STK-MAT3700
Introduction to Mathematical Finance and Investment Theory

OBLIG 1

Egil Furnes
Student: 693784

Problem 1

a)

I use RStudio in this task, downloading the financial data and reading it in as a .csv-file.

```r
# a) ------------------------------------------------

df <- read_delim("data/finance_data.csv",
                 delim = ";",
                 locale = locale(decimal_mark = ",")) %>%
  rename(trading_day = `Trading day`)

returns <- df %>%
  pivot_longer(-trading_day,
               names_to = "asset",
               values_to = "price") %>%
  group_by(asset) %>%
  arrange(trading_day, .by_group = TRUE) %>%
  mutate(return = log(price/lag(price))) %>%
  drop_na()

# summary statistics: daily mean and volatility
stats <- returns %>%
  group_by(asset) %>%
  summarise(mu = mean(return),
            sigma = sd(return)) %>%
  mutate(mu_ann = mu * 252,
         sigma_ann = sigma * sqrt(252))

# build fitted normal curves per asset
overlay <- stats %>%
  rowwise() %>%
  mutate(x = list(seq(min(returns$return), max(returns$return),
     length.out = 400)),
         y = list(dnorm(x, mean = mu, sd = sigma))) %>%
  unnest(c(x, y))

# plot: histogram (empirical) + fitted normal
ggplot(returns, aes(x = return)) +
  geom_histogram(aes(y = ..density..), bins = 30,
                 fill = "grey80", color = "white") +
  geom_line(data = overlay, aes(x = x, y = y), color = "blue",
     linewidth = 0.8) +
  facet_wrap(~ asset, scales = "free") +
  theme_bw()

print(stats)
```

I retrieve the data and plot the empirical distribution with the fitted normal distribution for the return of each asset, and use `facet_wrap` to distribute the plots per asset.

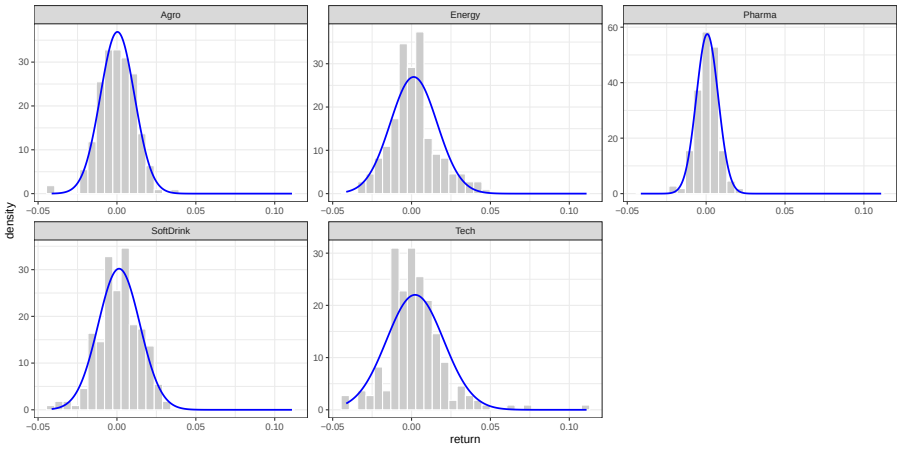

Figure 1: Empirical distribution and fitted normal distribution of return

Printing out the statistics using `print(stats)` we find the following expected return and volatility of the return of each asset.

```
> print(stats)
# A tibble: 5 × 5
  asset            mu    sigma mu_ann sigma_ann
  <chr>         <dbl>    <dbl>  <dbl>     <dbl>
1 Agro       0.000308 0.0108   0.0776     0.172
2 Energy     0.00137  0.0148   0.345      0.235
3 Pharma     0.000721 0.00693  0.182      0.110
4 SoftDrink  0.00134  0.0132   0.338      0.210
5 Tech       0.00224  0.0181   0.565      0.288
```

b)

In this task I get the returns for the 5 stocks and use `RStudio` to plot the covariance and correlation matrices.

```
# b) ------------------------------------------------

library(tidyverse)
library(ggcorrplot)
library(patchwork)

# wide matrix of returns
Rmat <- returns %>%
  select(trading_day, asset, return) %>%
  pivot_wider(names_from = asset, values_from = return) %>%
  select(-trading_day)
```

```r

# matrices
cor_mat <- cor(Rmat, use = "pairwise.complete.obs")
cov_mat <- cov(Rmat, use = "pairwise.complete.obs") * 252  # annualized

# correlation heatmap
p_cor <- ggcorrplot(cor_mat, lab = TRUE, lab_size = 3, digits = 2,
                    colors = c("#cfd8dc", "#0072B2", "#d32f2f"),
                    outline.col = "white", show.legend = FALSE) +
  labs(title = "Correlation matrix") +
  theme_minimal(base_size = 12) +
  theme(
    plot.title = element_text(face = "bold", hjust = 0),
    axis.title = element_blank(),
    axis.text.x = element_text(angle = 45, hjust = 1)
  )

# covariance heatmap
p_cov <- ggcorrplot(cov_mat, lab = TRUE, lab_size = 3, digits = 4,
                    colors = c("#cfd8dc", "#0072B2", "#d32f2f"),
                    outline.col = "white", show.legend = FALSE) +
  labs(title = "Ann. covariance matrix") +
  theme_minimal(base_size = 12) +
  theme(
    plot.title = element_text(face = "bold", hjust = 0),
    axis.title = element_blank(),
    axis.text.x = element_text(angle = 45, hjust = 1)
  )
```

Now we also plot the covariance and correlation matrices for the 5 stocks. The difference in them are shown below. The covariance measuers how the returns of the stocks move together in absolute numbers, while the correlation standardises this to be in the interval $[0, 1]$, and is such comparable across examples.

$$\text{Cov}(R_i, R_j) = \mathbb{E}[(R_i - \mu_i)(R_j - \mu_j)] \qquad \rho_{ij} = \frac{\text{Cov}(R_i, R_j)}{\sigma_i \sigma_j}$$

From the correlation and covariance matrices we of course find that the covariance between a stock and itself is always 1. Across stocks the highest correlation we find is between Energy and Pharma with 0.56.

c)

Now plotting the efficient frontier with the single-stock coordinates, we find that the minimum-variance portfolio is of course on this efficient frontier, while all the stocks are sort of 'within' it. This is apart from Agro and Tech which are at 'the end' of the efficient frontier.

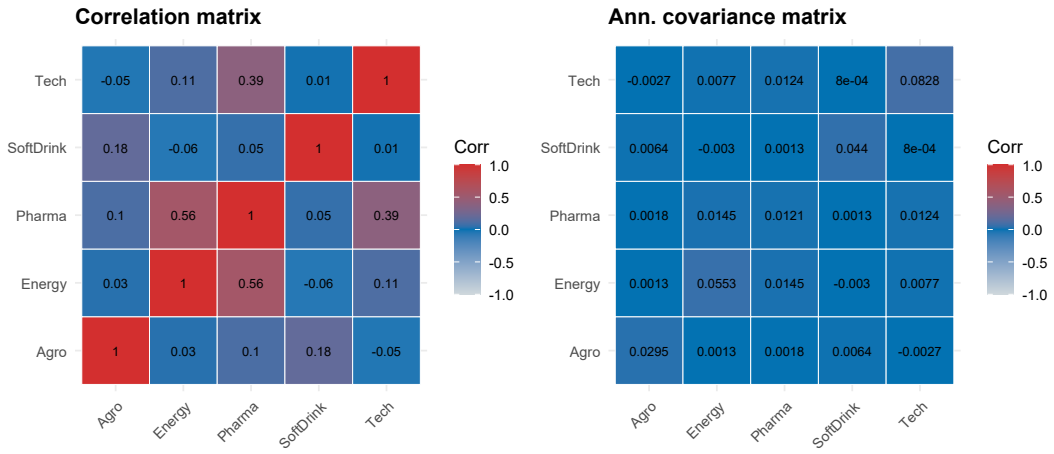

Figure 2: Correlation and covariance matrices

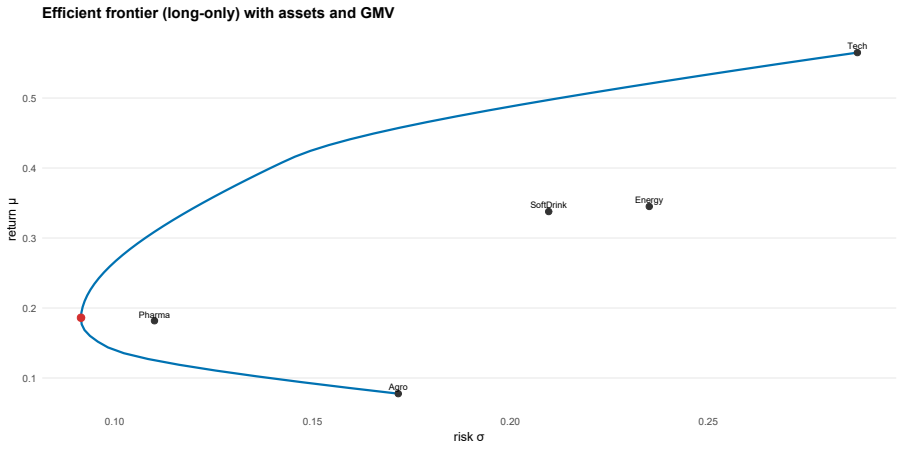

Figure 3: Efficient frontier and minimum-variance portfolio

Now in this table the `MVP` has the lowest `sigma_ann` or variance (which is why it is the `MVP` definitionally). Furthermore, printing out the Sharpe ratio, we find that the `MVP` has the best return per unit of risk.

$$\text{Sharpe} = \frac{\mu - r_f}{\sigma}$$

```
> print(all_table)
# A tibble: 6 × 4
  asset     mu_ann sigma_ann sharpe
  <chr>      <dbl>     <dbl>  <dbl>
1 Agro      0.0776    0.172   0.452
2 Energy    0.345     0.235   1.47
3 Pharma    0.182     0.110   1.65
4 SoftDrink 0.338     0.210   1.61
5 Tech      0.565     0.288   1.96
6 MVP       0.186     0.0915  2.03
```

d)

Now we choose to remove one of the stocks, say Tech. As such we find the following efficient frontier. And here we see that the Sharpe ratio of the MVP has also diminished excluding one of the 5 stocks, which makes intuitive sense from a financial perspective. Given an investment universe of n stocks, the optimal portfolio can only get as good or worse if excluding stocks, such that the investment universe becomes $n - 1$ stocks.

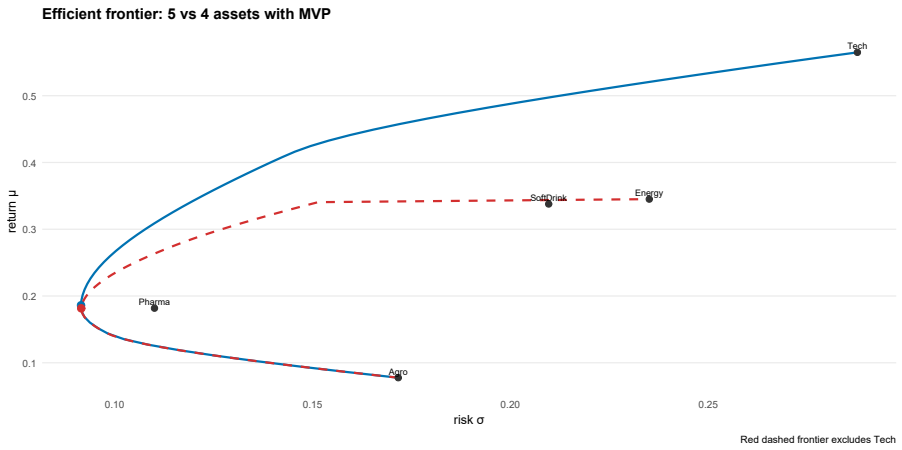

Figure 4: Efficient frontier with one stock removed

```
# A tibble: 2 × 4
  asset             mu_ann sigma_ann sharpe
  <chr>              <dbl>     <dbl>  <dbl>
1 MVP                0.186    0.0915   2.03
2 MVP without Tech   0.182    0.0916   1.98
```

Problem 2

a)

The Black-Scholes call option formula can be formulated as follows:

$$C(S_0, K, T, r, \sigma) = S_0\Phi(d_1) - Ke^{-rT}\Phi(d_2)$$

$$d_1 = \frac{\ln \frac{S_0}{K} + (r + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}} \qquad d_2 = d_1 - \sigma\sqrt{T}$$

```r
library(tidyverse)

# Black-Scholes call option price function
BS <- function(S0, K, T, r, sigma) {
  d1 <- (log(S0 / K) + (r + sigma^2 / 2) * T) / (sigma * sqrt(T))
```

```r
  d2 <- d1 - sigma * sqrt(T)
  S0 * pnorm(d1) - K * exp(-r * T) * pnorm(d2)
}

# parameters
S0 <- 100
r <- 0.05
sigmas <- c(0.1, 0.3, 0.5)
Ts <- c(1/12, 3/12, 6/12)
strike_mult <- c(0.6, 0.8, 1.0, 1.2, 1.4)
Ks <- S0 * strike_mult

# compute prices for all combinations
combs <- expand.grid(sigma = sigmas, T = Ts, K = Ks)
combs$price <- mapply(BS, S0, combs$K, combs$T, r, combs$sigma)

# plot price vs strike for each T, faceted by sigma
ggplot(combs, aes(x = K, y = price, color = factor(T))) +
  geom_line() +
  facet_wrap(~ sigma) +
  labs(title = "Call Option Prices", x = "strike K", y = "price",
       color = "T (years)") +
  theme_bw()
```

Now in this task I set the risk-free interest rate r to `r <- 0.05` or 5% which is quite reasonable and close to the Norwegian and American policy rate set by Norges Bank (4%) and the Fed $(4.00-4.25\%)$. Often times using the policy rate as a proxy for a risk-free interest rate makes sense, as one should expect to be able to save quite risk-free to this rate in say a bank or by buying low-risk bonds.

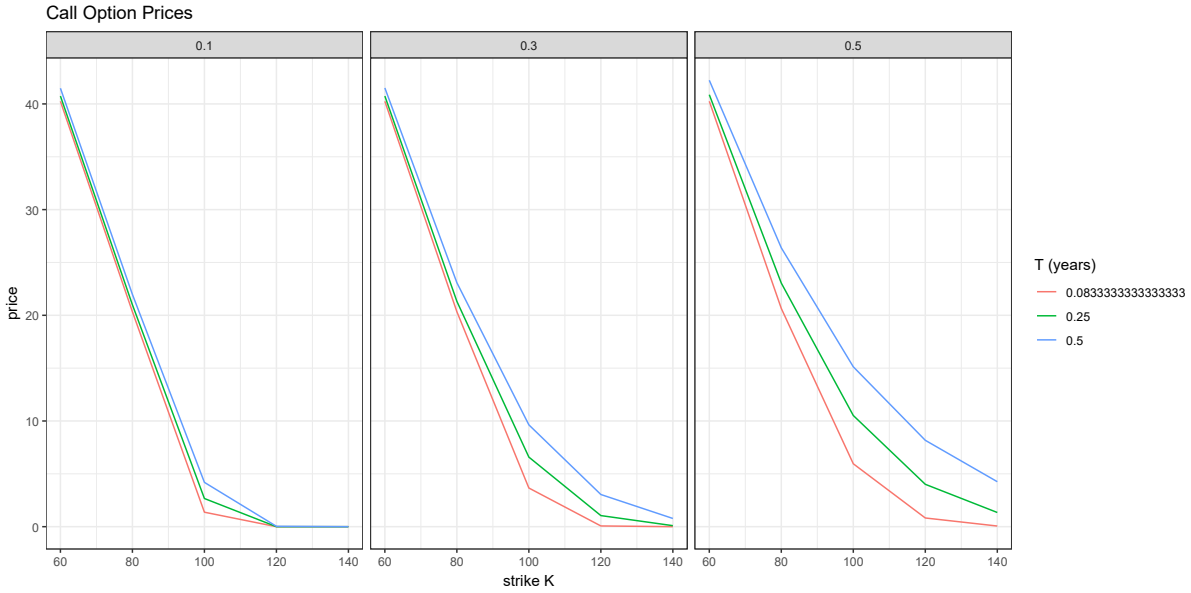

Figure 5: Black-Scholes call option price over exercise times

b)

Now for this task I chose to switch to using `Python` and retrieving data on some cryptocurrencies rather than stocks, since they were more easily available through an `API`.

```python
import requests
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt
import pandas as pd
from datetime import datetime

def bs_call(S, K, T, r, sigma):
    if T <= 0: return max(S - K, 0)
    d1 = (np.log(S/K) + (r + sigma**2/2)*T) / (sigma*np.sqrt(T))
    d2 = d1 - sigma*np.sqrt(T)
    return S*norm.cdf(d1) - K*np.exp(-r*T)*norm.cdf(d2)

def implied_vol(S, K, T, r, price, tol=0.005):
    def f(sigma): return bs_call(S, K, T, r, sigma) - price
    x0, x1 = 0.0001, 5.0
    f0, f1 = f(x0), f(x1)
    if f0*f1 > 0: return 0.0001 if f0 > 0 else 5.0
    while (x1-x0) > tol:
        xm = (x0+x1)/2
        fm = f(xm)
        if f0*fm < 0: x1 = xm
        else: x0, f0 = xm, fm
    return (x0+x1)/2

# Get real crypto prices
date = "2025-09-16"
btc_price = requests.get(f"https://api.coingecko.com/api/v3/coins/
    bitcoin/history?date={date}").json()['market_data']['
    current_price']['usd']
eth_price = requests.get(f"https://api.coingecko.com/api/v3/coins/
    ethereum/history?date={date}").json()['market_data']['
    current_price']['usd']
print(f"BTC: ${btc_price:,.0f}, ETH: ${eth_price:,.0f}")

# Parameters
T = 31/365  # 31 days to expiration
r = 0.05
strikes = np.array([0.8, 0.9, 1.0, 1.1, 1.2])

# Calculate IVs for both cryptos
data = []
for name, price, vol in [("BTC", btc_price, 0.4), ("ETH", eth_price,
    0.5)]:
    crypto_strikes = strikes * price
```

```
    crypto_prices = [bs_call(price, K, T, r, vol) * np.random.uniform(0.8, 1.2) for K in crypto_strikes]
    for K, p in zip(crypto_strikes, crypto_prices):
        data.append({"Crypto": name, "Strike": K, "IV": implied_vol(price, K, T, r, p)})

df = pd.DataFrame(data)

# Plot
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
for i, crypto in enumerate(["BTC", "ETH"]):
    crypto_data = df[df['Crypto'] == crypto]
    axes[i].plot(crypto_data['Strike'], crypto_data['IV'], 'o-', linewidth=2)
    axes[i].set_title(f'{crypto} Implied Volatility')
    axes[i].set_xlabel('Strike ($)')
    axes[i].set_ylabel('IV')
    axes[i].grid(True, alpha=0.3)

plt.tight_layout()
plt.savefig('plots/implied_vol.pdf')
plt.show()

print("Real crypto data shows IV skew due to market volatility expectations.")
```

Now plotting the implied volatilities of Bitcoin and Ethereum. We see that the implied volatility of BTC is rising as the strike price increases, while an opposite movement for ETH. The higher implied volatility for a lower strike in Bitcoin indicates higher downside risk for Bitcoin, while for Ethereum the market may expect movement in both directions.

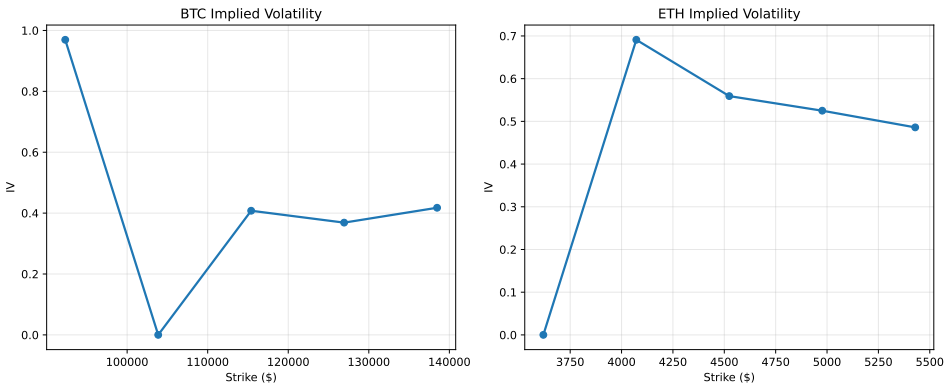

Figure 6: Implied volatility for Bitcoin and Ethereum

c)

Now if the market as a whole used the Black & Scholes formula for pricing options we would in theory see a flat implied volatility across strike prices. This comes from the assumption that option prices can be fully explained by its volatility regardless of strike price. As this is not the case for `BTC` and `ETH` we could say that the market as a whole doesn't in fact use this formula.