UP902897

Research paper no. 19

1. Simulation of the mean-variance model

I have decided to use Python across this report to conduct a simulation of the mean-variance model involving five stocks from Nasdaq stock exchange, namely Apple, Amazon, Facebook, Microsoft, Tesla. The stock symbols (tickers) have been found using Stock Screener (*Stock Screener*, n.d.)

First, we download the historical stock prices for one year on the transaction days between 01/01/2019 and 31/12/2019, as well as their closing prices on 31/12/2020 using Yahoo Finance.

```
import yfinance as yf
import numpy as np

tickers = ['TSLA','FB','AAPL','AMZN','MSFT'] # list of tickers for stocks in my portfolio

daily19_20 = yf.download(tickers,'2019-01-01','2020-01-01')['Adj Close']# dataframe of adjusted close prices for
                                                                        # each stock from 2018.12.31 to 2019.12.31
                                                                        # from yafoo finance
                                                                        # (2019.01.01 - not a trading day)
prices31_12_20 = yf.download(tickers,'2021-01-01','2021-01-01')['Close']  # dataframe of closing prices for
                                                                            # each stock on 31.12.2020
display(daily19_20, prices31_12_20)
```

Stock exchange: Nasdaq

Stocks: Apple, Amazon, Facebook, Microsoft, Tesla

31/12/2019 Adjusted Prices for each stock alphabetically:

2019-12-31	72.675339	1847.839966	205.250000	155.688324	83.666000

31/12/2020 Closing Prices for each stock alphabetically:

2020-12-31	132.690002	3256.929932	273.160004	222.419998	705.669983

The next step is to compute the mean of the return for each stock to get the mean vector and the covariance matrix of the return of the stocks. For this purpose, we use the formula for rate of return:

$$r = \frac{X_1 - X_0}{X_0}$$

Where, r - daily rate of return

X_0 - adjusted price current day

X_1 - adjusted price following day

We compute this for each trading day in 2019 starting from 31/12/2018 as 1/1/2019 was not a trading day to get 252 values (252 trading days in 2019) for each stock (dataframe named *daily_returns)*.

```
daily_returns = (daily19_20 - daily19_20.shift(1)) / daily19_20.shift(1)#using formula for rate of return from moodle,
                                                                        # r=(X1-X0)/X0
daily_returns
```

To find mean of the return for each stock we use:

$$E(r_i) = \sum_{k=1}^{n} r_{ik} p_k$$

$$= \sum_{k=1}^{n} \frac{1}{n} r_{ik}$$

$$= \frac{1}{n} \sum_{k=1}^{n} r_{ik}$$

Each r_i can take on the values r_{ik} for $k = 1, \dots, n$, with equal probability $\frac{1}{n}.(n = 252)$

Code:

```
# mean returns vector code:
r = daily_returns.iloc[1:] # delete first row with NaNs
n = len(r)
m = len(r.columns)
r_mean = []

for i in range(m):
    terms = 0
    for k in range(n):
        terms += r.iat[k,i]
    r_mean.append(terms / n)

r_mean = np.array(r_mean)

# we could use:
r_mean2 = np.array(r.mean(axis=0))
r_mean, r_mean2
# which gives same results
```

Output:

```
Out[3]: (array([0.0026646 , 0.00092645, 0.00193313, 0.00188339, 0.0013907 ]),
         array([0.0026646 , 0.00092645, 0.00193313, 0.00188339, 0.0013907 ]))
```

Covariance matrix is matrix whose (i, j) entry is the covariance computed using:

$$cov(r_i, r_j) = \frac{1}{n}\sum_{k=1}^{n}(r_{ik} - E[r_i])(r_{jk} - E[r_j])$$

Each pair (r_i, r_j) can take on the values (r_{ik}, r_{jk}) for $k = 1, \dots, n$, with equal probability $\frac{1}{n}$. Obtained mean vector and covariance matrix we multiply by 252.75 (average Nasdaq trading days per year) (*Trading Day*, n.d.) to get annual results.

```
Sigma = []
for i in range(m):
    for j in range(m):
        terms = 0
        for k in range(n):
            terms += (r.iat[k,i] - r_mean[i]) * (r.iat[k,j] - r_mean[j])
        cov = terms / (n) # biased estimator
        Sigma.append(cov)
Sigma = np.reshape(Sigma,(m,m))

# we could use:
Sigma2 = (r.cov())

display(Sigma, Sigma2)
# Results are slighly different because command .cov uses unbiased estimator (n-1)

#Annualised results
r_mean = r_mean * 252.75
Sigma = Sigma * 252.75

display(r_mean, Sigma)
```

Output of mean vector and covariance for the return of 5 stocks:

Apple	Amazon	Facebook	Microsoft	Tesla

```
array([0.67347669, 0.23416066, 0.48859785, 0.47602737, 0.35150046])

array([[0.0682527 , 0.03523713, 0.03433688, 0.03219264, 0.04252531],
       [0.03523713, 0.05243306, 0.03964468, 0.03246856, 0.03108779],
       [0.03433688, 0.03964468, 0.07763619, 0.02891381, 0.02996463],
       [0.03219264, 0.03246856, 0.02891381, 0.03927752, 0.02799044],
       [0.04252531, 0.03108779, 0.02996463, 0.02799044, 0.24300126]])
```

The last part is to optimise the following model:

$$min\ \boldsymbol{w}^T \sum \boldsymbol{w}$$

Constraints:

$$w^T \bar{\mathbf{r}} = \bar{r}$$

$$\boldsymbol{w}^T = \mathbf{1}$$

Where,

$\boldsymbol{w}^T$ - row vector of weights

$\boldsymbol{w}$ - column vector of weights

$\sum$ - covariance matrix

$\bar{\boldsymbol{r}}$ - column vector of annualised mean return

$\bar{r}$ - mean annual return required by investor

1 – column vector of m-number of ones

```
from scipy.optimize import minimize

w = np.random.random(m)# generates random number of n-number of weights to start optimisation
ones = np.ones(m)      # generates vector of ones
r_target = [0.05,0.1,0.2]   # list of chosen expected returns
w_target = []
var = []

def objective(w):
    return np.dot(w, np.dot(Sigma, w))# w is 1D vector, S is symmetric, no need to transpose in python
def constraint1(w):
    return np.dot(w,ones)-1
for rt in r_target:
    def constraint2(w):
        return np.dot(w,r_mean) - rt
    cons = ({'type' : 'eq' , 'fun' : constraint1},
            {'type' : 'eq' , 'fun' : constraint2})
    solution = minimize(fun = objective, x0 = w, method = 'SLSQP',constraints = cons)
    w = solution.x
    w_target.append(w)
    var.append(objective(w))
```

```
w_target = np.array(w_target).round(2)
var = np.array(var).round(6)
prices = np.array(daily19_20.values[-1])
shares = np.divide(w_target, prices)*1000000
shares = shares.round(1)

r_target, w_target, var, shares
```

Output:

```
Out[5]: ([0.05, 0.1, 0.2],
         array([[-0.53,  1.27, -0.13,  0.29,  0.1 ],
                [-0.45,  1.14, -0.09,  0.32,  0.09],
                [-0.3 ,  0.85, -0.04,  0.42,  0.07]]),
         array([0.071085, 0.063053, 0.050128]),
         array([[-7292.7,   687.3,  -633.4,  1862.7,  1195.2],
                [-6191.9,   616.9,  -438.5,  2055.4,  1075.7],
                [-4127.9,   460. ,  -194.9,  2697.7,   836.7]]))
```

Results for $\bar{r} = 0.05$:

(Apple, Amazon, Facebook, Microsoft, Tesla)

$\boldsymbol{w} = -0.53,\ 1.27,\ -0.13,\ 0.29,\ 0.1$ (2 dp)

$\boldsymbol{w}^T \sum \boldsymbol{w} = 0.071085$ (6 dp)

Number of shares: -7292.7, 687.3, -633.4, 1862.7, 1195.2 (1 dp)

Base on algorithm, to minimise risk, we should short sell Apple stocks for £530000 and Facebook for £130000 (660000 initial gain) and buy Amazon stocks for £1270000, Microsoft stocks for £290000 and Tesla stocks for £100000 (1660000 initial cost). The variance of our portfolio is 0.071368, therefore standard deviation (risk) is $\sqrt{0.071085} = 26.7\%$. We would have 687.3, 1862.7 and 1195.2 stocks in Amazon, Microsoft and Tesla respectively and owe 7292.7 and 633.4 stocks in Apple and Facebook respectively.

Results for $\bar{r} = 0.1$:

(Apple, Amazon, Facebook, Microsoft, Tesla)

$\boldsymbol{w} = -0.45,\ 1.14, -0.09,\ 0.32,\ 0.09$ (2 dp)

$\boldsymbol{w}^T \sum \boldsymbol{w} = 0.063053$ (6 dp)

Number of shares: -6191.9, 616.9, -438.5, 2055.4, 1075.7 (1 dp)

Results for $\bar{r} = 0.2$:

(Apple, Amazon, Facebook, Microsoft, Tesla)

$\boldsymbol{w} = -0.3\ ,\ 0.85, -0.04,\ 0.42,\ 0.07$(2 dp)

$\boldsymbol{w}^T \sum \boldsymbol{w} = 0.050128$ (6 dp)

Number of shares: -4127.9, 460, -194.9, 2697.7, 836.7 (1 dp)

The values indicate that on the mean-standard deviation diagram our portfolio for given mean annual return is below minimum variance point on minimum variance set because as mean return rises, standard deviation decreases.

```
value = np.dot(shares, prices31_12_20.T).round(2)
value
```

```
Out[6]: array([[2355518.51],
               [2284067.45],
               [2087674.33]])
```

Final value for our portfolio on 31/12/2020 is:

for $\bar{r} = 0.05$, £2355518.51

for $\bar{r} = 0.1$, £2284067.45

for $\bar{r} = 0.2$, £2087674.33

It indicates that while the expected return rises the value of our portfolio decreases but overall, the value of the portfolio is much higher than expected. At the most fundamental level, economic factors, supply and demand in the market determine stock price. We cannot rely only on the stock charts from the past to predict stock future prices.

2. Research paper analysis

"Financial portfolio management through the goal programming model: Current state-of-the-art" by Belaid Aouni, Cinzia Colapinto and Davide La Torre present a summary of Goal Programming (GP) models for portfolio management.

GP was introduced in 1955 by Charnes, Cooper and Ferguson and it combines multiple objectives and minimises any positive or negative deviations between achievement levels and investment goals.

There are numerous methods of GP:

- Lexicographic goal programming

  A financial investor describes the priority of the multiple objectives and they are optimised accordingly. The objectives less important will play a minor part in the decision-making process.

  Mathematical formulation:

  Min $L = [l_1(\delta^-, \delta^+), l_2(\delta^-, \delta^+), \dots, l_q(\delta^-, \delta^+)$

  St $f_i(x) + \delta_i^- - \delta_i^+ = g_i$ (for $i = 1, 2, \dots, p$)

  $\sum_{j=1}^{n} x_j = 1$

  $x \in F$

  $\delta_i^-, \delta_i^+ \geq 0$ (for $i = 1, 2, \dots, p$)

L denotes ordered vector of unwanted deviations, q denotes different priority levels, $\wp^{\pm}$ are positive, negative deviations, x is a proportion to be invested in the stock and F is set of feasible solutions.

- Weighted goal programming

In this GP variant positive and negative deviations are expressed through the weights w_i^+ and w_i^-.

Mathematical formulation:

Min $Z = \sum_{i=1}^{p}(w_i^+ \delta_i^+ + w_i^- \delta_i^-)$

St $f_i(x) + \delta_i^- - \delta_i^+ = g_i$ (for $i = 1, 2, \dots, p$)

$\sum_{j=1}^{n} x_j = 1$

$x \in F$

$\delta_i^-, \delta_i^+ \geq 0$ (for $i = 1, 2, \dots, p$)

- Polynomial goal programming

It includes skewness into investor`s decision-making process, which implies that asset returns are not normally distributed. That was pointed out by Arditti and Levy in 1975.

Mathematical formulation:

Min $Z = [(d_1)^{p_1} + (d_3)^{p_3}]$

St $\sum_{j=1}^{n} x_j E_j + d_1 = E^*$

$\sum_{i=1}^{n} x_j \left(r_j - E_j\right)^3 + d_3 = S^*$

$\sum_{j=1}^{n} \sum_{k=1}^{n} x_j x_k \sigma_{jk} = 1$

$\sum_{j=1}^{n} x_j = 1$

$x \in F$

$d_1, d_3 \geq 0$

p_1, p_3 are preference parameters towards excess returns and skewness. The first and the second constraints describe the difference between the achievement level and the corresponding goals for both the expected return and the skewness criteria (S, E).

- Stochastic goal programming

This model includes the uncertainty linked to the decision-making process. In SGP, we suppose that the investment goal amounts are stochastic and follow particular probability distribution.

Mathematical formulation:

Min $Z = \sum_{i=1}^{p}(\overline{\delta_i^-} + \overline{\delta_i^+})$

St $\sum_{j=1}^{n} a_{ij}x_j + \overline{\delta_i^-} - \overline{\delta_i^+} = \overline{g_i}$ (for $i = 1, 2, \dots, p$)

$x \in F$

$\overline{\delta_i^-}, \overline{\delta_i^+} \geq 0$ (for $i = 1, 2, \dots, p$)

Where $\overline{g_i} \in N(\mu_i, \sigma_i^2)$

It can be extended to SGP model with satisfaction function which includes the investor`s preferences for the portfolio selection. The satisfaction functions are defined on the interval [0, 1]. A value of 1 indicates total satisfaction of the investor.

- Fuzzy goal programming

This model is used where the investor cannot precisely specify the investment goals. The objective function is assumed to be linear and involves lower and upper acceptability degrees.

Mathematical model:

Max $Z = \lambda$

St $\frac{f_1(x)}{\Delta_i} + \delta_i^- - \delta_i^+ = \frac{g_i}{\Delta_i}$ (for $\forall i \in I$)

$\lambda + \delta_i^- + \delta_i^+ \leq 1$ (for $\forall i \in I$)

$x \in F$

$\lambda, \delta_i^-, \delta_i^+ \geq 0$ (for $\forall i \in I$)

Δ_i is the constant of deviation of the investor`s goal levels g_i.

- There are also other GP variants used in portfolio management over the years, however less popular and important.

The mean-variance model developed by Harry Markowitz in 1952 has been enhanced and fulfilled in several directions since then. It was based on simultaneous optimisation of the return of the portfolio and risk related to financial losses. Nevertheless, the portfolio choice cannot be reduced to only two attributes and taking into consideration numerous criteria models is now standard. GP models consider this multi-dimensional method and define the range of investor`s preferences allowing for give-and-take. It lets to change the model parameters which improves the decision-making process.

“Financial portfolio management through the goal programming model: Current state-of-the-art” by Belaid Aouni, Cinzia Colapinto and Davide La Torre was published in 16/4/2014, since then, it contributed to many other research papers involving financial field. To mention just a few it had an impact only in 2020:

- “Global minimum variance portfolios under uncertainty: a robust optimization approach” by S Caçador, JM Dias, P Godinho

- “Financial modelling with multiple criteria decision making: A systematic literature review” by de Almeida-Filho, de Lima Silva, Ferreira
- “A stochastic goal programming model to derive stable cash management policies” by Salas-Molina, Rodriguez-Aguilar, Pla-Santamaria
- “Risk-sensitive control of cash management systems” by F Salas-Molina
- “Multiple criteria cash management policies with particular liquidity terms” by Salas-Molina, Pla-Santamaria, Garcia-Bernabeu, Mayor-Vitoria

(*Google Scholar,* n.d.)

3. Bibliography

Stock Screener. (n.d.). Nasdaq. Retrieved March 29, 2021, from https://www.nasdaq.com/market-activity/stocks/screener

Trading Day. (n.d.). Wikipedia. Retrieved March 29, 2021, from https://en.wikipedia.org/wiki/Trading_day

Google Scholar. (n.d.). Google Scholar. Retrieved April 1, 2021, from https://scholar.google.it/scholar?oi=bibs&hl=en&cites=2962870391907951179&as_sdt=5

Aouni, B., Colapinto, C., & La Torre, D. (2014). Financial portfolio management through the goal programming model: Current state-of-the-art. European Journal of Operational Research, 234(2), 536–545. https://doi.org/10.1016/j.ejor.2013.09.040