2 - Utility and investor types

Espen Sirnes

2024-09-18

Table of contents

1 Expected Utility

In finance, the utility function is typically defined as the utility derived from a certain amount x. It is generally assumed that greater wealth yields higher utility, hence the utility function should be increasing, meaning $U'(x) > 0$. In situations involving risk, such as holding shares with uncertain future values, it becomes useful calculate the expected utility of a set of possible outcomes x_i with probabilities $P(x_i)$ and utilities $u(x_i)$

This builds on the general notion of expectations in statistics. For a set of outcomes x_i and probabilities $P(x_i)$ for example, the expected value is $X = \{x_0, x_1, \ldots, x_n\}$ is

$$\mathbb{E}[X] = \sum_{i=0}^{N} P(x_i)x_i \tag{1}$$

For example

```
import numpy as np

p = np.array([0.5, 0.2, 0.3])
x = np.array([3,10, 20])

np.sum(p*x)
```

```
9.5
```

In the same way, the descrete expected utility for a set of outcomes x_i with probabilities $P(x_i)$ is defined as follows:

$$U(X) = \mathbb{E}[u(X)] = \sum_{i=0}^{N} P(x_i)u(x_i) \tag{2}$$

where $P(x_i)$ is the probability that X assumes the value x_i, and $u(x_i)$ represents the utility derived if that event occurs.

For example, say utilities 2,-4 and 6 occur with probabilities 0.5,0.2 and 0.3, then the expected utility is

```
import numpy as np

p = np.array([0.5, 0.2, 0.3])
u = np.array([2, -4, 6])

U = np.sum(p*u)
U
```

```
1.9999999999999998
```

For a continuous distribution, the idea is the same. We multipliy each utility state $u(x_i)$ with the probability it occurs $f(x)dx$. The integral sign $\int_{-\infty}^{\infty} \dots$, is the equvalent of a sum for continous variables. The expected utility is therefore given by:

$$U(X) = \mathbb{E}[u(X)] = \int_{-\infty}^{\infty} u(x)f(x)\ dx,$$

where $f(x)$ is the density function of X, such as the normal distribution.

1.1 Expectation of the normal distribution

To make this a bit more tangible, let us assume f(x) represents the standard normal distribution (variance=1 and mean=0), and assume we collect the outcomes x into five intervals. According to the normal distribution, each outcome will then have these probabilities

Figure 1:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# User defined dx
dx = 2

# Create the figure and axis
fig, ax = plt.subplots()

# Define the x range from -2.5 to 2.5
x = np.arange(-4, 4 + dx, dx)

# Calculate the corresponding normal distribution
# values
p = norm.pdf(x, 0, 1)

# Plot the normal distribution curve in blue
x_fine = np.linspace(-3.5, 3.5, 1000)
p_fine = norm.pdf(x_fine, 0, 1)
ax.plot(x_fine, p_fine, label="Normal Distribution",
```

```
    color='blue')

# Plot the bars with blue color and dark blue border
bars = ax.bar(x, p, width=dx, align='center',
    alpha=0.6, color='blue', edgecolor='darkblue',
    label="Bars (f(x) with width dx)")

# Label each bar with its "mass" (height * dx)
for bar, height in zip(bars, p):
    mass = height * dx
    ax.text(bar.get_x() + bar.get_width() / 2,
        height, f'{mass:.3f}', ha='center',
        va='bottom')

# Add labels and title
ax.set_xlabel('x')
ax.set_title('Normal Distribution and '
    'corresponding discrete probabilities '
    ' (dx = {})'.format(dx))



# Show the plot
plt.show()
```

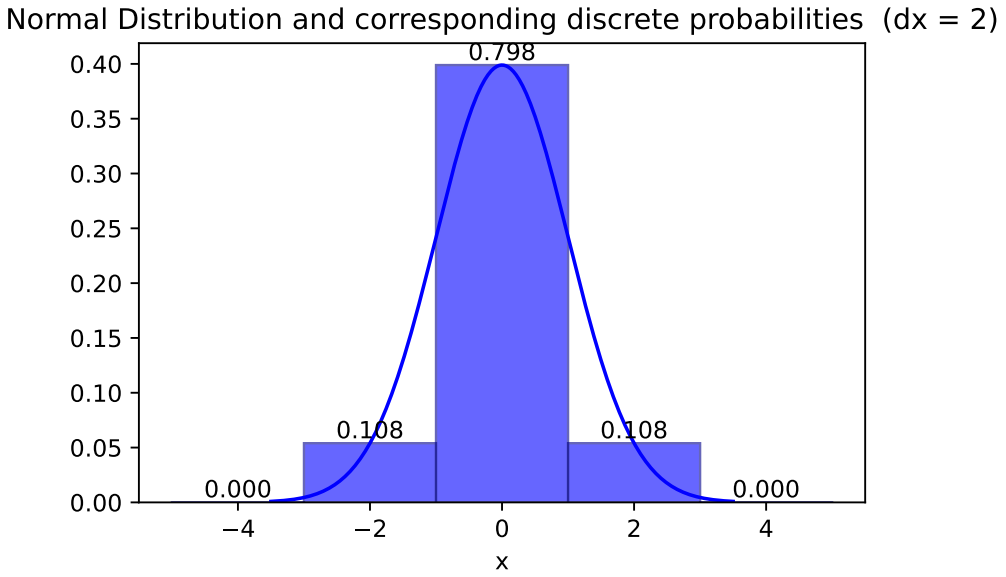


Figure 1: Descrete version of the normal distribution

The probabilities do not sum exactly to one, as expected for a probability distribution. This discrepancy arises due to the coarse nature of the discrete categories.

Coding Challenges:

- Challenge 1: Reduce the interval size dx and sum the probabilities to observe if the total approaches 1.
- Challenge 2: Now, assume the utility function is $f(x) = -\exp(-\rho x)$ (the natural logarithm). What is the expected utility for different values of dx?

As demonstrated above, decreasing the interval size dx brings the discrete distribution closer to the continuous one. The continuous distribution serves as an approximation when outcomes are real numbers, which is often the case in investment scenarios.

The primary advantage of continuous variables over discrete ones is that we can apply powerful mathematical tools, such as derivatives and integrals, from calculus.

1.2 A descrete example

A key outcome of this model is that with a concave utility function, an individual will always prefer a guaranteed cash flow over a speculative investment with the same expected return. This behavior is illustrated by the following example.

Consider an individual with a logarithmic utility function:

$$u(x) = -\exp(-\rho x),$$

which is increasing and concave, as shown by its first and second derivatives:

$$u'(x) = \rho u(x) > 0 u''(x) = -\rho^2 u(x) > 0 < 0.$$

Suppose there is a 50% chance of $x = 0.5$ and a 50% chance of $x = 1.5$. This setup implies that $X = \{0.5, 1.5\}$ with probability distribution $P(X) = \{0.5, 0.5\}$, leading to an expected value of $\mathbb{E}X = 1$. The expected utility is thus calculated as:

```python
import numpy as np

# Values and probabilities
x_values = np.array([-0.5, 0.5])
probabilities = np.array([0.5, 0.5])

#utility function
def u_func(x,rho):
    return -np.exp(-rho*x)


# Expected utility
expected_utility = np.sum(probabilities * u_func(x_values, 0.5))
expected_value = np.sum(probabilities * x_values)
utility_of_expected_value = u_func(expected_value, 0.5)

print(f"Expected utility: {expected_utility}")
print(f"Utility of expected value: {utility_of_expected_value}")
```

```
Expected utility: -1.0314130998795732
```

```
Utility of expected value: -1.0
```

By comparing the utility of the expected return of 1 with the expected utility, we observe:

$$u(\mathbb{E}X) = \ln(1) > \ln(\sqrt{0.75}) = \mathbb{E}u(X),$$

indicating that the utility of a certain outcome ($\ln(1)$) is preferred over the expected utility of a gamble ($\ln(\sqrt{0.75})$). This preference underscores risk aversion, as illustrated in the following plot:

```python
import numpy as np
import matplotlib.pyplot as plt
RHO = 1.5

# Exponential utility function
def u_func(x):
    return -np.exp(-RHO*x)

def x_func(u): # the inverse of the utility function
    return -np.log(-u)/RHO

# Values for wealth and utility
x_vals = np.linspace(-1.1, 1.1, 100)
u_x = u_func(x_vals)

# Gamble outcomes
x_gamble = [-1, 1]  # Outcomes of the gamble
p_gamble = [0.5, 0.5]  # Probabilities

# Certain outcome
x_certain = 1

# Expected utility of the gamble
expected_utility = np.sum(np.array(p_gamble) *
                          u_func(np.array(x_gamble)))

# Plotting the utility function using fig, ax objects
fig, ax = plt.subplots(figsize=(10, 6))
```

```
# Plot the utility function
ax.plot(x_vals, u_x, label=r'Utility Function: $u(x) = - \exp(-\rho x
        color='black')

# Plot the certain outcome
ax.axvline(x=x_certain, color='black', linestyle='--')
ax.text(x_certain, u_func(x_certain) + 0.1, "$1$",
            horizontalalignment='center', fontsize=12)

# Plot the gamble outcomes
ax.axvline(x=x_gamble[0], color='black', linestyle='--')
ax.axvline(x=x_gamble[1], color='black', linestyle='--')
ax.text(x_gamble[0], u_func(x_gamble[0]) - 0.3, "$0.5$",
        horizontalalignment='center', fontsize=12)
ax.text(x_gamble[1], u_func(x_gamble[1]) - 0.3, "$1.5$",
        horizontalalignment='center', fontsize=12)

# Plot a line connecting the points where the utility function crosse
ax.plot([x_gamble[0], x_gamble[1]],
        [u_func(x_gamble[0]), u_func(x_gamble[1])],
        color='blue', linestyle='-', marker='o',
        label='Any expected utility must lie on this line')

# Plot the expected utility
ax.axhline(y=expected_utility, color='gray', linestyle='--', label='
ax.text(-2, expected_utility, '$EU(X)$', verticalalignment='top', fo

# Risk premium - distance between expected utility and utility of ce
risk_premium = u_func(x_certain) - expected_utility
certainty_equivalence = x_func(expected_utility)
ax.annotate('', xy=(0, expected_utility),
            xytext=(certainty_equivalence, expected_utility),
            arrowprops=dict(facecolor='black', arrowstyle='<->'))
# Separate annotation for the label (π) without the arrow
ax.annotate(r'$\rho$', xy=(-0.3, expected_utility - 0.2), fontsize=1

# Labels and title
ax.set_title('Utility Function Demonstrating Risk Aversion')
```

```
ax.set_xlabel('Wealth (W)')
ax.set_ylabel('Utility (U)')
ax.legend()
ax.grid(True)

# Display the plot
plt.show()
```

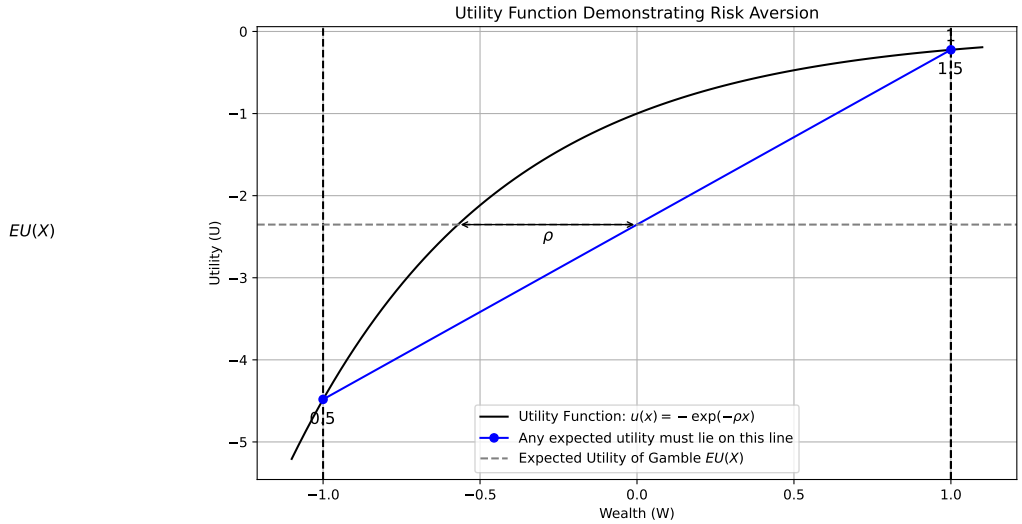


Figure 2: Graphical illustration of the risk premium

As we can see, the utility at wealth 1 is 0, while the mean utility is lower. If you draw a line between the utility of 0.5 and 1.5, you will see that the indicated expected utility, is where that line intersects 1.

Hence, with a concave uitlity function, a person will allways prefere a sure amount rather than a bet with the same expected payoff. In this case, the risk-averse investor will invariably prefer the certain payment of 1 over the gamble.

The difference indicated by ρ in the figure, is the certainty equivalence. It is the amount that you would need to compensate the investor, in order to take the gamble. In a real market, this is the premium that investors demand to hold risky assets to safe bills. It is usually about 2-4% on average for the whole market.

1.3 A continous example

Let us now calculate the expecte utility with more than two outcomes. We remember from the calculation of Figure 1 and calculation of descrete expectations in Equation 2 that a descrete version of the continous normal distribution can be calculated as

```
from scipy.stats import norm
dx = 2
x = np.arange(-4, 4 + dx, dx)
p = norm.pdf(x, 0, 1)
print(f'Outcomes:{x}')
print(f'Probabilities:{np.round(p*dx,3)}')
```

```
Outcomes:[-4 -2  0  2  4]
Probabilities:[0.    0.108 0.798 0.108 0.   ]
```

Expected utility for a utility function $u(x_i) = \ln(x_i)$ is then

```
expected_utility = np.sum(p*dx*u_func(x))

print(f'Approximate expected utility:{expected_utility}')
```

```
Approximate expected utility:-3.080118364013702
```

Now, the normal distribution is defined as

$$\frac{1}{\sigma\sqrt{2\rho}} e^{-\frac{(x-\mu)^2}{2\sigma}}$$

if we calculate the expected utility by taking the integral, it turns out that

$$\mathbb{E}[u(X)] = \int_{-\infty}^{\infty} u(x) f(x)\, dx = e^{-\rho(\mu - \frac{1}{2}\rho\sigma^2)} \tag{3}$$

Now, let us try to calculate this with our numbers. We have a standard standard normal distribution, so $\sigma = 1$ and $\mu = 0$. The risk aversion coefficient is $\rho = 1.5$. The exact and approximate expected utility is then

```
print(f'Exact expected utility:{-np.exp(0.5*RHO**2)}')
print(f'Approximate expected utility:{expected_utility}')
```

```
Exact expected utility:-3.080216848918031
Approximate expected utility:-3.080118364013702
```

The most important lesson here, is what is inside the brackets in Equation 3. This expression determines in effect the utility. The expression is

$$\mu - \frac{1}{2}\rho\sigma^2 \tag{4}$$

Hence, if returns are normally distributed, then investors would want to maximize the difference between the meand and the variance. This is true for any utility function, but it is not in general true if returns are distributed differently.

Coding Challenges:

- Challenge 1: Calculate the approximate expected utility with some other parameters and a non standard normal distribution ($\sigma <> 1$ and $\mu <> 1$, and see if the approxmate is still a good approximation.

2 Optimal Portfolio with One Asset

With the knowledge from Equation 4, we can actually calculate the optimal level of investment. We will later see that the method for doing that, is very similar to how we calcualte optmal portfolios with many different assets. For now, we will assume we only consider one asset, for example the market index. Say you buy shares for w NOK. Then, we can use the method in the previoius section to find that the expected utility depends on

$$w\mu - \frac{w^2}{2}\rho\sigma^2 \tag{5}$$

Taking the derivative of Equation 5, which we assume is a concave and increasing function, we get

$$\frac{d(w\mu - \frac{w^2}{2}\rho\sigma^2)}{dw} = \mu - w\rho\sigma^2 \tag{6}$$

Setting Equation 7 equal to zero and solving for w gives

$$w = \frac{\mu}{\rho\sigma^2} \tag{7}$$

From this, we conclude: 1. Higher risk aversion leads to lesser investment. 2. Greater expected returns encourage more investment. 3. Increased risk (σ^2) discourages investment.

In the next lecture, we will extend these principles to portfolios with multiple assets using matrix algebra.

2.1 Investor Types

The shape of a utility function critically influences how individuals respond to risk. This subsection discusses three primary types of investors based on their risk preferences and the corresponding shapes of their utility functions.

- Risk-Averse Investors: These individuals have concave utility functions, as in the example above, indicating a preference for certain outcomes over uncertain ones with the same expected value. Commonly modeled in financial theory, risk-averse investors prioritize minimizing risk over maximizing returns. They tend to diversify their portfolios across various asset classes to reduce volatility. Even with the option of unlimited borrowing, they typically opt to limit their investment exposure.
- Risk-Neutral Investors: For risk-neutral individuals, volatility is inconsequential. Their utility functions are linear, reflecting indifference to the level of risk associated with any investment. They focus solely on maximizing expected returns and are likely to invest in the asset with the highest expected payoff, irrespective of the associated risks. This type of investor is willing to allocate as much capital as possible to maximize potential gains.
- Risk-Loving (Risk-Seeking) Investors: Risk lovers have convex utility functions and engage in behaviors akin to gambling, where the expected return is typically negative. They derive satisfaction from the risk itself and often pursue investments that offer the highest possible returns, irrespective of the high levels of risk involved. Such behavior is commonly seen in speculative ventures and high-stakes gambling.

Interestingly, it is not uncommon for individuals to display traits of both risk-averse and risk-seeking behaviors, a phenomenon that may seem paradoxical. For instance, the same person might purchase insurance (a risk-averse action) while also indulging in lottery gambling (a risk-seeking behavior). This can be explained by the utility function's varying shape at different levels of wealth or stakes: a person might be risk-seeking with small, disposable amounts of money but risk-averse with larger, life-impacting sums. This dual nature influences how individuals choose to allocate their investments across different risk levels.

2.2 Assumptions in a Financial Market

In financial theory, the assumption typically made about market participants is that they are predominantly risk averse. This assumption is crucial as portfolio optimization and related strategies largely rely on this characteristic. Risk-neutral or risk-loving investors, who either disregard risk or actively seek it, are considered exceptions rather than the norm in these models. This foundational assumption allows for the development of investment strategies that aim to maximize returns while minimizing risk, aligning with the preferences of risk-averse individuals.

3 The Risk Premium

We understand that a risk-averse individual has a concave utility function, indicated by $u''(x) < 0$. But how do we quantify the degree of risk aversion?

A direct approach might be to use the second derivative of the utility function, $u''(x)$, as a measure of risk aversion. However, this method has limitations, particularly at higher wealth levels. The issue arises because the curvature of the utility function tends to flatten as wealth increases, implying a decrease in relative risk aversion. If we solely relied on $u''(x)$ for measuring risk aversion, it might inaccurately suggest that wealthier individuals become nearly risk-neutral, due to the diminishing curvature in their utility functions. This phenomenon can be explained by the principle of decreasing marginal utility, which asserts that the incremental value or utility derived from each additional unit of wealth diminishes at higher wealth levels.

3.1 Absolute Risk Aversion (ARA)

A more nuanced measure of risk aversion is defined through the Absolute Risk Aversion (ARA) index, given by:

$$\rho_{ARA}(x) = -\frac{u''(x)}{u'(x)}$$

This metric, also known as the Arrow-Pratt measure of risk aversion, remains robust against the non-linear scaling of wealth. As wealth increases, both the second derivative $u''(x)$ and the first derivative $u'(x)$ typically decrease, but their ratio, representing the ARA, adjusts proportionally. Thus, the measure of risk aversion does not necessarily approach zero as wealth grows, providing a more reliable indicator across different wealth levels.

This measure is intrinsically linked to the concept of the risk premium.

3.1.1 Constant Absolute Risk Aversion (CARA)

If ARA is constant, we refer to it as Constant Absolute Risk Aversion (CARA). The utility function aligning with CARA can be derived by solving the differential equation $ _{CARA} = -u' ‘/u’ $, which gives:

$$u(x) = -e^{-x \cdot \rho_{CARA}}$$

This is the exact same utility function as in the previoius examples. Although the CARA utility function is negative, it should not be interpreted as the individual deriving negative utility from wealth. Instead, it serves as a ranking mechanism among different wealth levels, ensuring $ u(x) > u(y) $ whenever $ x > y $, and $ u(x) > u(y) $ whenever $ var(x) > var(y)$ and $x = y$

This utility specification simplifies calculations and is extensively used due to its mathematical tractability. It also implies that for a CARA utility function, the risk premium can be calculated exactly as $\frac{1}{2}\pi_{ARA}\sigma^2$.

3.2 Relative Risk Aversion (RRA)

Relative Risk Aversion (RRA) is more applicable when the risky decision involves a proportion of an individual's wealth, rather than a fixed amount.

This measure is particularly relevant when returns are expressed as a percentage of wealth, aligning more closely with practical financial scenarios. RRA is mathematically defined as:

$$\pi_{RRA}(x) = -x\frac{u''(x)}{u'(x)}$$

The choice between using Absolute Risk Aversion (ARA) and RRA depends on the nature of the risk involved. RRA is preferred when analyzing bets that are proportional to wealth, while ARA is more suitable for fixed-level bets.

The constant RRA utility, is a Cobb-Douglas-type function:

$$u_{CRRA}(x) = x^{1-\pi_{CARA}}$$

4 The Normal Distribution and risk

As mentioned, if the distribution of the returns is normal, the investor should allways maximize the difference between mean and variance. However, if the returns are distributed differently, that is not the case.

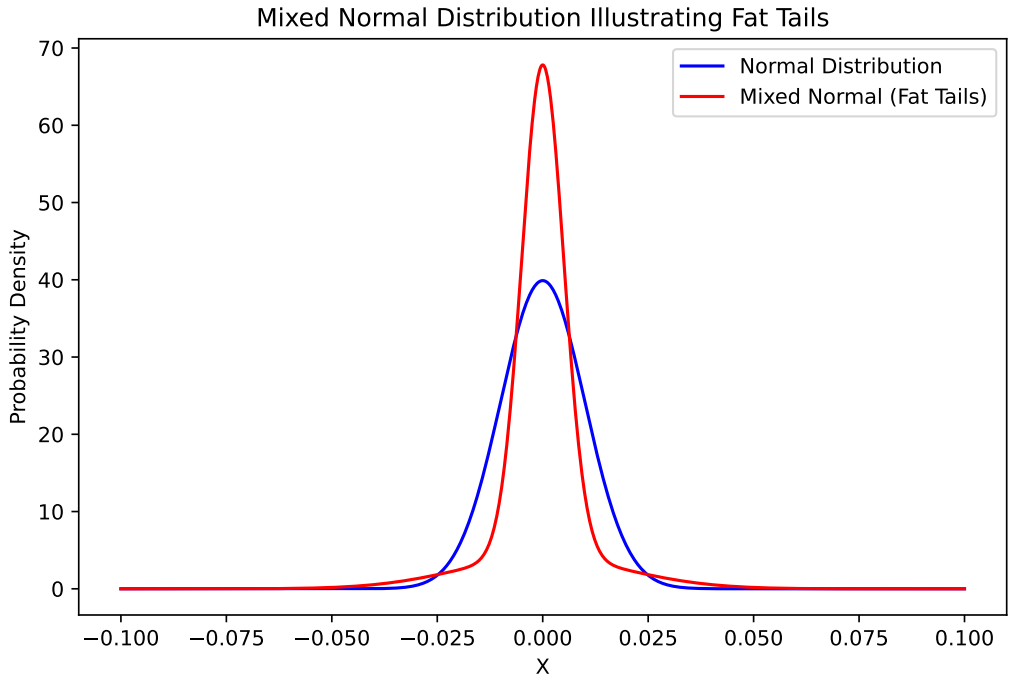
It is well known that returns in financial markets are not normally distributed.For example, under the assumption that returns on the Oslo Stock Exchange follow a normal distribution, statistically extreme changes exceeding 10% would be exceedingly rare—estimated to occur once every 17,000 years. Yet, during the 2008 financial crisis, such anomalies were observed twice.

This is called “fat tails”. It means that extreme events are much more likely in actual financial markets, than in the normal distribution world.

However, in most cases, we can represent the empirical distribution with a “mixed normal distribution”. A mixed normal distribution, is a linear combination of normal distirbutino with different mean and varinaces.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
```

```
#Set range
MAX_MIN = 0.1
x_vals = np.linspace(-MAX_MIN, MAX_MIN, 1000)

# Create the plot using fig and ax
fig, ax = plt.subplots(figsize=(8, 5))

#Calculate mean and std:
rmean = 0
rstd = 0.01

# Normal distribution and mixed normal distribution:
norm_dist    =     norm.pdf(x_vals, loc=rmean, scale=rstd)
fat_tails = (0.8 * norm.pdf(x_vals, loc=rmean, scale=rstd*0.5)
           + 0.2 * norm.pdf(x_vals, loc=rmean, scale=rstd*2))

# Plotting
ax.plot(x_vals, norm_dist, label="Normal Distribution", color='blue'
ax.plot(x_vals, fat_tails, label="Mixed Normal (Fat Tails)", color='

# Set title, labels, and grid and legend
ax.set_title('Mixed Normal Distribution Illustrating Fat Tails')
ax.set_xlabel('X')
ax.set_ylabel('Probability Density')
ax.legend()
```



Coding Challenges:

- Challenge 1: Use data from https://titlon.uit.no to create a histogram, and overlay the normal distribution to see if the empirical distribution has fat tails.
- Challenge 2: Try to fit a mixed normal distribution to the empirical distribution.

Solution:

We start by fetching data for the OSEBX index from titlon.uit.no

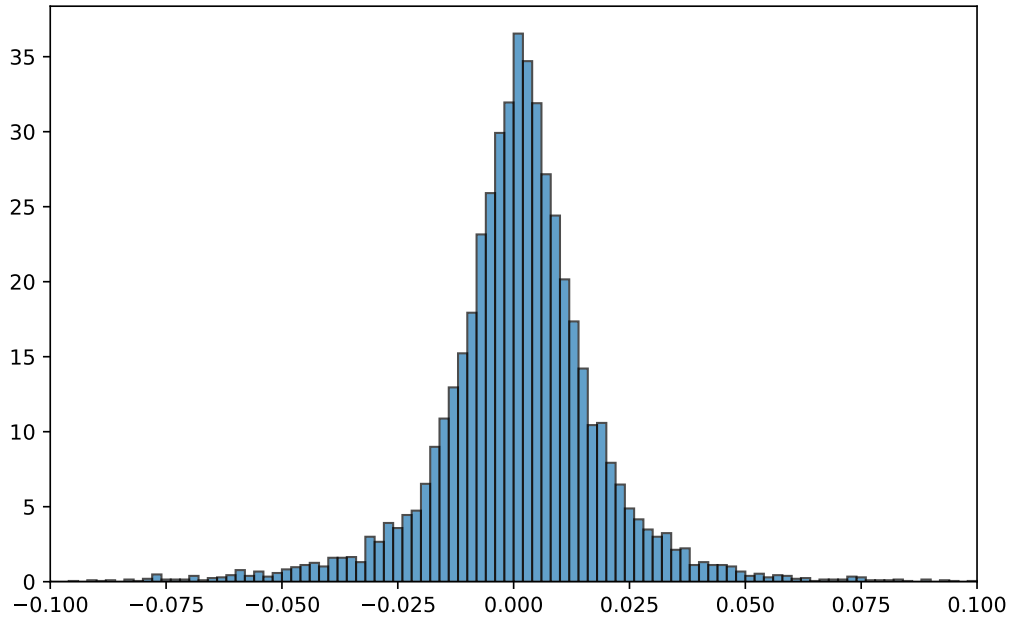
```
import pandas as pd
#Query script for MySQL client
import pymysql
con = pymysql.connect(host='titlon.uit.no',
                      user="esi000@uit.no",
                      password  = "39oQ!Fjzpuh$OZ2FpewRp",
                      database='OSE')
crsr=con.cursor()
crsr.execute("""
    SELECT
```

```
    `Date`, `OSEBXLinked`, `OBXLinked`, `OSEFX`, `ID`
    FROM `OSE`.`equityindex_linked`
    WHERE year(`Date`) >= 1980

""")
r=crsr.fetchall()
df=pd.DataFrame(list(r),
        columns=[i[0] for i in crsr.description])
df
#YOU NEED TO BE CONNECTED TO YOUR INSTITUTION VIA VPN,
#OR BE AT THE INSTITUTION, FOR THIS CODE TO WORK

pd.to_pickle(df,'output/index.df')
df = None #in order to avoid memory problems
```

We can then calculate the return series and creating a histogram. Then period mean and standard deviation is also calculated.

```
# loading the data
df = pd.read_pickle('output/index.df')

# Creating the plot using fig and ax
fig, ax = plt.subplots(figsize=(8, 5))

#fixing Date, setting index and calculate returns
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
df['Returns'] = df['OSEBXLinked'].pct_change()

#Calculate mean and std:
rmean = np.mean(df['Returns'])
rstd = np.std(df['Returns'])

# Define the histogram intervals (dx = 0.01)
dx = 0.002
bins = int((0.2 - (-0.2)) / dx)

# Create the plot
ax.hist(df['Returns'].dropna(), bins=bins,
```

```
        range=(-0.2, 0.2), density=True,
        edgecolor='k', alpha=0.7)
df = None

# Set x-axis limits
ax.set_xlim(-MAX_MIN, MAX_MIN)
```



We can now recalculate the normal distribution and the mixed normal, but based on the estimated mean and standard deviation, and add that to the existing chart.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

#Defining the range of x-values:
MAX_MIN = 0.1
x_vals = np.linspace(-MAX_MIN, MAX_MIN, 1000)

# Calculate normal dist and the mixed normal dist:
norm_dist    =     norm.pdf(x_vals, loc=rmean, scale=rstd)
fat_tails = (0.8 * norm.pdf(x_vals, loc=rmean, scale=rstd*0.4) +
```

```
            0.2 * norm.pdf(x_vals, loc=rmean, scale=rstd*2))

# Plotting normal dist and mixed normal dist
ax.plot(x_vals, norm_dist,
            label="Normal Distribution", color='blue')
ax.plot(x_vals, fat_tails,
            label="Mixed Normal (Fat Tails)", color='red')

# Set title, labels and legend
ax.set_title('Mixed Normal Distribution'
                'Illustrating Fat Tails')
ax.set_xlabel('X')
ax.set_ylabel('Probability Density')
ax.legend()
fig
```