Response for Project 3

Problem 1

Calculate and compare the expected value and standard deviation of Pt

I first define 3 ways of price simulation

- Brownian Motion

```
def brownianPrice(Pt_1, ret):
    return Pt_1 + ret
```

- Arithmetic Return System

```
def arithmeticPrice(Pt_1, ret):
    return Pt_1 * (1 + ret)
```

- Log Return or Geometric Brownian Motion

```
def geometricBrownPrice(Pt_1, ret):
    return Pt_1 * np.exp(ret)
```

Next, I simulate 10000 returns following the normal distribution with mean 0 and sigma 0.1 and set the initial price to be 100.

Then, I calculate Pt according the 3 ways of price simulation and calculate their mean and standard deviation. Here is the result.

	mean	std
brownian	99.998926	0.099836
arithmatic	99.892642	9.983573
geometric	100.391081	10.026342

Theoretically,

$$Brownian: E(P_t) = E(P_{t-1} + r_t) = E(P_{t-1}) + E(r_t) = P_{t-1} + 0 = P_{t-1}$$

$$Arithmetic: E(P_t) = E(P_{t-1}(1 + r_t)) = E(P_{t-1} + P_{t-1}r_t) = P_{t-1}$$

$$LogReturn: E(P_t) = E(P_{t-1}e^{r_t}) = P_{t-1}e^{E(r_t)} = P_{t-1}$$

$$Brownian: Var(P_t) = Var(P_{t-1} + r_t) = Var(P_{t-1}) + Var(r_t) + 2Cov(P_{t-1}, r_t) = Var(r_t)$$

$$\sigma_{P_t} = \sigma_{r_t}$$

$$Arithmetic: Var(P_t) = Var(P_{t-1} + P_{t-1}r_t) = Var(P_{t-1}) + Var(P_{t-1}r_t)$$

$$= Var(P_{t-1}) + P_{t-1}^2 Var(r_t) = P_{t-1}^2 Var(r_t)$$

$$\sigma_{P_t} = P_{t-1}\sigma_{r_t}$$

$$LogReturn: Var(P_t) = Var(P_{t-1}e^{r_t}) = P_{t-1}^2 Var(e^{r_t}) = P_{t-1}^2 Var(r_t)$$

$$\sigma_{P_t} = P_{t-1}\sigma_{r_t}$$

Since I set the initial price to be 100, and the sigma to be 0.1, the expectation of Pt should be around 100, the standard deviation of brownian motion Pt should be around 0.1, and the standard deviation of arithmetic and geometric Pt should be around 10. Therefore, the means and standard deviations match my expectations.

Problem 2

Implement a function similar to the “return_calculate()” in this week’s code

```
def return_calculate(prices, method="DISCRETE", dateColumn="date"):
    if dateColumn not in prices.columns:
        print("dateColumn: ", dateColumn, " not in DataFrame: ", prices)
        return
    rets = prices.copy()
    if method.upper() == "DISCRETE":
        for column in prices.columns[1:]:
            rets[column] = prices[column] / prices[column].shift() - 1
    elif method.upper() == "LOG":
        for column in prices.columns[1:]:
            rets[column] = np.log(prices[column] / prices[column].shift())
    else:
        print("method: ", method, " must be DISCRETE or LOG")
```

```
    return rets.iloc[1:, :]
```

Calculate VaR using a normal distribution, a normal distribution with an Exponentially Weighted variance, a MLE fitted T distribution, and a Historic Simulation

VaR with a normal distribution is basically the percentile of a normal distribution given the significance level, mean, and standard deviation.

```
def normalVaR(alpha, mu, sigma):
    return -norm.ppf(alpha, loc=mu, scale=sigma)
```

VaR with a exponentially weighted variance normal distribution is the percentile of a normal distribution given the significance level, mean, and a exponentially weighted standard deviation.

```
def EWNormalVaR(alpha, lam, ret):
    weight = exponentialWeights(len(ret), lam)
    sigma = np.sqrt(expCovForPair(weight, ret, ret))
    mu = np.mean(ret)
    return normalVaR(alpha, mu, sigma)
```

VaR with MLE fitted T distribution is the percentile of a generalized T distribution given the MLE estimated degree of freedom and standard deviation.

```
def mleTVaR(alpha, ret):
    def negLogLikeForT(initialParams):
        df, sigma = initialParams
        return -t(df=df, scale=sigma).logpdf(ret).sum()
    initialParams = np.array([1, 1])
    df, sigma = minimize(negLogLikeForT, initialParams, method="BFGS").x
    return -t.ppf(alpha, df, loc=ret.mean(), scale=sigma)
```

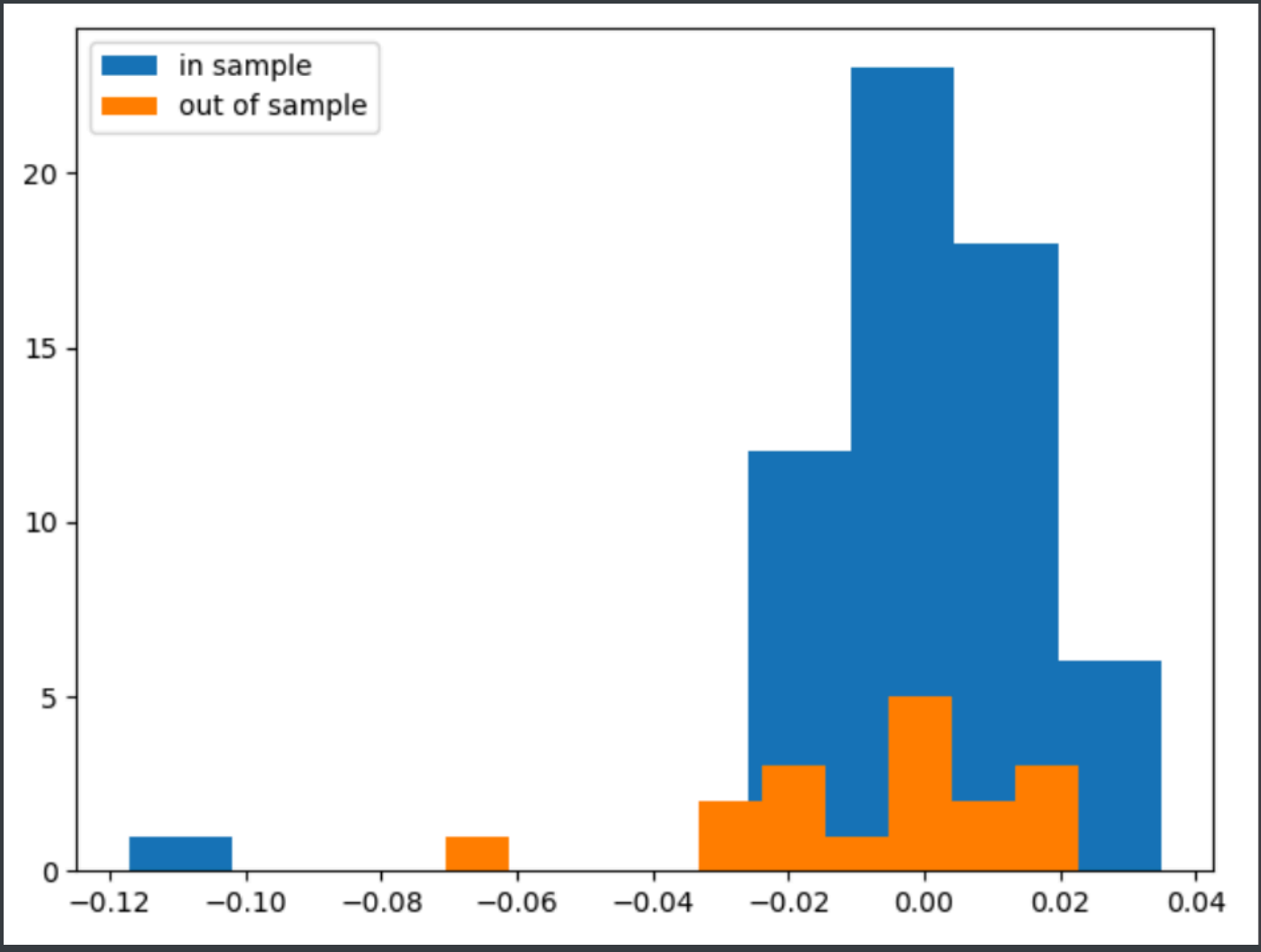
VaR with a historical simulation is basically the quantile of all the historical data given a significance level.

```
def historicVaR(rets, alpha):
    return -np.quantile(rets, q=alpha)
```

Compare the 4 values

```
INTC normal VaR:  3.41 %
INTC EW normal VaR:  2.61 %
INTC MLE with T VaR:  2.73 %
INTC historical VaR:  2.07 %
```

Look at the empirical distribution of returns, in sample and out of sample



Discuss the ability of these models to describe the risk in this stock

From the graph I can observe two outliers in the in-sample data and the out-of-sample data. However, all VaR models do not capture these two extreme loss. In this case, VaR with normal distribution performs the best since it has the largest VaR. VaR with historical simulation performs the worst since historical data may not represent out-of-sample data very well.

Problem 3

I choose delta normal and historical simulation to calculate VaR.

```
def deltaNormalVaR(alpha, rets, currPrices, holdings, dateColumn='Date'):
    rets = rets.drop(dateColumn, axis=1)
    presentValue = (currPrices * holdings).sum()
    weights = currPrices * holdings / presentValue
    sigma = np.cov(rets.T)
    portfolioSigma = np.sqrt(weights.T @ sigma @ weights)
    return -presentValue * norm.ppf(alpha) * portfolioSigma
```

```
def historicVaR(alpha, rets, currPrices, holdings, dateColumn='Date'):
    rets = rets.drop(dateColumn, axis=1)
    simulatedPrices = (1 + rets) * currPrices
    simulatedValues = simulatedPrices @ holdings
    simulatedValues = simulatedValues.sort_values()
    simulatedValues = simulatedValues.reset_index(drop=True)
    presentValue = (currPrices * holdings).sum()
    return presentValue - simulatedValues[int(alpha*len(rets))]
```

	historicVaR	deltaNormalVaR
A	5298.490894	6003.221296
B	5576.130254	4886.596045
C	3307.758237	3679.556066
Total	12460.873738	14100.550125

Discuss your methods, why you chose those methods, and your results

Delta normal method assumes that all asset returns are normally distributed. It gives each stock a weight based on its position in a portfolio. The normality assumption is also the drawback since in reality stock returns are fat-tailed.

Historical simulation method has almost no assumption. The distribution of returns can be non-normal, and securities can be non-linear. One drawback is that the distribution of historical data may not represent the future distribution.

I choose these two methods because I want to compare the prediction power of parametric and non-parametric methods.

Generally speaking, my result shows that delta normal method gives higher VaRs compared to those given by the historical-simulation method. Considering the fat-tail charasteristics of the distributions of stock returns, I think deltaNormalVaR does a better job than the historicVaR.