Modern portfolio theory

INTRODUCTION TO PORTFOLIO ANALYSIS IN PYTHON

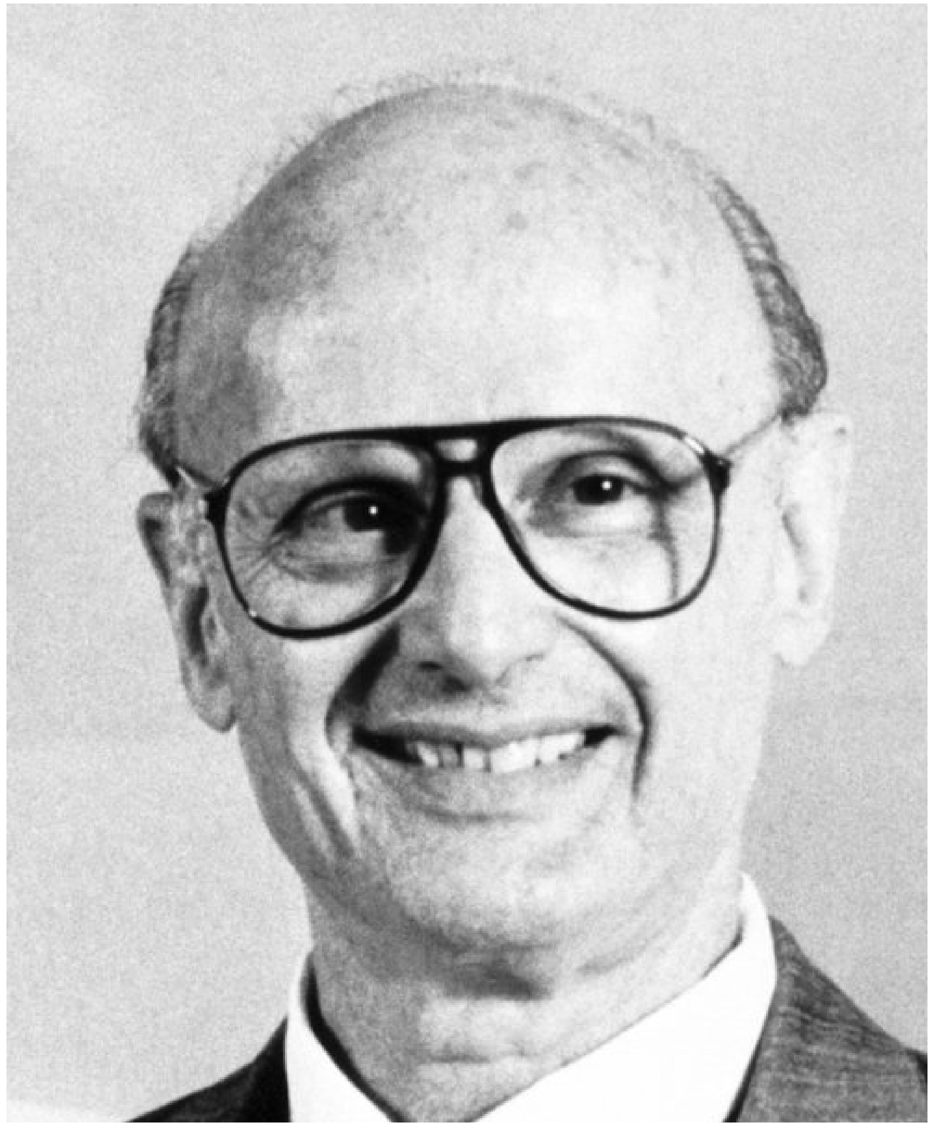
Charlotte Werger
Data Scientist

DataCamp

Creating optimal portfolios

What is Portfolio Optimization?

Meet Harry Markowitz



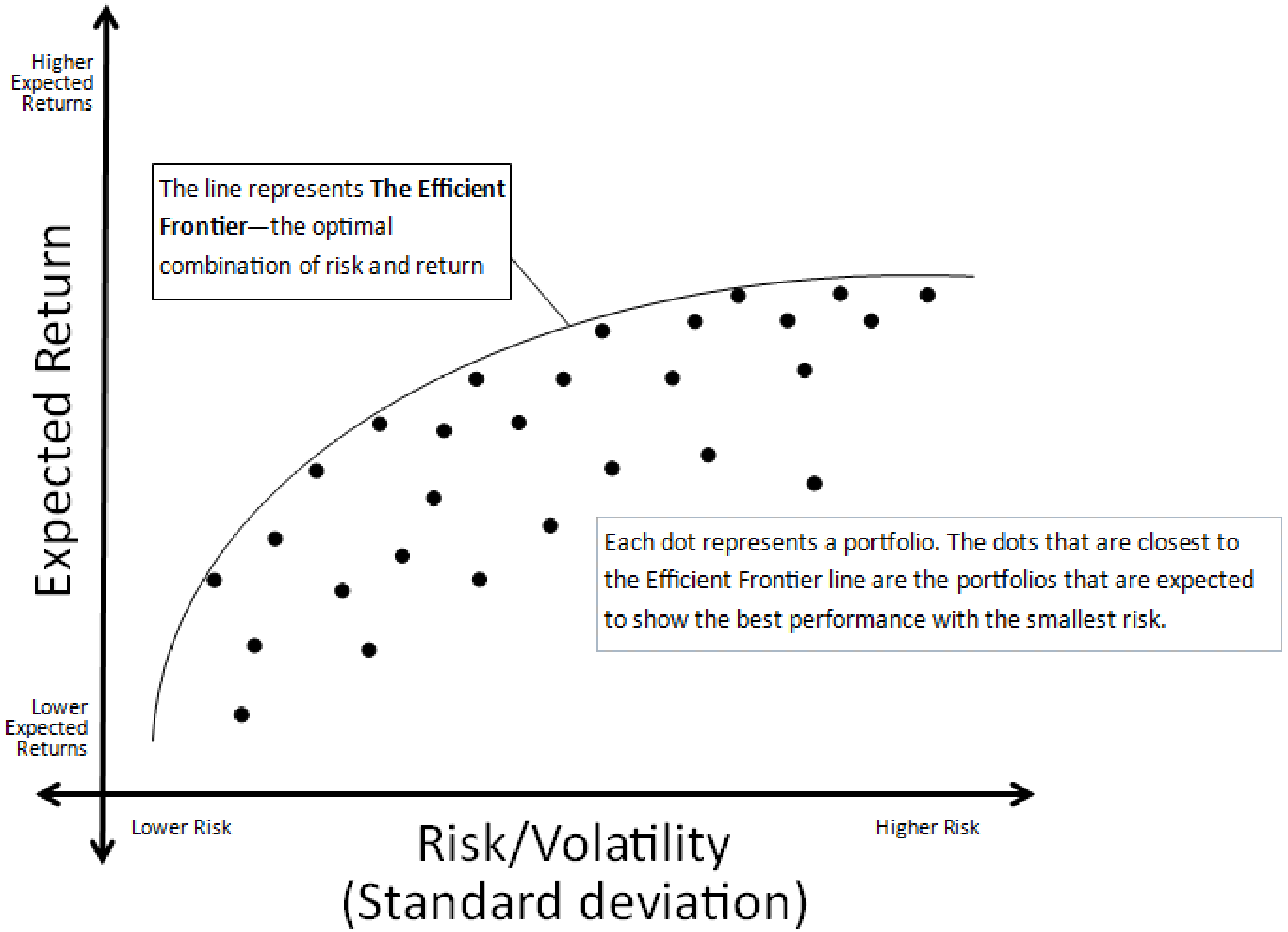
The optimization problem: finding optimal weights

$$
\begin{aligned}
\underset{\omega}{\text{minimise}} \quad & \omega^T \Sigma \omega \\
\text{subject to} \quad & \omega^T \mu \geq \mu^* \\
& \omega^T \mathbf{1} = 1 \\
& \omega_i \geq 0
\end{aligned}
$$

In words:

- Minimize the portfolio variance, subject to:
- The expected mean return is at least some target return
- The weights sum up to 100%
- At least some weights are positive

Varying target returns leads to the Efficient Frontier



PyPortfolioOpt for portfolio optimization

```
from pypfopt.efficient_frontier import EfficientFrontier
from pypfopt import risk_models
from pypfopt import expected_returns
```

```
df=pd.read_csv('portfolio.csv')
df.head(2)
                XOM        RRC        BBY         MA        PFE
date
2010-01-04  54.068794  51.300568  32.524055  22.062426  13.940202
2010-01-05  54.279907  51.993038  33.349487  21.997149  13.741367
```

```
# Calculate expected annualized returns and sample covariance
mu = expected_returns.mean_historical_return(df)
Sigma = risk_models.sample_cov(df)
```

Get the Efficient Frontier and portfolio weights

```
# Calculate expected annualized returns and risk
mu = expected_returns.mean_historical_return(df)
Sigma = risk_models.sample_cov(df)
```

```
# Obtain the EfficientFrontier
ef = EfficientFrontier(mu, Sigma)
```

ef contains all the risk and return optimized portfolio.

```
# Select a chosen optimal portfolio
ef.max_sharpe()
```

Different optimizations

```
# Select the maximum Sharpe portfolio
ef.max_sharpe()
```

```
# Select an optimal return for a target risk
ef.efficient_risk(2.3)
```

```
# Select a minimal risk for a target return
ef.efficient_return(1.5)
```

Calculate portfolio risk and performance

```
# Obtain the performance numbers
ef.portfolio_performance(verbose=True, risk_free_rate = 0.01)
```

make sure the performance numbers are printed

```
Expected annual return: 21.3%
Annual volatility: 19.5%
Sharpe Ratio: 0.98
```

Let's optimize a portfolio!

INTRODUCTION TO PORTFOLIO ANALYSIS IN PYTHON

DataCamp

Maximum Sharpe vs. minimum volatility

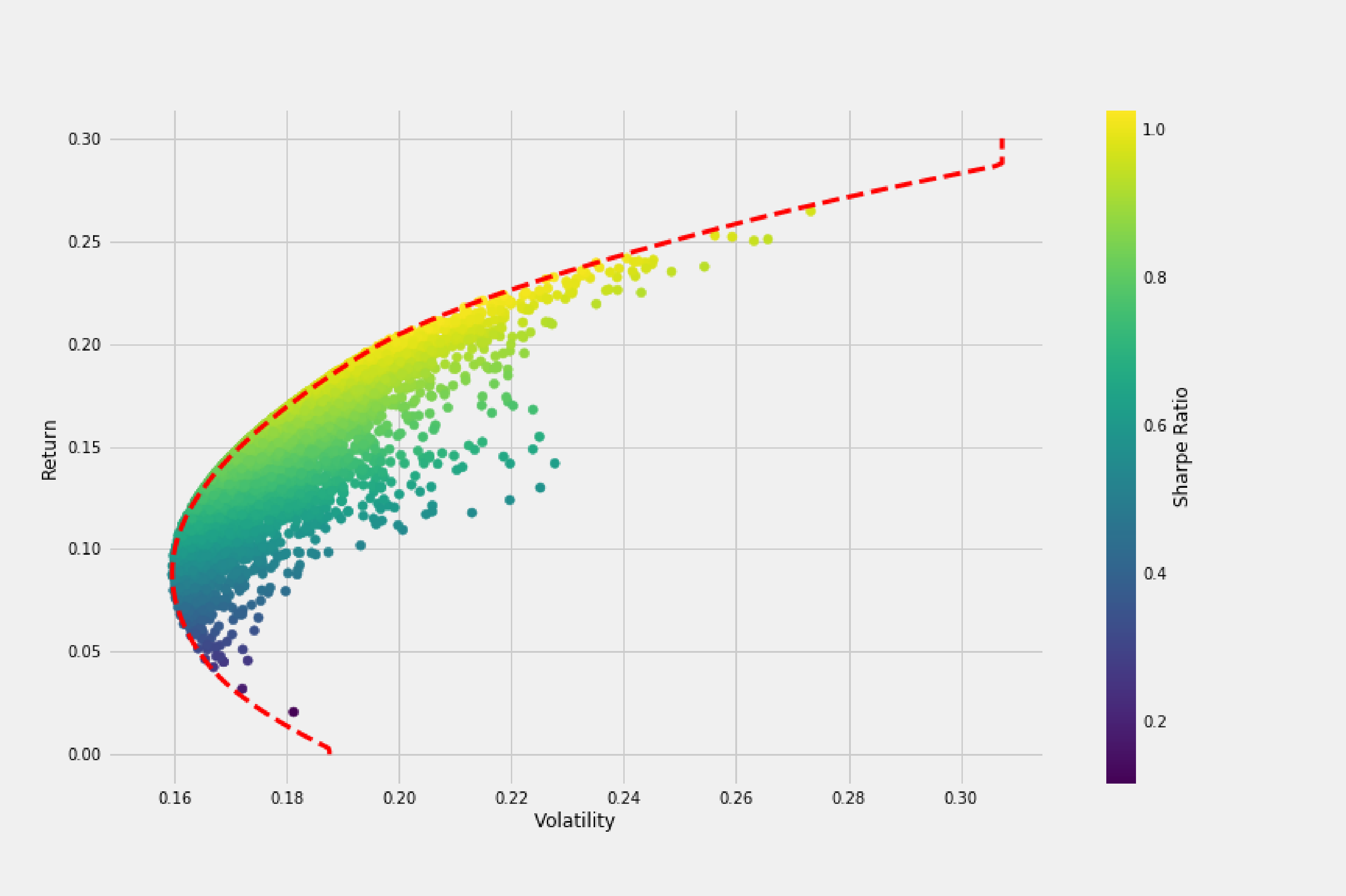
INTRODUCTION TO PORTFOLIO ANALYSIS IN PYTHON

Charlotte Werger
Data Scientist

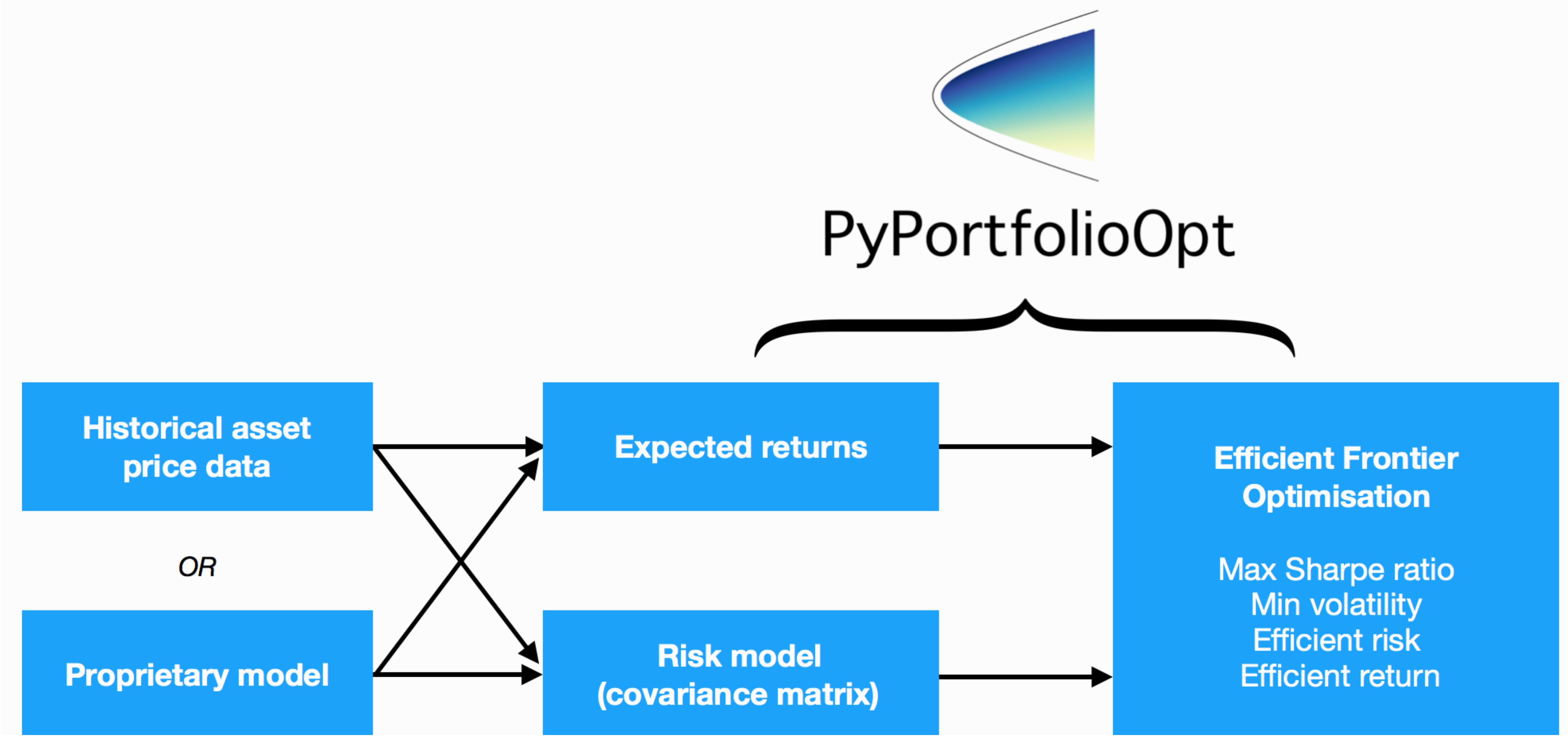
DataCamp

Remember the Efficient Frontier?

- Efficient frontier: all portfolios with an optimal risk and return trade-off
- Maximum Sharpe portfolio: the highest Sharpe ratio on the EF
- Minimum volatility portfolio: the lowest level of risk on the EF



Adjusting PyPortfolioOpt optimization



Maximum Sharpe portfolio

Maximum Sharpe portfolio: the **highest Sharpe ratio** on the EF

```
from pypfopt.efficient_frontier import EfficientFrontier
```

```
# Calculate the Efficient Frontier with mu and S
ef = EfficientFrontier(mu, Sigma)
raw_weights = ef.max_sharpe()
```

Use max_sharp() to obtain the maximum Sharpe ratio portfolio, and
store those portfolio weights to raw_weights.
determine the raw weights using the Maximum Sharpe optimizer

```
# Get interpretable weights
cleaned_weights = ef.clean_weights()
```

get the clean weights for the best interpretation of the portfolio weights

```
{'GOOG': 0.01269,'AAPL': 0.09202,'FB': 0.19856,
 'BABA': 0.09642,'AMZN': 0.07158,'GE': 0.02456,...}
```

Maximum Sharpe portfolio

```
# Get performance numbers
ef.portfolio_performance(verbose=True)
```

For stocks have negative historic performance, adding them don't actually lower the risk of the portfolio.
That means that when we optimize for risk and return, apparently it is optimal not to invest in those.

```
Expected annual return: 33.0%
Annual volatility: 21.7%
Sharpe Ratio: 1.43
```

Minimum Volatility Portfolio

Minimum volatility portfolio: the **lowest level of risk** on the EF

```
# Calculate the Efficient Frontier with mu and S
ef = EfficientFrontier(mu, Sigma)
```

```
raw_weights = ef.min_volatility()
```

The minimum volatility portfolio spreads the risk more by investing in all the stocks, although some have a negative performance, this leads to an overall lower performance, but also to lower risk.

```
# Get interpretable weights and performance numbers
cleaned_weights = ef.clean_weights()
```

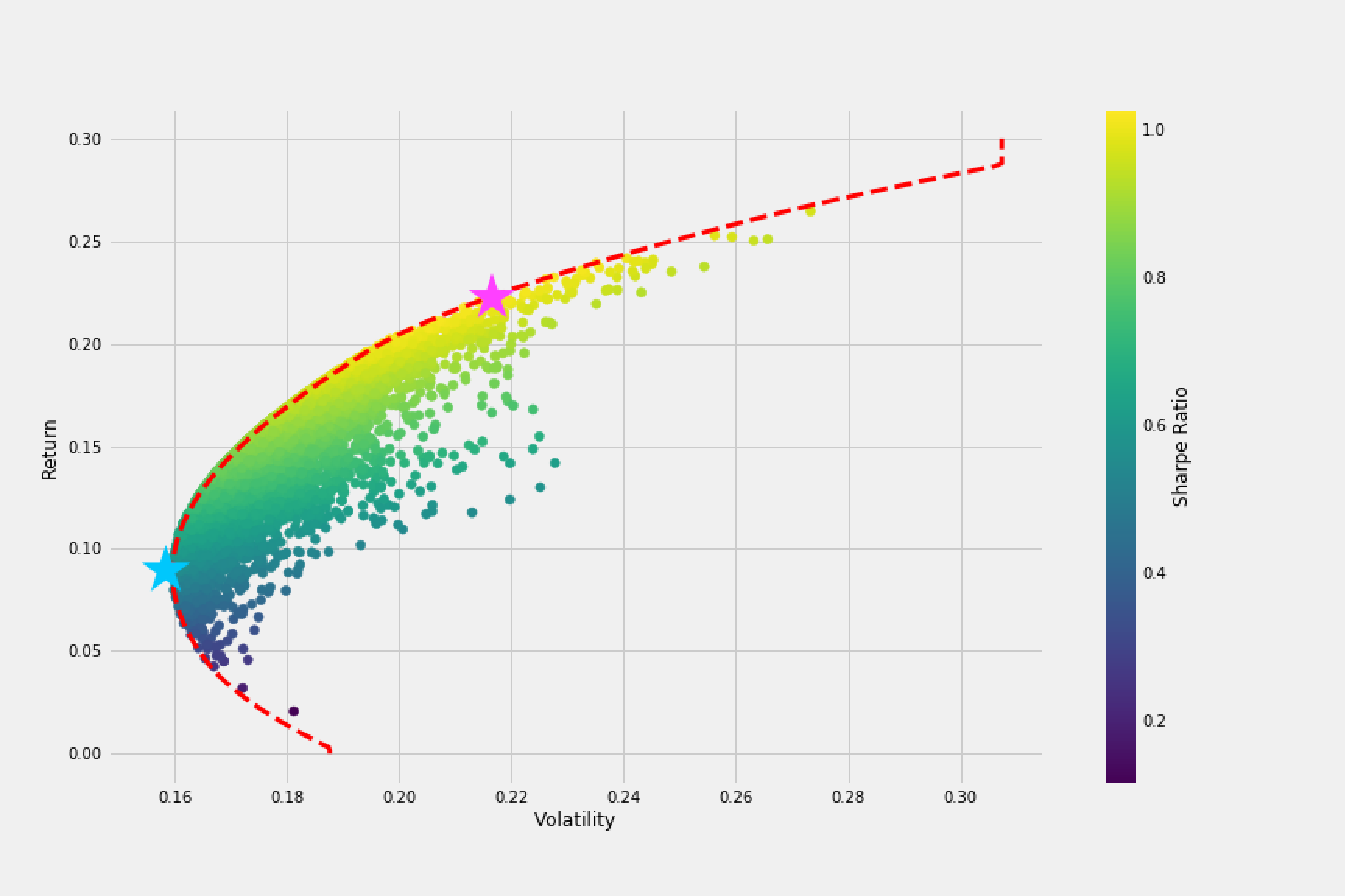
```
{'GOOG': 0.05664, 'AAPL': 0.087, 'FB': 0.1591,
'BABA': 0.09784, 'AMZN': 0.06986, 'GE': 0.0123,...}
```

Minimum Volatility Portfolio

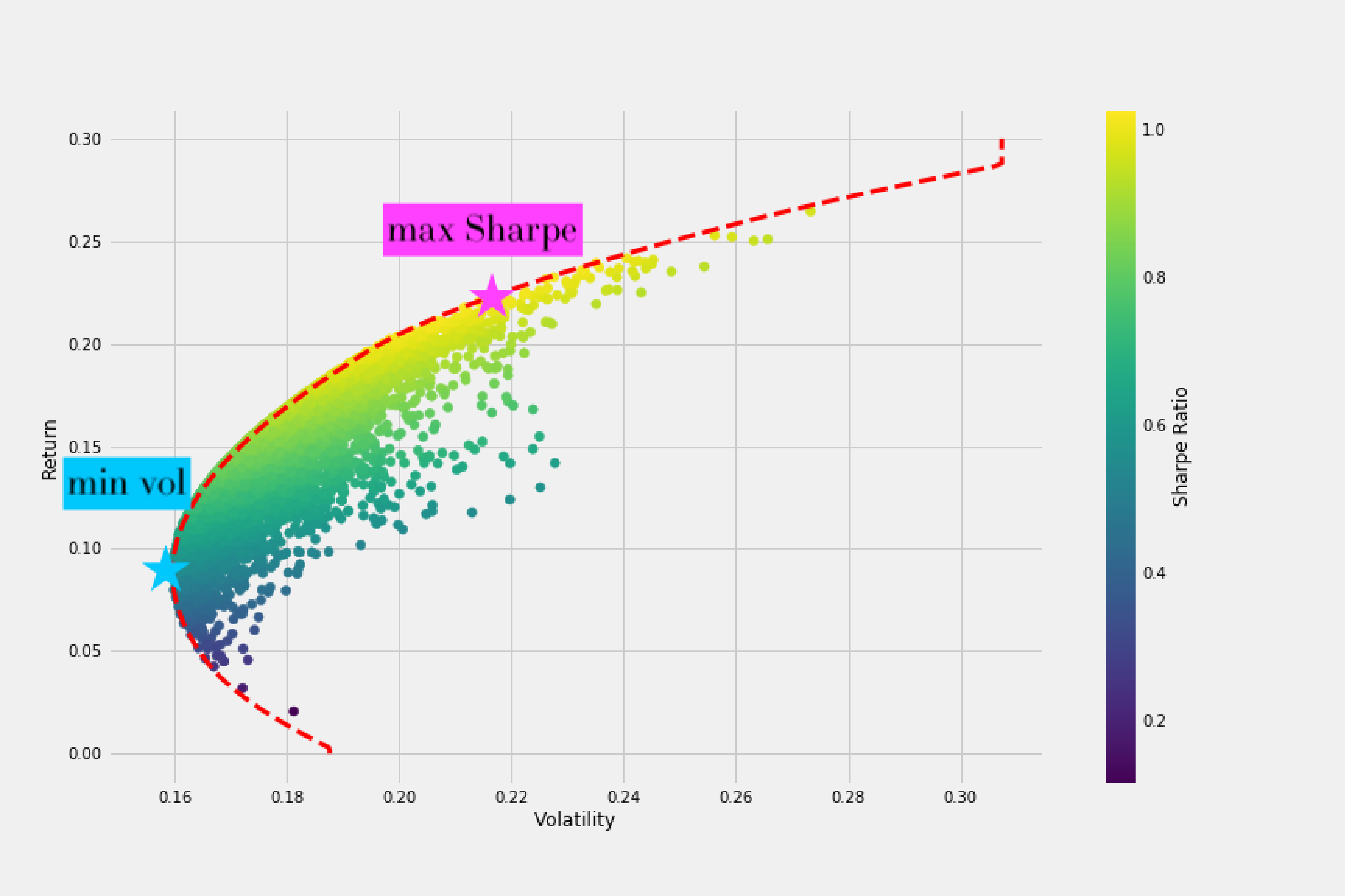
```
ef.portfolio_performance(verbose=True)
```

```
Expected annual return: 17.4%
Annual volatility: 13.2%
Sharpe Ratio: 1.28
```

Let's have another look at the Efficient Frontier



Maximum Sharpe versus Minimum Volatility



Let's practice!

INTRODUCTION TO PORTFOLIO ANALYSIS IN PYTHON

Alternative portfolio optimization

INTRODUCTION TO PORTFOLIO ANALYSIS IN PYTHON

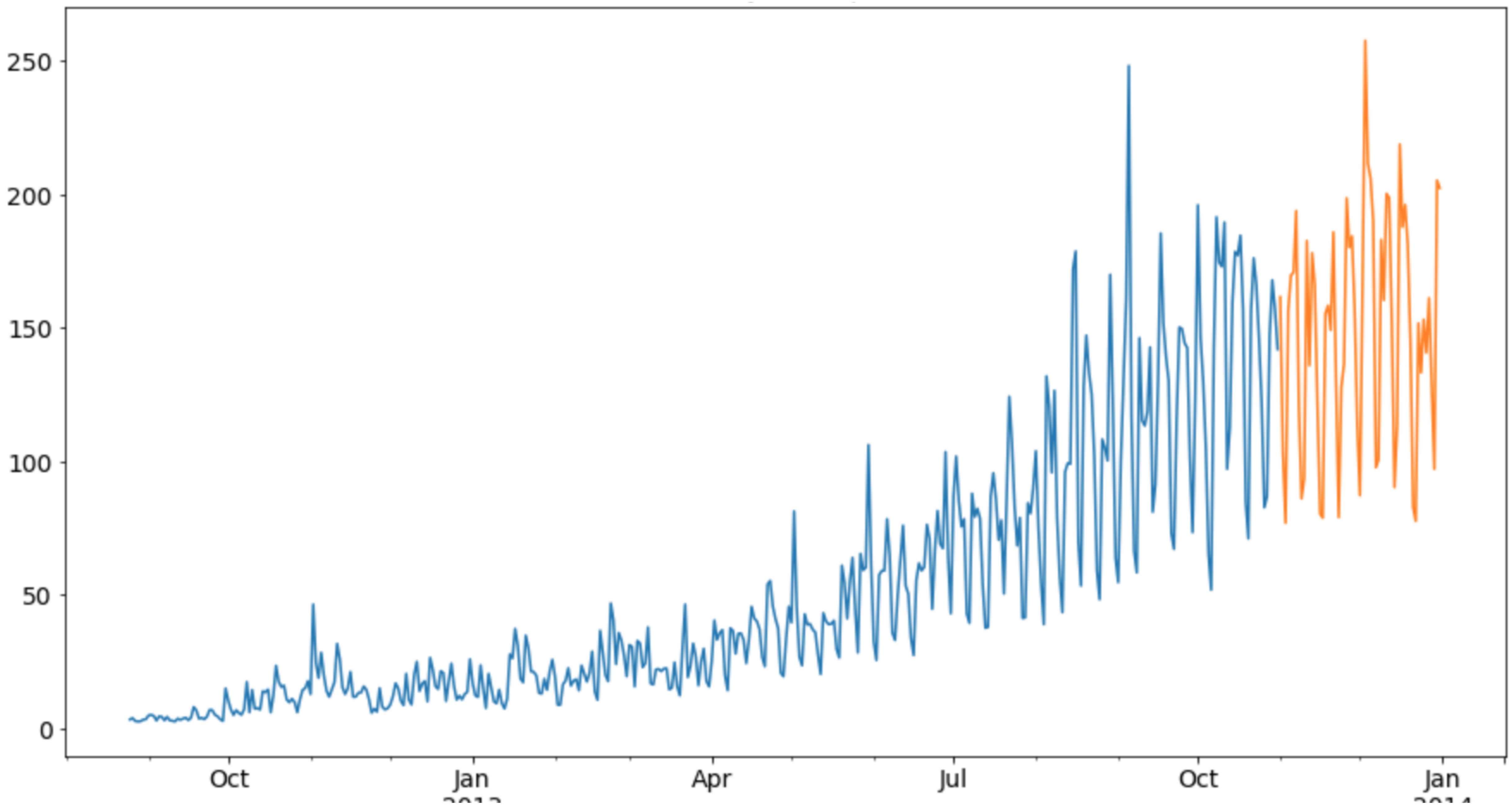
Charlotte Werger
Data Scientist

DataCamp

Expected risk and return based on historic data

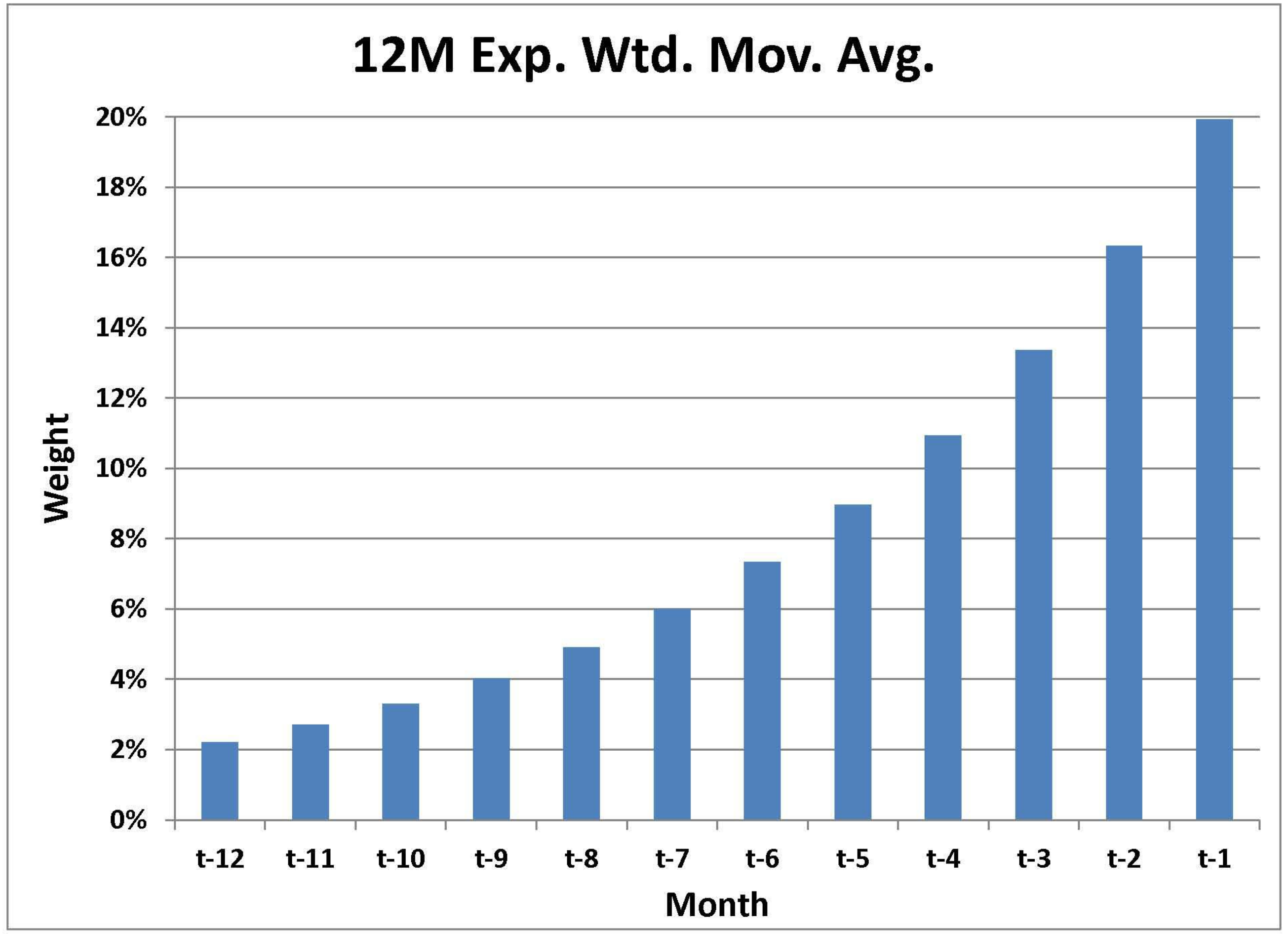
- Mean historic returns, or the historic portfolio variance are **not perfect estimates** of mu and Sigma
- Weights from portfolio optimization therefore **not guaranteed to work well** on future data

Historic data



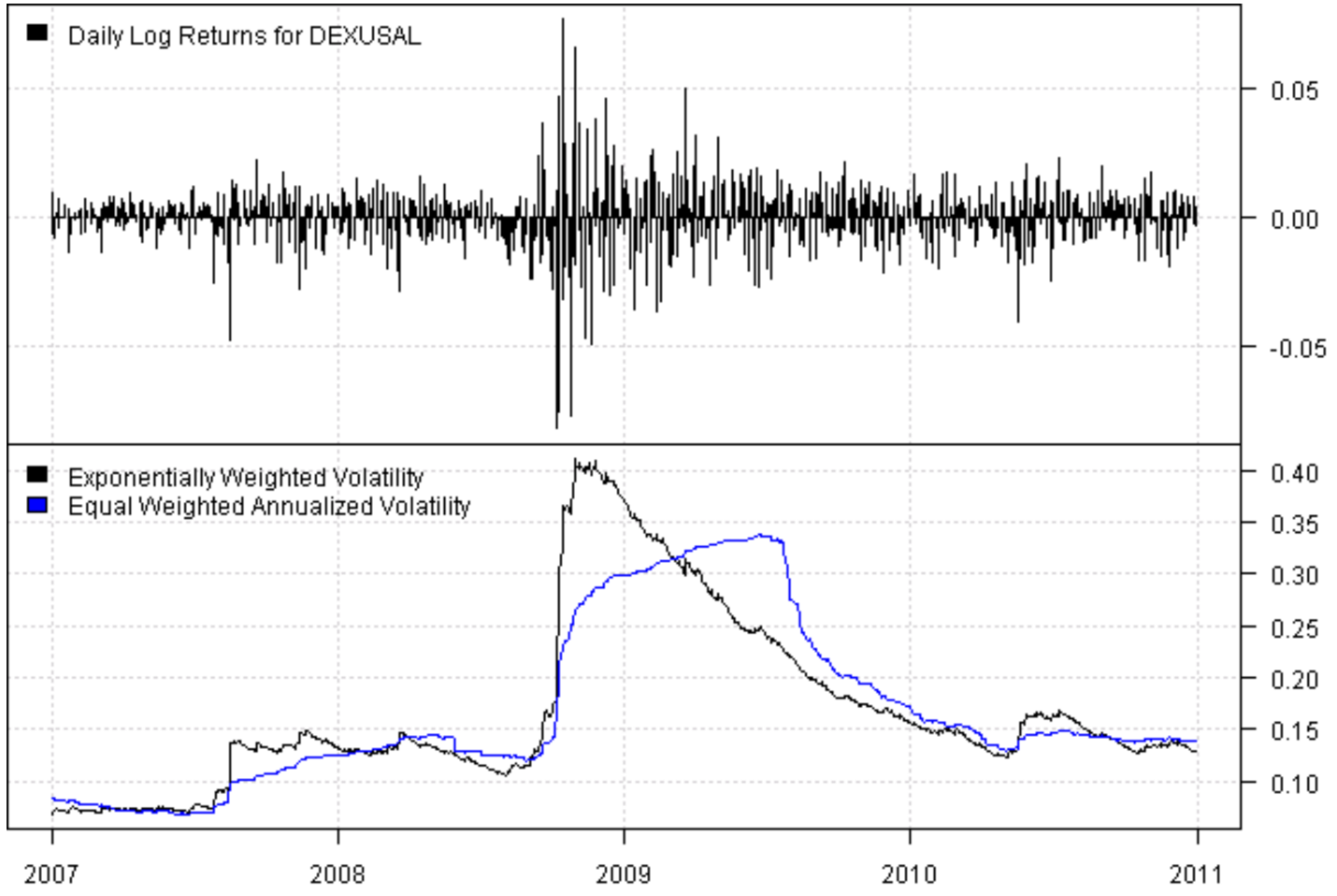
Exponentially weighted returns

- Need better measures for risk and return
- Exponentially weighted risk and return assigns more importance to the most recent data
- Exponential moving average in the graph: most weight on t-1 observation



Exponentially weighted covariance

- The exponential covariance matrix: gives more weight to recent data
- In the graph: exponential weighted volatility in black, follows real volatility better than standard volatility in blue



[1] Source: http://systematicinvestor.github.io/Exponentially-Weighted-Volatility-RCPP

Exponentially weighted returns

```
from pypfopt import expected_returns
```

```
# Exponentially weighted moving average
mu_ema = expected_returns.ema_historical_return(df,
                                span=252, frequency=252)
print(mu_ema)
```

span: a look back window over which you calculate the return

```
symbol
XOM     0.103030
BBY     0.394629
PFE     0.186058
```

Exponentially weighted covariance

```
from pypfopt import risk_models
```

```
# Exponentially weighted covariance
Sigma_ew = risk_models.exp_cov(df, span=180, frequency=252)
```

It does not need to be the same span as for the exponential return calculation.

Using downside risk in the optimization

- Remember the Sortino ratio: it uses the variance of negative returns only
- PyPortfolioOpt allows you to use **semicovariance** in the optimization, this is a measure for downside risk:

$$\text{Downside risk} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\text{return} - \text{target return})^2 \, f(t)}$$

$$f(t) = 1 \text{ if return} < \text{target return}$$

Semicovariance in PyPortfolioOpt

```
Sigma_semi = risk_models.semicovariance(df,
                        benchmark=0, frequency=252)
```

You need to define a benchmark, if the return falls below this benchmark, it's considered a negative return and used in the variance calculation.

```
print(Sigma_semi)
```

	XOM	BBY	MA	PFE
XOM	0.018939	0.008505	0.006568	0.004058
BBY	0.008505	0.016797	0.009133	0.004404
MA	0.006568	0.009133	0.018711	0.005373
PFE	0.004058	0.004404	0.005373	0.008349

Let's practice!

INTRODUCTION TO PORTFOLIO ANALYSIS IN PYTHON

DataCamp

Recap

INTRODUCTION TO PORTFOLIO ANALYSIS IN PYTHON

Charlotte Werger
Data Scientist

DataCamp

Chapter 1: Calculating risk and return

- A portfolio as a collection of weight and assets
- Diversification
- Mean returns versus cumulative returns
- Variance, standard deviation, correlations and the covariance matrix
- Calculating portfolio variance

Chapter 2: Diving deep into risk measures

- Annualizing returns and risk to compare over different periods
- Sharpe ratio as a measured of risk adjusted returns
- Skewness and Kurtosis: looking beyond mean and variance of a distribution
- Maximum draw-down, downside risk and the Sortino ratio

Chapter 3: Breaking down performance

- Compare to benchmark with active weights and active returns
- Investment factors: explain returns and sources of risk
- Fama French 3 factor model to breakdown performance into explainable factors and alpha
- Pyfolio as a portfolio analysis tool

Chapter 4: Finding the optimal portfolio

- Markowitz' portfolio optimization: efficient frontier, maximum Sharpe and minimum volatility portfolios
- Exponentially weighted risk and return, semicovariance

Continued learning

- Datacamp course on Portfolio Risk Management in Python
- Quantopian's lecture series: **https://www.quantopian.com/lectures**
- Learning by doing: Pyfolio and PyPortfolioOpt

End of this course

INTRODUCTION TO PORTFOLIO ANALYSIS IN PYTHON

DataCamp