2DF40 Financial Mathematics

Assignment 2

Group 16

Sergiu Marin (0935191)
Bogdan Floris (0935036)

March 16, 2018

Introduction

The whole assignment is implemented in Python 3 using the standard data analysis libraries **NumPy**, **Pandas** and **Matplotlib**. The whole code for all the tasks is present in the appendix and it is organized into functions and commented. We will further explain it starting with the init function in this section and the other functions for all the tasks in the subsequent sections.

The init function is used to import the data from the *.csv* files. The *_history* and *_future* data frames contain the stock price information as found in the original files. *_rf* contains the risk free asset return. *_returns* contains the returns calculated using the data from *_history* with the function *pct_change*() that implemented the formula $(S_k - S_{k-1})/S_{k-1}$ exactly. We also drop the first row since it only contains nulls. *_gains* contains the returns calculated by adding 1 to the returns and putting them in the logarithm function.

Using this data, we can calculate everything needed in the four tasks.

Task 1

Using the *_returns* data frame and Pandas specific functions like *mean*(), *var*() and *cov*(), we can find the means, variances of the five stocks and covariance matrix. We only need to multiply each value by 252 to transform them in units of years and we are done. The results are presented in Table 1 and Table 2. Besides that, we also implemented three functions *_mean*(), *_var*() and *_cov*() in order to the describe the estimation procedure. The mean and var functions take as argument the name of a stock and calculate their mean and variance using the estimation formulas presented in the slides. The cov function takes as arguments two stocks and calculates their covariance again using the formula given in the slides.

Table 1: Means and Variances

	Mean	Variance
MU	0.432791	0.186241
AABA	0.291890	0.082653
AAPL	0.229464	0.052943
VEON	-0.139981	0.183946
INTC	0.193229	0.046199

Table 2: Covariance Matrix

	MU	AABA	AAPL	VEON	INTC
MU	0.186241	0.042748	0.027578	0.043588	0.038723
AABA	0.042748	0.082653	0.018364	0.029228	0.018845
AAPL	0.027578	0.018364	0.052943	0.021624	0.016306
VEON	0.043588	0.029228	0.021624	0.183946	0.021985
INTC	0.038723	0.018845	0.016306	0.021985	0.046199

Task 2

To be able to calculate the two portfolios: the minimum risk one and the tangency one, we first need to get the means and the covariance matrix from task 2 and then calculate the inverse of the matrix using the NumPy inverse function. Now, the weights, means and variances are easily calculated using the matrix multiplication formulas given in the slides using the NumPy *dot*() function. Table 3 presents these values.

These two portfolios are presented in Figure 1. The blue rhomb represents the minimum variance portfolio and the red one represents the tangency one. Moreover, the green one is used to represent the risk free asset. Besides only implementing the efficient frontier using the two portfolios and the two fund theorem, we have devised a Monte-Carlo technique that simulates the whole feasible set, with shorting allowed. This is done by generating five random numbers between -1 and 1, normalizing them so they sum up to 1, calculating their mean and standard deviation and adding it to a portfolio list. We have generated 30000 of those portfolios, but we have restricted their means to be between -0.3 and 1 and their standard deviation to be less than 0.6 because going besides those values would make the picture unreadable. All these portfolios form the most important part of the feasible set. The efficient frontier can be traced by starting from the blue rhomb and moving upwards along the feasible set. Along the way, we will also hit the tangency portfolio in red.

The capital market line can be traced using a Monte-Carlo technique again. We generate a random standard deviation between 0 and 0.6 and using the capital market line formula, we can calculate the mean. We plot each one of those points in black and we obtain the capital market line.

Table 3: Portfolios

	w_MU	w_AABA	w_AAPL	w_VEON	w_INTC	Mean	Var
Min-Var	-0.040468	0.180266	0.360756	0.0456550	0.453790	0.199179	0.029680
Tangency	0.215252	0.366189	0.450581	-0.316653	0.284629	0.402762	0.060798

Task 3

We get the weights of the averaged portfolio by summing the weights of each stock and dividing it by 2. The weights are in order of the stocks: 0.1259, 0.3012, 0.4192, -0.1901 and 0.3436. To get the returns and the gains of this portfolio, we use the *_history* data frame by first taking the dot product of the prices in each day with the weights of the portfolio. Then we calculate the returns using the *pct_change*() function used before and gains by adding 1 to the returns and taking the natural logarithm. All that is left is to plot the returns and the gains. The histograms are depicted in Figure 2 and Figure 3. Both the returns and the gains look normally distributed.

Task 4

Using the returns for the portfolio that we calculated in task 3, we can get the annual mean and variance using the standard *mean*() and *var*() functions and multiplying by 252. Using these values, we can compute the Sharpe Ratio by subtracting from the mean the risk free rate and then dividing by the standard deviation. The Sharpe Ratio for the portfolio is 1.21, which is a good one. Using the same idea, we can calculate the Sharpe Ratio from the $\_future$ data. We get 0.88, so it means our portfolio performed worse than we expected.

Next, we need to get the value at risk. We need to get the the value of our portfolio on 29th of December 2017 and on the 23rd of February 2018. There are 37 days between these dates so the variable *value_at_*0 and *value_at_*37 contain those values. We also get the distribution $G_{37} = ln(S_{37}/S_0)$, which is approximately the same as a normal distribution with mean 37 times the mean of the returns from the future data frame and variance 37 times the variance of the returns from the future data frame. Using this distribution we get the z score and with that we can compute the value at risk with the formula from the slides. We get

Figure 1: Feasible Set, Efficient Frontier and the CML

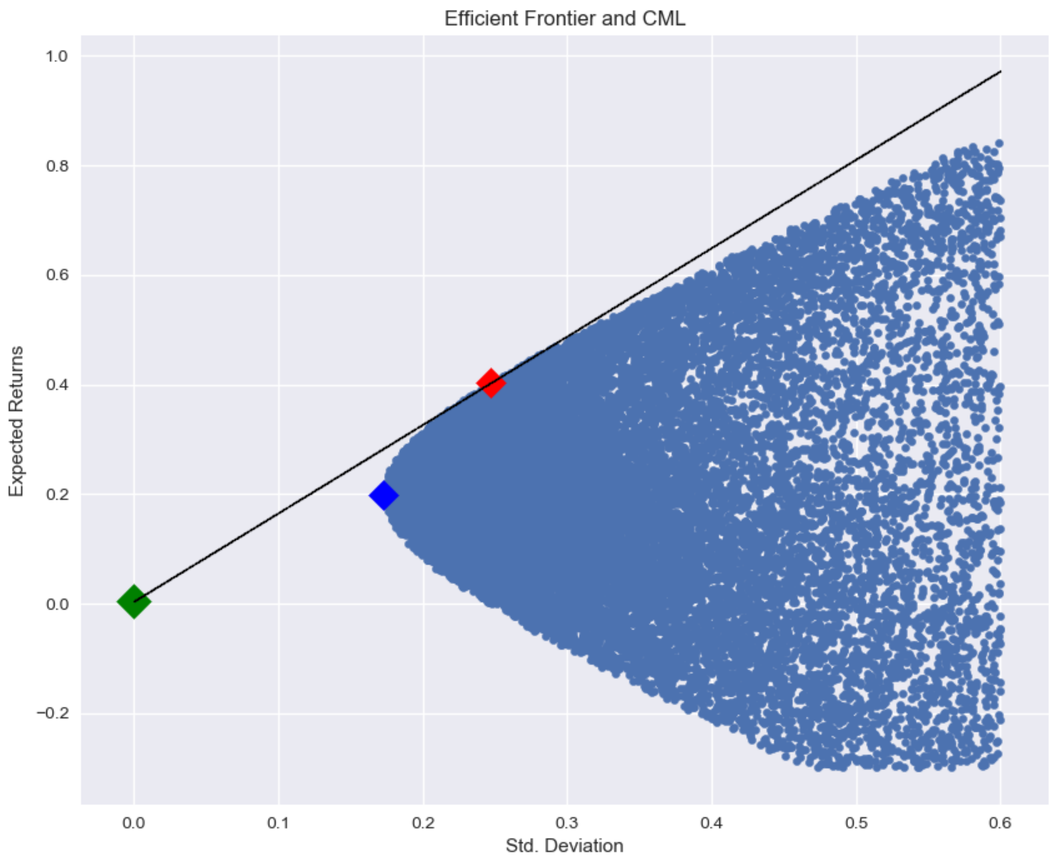


Figure 2: Returns

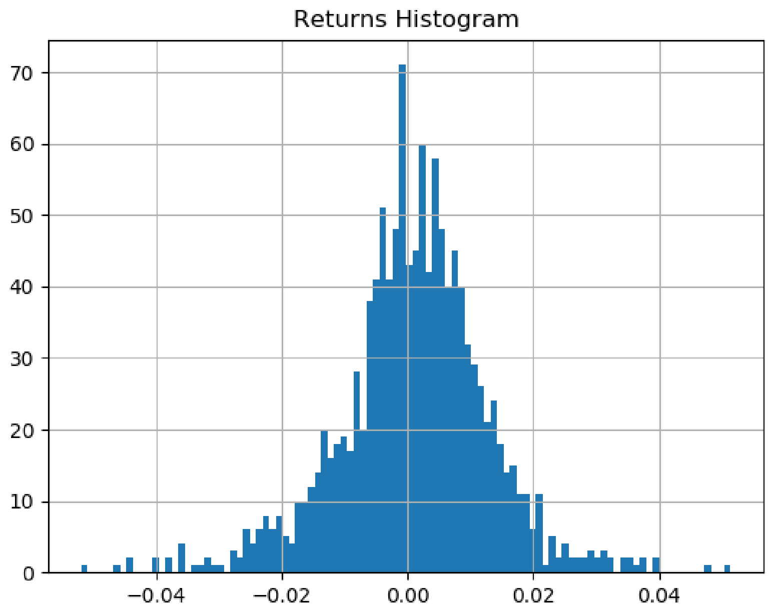
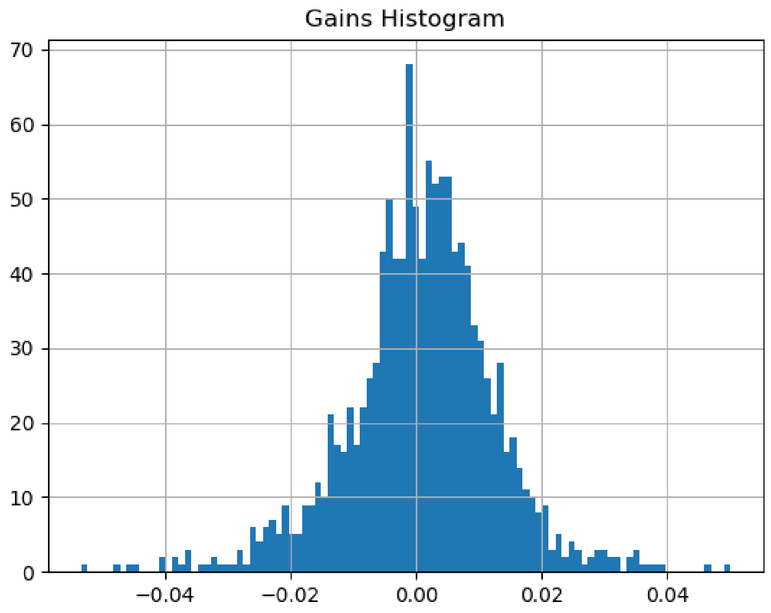


Figure 3: Gains



that $VaR = 102.77$. That means that we are 99% certain that we will not lose more than 102.77 Euros in the next 37 days.

A Appendix A

Python code for reading the CSV files

```
class Assignment2:
    def __init__(self):
        # risk free asset return
        self._rf = 0.005

        # read data
        self._history = pd.read_csv('history.csv')
        self._future = pd.read_csv('future.csv')

        # change index, calculate returns and drop the first row
        # (since we have n - 1 returns for n days)
        self._returns = \
            self._history.set_index('date').pct_change().drop('2/25/2013')

        # set the gains
        self._gains = np.log(self._returns + 1)
```

Python code for task 1

```
def task_1(self):
        print(self._returns.mean() * 252)
        print(self._returns.var() * 252)
    print(self._returns.cov() * 252)

def _mean(self, stock):
        summation = 0
        counter = 0
        for val in self._returns[stock]:
        summation += val
        counter += 1
    return summation / counter

def _var(self, stock):
        summation = 0
        counter = 0
        mean = self._mean(stock)

        for val in self._returns[stock]:
                summation += (val - mean)**2
        counter += 1

    return summation / (counter - 1)

def _cov(self, stock1, stock2):
        summation = 0
        counter = 0

        mean1 = self._mean(stock1)
        mean2 = self._mean(stock2)

        for val1, val2 in zip(self._returns[stock1], self._returns[stock2]):
        summation += (val1 - mean1) * (val2 - mean2)
        counter += 1

        return summation / (counter - 1)
```

Python code for task 2

```python
def task_2(self):
        # generate the mean return per stock (for a year)
        means = np.asarray(self._returns.mean() * 252)

        # generate the covariance matrix and its inverse;
        # also generate the row of ones
        cov_matrix = (self._returns.cov() * 252).as_matrix()
    inverse_cov_matrix = np.linalg.inv(cov_matrix)
    ones = np.asarray([1] * len(means))

    # weights, mean and var of the minimum variances portofolio
    weights_min = np.dot(
        ones, inverse_cov_matrix) / np.dot(
                np.dot(ones, inverse_cov_matrix), ones)

        print("Minimum variance portofolio weights:")
    for weight in weights_min:
        print(weight, end=' ')

        print('\n')

    min_portofolio = self.get_mean_var(weights_min, means, cov_matrix, 'min')

    print('Mean:', min_portofolio[0])
    print('Var:', min_portofolio[1])

    # weights of the tangency portofolio
    weights_tan = np.dot(
        means - self._rf, inverse_cov_matrix) / np.dot(
                np.dot(means - self._rf, inverse_cov_matrix), ones)
        print('Tangency portofolio:')

    for weight in weights_tan:
        print(weight, end=' ')

    print('\n')

        tan_portofolio = self.get_mean_var(weights_tan, means, cov_matrix, 'tan')

    print('Mean:', tan_portofolio[0])
    print('Var:', tan_portofolio[1])

    # visualize the portofolios, the effcient frontier and the CML
    num_assets = len(means)
    num_portofolios = 30000
    port_returns = []
    port_stds = []

    for _ in range(num_portofolios):
        weights = np.random.uniform(-1.0, 1.0, num_assets)
        weights /= np.sum(weights)
        port_returns.append(np.dot(weights, means))
        port_stds.append(np.sqrt(np.dot(weights, np.dot(cov_matrix, weights))))

        portofolio = {'Returns': port_returns,
                     'Stds': port_stds}

    port = pd.DataFrame(portofolio)
    mask = ((port['Returns'] <= 1.0) & (port['Returns'] >= -0.3)) & (port['Stds'] <= 0.6)
    port = port[mask]

        plt.style.use('seaborn')
    # feasible region
    ax = port.plot.scatter(x='Stds', y='Returns', figsize=(10, 8), grid=True)
    # tangency portofolio
    plt.scatter(
        x=np.sqrt(tan_portofolio[1]), y=tan_portofolio[0], c='red', marker='D', s=150)
    # minimum variance portofolio
    plt.scatter(
        x=np.sqrt(min_portofolio[1]), y=min_portofolio[0], c='blue', marker='D', s=150)
    # risk free asset
    plt.scatter(x=0, y=self._rf, c='green', marker='D', s=200)
    # capital market line
    cpm_means = []
    cpm_stds = []

    for _ in range(10000):
        std = np.random.uniform(0, 0.6)
        cpm_stds.append(std)
        cpm_means.append(self._rf + std * (
                tan_portofolio[0] - self._rf) / np.sqrt(tan_portofolio[1]))

    cpm = {'Returns': cpm_means,
            'Stds': cpm_stds}

    cpm_df = pd.DataFrame(cpm)
    cpm_df.plot.scatter(ax=ax, x='Stds', y='Returns', c='black', s=0.25)

    plt.xlabel('Std. Deviation')
    plt.ylabel('Expected Returns')
    plt.title('Efficient Frontier and CPM')
    plt.show()
```

Python code for task 3

```
def task_3(self):
        """Task 3: avearge of the min and tan portofolios"""
    # get the means
    means = np.asarray(self._returns.mean() * 252)
    # get the covariance matrix and the inverse
    cov_matrix = (self._returns.cov() * 252).as_matrix()
    inverse_cov_matrix = np.linalg.inv(cov_matrix)
    ones = np.asarray([1] * len(means))

    # weights of the minimum variances portofolio
    weights_min = np.dot(
        ones, inverse_cov_matrix) / np.dot(
            np.dot(ones, inverse_cov_matrix), ones)
    # weights of the tangency portofolio
    weights_tan = np.dot(
        means - self._rf, inverse_cov_matrix) / np.dot(
            np.dot(means - self._rf, inverse_cov_matrix), ones)
    # weights of the averaged portofolio
    weights_port = (weights_min + weights_tan) / 2

    # calculation of the returns and gains
    self._history = self._history.set_index('date')
    self._history['Returns'] = np.dot(self._history, weights_port)
    self._history = self._history.pct_change().drop('2/25/2013')
    self._history['Gains'] = np.log(self._history['Returns'] + 1)

    # histograms of the returns and gains
    self._history.hist(column='Returns', bins=100)
    plt.title('Returns Histogram')
    self._history.hist(column='Gains', bins=100)
    plt.title('Gains Histogram')
    plt.show()
```

Python code for task 4

```
def task_4(self):
    """Task 3: avearge of the min and tan portofolios"""
    # get the means
    means = np.asarray(self._returns.mean() * 252)
    # get the covariance matrix and the inverse
    cov_matrix = (self._returns.cov() * 252).as_matrix()
    inverse_cov_matrix = np.linalg.inv(cov_matrix)
    ones = np.asarray([1] * len(means))

    # weights of the minimum variances portofolio
    weights_min = np.dot(
        ones, inverse_cov_matrix) / np.dot(
            np.dot(ones, inverse_cov_matrix), ones)
    # weights of the tangency portofolio
    weights_tan = np.dot(
        means - self._rf, inverse_cov_matrix) / np.dot(
            np.dot(means - self._rf, inverse_cov_matrix), ones)
    # weights of the averaged portofolio
    weights_port = (weights_min + weights_tan) / 2

    # calculation of the returns and gains for history.csv
    self._history = self._history.set_index('date')
    self._history['Returns'] = np.dot(self._history, weights_port)

    # value at 0 (12/29/2017) needed to compute the VAR
    value_at_0 = self._history['Returns'].values[1222]

    self._history = self._history.pct_change().drop('2/25/2013')
    self._history['Gains'] = np.log(self._history['Returns'] + 1)

    # compute the mean return per year, and the variance per year for history.csv
    portfolio_mean_return_history = 252 * self._history['Returns'].mean()
    portfolio_variance_history = 252 * self._history['Returns'].var()

    # compute the sharpe ratio for history.csv
    sharpe_ratio_history = (portfolio_mean_return_history - self._rf) / sqrt(portfolio_va
    print("Sharpe ratio for the portfolio (history.csv) = ", sharpe_ratio_history)

    # calculate returns and gains for future.csv
    self._future = self._future.set_index('date')
    self._future['Returns'] = np.dot(self._future, weights_port)

    # value at 37 (after 37 days) to compute the var (date is 02/23/18)
    value_at_37 = self._future['Returns'].values[36]

    self._future = self._future.pct_change().drop('1/2/2018')
    self._future['Gains'] = np.log(self._future['Returns'] + 1)

    # compute the mean return per year, and the variance per year for future.csv
    portfolio_mean_return_future = 252 * self._future['Returns'].mean()
    portfolio_variance_future = 252 * self._history['Returns'].var()

    # compute the sharpe ratio for future.csv
    sharpe_ratio_future = (portfolio_mean_return_future - self._rf) / sqrt(portfolio_vari

    print("Sharpe ratio for the portfolio (future.csv) = ", sharpe_ratio_future)

    # compute the value at risk
    # note that we already compute the value of the portfolio at time 0 (12/29/17)
    # and at time 37 (02/23/18)
    G_37 = np.log(value_at_37 / value_at_0)
    mean_normal_dist = 37 * portfolio_mean_return_future
    var_normal_dist = 37 * portfolio_variance_future

    z_score = st.norm.ppf(0.99, mean_normal_dist, sqrt(var_normal_dist))

    print("mean of the normal distribution = ", mean_normal_dist)
    print("standard deviation of the normal distribution = ", sqrt(var_normal_dist))
    print("z_epsilon = ", z_score)

    VAR = value_at_0 * (1 - np.exp((-1) * z_score * sqrt(portfolio_variance_future) + por

    # compute the value at risk
    print("value at risk = ", VAR)
```