Financial returns

INTRODUCTION TO PORTFOLIO RISK MANAGEMENT IN PYTHON

Dakota Wixom
Quantitative Analyst | QuantCourse.com

Course overview

Learn how to analyze investment return distributions, build portfolios and reduce risk, and identify key factors which are driving portfolio returns.

- Univariate Investment Risk

- Portfolio Investing

- Factor Investing

- Forecasting and Reducing Risk
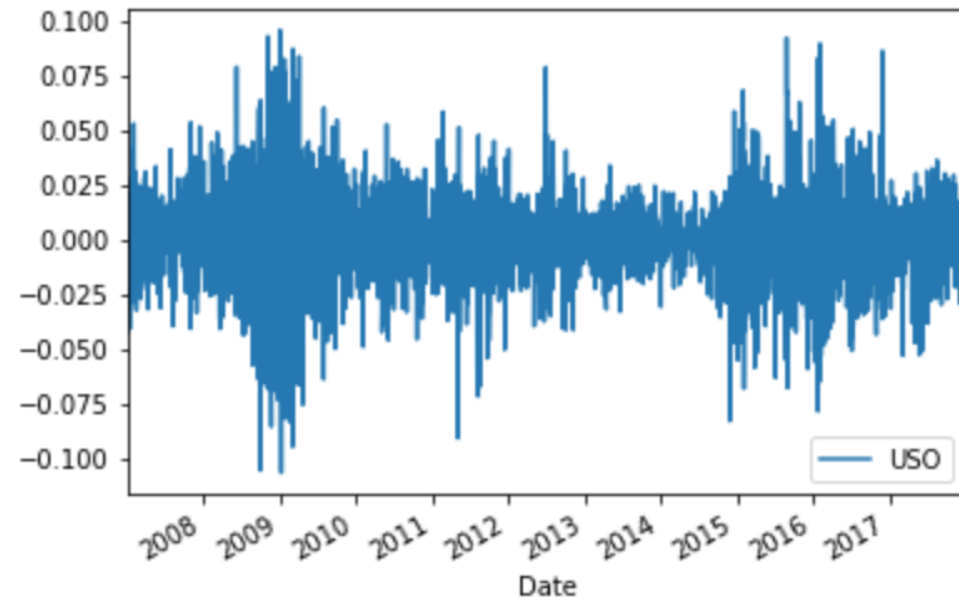
Investment risk

What is Risk?

- Risk in financial markets is a measure of uncertainty

- Dispersion or variance of financial returns
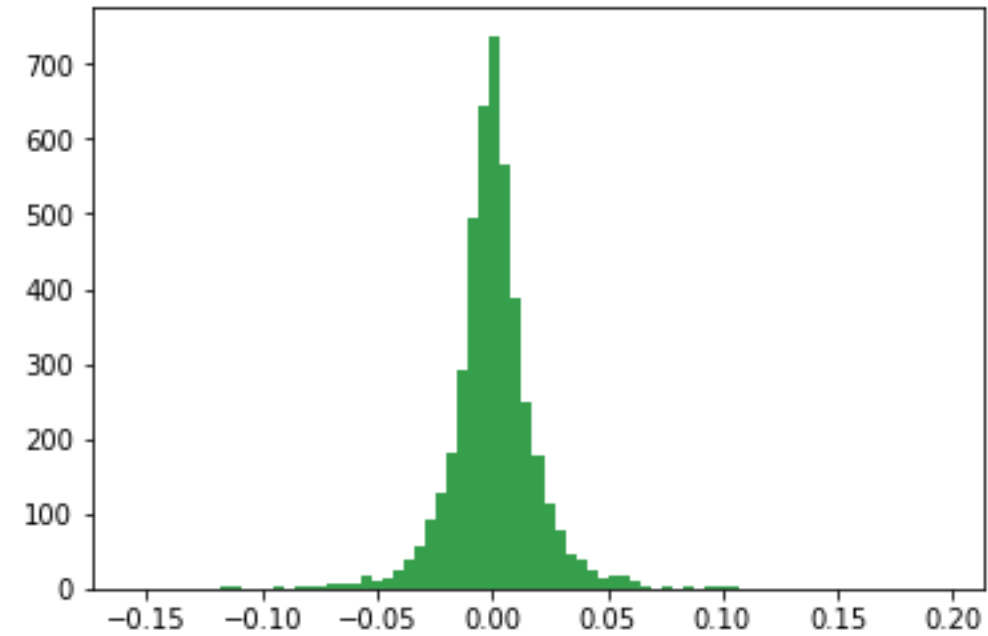
How do you typically measure risk?

- Standard deviation or variance of daily returns

- Kurtosis of the daily returns distribution

- Skewness of the daily returns distribution

- Historical drawdown

Financial risk

Returns

Probability

A tale of two returns

- Returns are derived from stock prices

- **Discrete returns** (simple returns) are the most commonly used, and represent periodic (e.g. daily, weekly, monthly, etc.) price movements

- **Log returns** are often used in academic research and financial modeling. They assume continuous compounding.

Discrete vs. Log Returns

Log returns are always smaller than discrete returns
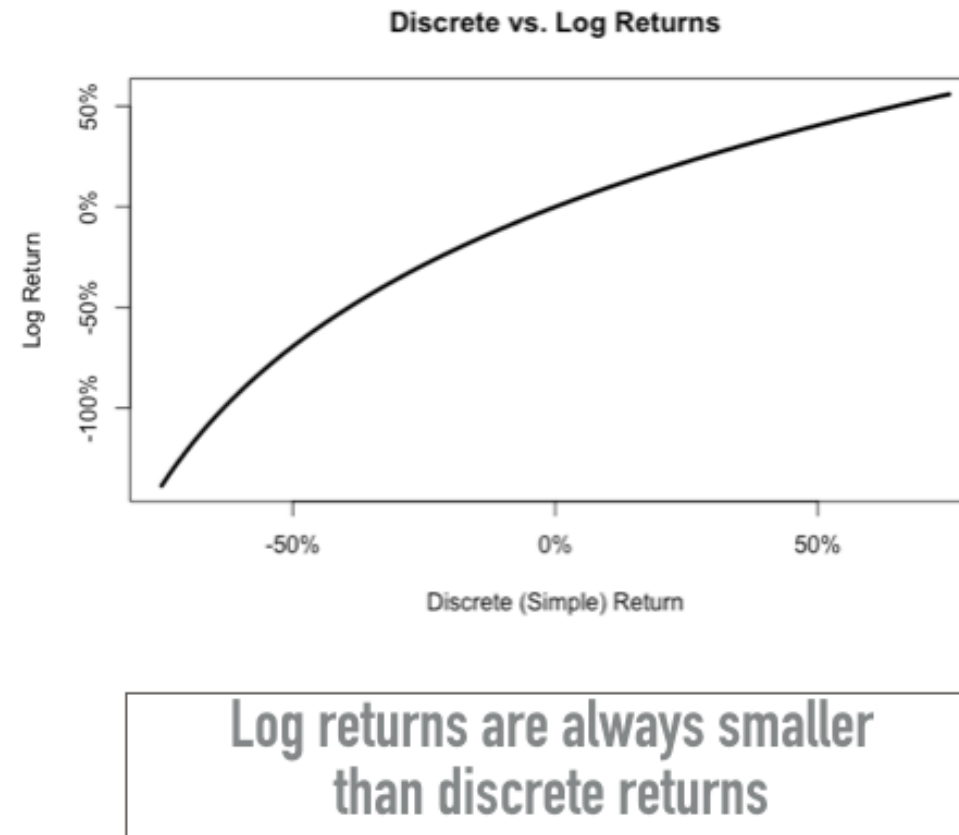
Calculating stock returns

- Discrete returns are calculated as the change in price as a percentage of the previous period's price

Calculating Discrete Returns

$$R_{t_2} = \frac{(P_{t_2} - P_{t_1})}{P_{t_1}}$$

Calculating log returns

- Log returns are calculated as the difference between the log of two prices

- Log returns *aggregate across time*, while discrete returns *aggregate across assets*

$$Rl = \ln(\frac{P_{t_2}}{P_{t_1}})$$

or equivalently

$$Rl = \ln(P_{t_2}) - \ln(P_{t_1})$$

Calculating stock returns in Python

Step 1:

Load in stock prices data and store it as a pandas DataFrame organized by date:

```python
import pandas as pd
StockPrices = pd.read_csv('StockData.csv', parse_dates=['Date'])
StockPrices = StockPrices.sort_values(by='Date')
StockPrices.set_index('Date', inplace=True)
```

Calculating stock Returns in Python

Step 2:

Calculate daily returns of the adjusted close prices and append the returns as a new column in the DataFrame.

```python
StockPrices["Returns"] = StockPrices["Adj Close"].pct_change()
StockPrices["Returns"].head()
```

Date	Open	High	Low	Close	Adj Close	Volume	Returns
2000-01-03	58.68750	59.3125	56.00000	58.28125	42.641369	53228400	NaN
2000-01-04	56.78125	58.5625	56.12500	56.31250	41.200928	54119000	-0.033780
2000-01-05	55.56250	58.1875	54.68750	56.90625	41.635361	64059600	0.010544
2000-01-06	56.09375	56.9375	54.18750	55.00000	40.240646	54976600	-0.033498
2000-01-07	54.31250	56.1250	53.65625	55.71875	40.766510	62013600	0.013068

Visualizing return distributions

```python
import matplotlib.pyplot as plt
plt.hist(StockPrices["Returns"].dropna(), bins=75, density=False)
plt.show()
```

Let's practice!

datacamp
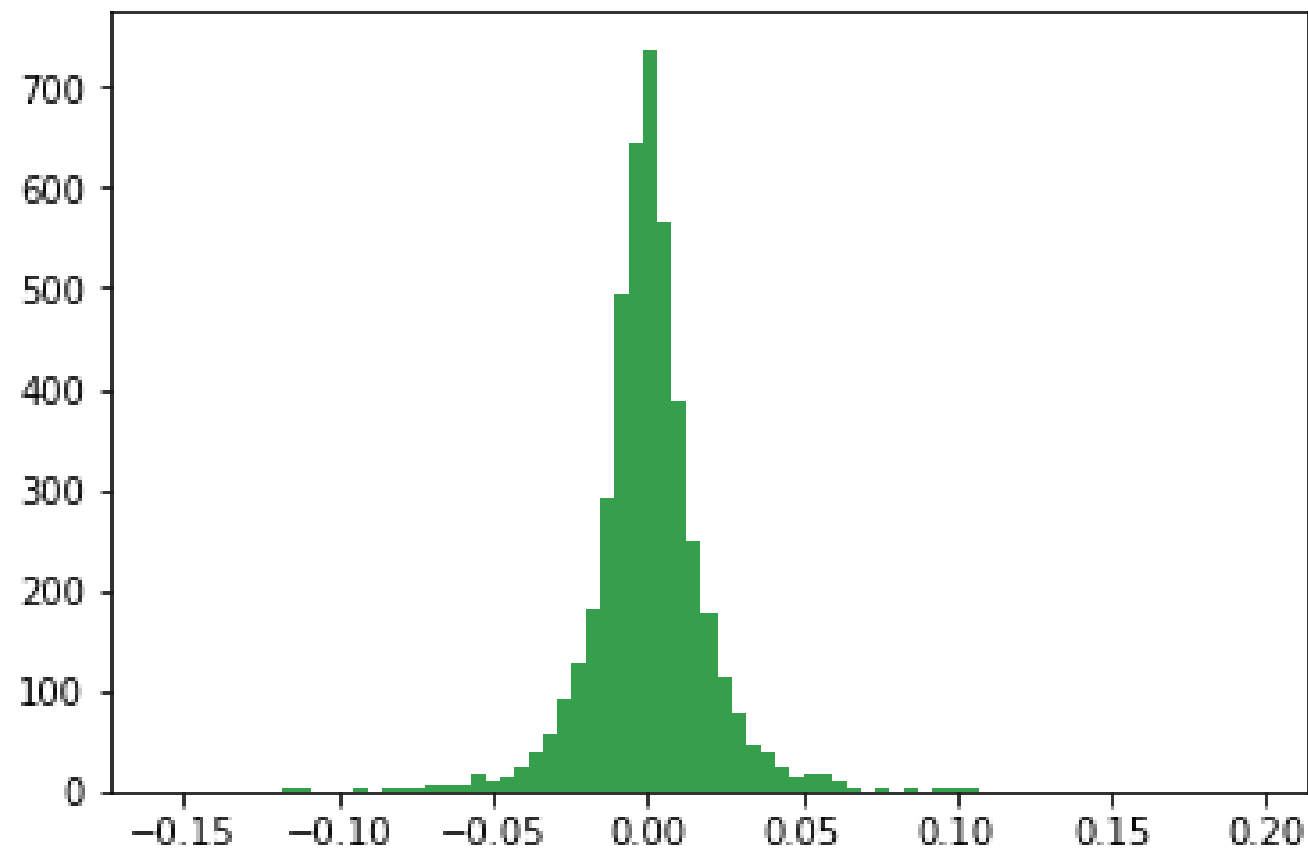
Mean, variance, and normal distribution

INTRODUCTION TO PORTFOLIO RISK MANAGEMENT IN PYTHON

Dakota Wixom

Quantitative Analyst | QuantCourse.com

datacamp

Moments of distributions

Probability distributions have the following moments:

- 1) Mean (μ)

- 2) Variance (σ^2)

- 3) Skewness

- 4) Kurtosis

There are many types of distributions. Some are normal and some are non-normal. A random variable with a **Gaussian distribution** is said to be *normally distributed*.

Normal Distributions have the following properties:

- Mean = μ

- Variance = σ^2

- Skewness = 0

- Kurtosis = 3

Probability Density Function of Normal Distributions

$\mu=0, \quad \sigma^2=0.2,$ ——
$\mu=0, \quad \sigma^2=1.0,$ ——
$\mu=0, \quad \sigma^2=5.0,$ ——
$\mu=-2, \quad \sigma^2=0.5,$ ——
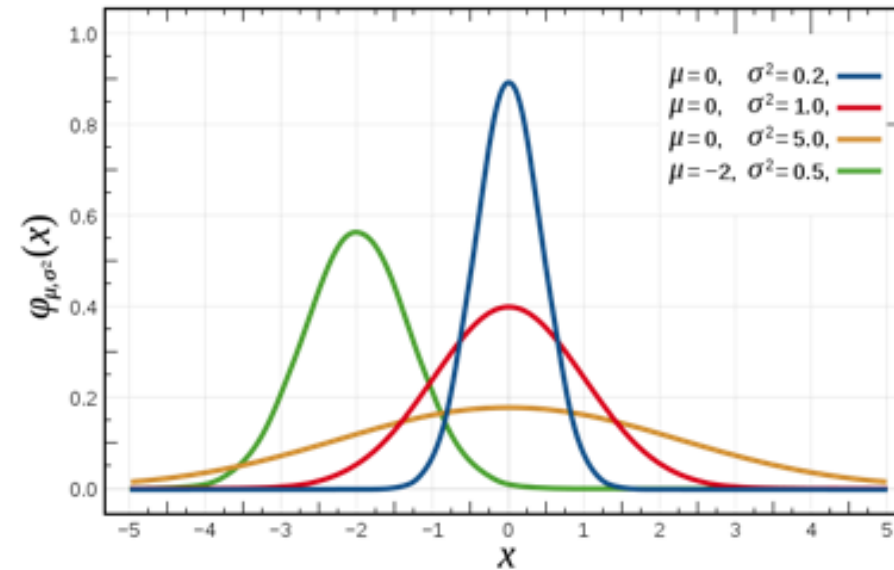
Probability Density Function Equation of a Standard Normal Distribution

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$
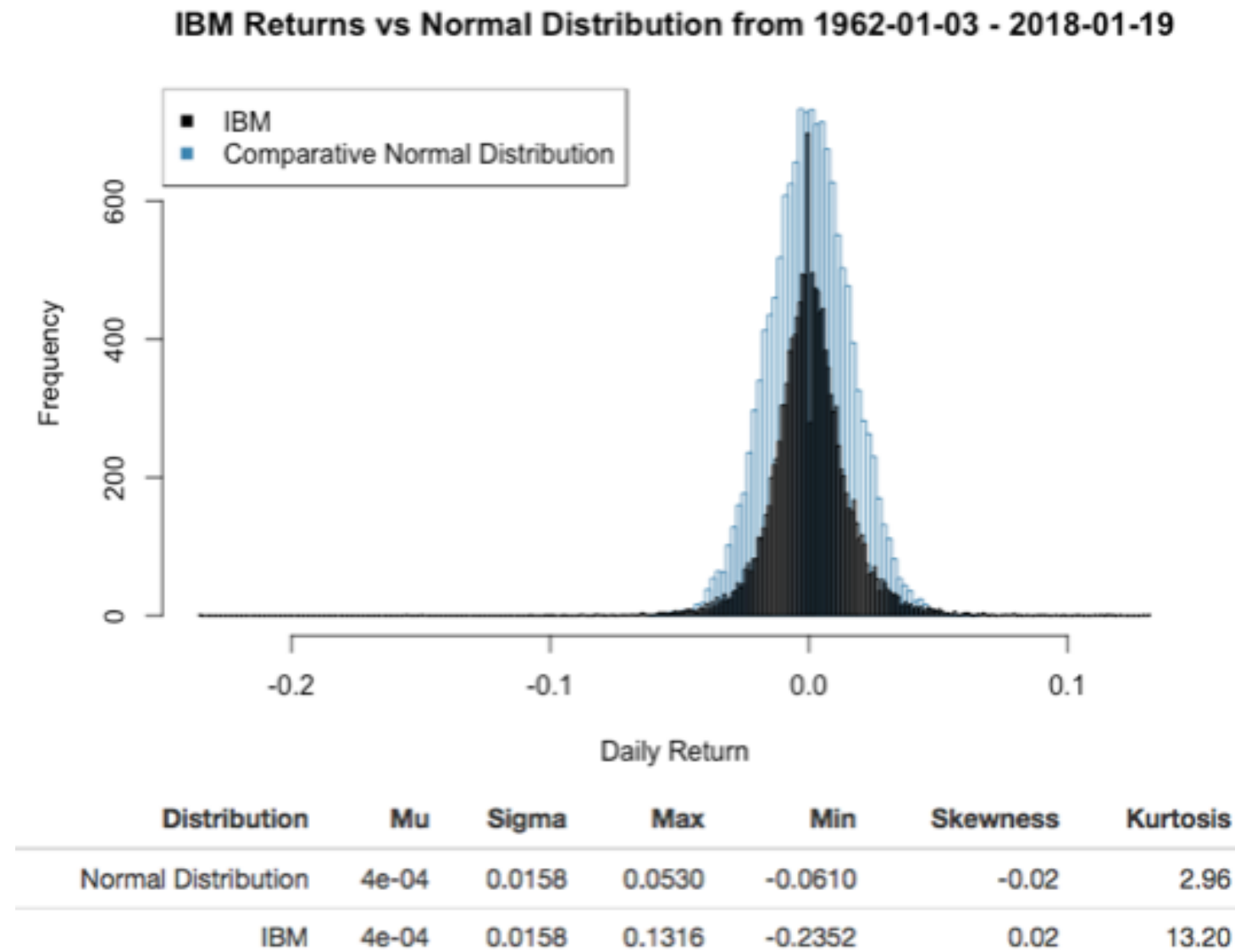
The standard normal distribution

The **Standard Normal** is a special case of the Normal Distribution when:

- $\sigma = 1$

- $\mu = 0$

Comparing against a normal distribution

- Normal distributions have a skewness near 0 and a kurtosis near 3.

- Financial returns tend not to be normally distributed

- Financial returns can have high kurtosis

Comparing against a normal distribution

IBM Returns vs Normal Distribution from 1962-01-03 - 2018-01-19

Distribution	Mu	Sigma	Max	Min	Skewness	Kurtosis
Normal Distribution	4e-04	0.0158	0.0530	-0.0610	-0.02	2.96
IBM	4e-04	0.0158	0.1316	-0.2352	0.02	13.20

Calculating mean returns in python

To calculate the average daily return, use the `np.mean()` function:

```python
import numpy as np
np.mean(StockPrices["Returns"])
```

```
0.0003
```

To calculate the average annualized return assuming 252 trading days in a year:
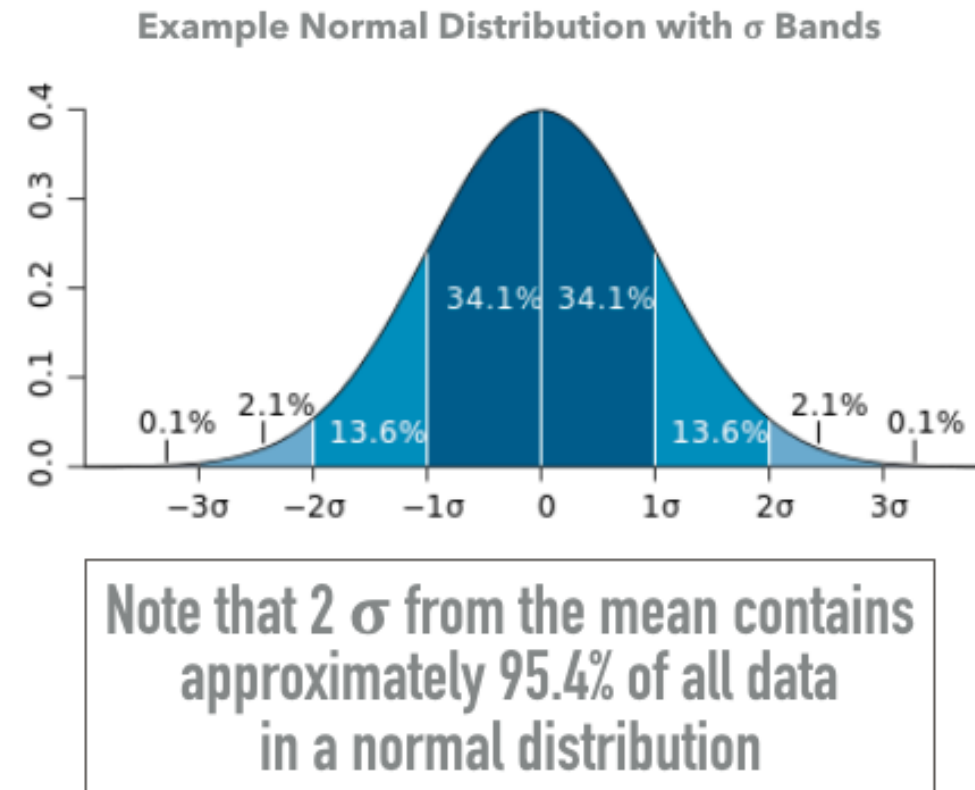
```python
import numpy as np
((1+np.mean(StockPrices["Returns"]))**252)-1
```

```
0.0785
```

Standard deviation and variance

Standard Deviation (Volatility)

- Variance = σ^2

- Often represented in mathematical notation as σ, or referred to as *volatility*

- An investment with higher σ is viewed as a higher risk investment

- Measures the dispersion of returns

Example Normal Distribution with σ Bands

Note that 2 σ from the mean contains approximately 95.4% of all data in a normal distribution

Standard deviation and variance in Python

Assume you have pre-loaded stock returns data in the `StockData` object. To calculate the periodic standard deviation of returns:

```python
import numpy as np
np.std(StockPrices["Returns"])
```

```
0.0256
```

To calculate variance, simply square the standard deviation:

```python
np.std(StockPrices["Returns"])**2
```
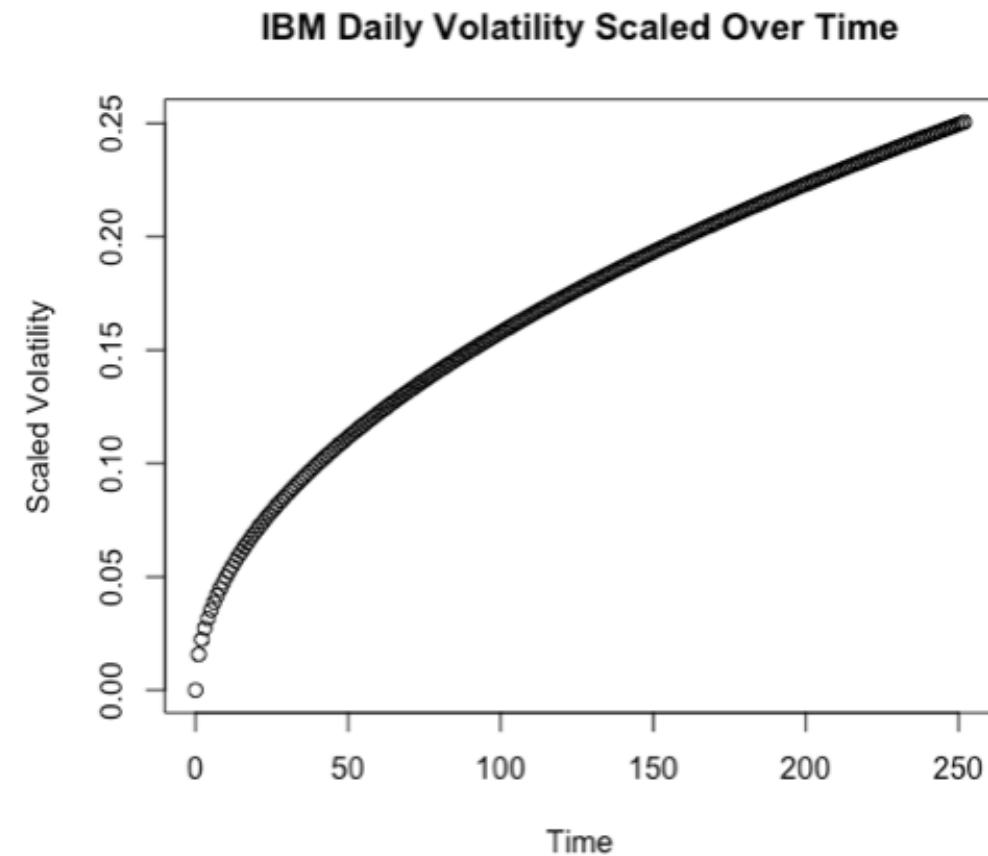
```
0.000655
```

Scaling volatility

- Volatility scales with the square root of time

- You can normally assume 252 trading days in a given year, and 21 trading days in a given month

Example Volatility Scaling Equations

$$\sigma_{Annual} = \sigma_{Daily} * \sqrt{(252)}$$

$$\sigma_{Monthly} = \sigma_{Daily} * \sqrt{(21)}$$

IBM Daily Volatility Scaled Over Time

Scaling volatility in Python

Assume you have pre-loaded stock returns data in the `StockData` object. To calculate the annualized volatility of returns:

```python
import numpy as np
np.std(StockPrices["Returns"]) * np.sqrt(252)
```

```
0.3071
```

Let's practice!

datacamp

Skewness and kurtosis

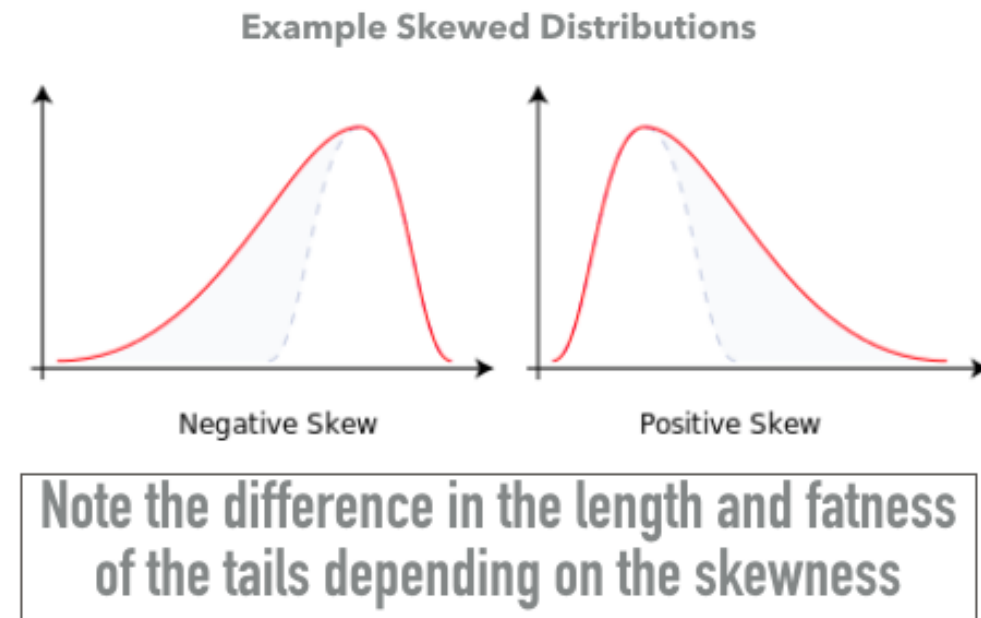## INTRODUCTION TO PORTFOLIO RISK MANAGEMENT IN PYTHON

Dakota Wixom
Quantitative Analyst | QuantCourse.com

Skewness is the third moment of a distribution.

- **Negative Skew:** The mass of the distribution is concentrated on the right. Usually a right-leaning curve

- **Positive Skew:** The mass of the distribution is concentrated on the left. Usually a left-leaning curve

- In finance, you would tend to want positive skewness

Example Skewed Distributions

Negative Skew   Positive Skew

Note the difference in the length and fatness of the tails depending on the skewness

Skewness in Python

Assume you have pre-loaded stock returns data in the
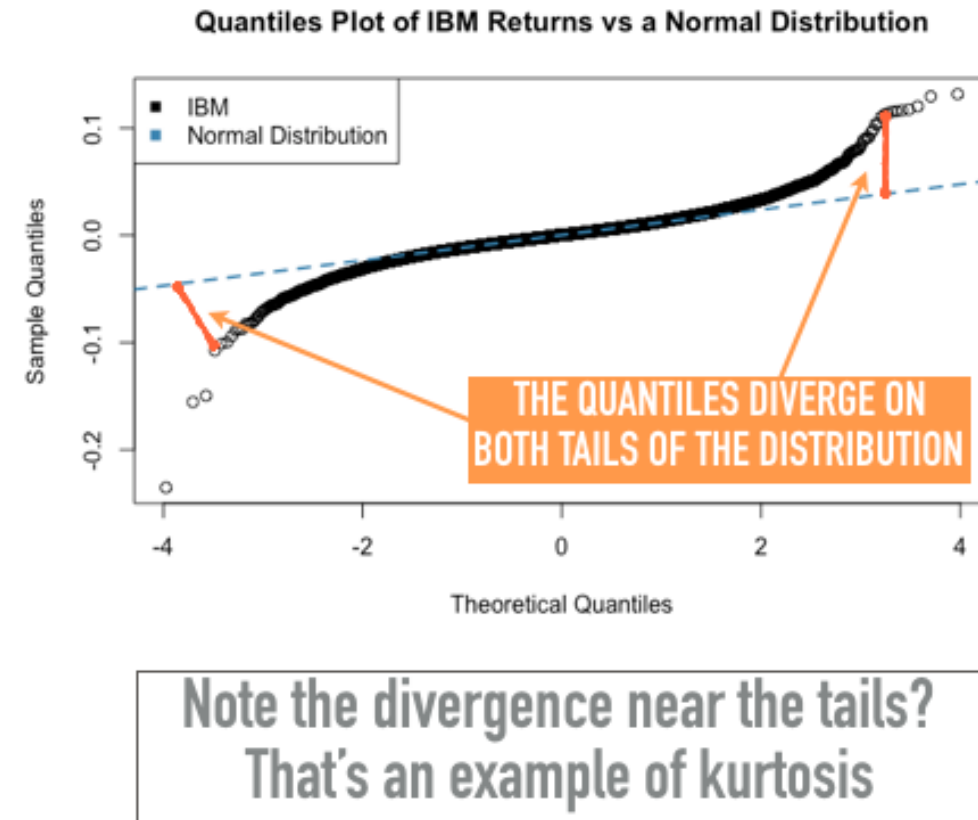`StockData` object.

To calculate the skewness of returns:

```python
from scipy.stats import skew
skew(StockData["Returns"].dropna())
```

```
0.225
```

Note that the skewness is higher than 0 in this example,
suggesting non-normality.

Kurtosis is a measure of the thickness of the tails of a distribution

- Most financial returns are leptokurtic

- **Leptokurtic:** When a distribution has positive excess kurtosis (kurtosis greater than 3)

- **Excess Kurtosis:** Subtract 3 from the sample kurtosis to calculate "Excess Kurtosis"

Quantiles Plot of IBM Returns vs a Normal Distribution

THE QUANTILES DIVERGE ON BOTH TAILS OF THE DISTRIBUTION

Note the divergence near the tails? That's an example of kurtosis

Excess kurtosis in Python

Assume you have pre-loaded stock returns data in the `StockData` object. To calculate the **excess kurtosis** of returns:

```python
from scipy.stats import kurtosis
kurtosis(StockData["Returns"].dropna())
```

```
2.44
```

Note the excess kurtosis greater than 0 in this example, suggesting non-normality.

Testing for normality in Python

How do you perform a statistical test for normality?

The null hypothesis of the **Shapiro-Wilk test** is that the data are normally distributed.

```python
# Run the Shapiro-Wilk normality test in Python
from scipy import stats
p_value = stats.shapiro(StockData["Returns"].dropna())[1]
if p_value <= 0.05:
    print("Null hypothesis of normality is rejected.")
else:
    print("Null hypothesis of normality is accepted.")
```

The p-value is the second variable returned in the list. If the p-value is less than 0.05, the null hypothesis is rejected because the data are most likely non-normal.

Let's practice!

INTRODUCTION TO PORTFOLIO RISK MANAGEMENT IN PYTHON

datacamp