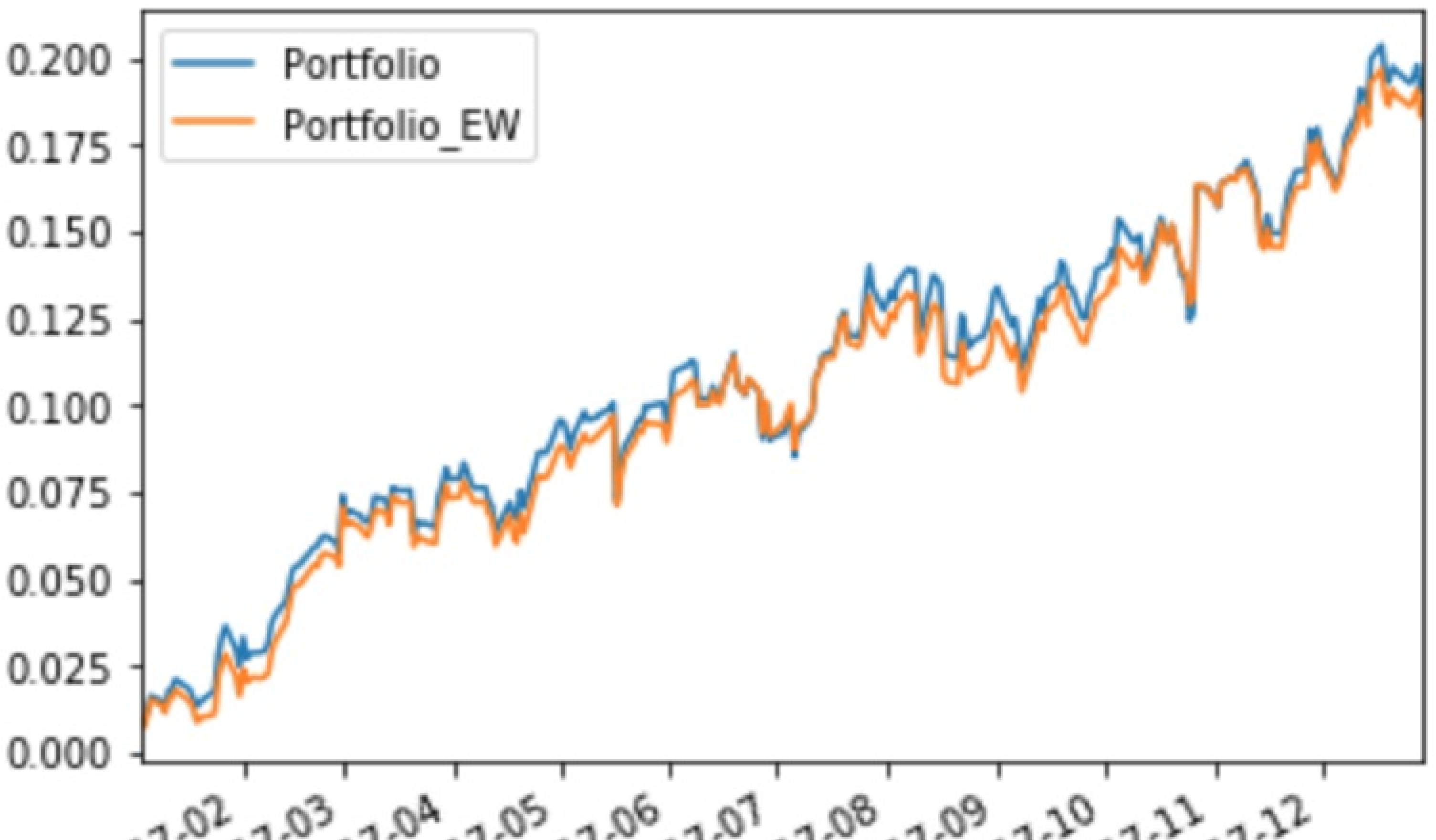
```
In [1]: StockPrices["Returns"] = StockPrices["Adj Close"].pct_change()
In [2]: StockReturns = StockPrices["Returns"]
In [3]: StockReturns.plot()
```



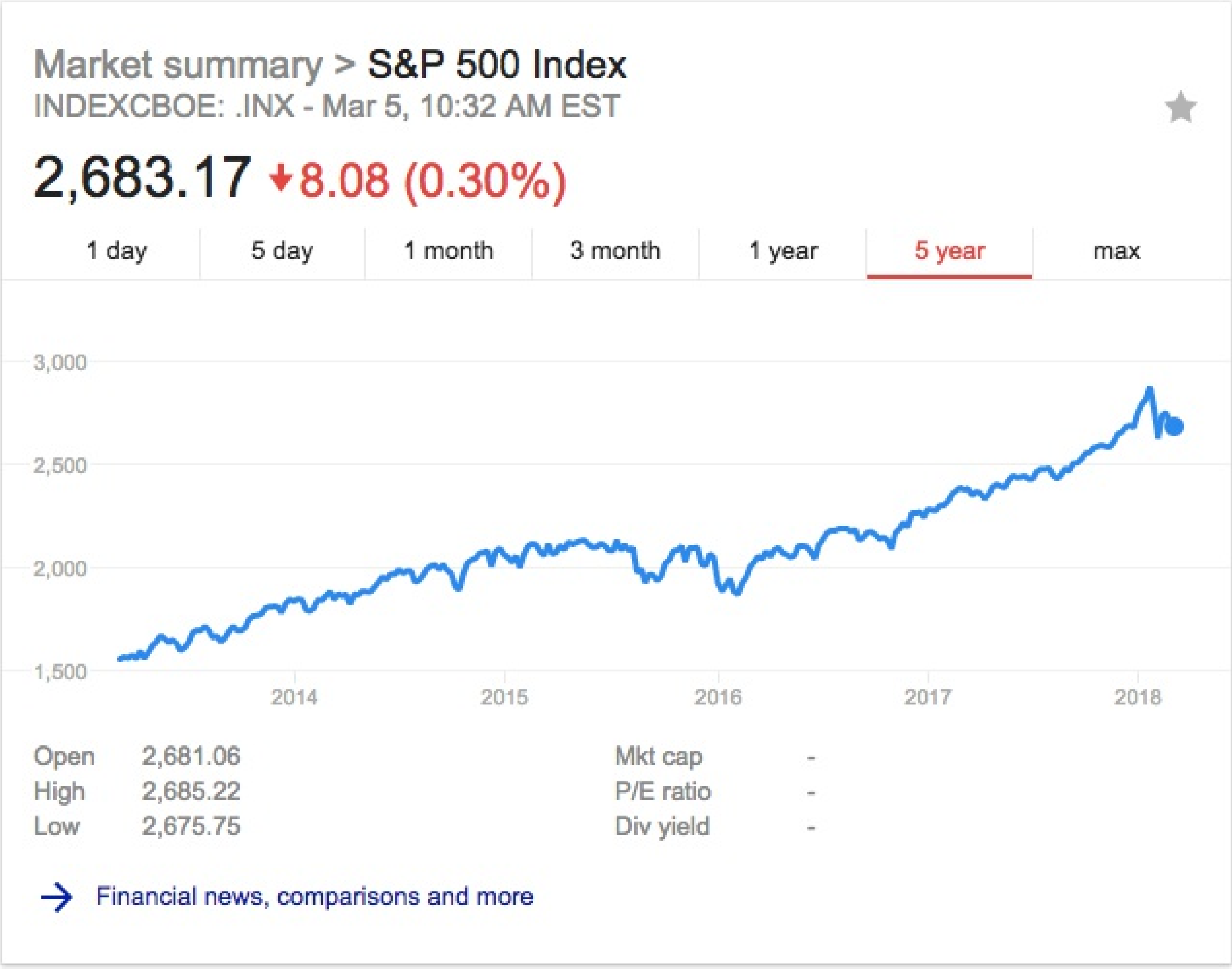
Plotting Portfolio Cumulative Returns

In order to plot the cumulative returns of multiple portfolios:

```
In [1]: import matplotlib.pyplot as plt
In [2]: CumulativeReturns = ((1+StockReturns).cumprod()-1)
In [3]: CumulativeReturns[["Portfolio","Portfolio_EW"]].plot()
Out [3]:
```



Market Capitalization



Market Capitalization

Market Capitalization: The value of a company's publically traded shares.

Also referred to as **Market Cap**.

Market-Cap Weighted Portfolios

In order to calculate the market cap weight of a given stock n:

$$w_{mcap_n} = \frac{mcap_n}{\sum_{i=1}^{n} mcap_i}$$

Market-Cap Weights in Python

To calculate market cap weights in python, assuming you have data on the market caps of each company:

```
In [1]: import numpy as np
In [2]: market_capitalizations = np.array([100, 200, 100, 100])
In [3]: mcap_weights = market_capitalizations/sum(market_capitalizations)
In [4]: mcap_weights
Out [4]: array([0.2, 0.4, 0.2, 0.2])
```

INTRO TO PORTFOLIO RISK MANAGEMENT IN PYTHON

Let's practice!

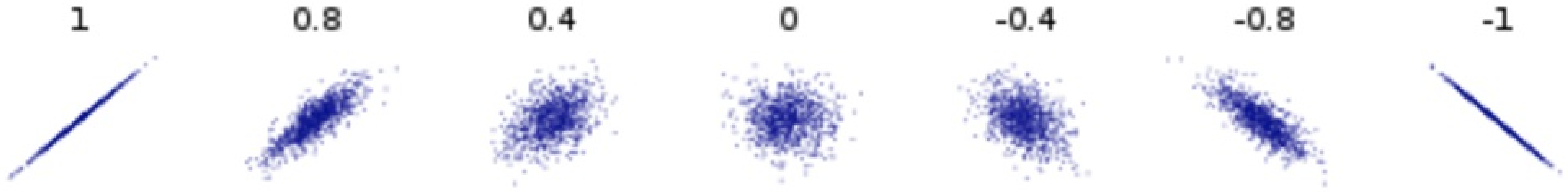
INTRO TO PORTFOLIO RISK MANAGEMENT IN PYTHON

Correlation and Co-Variance

Dakota Wixom
Quantitative Analyst | QuantCourse.com

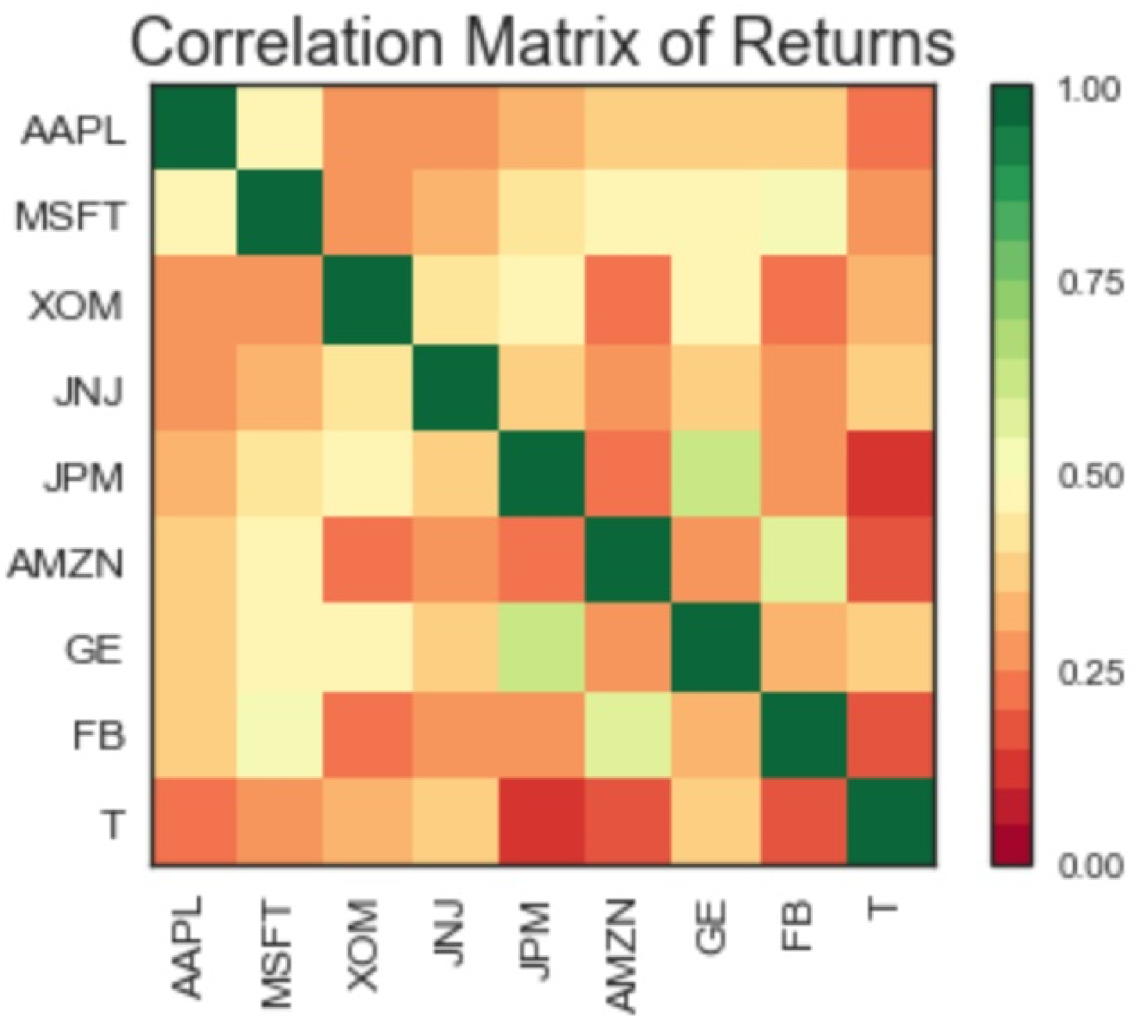
Pearson Correlation

EXAMPLES OF DIFFERENT CORRELATIONS BETWEEN TWO RANDOM VARIABLES:



Pearson Correlation

A HEATMAP OF A CORRELATION MATRIX:



Correlation Matrix in Python

Assuming StockReturns is a pandas DataFrame of stock returns, you can calculate the correlation matrix as follows:

```
In [1]: correlation_matrix = StockReturns.corr()
In [2]: print(correlation_matrix)
Out [2]:
```

	AAPL	MSFT	XOM
AAPL	1.000000	0.494160	0.272386
MSFT	0.494160	1.000000	0.289405
XOM	0.272386	0.289405	1.000000

Portfolio Standard Deviation

Portfolio standard deviation for a two asset portfolio:

$$\sigma_p = \sqrt{w_1^2\sigma_1^2 + w_2^2\sigma_2^2 + 2w_1w_2\rho_{1,2}\sigma_1\sigma_2}$$

- σ_p: Portfolio standard deviation
- w: Asset weight
- σ: Asset volatility
- $\rho_{1,2}$: Correlation between assets 1 and 2

The Co-Variance Matrix

To calculate the co-variance matrix (Σ) of returns X:

$$\Sigma = \begin{bmatrix} \mathrm{E}[(X_1-\mu_1)(X_1-\mu_1)] & \mathrm{E}[(X_1-\mu_1)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_1-\mu_1)(X_n-\mu_n)] \\ \mathrm{E}[(X_2-\mu_2)(X_1-\mu_1)] & \mathrm{E}[(X_2-\mu_2)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_2-\mu_2)(X_n-\mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n-\mu_n)(X_1-\mu_1)] & \mathrm{E}[(X_n-\mu_n)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_n-\mu_n)(X_n-\mu_n)] \end{bmatrix}.$$

The Co-Variance Matrix in Python

Assuming StockReturns is a pandas DataFrame of stock returns, you can calculate the covariance matrix as follows:

```
In [1]: cov_mat = StockReturns.cov()
In [2]: cov_mat
Out [2]:
```

	AAPL	MSFT	XOM	JNJ	JPM	AMZN	GE	FB	T
AAPL	0.000216	0.000104	0.000048	0.000033	0.000080	0.000097	0.000059	0.000093	0.000031
MSFT	0.000104	0.000204	0.000050	0.000040	0.000098	0.000133	0.000076	0.000132	0.000034
XOM	0.000048	0.000050	0.000145	0.000042	0.000090	0.000055	0.000059	0.000049	0.000037
JNJ	0.000033	0.000040	0.000042	0.000072	0.000047	0.000043	0.000037	0.000043	0.000028
JPM	0.000080	0.000098	0.000090	0.000047	0.000241	0.000073	0.000106	0.000073	0.000020
AMZN	0.000097	0.000133	0.000055	0.000043	0.000073	0.000350	0.000058	0.000197	0.000034
GE	0.000059	0.000076	0.000059	0.000037	0.000106	0.000058	0.000117	0.000065	0.000037
FB	0.000093	0.000132	0.000049	0.000043	0.000073	0.000197	0.000065	0.000319	0.000025
T	0.000031	0.000034	0.000037	0.000028	0.000020	0.000034	0.000037	0.000025	0.000083

Annualizing the Covariance Matrix

To annualize the covariance matrix:

```
In [2]: cov_mat_annual = cov_mat*252
```

Portfolio Standard Deviation using Covariance

The formula for portfolio volatility is:

$$\sigma_{Portfolio} = \sqrt{w_T \cdot \Sigma \cdot w}$$

- $\sigma_{Portfolio}$: Portfolio volatility
- Σ: Covariance matrix of returns
- w: Portfolio weights (w_T is transposed portfolio weights)
- $\cdot$ The dot-multiplication operator

Matrix Transpose

Examples of **matrix transpose** operations:

$$\begin{bmatrix} 1 & 2 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

Dot Product

The **dot product** operation of two vectors **a** and **b**:

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

Portfolio Standard Deviation using Python

To calculate portfolio volatility assumy a weights array and a covariance matrix:

```
In [1]: import numpy as np
In [2]: port_vol = np.sqrt(np.dot(weights.T, np.dot(cov_mat, weights)))
In [3]: port_vol
Out [3]: 0.035
```

INTRO TO PORTFOLIO RISK MANAGEMENT IN PYTHON

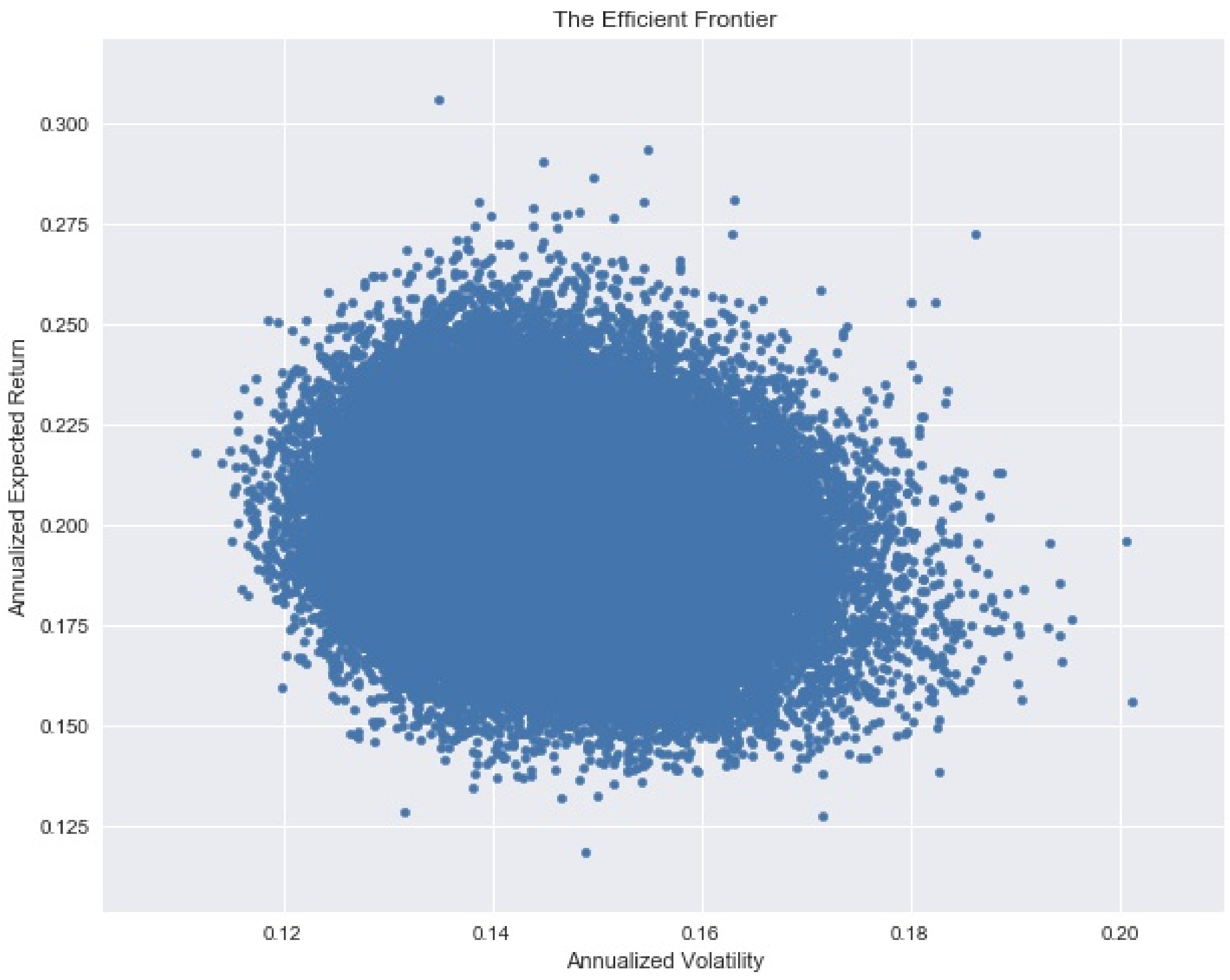
Let's practice!

INTRO TO PORTFOLIO RISK MANAGEMENT IN PYTHON

Markowitz Portfolios

Dakota Wixom
Quantitative Analyst | QuantCourse.com

100,000 Randomly Generated Portfolios



Sharpe Ratio

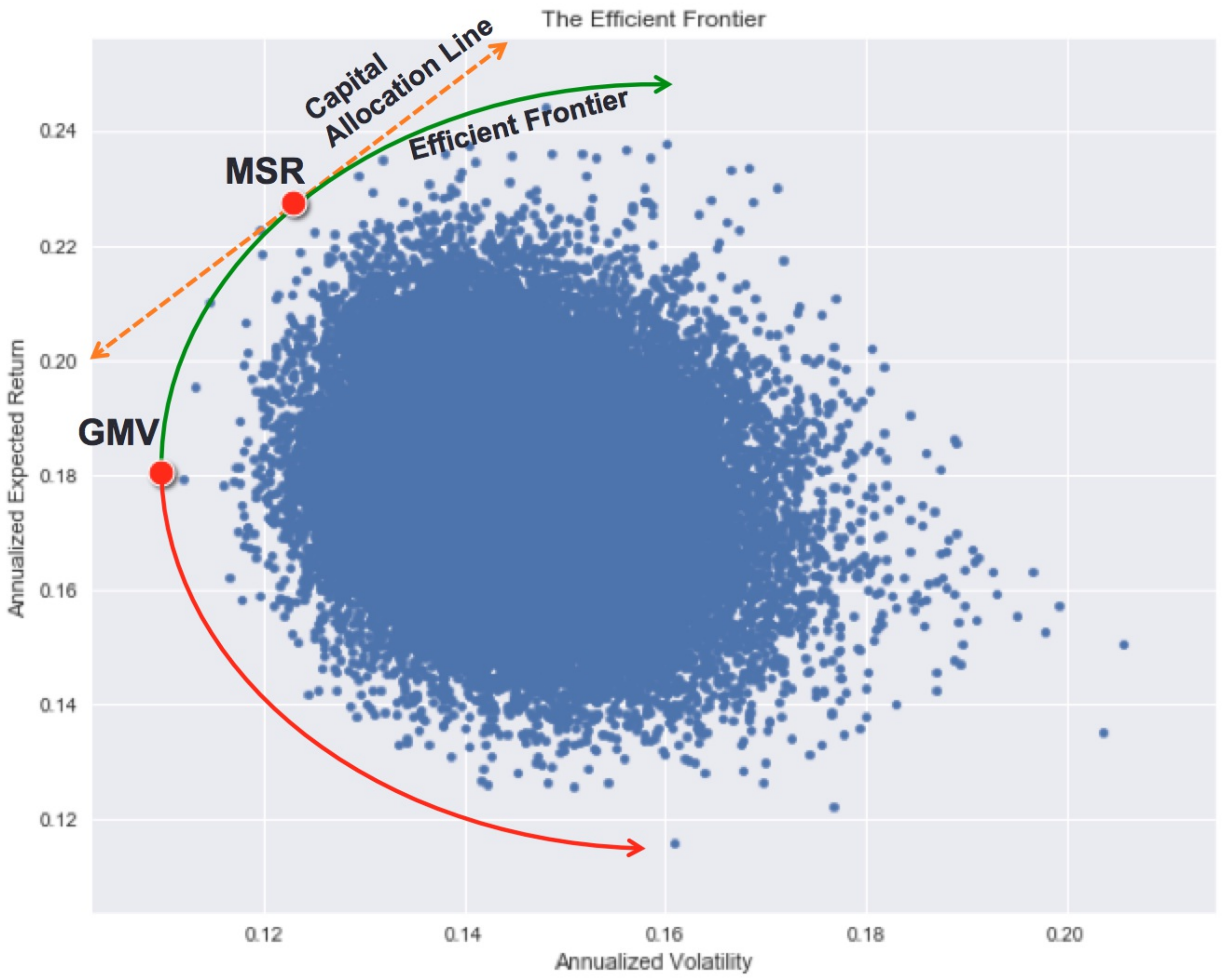
The Sharpe ratio is a measure of **risk-adjusted return**.

To calculate the 1966 version of the Sharpe ratio:

$$S = \frac{R_a - r_f}{\sigma_a}$$

- S: Sharpe Ratio
- R_a: Asset return
- r_f: Risk-free rate of return
- σ_a: Asset volatility

The Efficient Frontier

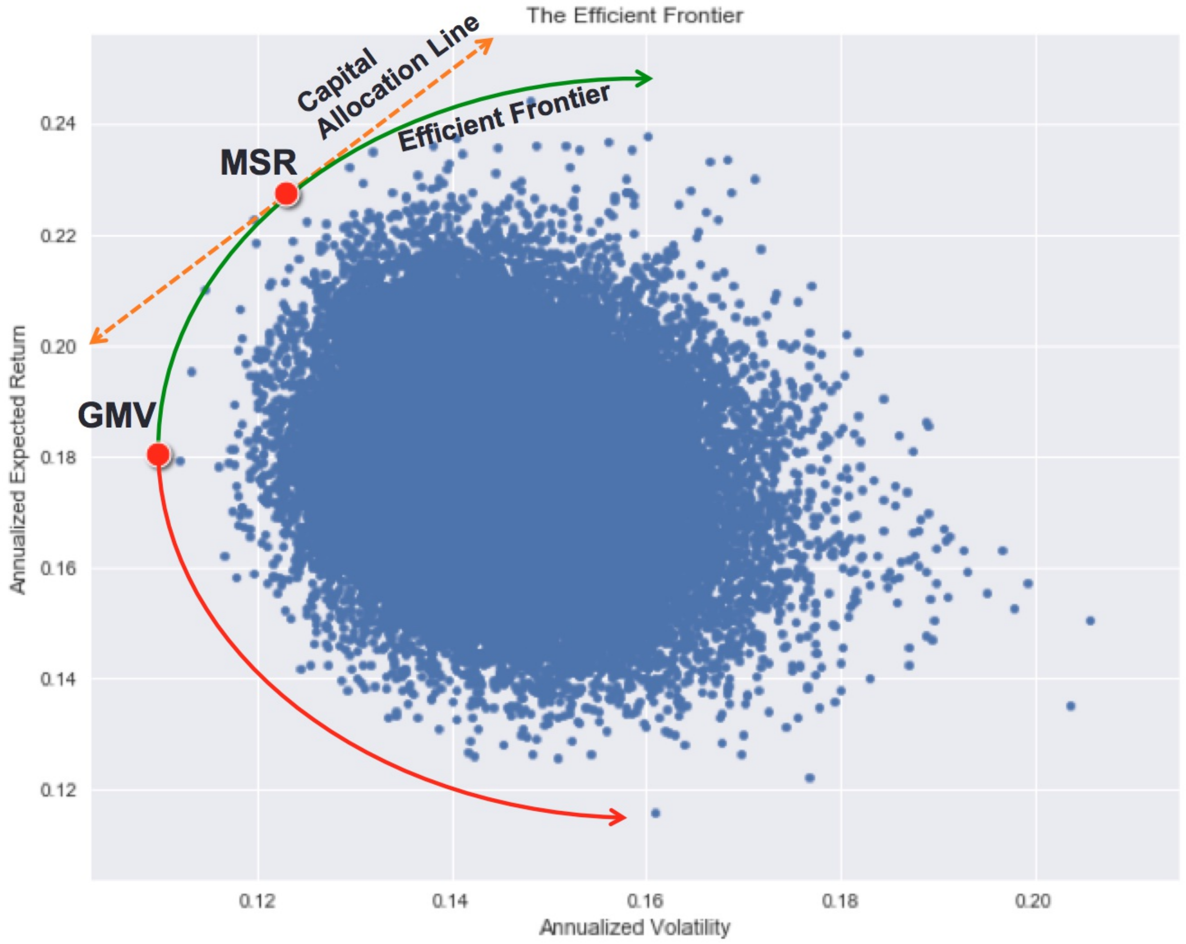


The Markowitz Portfolios

Any point on the efficient frontier is an optimium portfolio.

These two common points are called **Markowitz Portfolios**:

- **MSR: M**ax **S**harpe **R**atio portfolio
- **GMV: G**lobal **M**inimum **V**olatility portfolio



Choosing a Portfolio

How do you choose the best Portfolio?

- Try to pick a portfolio on the bounding edge of the efficient frontier
- Higher return is available if you can stomach higher risk

Selecting the MSR in Python

Assuming a DataFrame df of random portfolios with Volatility and Returns columns:

```
In [1]: numstocks = 5
In [2]: risk_free = 0
In [3]: df["Sharpe"] = (df["Returns"]-risk_free)/df["Volatility"]
In [4]: MSR = df.sort_values(by=['Sharpe'], ascending=False)
In [5]: MSR_weights = MSR.iloc[0,0:numstocks]
In [6]: np.array(MSR_weights)
Out [6]: array([0.15, 0.35, 0.10, 0.15, 0.25])
```

Past Performance is Not a Guarantee of Future Returns

Even though a Max Sharpe Ratio portfolio might sound nice, in practice, returns are extremely difficult to predict.

Selecting the GMV in Python

Assuming a DataFrame df of random portfolios with Volatility and Returns columns:

```
In [1]: numstocks = 5
In [2]: GMV = df.sort_values(by=['Volatility'], ascending=True)
In [3]: GMV_weights = GMV.iloc[0,0:numstocks]
In [4]: np.array(GMV_weights)
Out [4]: array([0.25, 0.15, 0.35, 0.15, 0.10])
```

INTRO TO PORTFOLIO RISK MANAGEMENT IN PYTHON

Let's practice!