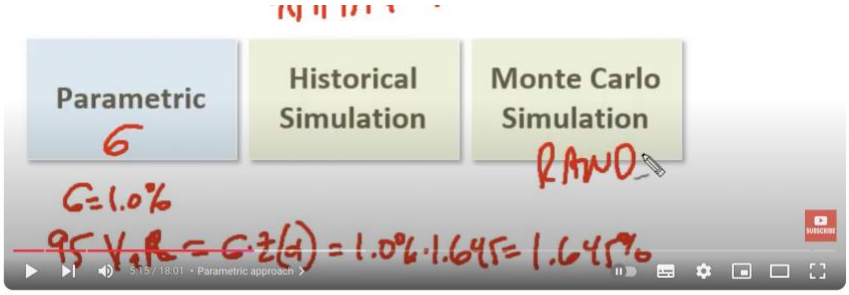
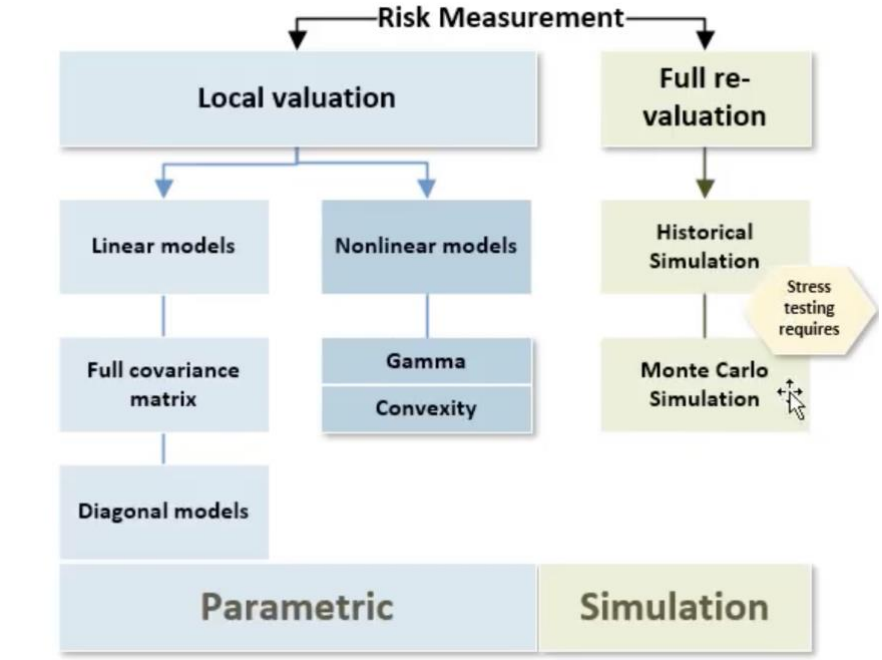
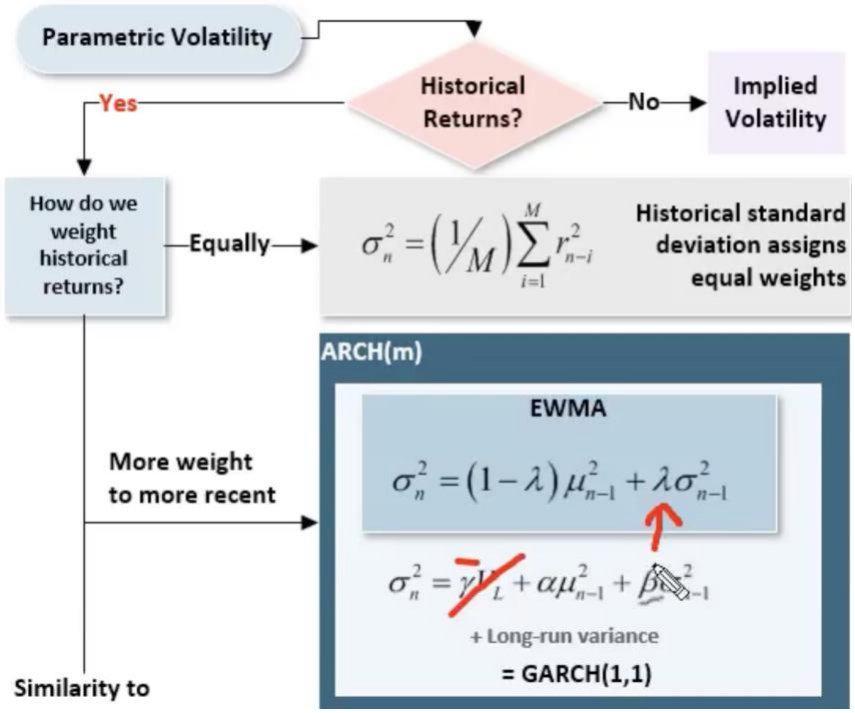
Financial Asset & Return Analysis

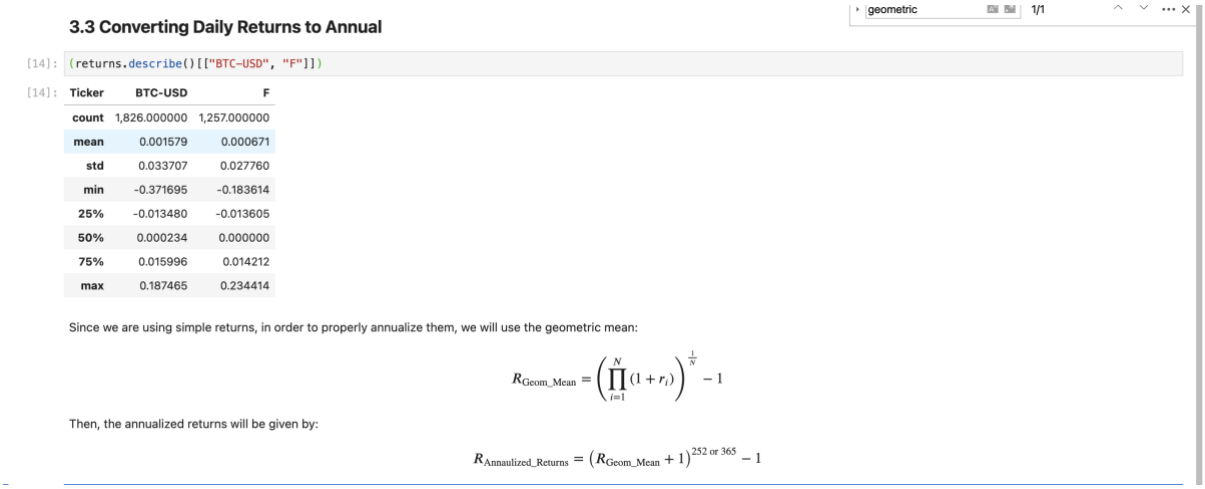


Three approaches to value at risk (VaR) and volatility (FRM T4-1)





L1: Financial Asset Data



3.3 Converting Daily Returns to Annual

[14]:
```
(returns.describe()[["BTC-USD", "F"]])
```

[14]:

Ticker	BTC-USD	F
count	1,826.000000	1,257.000000
mean	0.001579	0.000671
std	0.033707	0.027760
min	-0.371695	-0.183614
25%	-0.013480	-0.013605
50%	0.000234	0.000000
75%	0.015996	0.014212
max	0.187465	0.234414

Since we are using simple returns, in order to properly annualize them, we will use the geometric mean:

$$R_{\text{Geom\_Mean}} = \left(\prod_{i=1}^{N}(1+r_i)\right)^{\frac{1}{N}} - 1$$

Then, the annualized returns will be given by:

$$R_{\text{Annaulized\_Returns}} = \left(R_{\text{Geom\_Mean}} + 1\right)^{252 \text{ or } 365} - 1$$

L2:

2. Variance and Standard Deviation

Investors have to keep volatility in mind as well when choosing an investment. For example, a pension fund may need to be extra careful with its money and will want to ensure they aren't getting into any extremely volatile investments. There are also hedge funds, which short stocks and even trade volatility with options. As you can see, whether you're risk-seeking or risk-averse, the volatility (risk) of an investment is something you should care about.

A simple measure of volatility is the variance. Variance is used to see how far away each data point in a set is away from the mean. Variance is calculated with the following steps:

- Take the difference between each data point and the mean
- Square each difference so that they're all positive values
- Sum up the squared results
- Divide this by the count of data points minus one

$$\sigma^2 = \frac{\sum(x_i - \overline{x})^2}{n - 1}$$

where

σ^2 = sample variance
x_i = value of one observation
$\overline{x}$ = mean of all observations
n = number of observations

The larger the variance, the further spread out it is from the mean. Variance treats all deviations from the mean the same way, regardless of whether they are less than or greater than the mean. A variance of zero would indicate that each data point is the same.

Standard deviation is easy to calculate once you have the variance. All you have to do is take the square root of the variance:

3. Covariance and Correlation

We are able to compare the performance of stocks that have different price levels by using returns and standard deviation. How can we look at the joint performance of two stocks? For this, we turn to covariance and correlation. We will start with covariance:

$$\frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{N - 1}$$

where

x_i = value of one observation of x
y_i = value of one observation of y
$\overline{x}$ = mean of x
$\overline{y}$ = mean of y
N = number of observations

Covariance provides us some insight into how two variables move together. A positive covariance between stock returns would indicate that when one stock goes up, so does the other and vice versa. A negative covariance would mean that the two stocks move inversely, i.e., when one goes up, the other goes down.

Looking at the covariance matrix below for our three assets, we can see that these assets have a positive relationship with each other. It's hard to understand much more than that with these numbers, given that the units are not standardized.

3.1 Using a Covariance Matrix

While covariance is useful for determining the direction of two variables jointly, we can turn to correlation for a more standardized version of this. For now, when discussing correlation, we will use the Pearson's correlation coefficient. The other types of correlation will be discussed in a future lesson.

[11]:
```
df.cov()
```

[11]:

Ticker	SP500	NASDAQ	Bitcoin
Ticker			
SP500	0.000181	0.000203	0.000200
NASDAQ	0.000203	0.000254	0.000253
Bitcoin	0.000200	0.000253	0.001726

Here's how to interpret it:

- Diagonal values: These represent the variance of each asset. For example, 0.000149 is the variance of the S&P 500 daily log returns.
- Off-diagonal values: These represent the covariance between two different assets. For example, 0.000160 is the covariance between the daily log returns of the S&P 500 and NASDAQ.

All the values are positive, indicating a positive relationship between the returns of these assets. This means that when the return of one asset is positive, the return of the other assets tends to be positive as well.

3.2 Using the Pearson Correlation Coefficient Formula

The Pearson correlation coefficient is a measure of the strength of a linear relationship between two variables. The formula is as follows:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

where

cov = Covariance
σ_x = Standard Deviation of X
σ_y = Standard Deviation of Y

Correlation is used in statistics to quantify the degree to which two variables move in a linear relation to each other. Correlation can range from –1 to 1 inclusive. A correlation of 1 means perfect correlation; the variables move exactly in tandem with one another. Perfect tandem for positive correlation means that if we know variable X increases, then variable Y also increases. A correlation of -1 indicates perfect inverse correlation. This is also perfect tandem, but in the opposite direction. Here, if we know variable X increases, then variable Y decreases. A correlation of 0 indicates there is no linear distinguishable relationship between two variables; therefore, it would be impossible to make predictions of one variable given the other. In this case, if we know variable X increases, then variable Y is equally likely to increase or decrease.

A benefit of correlation over covariance is that correlation is capped from -1 to 1 while covariance can be from -inf to inf. This makes covariance a harder statistic to understand intuitively. Correlation is also proportional, which will be shown in the video below.

Keep in mind that a lot of models and financial concepts assume a constant correlation, but this is rarely the case. Correlation changes over time and will likely even change if you adjust the time range from which you're measuring correlation.

When comparing the correlations of the assets we've been using so far, you can see it paints a clearer picture than the covariance did.

9]:
```
round(df.corr(), 3)
```

3.3 The Sharpe Ratio

Are there any statistics we can use to quantify not just return but also risk? For this, we turn to the Sharpe ratio.

The Sharpe ratio allows an investor to understand the relationship between the return of an investment and its volatility. The formula is as follows:

$$\text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma_p}$$

where

R_p = Return of the portfolio
R_f = Risk-free rate
σ_p = Standard deviation of the portfolio

In many cases with interest rates so low, investors will assume the risk-free rate to be 0, making the ratio:

$$\text{Sharpe Ratio} = \frac{R_p}{\sigma_p}$$

Notice anything about the denominator? Yes, that's right: we use the standard deviation here to represent risk.

This measure is used as a way of scaling the return of an investment depending on how much risk is taken. In other words, the higher the standard deviation, the more the risk-weighted return is reduced. Let's use the same example as above with S&P 500 and Bitcoin daily returns over the last five years.

[ ]:
```
# Calculates the Sharpe Ratios for both the S&P 500 and Bitcoin
Sharpe_Ratio_SP500 = df["SP500"].mean() / df["SP500"].std()
Sharpe_Ratio_Bitcoin = df["Bitcoin"].mean() / df["Bitcoin"].std()

# Print the results
print("Sharpe Ratio of S&P 500: ", round(Sharpe_Ratio_SP500, 5))
```

4. Semivariance

How can we refine the Sharpe ratio to give an even better measure of risk-adjusted returns? Semivariance is the answer. Semivariance, also known as downside risk, is a more refined version of a standard deviation. Standard deviation looks at both the upside and downside risk of an investment. Most investors, unless you're trading short, care much more about the downside risk than the upside risk. In other words, if you bought a stock and are looking at the Sharpe ratio, you wouldn't want this number to be penalized for how far it moves to the upside. Most of the time, an investor will be much more concerned with the downside risk of a stock.

$$\text{Semivariance} = \frac{1}{n}\sum_{r_i<\bar{r}}^{n}(r_i - \bar{r})^2$$

where

r_i = value of one observation
$\bar{r}$ = mean of all observations
n = number of observations

Semivariance can be thought of as an estimator of variance of the returns that are less than their average. This can be used to estimate the average loss a portfolio could incur, assuming normal distributions of returns.

Conversely, if we are short a security, we could still use semivariance, but this time, it would focus on the returns that are positive. If we are short, drops in the price create upside, so we would not want to penalize this volatility in our Sharpe ratio. However, if there are large deviations in the upward direction, then this will contribute to semivariance. In short, semivariance uses either the positive or negative returns.

Now let's compute semivariance for both the S&P 500 and Bitcoin:

[ ]:
```
# Calculate the mean return for each
sp500mean = df["SP500"].mean()
BTCmean = df["Bitcoin"].mean()
```

Now let's compute semivariance for both the S&P 500 and Bitcoin:

```
# Calculate the mean return for each
sp500mean = df["SP500"].mean()
BTCmean = df["Bitcoin"].mean()

# Calculate semivariance for each
sp500_semivariance = ((df[df["SP500"] < sp500mean]["SP500"] - sp500mean) ** 2).mean()
BTC_semivariance = ((df[df["Bitcoin"] < BTCmean]["Bitcoin"] - BTCmean) ** 2).mean()

# Print the semivariance results
print("Semivariance of S&P 500: ", round(sp500_semivariance, 5))
print("Semivariance of Bitcoin: ", round(BTC_semivariance, 5))
```

```
Semivariance of S&P 500:  0.00021
Semivariance of Bitcoin:  0.00181
```

This output shows that Bitcoin has a much higher semivariance (0.00181) compared to the S&P 500 (0.00021). This indicates that Bitcoin has experienced significantly larger negative deviations from its average return, suggesting higher downside risk.

5. How Are Stock Returns Distributed?

Many models and theories surrounding stocks assume a normal distribution. We will try to determine that here with a data-based analysis. Properties of a Gaussian distribution are as follows:

- Mean, median, and mode are all the same.
- The data is symmetrical, meaning there are equal counts of observations on both sides of the mean.
- In normally distributed data, 68.25% of all cases fall within +/- one standard deviation from the mean, 95% of all cases fall within +/- two standard deviations from the mean, and 99.7% of all cases fall within +/- three standard deviations from the mean.

Let's start by pulling 20 years of daily price data for the S&P 500. We'll use similar methods we've used in the last few lessons to pull this data and will calculate the log returns here.

One quick way of doing this is to determine how many data points we have on either side of the mean here. We have a bit more than 5,000 data points. The below code takes the count of data points greater than the mean and divides it by the total number of data points. This will give us the percentage of data points greater than the mean.

[32]:
```
# Starting and end dates
start = datetime.date(2004, 8, 1)
end = datetime.date(2024, 8, 1)

# Get the data
prices = pd.DataFrame(yfin.download(["^GSPC"], start, end)["Adj Close"])

# Rename column to make names more intuitive
prices = prices.rename(columns={"Adj Close": "SP500"})
df = np.log(prices) - np.log(prices.shift(1))
df = df.iloc[1:, 0:]
```

99.7% data will fall in 3 standard deviations.

5.1 Are Returns Symmetric?

One quick way of doing this is to determine how many data points we have on either side of the mean here. We have a bit more than 5,000 data points here. The below code takes the count of data points greater than the mean and divides it by the total number of data points. This will give us the percentage of data points greater than the mean.

0]:
```
(len(df[df.SP500 > df.SP500.mean()])) / (len(df))
```

0]:
```
0.5235446056030201
```

The output will be a number between 0 and 1, representing the proportion of S&P 500 daily returns that are greater than the mean return. This can give you an idea of the skewness of the return distribution. A value close to 0.5 suggests a roughly symmetrical distribution, while a value significantly greater than 0.5 suggests a negative skew (more returns below the mean).

We're getting about 52.6% of data points being greater than the mean, which shows we have a slightly negative skew to this dataset. We can't rule out symmetric returns based on this since it is only a sample of data and is reasonably close to the 50% mark. This makes it hard to say for certain whether S&P 500 returns are symmetric or not, but it is still a reasonable assumption to make here.

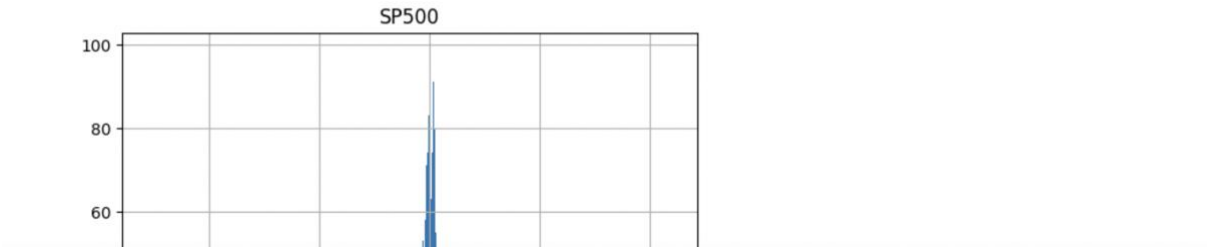
6. Are Stock Returns Normally Distributed?

The normal distribution is one of the most common distributions used in modeling random variables. Indeed, many phenomena in the natural and social sciences can be modeled by normal distribution. One of the great advantages of the normal distribution is its simplicity. We can completely describe a normal distribution through two numbers: one for the center of the distribution and one for the uncertainty about that center. The first number refers to the mean, and the second number refers to the standard deviation.

Once we have these two numbers, we can draw inferences, estimate percentiles, compute probabilities that a point falls within a region, and more. If our data is well represented by the normal distribution, then we can confidently use the mean and standard deviation to report our portfolio expected returns and volatilities. If our data is not well represented by the normal distribution, then we need to find other distributions that are more suitable. Thus, when we have a distribution of stock returns, for example, we'll want to start this assessment by visualizing the returns to see if they appear to be normal. Of course, we can follow this up with more quantitative assessments by running statistical tests.

We can visualize the data using the `hist()` method. We pass in `bins = 100` as a parameter to determine the amount of buckets to place the data in. The more bins you have, the more granular the data will look in a histogram. Increasing the bins too much may result in slightly noisy data, which will make it tougher to determine a normal distribution.

[73]:
```
df.hist(bins=1000);
```



6.1 Conducting a Normality Test

We can use the `normaltest()` method to determine if the sample data could fit a normal distribution. This method uses D'Agostino and Pearson's normality test, which combines skew and kurtosis to produce an omnibus test of normality.

[17]:
```
stats.normaltest((np.array(df.SP500)))
```

[17]:
```
NormaltestResult(statistic=1234.7310969195069, pvalue=7.61288004534643e-269)
```

The output of this code includes two values:

- Statistic: A test statistic that measures the deviation from normality. Higher values indicate a greater deviation.
- p-value: The probability of observing the data if it were truly normally distributed. A small p-value (typically less than 0.05) suggests that the data is not normally distributed.

The output above shows the results of the D'Agostino and Pearson normality test on our S&P 500 log return data. The statistic of 1234.73 is very high, indicating a substantial deviation from a normal distribution. The extremely small p-value (7.61e-269) provides strong evidence to reject the null hypothesis that the data is normally distributed. In simpler terms, this test strongly suggests that the S&P 500 daily log returns in our dataset do not follow a normal distribution.

6.2 Testing Skewness and Kurtosis

As one added testing step, we can test the skewness and kurtosis of our distribution using the Jarque-Bera test. The test statistic will always be greater than zero. The further the test statistic is from zero, the more likely the sample data does not match a normal distribution.

Lucky for us, Python has another library for us to use here, which really simplifies the analysis. From the `scipy.stats` library, we can apply the `jarque_bera()` method directly to our data to get the test statistic.

[18]:
```
stats.jarque_bera((np.array(df.SP500))).pvalue
```

[18]:
```
0.0
```

A small p-value (typically less than 0.05) indicates that the data is likely not normally distributed. This output indicates that the p-value from the Jarque-Bera test is 0.0 (or extremely close to zero). This strongly suggests that the S&P 500 daily log returns in your dataset are not normally distributed.

6.3 Where Does Our Gaussian Distribution Break Down?

According to the normality test, our data is not normally distributed despite the histogram looking like it may be. So why does the data fail the normality test? The answer likely comes down to fat tails. Fat tails essentially mean that extreme events (+/-3 standard deviations away from the mean) are more likely than the normal distribution would imply.

To determine how many standard deviations away from the mean a specific number is, we need to use

$$\frac{X - \bar{X}}{\text{Sample standard deviation}}$$

Let's do this for the min and max of the sample data:

19]:
```
# minimum and maximum daily log returns
dfMax = df.SP500.max()
dfMin = df.SP500.min()

# Print maximum and minimum daily log returns
print("Maximum return of sample data is: ", round(dfMax, 5))
print("Minimum return of sample data is: ", round(dfMin, 5))
print(' - - - - - - - - - -')

# Calculates the number of standard deviations from the mean return
num_dev_max = (df.SP500.max() - df.SP500.mean()) / df.SP500.std()
num_dev_min = (df.SP500.min() - df.SP500.mean()) / df.SP500.std()

# Print num_dev_max and num_dev_min
print("Number of standard deviations from the mean for the maximum return: ", round(num_dev_max, 5))
print("Number of standard deviations from the mean for the minimum return: ", round(num_dev_min, 5))
```

```
Maximum return of sample data is:  0.10957
Minimum return of sample data is:  -0.12765
 - - - - - - - - - -
```


```
Maximum return of sample data is:  0.10957
Minimum return of sample data is:  -0.12765
 - - - - - - - - - -
```

```
Number of standard deviations from the mean for the maximum return:  9.0497
Number of standard deviations from the mean for the minimum return:  -10.60025
```

This output provides valuable insights into the distribution of S&P 500 daily log returns, particularly highlighting the presence of outliers or fat tails:

- Maximum and Minimum Returns: The maximum daily return of approximately 10.957% and the minimum daily return of approximately -12.765% show the range of returns observed in our data.
- Z-scores: The z-scores for the maximum and minimum returns are 9.05 and -10.6, respectively. These are extremely high z-scores, indicating that both the maximum and minimum returns are significant outliers, far away from the mean in terms of standard deviations.

In a normal distribution, you would rarely observe data points more than 3 standard deviations away from the mean. These high z-scores suggest that the S&P 500 daily returns have fatter tails than a normal distribution, meaning extreme events (large positive or negative returns) are more likely than would be expected under normality.

This information is crucial for risk management and modeling, as relying on the assumption of normality can lead to underestimating the probability of extreme events and potential losses.

These standard deviations are humongous when compared to the normal distribution. We can see this analytically when we plug in the z score to the `norm.cdf()` method to determine the probability this value could be in a normal distribution:

20]:
```
stats.norm.cdf(-10.60)
```

20]:
```
1.4899011272964664e-26
```

This implies that the chance we could have a move as small as -12.77% is 1.49e-26. This probability is so low that we would never expect an event like this to happen in our lifetime. We have multiple events like this, as illustrated by the minimum and maximum.

Going further with this idea, based on normal distribution z tables, we would expect 99.7% of our data points to be within +/- 3 standard deviations from the mean. Let's determine this for our sample data. First off, we need to find the cut-off values at +/- 3 standard deviations:

Going further with this idea, based on normal distribution z tables, we would expect 99.7% of our data points to be within +/- 3 standard deviations from the mean. Let's determine this for our sample data. First off, we need to find the cut-off values at +/- 3 standard deviations:

]:
```
# Calculates the upper and lowers bounds
upper = (3 * df.SP500.std()) + df.SP500.mean()
lower = (-3 * df.SP500.std()) + df.SP500.mean()

# Print the results
print("Upper bound: ", round(upper, 5))
print("Lower bound: ", round(lower, 5))
```

```
Upper bound:  0.03654
Lower bound:  -0.0359
```

The above two calculations would imply that 99.7% of all of our data points should be in between -0.0359 and 0.03654.

Since we have 5,031 data points, we would expect about 15 (i.e., 3% of 5,031) of them to be outside of that range if normality was held. Now let's see how many we actually have. Let's filter the `df` DataFrame to select rows where the S&P 500 daily log returns are outside the range of upper bound and lower bound. Then, we'll count the number of the data points that fall in this range:

]:
```
# Calculates the number of data points
len(df[(df["SP500"] < lower) | (df["SP500"] > upper)])
```

]: 84

That's a significant number of outliers considering that in a normal distribution, you would expect only about 0.3% of the data points to fall outside the range defined by 3 standard deviations from the mean.

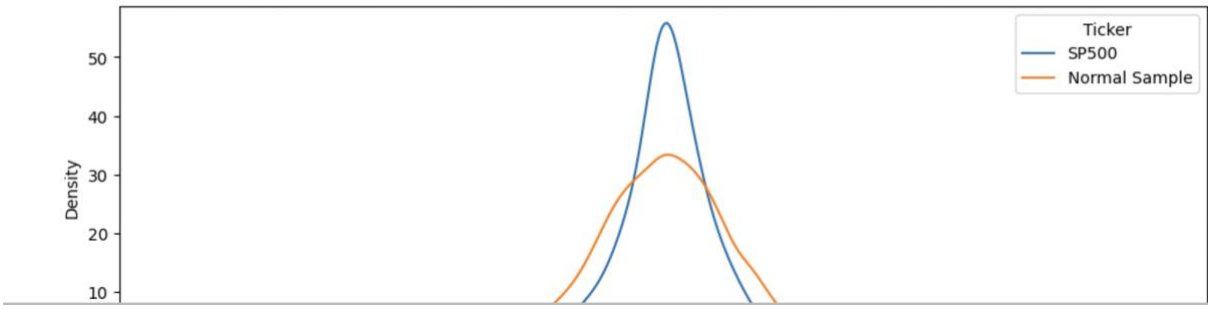
This finding further supports the conclusion from the normality tests that the S&P 500 daily log returns are not normally distributed and exhibit fat tails. The presence of these outliers highlights the importance of considering alternative distributions and risk measures when modeling and analyzing financial data.

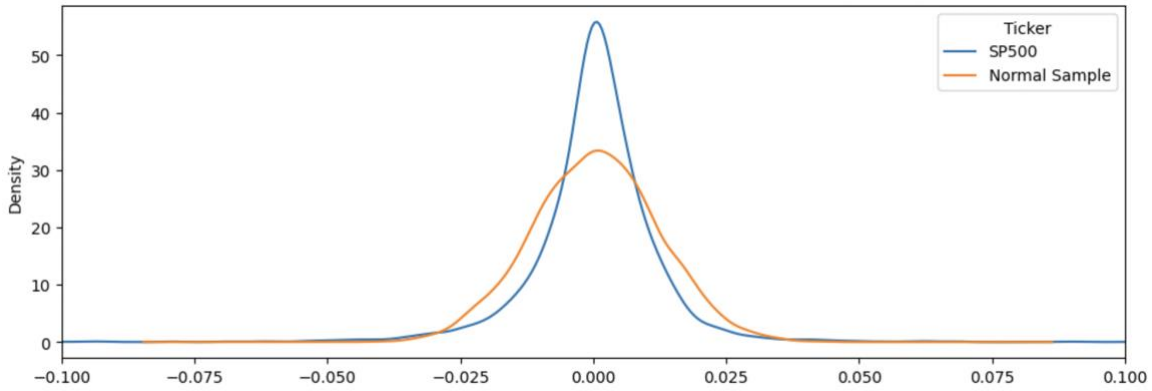
7. Non-Gaussian Distributions

One potential alternative distribution we could use to forecast stock returns is the Student's t-distribution. This is very similar to a normal distributior except it has heavier tails. Theoretically, this sounds perfect for daily returns based on what we've seen up to this point.

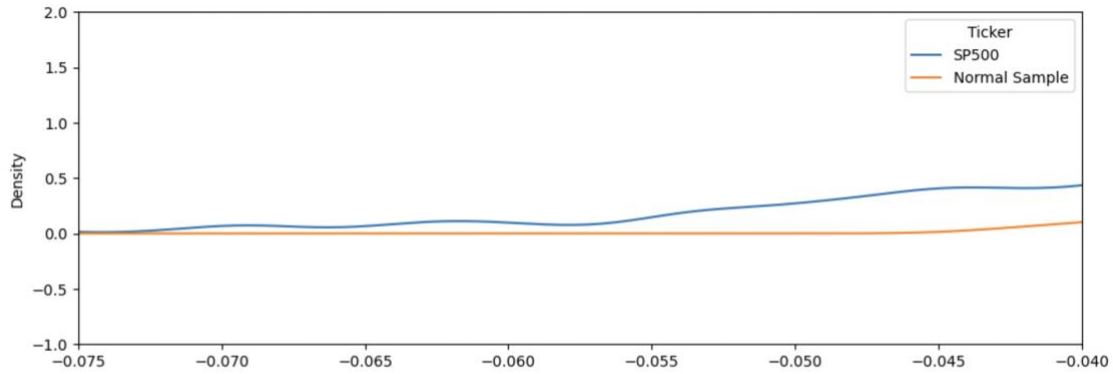
Let's proceed with a visual inspection of the distribution of our data by superimposing the normal distribution on the kernel density estimation (KDE) of S&P 500 returns:

[25]:
```
# Sampling from normal distribution
np.random.seed(222)
normal_dist = stats.norm.rvs(size=len(df["SP500"]), loc = df["SP500"].mean(), scale = df["SP500"].std())

# Creating an additional column in df in order to use the KDE plot functionality of pandas
df['Normal Sample'] = normal_dist

# Plotting the KDE plots
df[['SP500', 'Normal Sample']].plot(kind = 'kde', xlim = (-0.1, 0.1), figsize = (12,4));
```





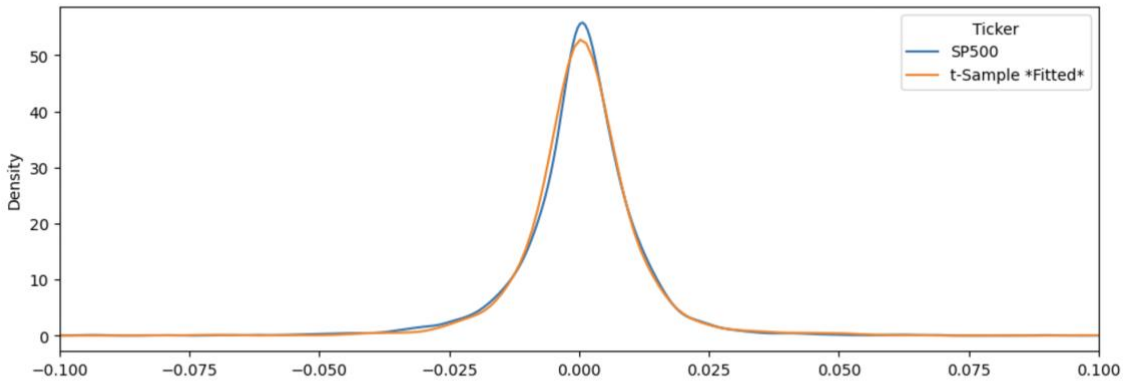
[28]:
```
# Observing the tails
df[['SP500', 'Normal Sample']].plot(kind = 'kde', xlim = (-0.075, -0.04), ylim = (-1, 2), figsize = (12,4));
```



The resulting plot visually shows the differences in the tails of the two distributions. We observe that the actual S&P 500 returns have a fatter left tail than the normal distribution, indicating a higher probability of large negative returns. This visualization provides further evidence of the non-normality of the S&P 500 returns and the presence of fat tails.

At this stage, we will calibrate the parameters of the Student's t-distribution using Maximum Likelihood Estimation (MLE) to align the distribution closely with our observed data. The following code fits a Student's t-distribution to the S&P 500 daily log returns using Maximum Likelihood Estimation (MLE) and then creates a KDE plot to compare the fitted t-distribution with the actual data.

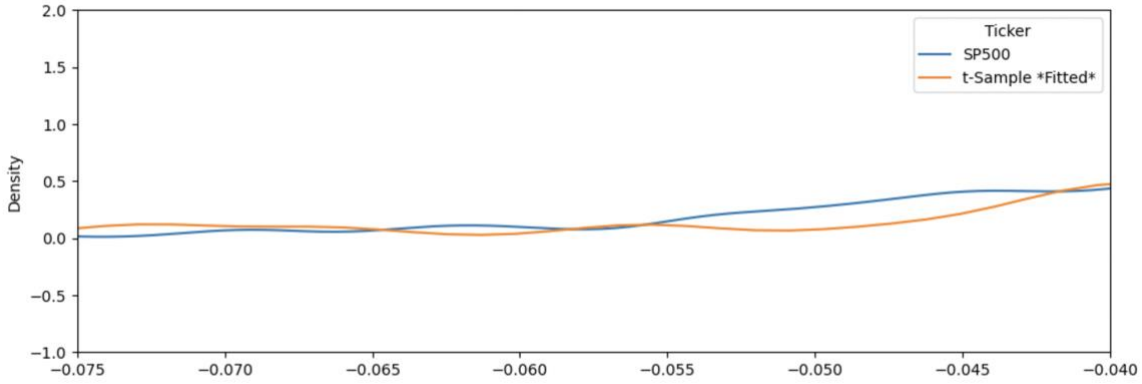
[31]:
```
# Fit the t-distribution using MLE
params = stats.t.fit(df.SP500)

# We plot the fitted distribution against the kde of the data
df['t-Sample *Fitted*'] = stats.t.rvs(*params, size = len(df))
df[['SP500', 't-Sample *Fitted*']].plot(kind = 'kde', figsize = (12,4), xlim = (-0.1, 0.1));
```



From the visual inspection, it's evident that the synthetic data generated from the fitted Student's t-distribution offers a more accurate approximation of our actual data compared to what we get from a normal distribution.

Let's check the tails once more. We will again zoom in to observe proximity between the left tail of the actual S&P 500 daily log returns with the left tail of the fitted Student's t-distribution:

2]:
```
# Plot left tail area
df[['SP500', 't-Sample *Fitted*']].plot(kind = 'kde', figsize = (12,4), xlim = (-0.075, -0.04), ylim = (-1, 2));
```



This time, the tails seem to be better explained by the t-distribution. Here, we can observe that the fitted t-distribution captures the fat tail of the S&P 500 returns more accurately than the normal distribution. This further demonstrates the suitability of the t-distribution for modeling financial data that exhibits fat tails and the potential for extreme events.

L3: Special Matrices for Equity Analysis

A **symmetric matrix** is a square matrix whose transpose is the same as the matrix itself. A square matrix is a matrix that has the same number of rows as the number of columns. The following example demonstrates the concept of a symmetric matrix.

$$A = \begin{bmatrix} m & a & b \\ a & n & c \\ b & c & p \end{bmatrix}$$

$$A^T = \begin{bmatrix} m & a & b \\ a & n & c \\ b & c & p \end{bmatrix}$$

In the above example, the second matrix is the transpose of A matrix. The elements of the original matrix and its transpose are the same. In mathematical notation, for matrix A, if $A = A^T$, then A is symmetric. To be more specific, the element at position (i,j) is equal to the element at position (j,i) for all i and j.
Here are some key properties of a symmetric matrix:

1. The eigenvalues of the matrix are real numbers.
2. The eigenvectors of the matrix are orthogonal to each other.
3. The matrix is diagonalizable.
4. The rank of the matrix is equal to the number of non-zero eigenvalues.

When two vectors are **orthogonal**, it means that the vectors are perpendicular to each other. It also means the inner product of the two vectors is 0. These two vectors are linearly independent. Next, we will talk about the diagonalizability of a symmetric matrix.

1.2. Diagonalization of Symmetric Matrices

Matrix A is diagonalizable if there exists a diagonal matrix Λ such that

$$A = B\Lambda B^{-1}$$

If A is a symmetric matrix, the values on the main diagonal of Λ are eigenvalues of A. B is a matrix whose columns are eigenvectors of A.
Let's look at one example of diagonalization of a symmetric matrix.
We have a symmetric matrix M as follows:

$$M = \begin{bmatrix} 6 & -2 & -1 \\ -2 & 6 & -1 \\ -1 & -1 & 5 \end{bmatrix}$$

In order to obtain eigenvalues, we need to set the characteristic polynomial to 0, or solve for the characteristic equation as follows:

$$det(M - \lambda I) = det(\begin{bmatrix} 6-\lambda & -2 & -1 \\ -2 & 6-\lambda & -1 \\ -1 & -1 & 5-\lambda \end{bmatrix})$$

$$\Downarrow$$

$$(6-\lambda)(6-\lambda)(5-\lambda) + (-2)(-1)(-1) + (-2)(-1)(-1) - (-1)(-1)(6-\lambda) - (-2)(-2)(5-\lambda) - (-1)(-1)(6-\lambda) = 0$$

$$\Downarrow$$

Now we can write down the diagonalized symmetric matrix M.

$$M = B\Lambda B^{-1}$$

where

$$\Lambda = \begin{bmatrix} 8 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

$$B = \begin{bmatrix} -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{3}} \\ 0 & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{3}} \end{bmatrix}$$

Because B is an orthogonal matrix, $B^T = B^{-1}$. We can also write the diagonalized symmetric matrix as follows:

$$M = B\Lambda B^T$$

Transforming a symmetric matrix to the diagonalized form will make the computation of matrices more efficient. It also helps decompose a matrix into simpler forms for analysis. Diagonalization of a matrix is also an essential tool to help solve the linear transformation issue.
Now we have learned many properties of symmetric matrices. Let's take a look at some common symmetric matrices, like the covariance matrix and the correlation matrix. We'll look at the covariance matrix first.

2. Triangular Matrix

Another commonly used matrix is the triangular matrix. A triangular matrix is a square matrix where all the entries on one side of the main diagonal are zero. There are two types of triangular matrices: lower triangular matrix and upper triangular matrix.

2.1. Lower Triangular Matrix

A **lower triangular matrix** is a square matrix whose entries above the main diagonal are zero. The main diagonal and entries below it can be any value. Below is a demonstration of a lower triangular matrix.

$$B = \begin{bmatrix} m & 0 & 0 \\ a & n & 0 \\ b & c & p \end{bmatrix}$$

2.2. Upper Triangular Matrix

An **upper triangular matrix** is a square matrix whose entries below the main diagonal are zero. The main diagonal and entries above it can be any value. Below is a demonstration of an upper triangular matrix.

$$B = \begin{bmatrix} m & a & b \\ 0 & n & c \\ 0 & 0 & p \end{bmatrix}$$

2.3. Properties of a Triangular Matrix

Here are some key properties of a triangular matrix:

1. The determinant is the product of the diagonal elements.
2. The inverse of a triangular matrix (if it exists) is also triangular.
3. The product of two upper (or lower) triangular matrices is upper (or lower) triangular.

The above properties are all very easy to prove using matrix operations, so we will leave these proofs to the readers.

3. Symmetric Positive Definite (PD) Matrices and Symmetric Positive Semi-Definite (SPD) Matrices

In this section, we are going to introduce two important types of matrices: symmetric positive definite (PD) matrices and symmetric positive semi-definite matrices (SPD).

3.1. Symmetric Positive Definite (PD) Matrices

A symmetric matrix A is positive definite if $x^T Ax > 0$ for all non-zero vectors x.
Here are some key properties of a symmetric positive definite matrix:

1. All eigenvalues of the matrix are positive
2. The determinant of the matrix is positive
3. The matrix has full rank
4. The matrix is invertible
5. The summation and multiplication of two symmetric positive definite matrices is also a symmetric positive definite matrix

3.1. Symmetric Positive Definite (PD) Matrices

A symmetric matrix A is positive definite if $x^T Ax > 0$ for all non-zero vectors x.
Here are some key properties of a symmetric positive definite matrix:

1. All eigenvalues of the matrix are positive
2. The determinant of the matrix is positive
3. The matrix has full rank
4. The matrix is invertible
5. The summation and multiplication of two symmetric positive definite matrices is also a symmetric positive definite matrix

These properties of symmetric positive definite matrices—full rank, positive determinant, and positive eigenvalues—are intimately connected. These characteristics make symmetric positive definite matrices particularly useful in various mathematical and applied contexts, providing guarantees of non-singularity, stability, and convergence in many algorithms and methods.
Usually, a quick way to identify if a symmetric matrix is positive definite is to calculate its determinant and check if the determinant is positive. For example, let's take a look at the following symmetric matrix.

$$\begin{bmatrix} 3 & 4 \\ 4 & 5 \end{bmatrix}$$

This is a symmetric matrix. However, the determinant is $(3 \times 5) - (4 \times 4) = -1$, this matrix is not positive definite.

3.2. Symmetric Positive Semidefinite (SPD) Matrices

A symmetric matrix A is positive semidefinite if $x^T Ax \geq 0$ for all vectors is x.
Here are the key properties of a symmetric positive semidefinite matrix:

1. All eigenvalues of the matrix are non-negative
2. The determinant of the matrix is non-negative
3. The matrix may not have full rank
4. The matrix is not always invertible

We usually also use the determinant of a symmetric matrix to quickly check if it is also a positive semidefinite. For the following matrix example,

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

we can check that the determinant is 0. Hence, this matrix is a positive semidefinite matrix.
Going back to covariance matrices and correlation matrices, they are both symmetric positive semidefinite but not necessarily symmetric positive definite. The following Python code can be used to check if a matrix is a symmetric positive definite or a symmetric positive semidefinite.

```python
def is_positive_definite(matrix):
    return np.all(np.linalg.eigvals(matrix) > 0)

A = stock_returns_correlation_matrix

print("Is A positive definite?", is_positive_definite(A))
print("Eigenvalues:", np.linalg.eigvals(A))
print("Determinant:", np.linalg.det(A))
print("Rank:", np.linalg.matrix_rank(A))
```

covariance matrices and correlation matrices, they are both symmetric positive semidefinite but not necessarily symmetric positive definite.

4. Cholesky Factorization

4.1. Definition of Cholesky Factorization

Cholesky factorization is a method to decompose a symmetric matrix. Cholesky factorization states that a symmetric positive definite matrix S can be decomposed as follows:

$$S = LL^T$$

where L is a lower triangular matrix or a Cholesky factor.

Let's look at a numeric example. Suppose we have the following 3×3 symmetric positive definite matrix.

$$S = \begin{bmatrix} s_{11} & s_{21} & s_{31} \\ s_{21} & s_{22} & s_{32} \\ s_{31} & s_{23} & s_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ 1_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix} = LL^T$$

$$\Downarrow$$

$$LL^T = \begin{bmatrix} l_{11}^2 & l_{11}l_{21} & l_{11}l_{13} \\ l_{21}l_{11} & l_{21}^2 + l_{22}^2 & l_{21}l_{31} + l_{22}l_{32} \\ l_{31}l_{11} & l_{21}l_{31} + l_{22}l_{32} & l_{31}^2 + l_{32}^2 + l_{33}^2 \end{bmatrix}$$

S is symmetric positive definite

From the above matrices, we can get the following formulae for the lower triangular matrix.

$$l_{11} = \sqrt{s_{11}}$$
$$l_{21} = \frac{s_{21}}{l_{11}}$$
$$l_{31} = \frac{s_{31}}{l_{11}}$$
$$l_{22} = \sqrt{s_{22} - l_{21}^2}$$
$$l_{32} = \frac{s_{23} - l_{31} l_{31}}{l_{22}}$$
$$l_{33} = \sqrt{s_{33} - l_{31}^2 - l_{32}^2}$$

To get the above solution, you must start to solve the first row of LL^T. Then, you move on to the second row and the next row until you solve all the elements in L.
Let's look at a numeric example. Suppose we have the following symmetric positive definite matrix (please verify),

$$S = \begin{bmatrix} 25 & 15 & -5 \\ 15 & 18 & 0 \\ -5 & 0 & 11 \end{bmatrix}$$

Let's find the Cholesky factor using the above equations we derived.

$$l_{11} = \sqrt{25} = 5$$
$$l_{21} = \frac{15}{5} = 3$$
$$l_{31} = \frac{-5}{5} = -1$$
$$l_{22} = \sqrt{18 - 9} = 3$$
$$l_{32} = \frac{0 - (3)(-1)}{3} = 1$$
$$l_{33} = \sqrt{11 - (-1)^2 - (1)^2} = 3$$

Hence, the Cholesky factor of S is

$$L_s = \begin{bmatrix} 5 & 0 & 0 \\ 3 & 3 & 0 \\ -1 & 1 & 3 \end{bmatrix}$$

Python Numpy has a function to calculate the Choleksy factor for a matrix. We can use it to verify our manual calculation.

6]:
```
S= np.array([[25,15,-5],
             [15,18,0],
             [-5,0,11]])

L_numpy = np.linalg.cholesky(S)
print("\nNumPy Cholesky factor:")
print(L_numpy)
```

```
NumPy Cholesky factor:
[[ 5.  0.  0.]
 [ 3.  3.  0.]
 [-1.  1.  3.]]
```

Choleksy factorization algorithum greatly improves the computer efficiency to solve for linear equations. It can also be used to compute the inverse of a matrix more efficiently than some other methods. Choleksy factorization is also numerically stable, making it useful for computations involving ill-conditioned matrices.

4.2. Using Cholesky Factorization for Monte Carlo Simulation

Cholesky factorization is useful in Monte Carlo simulations or other optimization tasks. In this section, we will use Cholesky factorization in a Monte Carlo simulation to generate correlated random variables. Let's use the Apple and Ford Motor stock return correlation matrix as an example. The process involves the following steps:

1. Compute the Cholesky factorization of the stock return correlation matrix to get the factor L.
2. Generate independent standard normal random variable Z.
3. Compute X = LZ to get correlated random variables.

First, let's calculate the correlation of Apple and Ford Motor stock returns.

[ ]:
```
# Create a new dataframe with just Apple and Ford Motor stock returns from the dataframe from the last example
two_stock_returns = stocks_returns[["AAPL", "F"]]
original_correlation = two_stock_returns.corr()
original_correlation
```

Now, let's implement the simulation using Cholesky factorization as described above.

[ ]:
```
# Create a function to generate random samples from correlated variables
def generate_correlated_samples(n_samples, correlation_matrix):
    # Compute Cholesky factorization
    L = np.linalg.cholesky(correlation_matrix)

    # Generate independent standard normal samples
    Z = np.random.standard_normal((correlation_matrix.shape[0], n_samples))

    # Generate correlated samples
    X = L @ Z

    return X
```

```
# Generate samples
n_samples = 10000
X = generate_correlated_samples(n_samples, original_correlation)
X = pd.DataFrame(generate_correlated_samples(n_samples, original_correlation).T, columns = ["APPL","F"])
# Compute sample correlation
sample_correlation = X.corr()
sample_correlation
```

43]:

	APPL	F
APPL	1.000000	0.365844
F	0.365844	1.000000

Our next step is to visualize the result from original data and sampled data using a Monte Carlo simulation.

44]:

```
# Visualize results
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

sns.heatmap(original_correlation, annot=True, cmap='coolwarm', vmin=-1, vmax=1, ax=ax1)
ax1.set_title('Original Correlation Matrix')
ax1.set(xlabel='', ylabel='')


sns.heatmap(sample_correlation, annot=True, cmap='coolwarm', vmin=-1, vmax=1, ax=ax2)
ax2.set_title('Sample Correlation Matrix')
ax2.set(xlabel='', ylabel='')

plt.tight_layout()
plt.show()

# Scatter plot of the stimulated Apple and Ford Motor stock returns
plt.figure(figsize=(8, 6))
plt.scatter(X["APPL"], X["F"], alpha=0.1)
plt.title('Scatter Plot of the Simulated Apple and Ford Motor stock returns')
```

L4: Ethical practices with data

1. Intentionally Creating Misleading Data

One day, an artist named Simon Weckert collected 99 smart phones (Barret). He put them into a wagon that he then proceeded to walk along the streets of Munich, ironically the street near Google headquarters in Munich—-he was hacking the Google Maps app! About an hour into his walk, Google Maps showed long, snarling traffic—despite there not being any cars in the area. The artist claims he was pointing out the simplicity behind these systems. There was really no harm done in this clever prank. This makes us question our assumptions when we receive data, like traffic conditions.

If this exercise were done in financial markets, it would be an entirely different story. It is unethical to intentionally create misleading data. Just as we may automatically follow the conditions on a traffic app, a RoboAdvisor may automatically follow the conditions in the marketplace (D'Acunto and Rossi). They may react without questioning data because they focus on speed and efficiency rather than authenticating the results.

One egregious way to intentionally mislead a market is to spoof it. One such person who did that in 2010 was Navinder Singh Sarao. He was held responsible for initiating a market downturn known as the Flash Crash of 2010. Sarao pled guilty to spoofing the market and paid with a prison sentence and fines.

What is spoofing? Spoofing is an extreme type of bluff. Spoofing occurs when a trader places a large order that he has no intention of getting filled. Instead, the intention is to manipulate the market in one's favor. For example, a spoofer placing a large buy order would place a huge bid order well below the current market bid price. When market makers and other traders, including algorithmic traders, sense a motivated buyer, they pull their ask prices and raise them. These price increases trigger other traders to do the same. Yet, this was the spoofer's motive all along—a manipulation of price in the desired direction.

In Mr. Sarao's case, he spoofed the market by placing a large sell order (above the ask price) for the E-mini S&P futures contract. His order caused the algorithms to lower their prices, which in turn caused other traders and algorithms to lower their prices, and so on. Prices plummeted. The actions and reactions caused the Flash Crash, an intraday market crash that wiped out more than $1 trillion in market value. Other markets did not exhibit the drop, and so the market learned that this was a one-off accident. Within the hour, most of the market corrected.

It is unethical to crate misleading data. Markets trade in real time and can be misled by spoofers. Fortunately, regulation and greater controls exist at many exchanges today that minimize the reoccurrence of a trader spoofing the market and causing another flash crash. It is still a criminal offense to spoof the market.

2. Propagating Fake Reviews

Next time you're in London, you may want to dine at what was once the top-rated restaurant on tripadvisor.com: The Shed (Butler). With pictures of dishes, an outside garden, and dozens of reviews, The Shed seemed to be the trendy place to eat. The only problem is that the restaurant didn't exist. The initial reviews were all fake. Ironically, The Shed was the go-to place. Phone calls into the restaurant only made it to the answering machine. When people realized how trendy the restaurant was, they wrote reviews even though they never even went there. Part of the momentum of going viral was hiring celebrities that posed with dishes (that weren't even from the restaurant) to lend an air of credibility and panache.

Reading reviews is not the same as performing due diligence. As more details of the FTX scandal become public, some celebrities are coming forward to admit that they were investors. Many people are familiar with the movie star Kevin Bacon. He was an investor in FTX and claims he lost millions of dollars. Even celebrities can become victims of the halo effect or appeal to authority. Marketers know the power of celebrity endorsements, to the point where many athletes and musicians earn millions more from product endorsements than they earn in wages.

Do you wonder how many restaurant reviews are fake? There are parties with vested interests who are tempted to post glowing review after glowing review. A quick click into that reviewer shows they reviewed nothing else. The same can be said for forums that review stocks. Some people join these popular forums to tout stocks (or to bash them if they have short positions). Good ethics avoids misrepresentation.

3. Failing to Disclose Fees or Conflicts of Interest

According to a data survey by the data research company Morning Consult, only 36% of adults trusted investment and wealth managers. The rest either did not trust them or had no opinion about them. Part of the problem is that financial advisors fail to disclose fees or reveal conflicts of interest. For example, an advisor may recommend a mutual fund A over B for a client. Although B may be more suitable for the investment profile and risk tolerance, the advisor is paid a much better commission for selling mutual fund A. Failing to disclose fees or reveal conflicts of interest violates a fiduciary trust.

In the investment world, these are criminal activities. For example, two robo-advisors, one which had more than $11 billion in client money, made false statements about a strategy involving tax-loss harvesting (U.S. Securities and Exchange Commission, "SEC Charges Two Robo-Advisers"). These false disclosures misled investors into thinking the funds had superior performance (higher returns, less risk, smaller fees) and naturally makes them appear more attractive than other funds. Other times, managers fail to disclose fees (U.S. Securities and Exchange Commission, "SEC Charges Investment Adviser").

Good ethics show all data associated with investment: historical results, fees, and risks. Regulation helps. Some industries protect the consumer/investor better than others.

4. Using Simulated Data Instead of Real Data without Explicit Disclosure

Another way to lie with data is to use simulated data while trying to pass it off as real data. For example, a hedge fund raised millions of dollars by falsely claiming that it had a lengthy track record based on actual trades using real money (U.S. Securities and Exchange Commission, "SEC Charges Father-and-Son Hedge Fund Managers"). In reality, the results were from back-tested hypothetical simulations. Simulations have details that can skew a study one way or another. For example, a simulation of a back-test may ignore a bid-ask spread and market impact. Misrepresenting the type of data is illegal, and the managers agreed to pay millions to settle the fraud case. Using simulated returns in and of itself is not illegal or unethical. When using simulated data, you should clearly indicate that the results are not real, but merely a product of back-tests or scenarios. Even when a fund has a track record going back years, a disclaimer is added that past performance is no guarantee of future results. Misleading is lying.