Quantitative Portfolio Management

Assignment #5

Raman Uppal
EDHEC Business School

November 2023

Instructions for each assignment . . . I

- Assignment #1 should be done individually.
- The other assignments are to be done in groups of 4 or 5 students.
  - This means that groups of 1, 2, 3, 6, etc. are not allowed.
  - Diversity in groups is strongly encouraged
    (people from different countries, different genders, different finance knowledge, and different coding ability, etc.)

Instructions for each assignment ... II

- Each assignment should be emailed as a Jupyter file
  - To Raman.Uppal@edhec.edu
  - The subject line of the email should be: "QPM: Assignment n," where $n = \{1, 2, \ldots, 8\}$.
  - Assignment n is due before Lecture n, where $n = \{1, 2, \ldots, 8\}$.
  - Assignments submitted late will not be accepted (grade = 0), so please do not email me assignments after the deadline.

Instructions for each assignment . . . III

- The Jupyter file should include the following (use Markdown):
  - Section "0" with information about your submission:
    - Line 1: QPM: Assignment n
    - Line 2: Group members: listed alphabetically by last name, where the last name is written in CAPITAL letters
    - Line 3: Any comments/challenges about the assignment
  - Section "k" where $k = \{1, 2, \ldots\}$.
    - First type Question k of Assignment n.
    - Then, below the question, provide your answer.
    - Your code should include any packages that need to be imported.

Initial step to prepare the data for this assignment

- The data we will be using is the same that we used for the previous assignment. For convenience, I have typed again the instructions.
  - Make sure you have already imported "pandas" and "yfinance."
  - Download from Wikipedia (or any other source) a table that lists the companies that comprise the S&P 500. (See "Helpful links" provided at the end of the assignment.)
  - From this table, extract the list of ticker symbols.
  - Set the start date and end date to be
    - start_date = "2000-01-01"
    - end_date = "2022-12-31"
  - Build a dataframe that contains the stock prices for the S&P 500 companies. (If there are errors for some company names, it is fine to ignore the company names with errors.)
  - Drop the columns that have only "NaN" entries.
  - Drop also the companies with more than 100 missing observations.

Questions for Assignment 5 . . . I

- Select the following 10 companies (these are the first 10 companies with no missing data):
  "MMM","AOS","ABT","ADM","ADBE","ADP","AES","AFL","A","AKAM"
- So, just like for the last assignment, our dataset for this assignment will consist of monthly returns for these 10 companies.
- To reduce the work required for this assignment, please continue to assume that the risk-free rate of return is zero.

Questions for Assignment 5 . . . II

Q5.1 Choose the estimation window to be $T^{\text{est}} = 60$ months of monthly returns. Call this the estimation sample. Use the estimation sample to compute the following two portfolio strategies:

a. mean-variance portfolio with nonnegativity constraints on the weights (when a risk-free rate is available, and set this rate to 0); we will refer to this portfolio as "MVP-C."

b. global minimum variance (GMV) portfolio with nonnegativity constraints; we will refer to this portfolio as "GMV-C".

- For each of the two portfolios, rescale the weights in the risky assets so that they sum to 1; that is, you are "fully invested" in just the risky assets.

Questions for Assignment 5 ...III

- So, compared to the previous assignment, the only change is that
  - we have replaced the unconstrained strategies
  - by strategies that have nonnegativity constraints on the weights, which rule out short selling.
- The remaining instructions are the same as for last week.

Q5.2 Now use a rolling window of $T^{\text{est}} = 60$ months to estimate the portfolio weights for the two strategies listed above for each of the $T - T^{\text{est}}$ months. That is, repeat the calculations of the previous question for all the dates *after* the first 60 months.

Q5.3 Use the time-series of portfolios weights for each of the two portfolio strategies, to compute the out-of-sample portfolio returns. That is, for each of the two portfolio strategies that you estimate at each date t, compute its out-of-sample return in month $t + 1$.

Q5.4 Now, compute the Sharpe ratio of the out-of-sample returns for the two portfolio strategies. Which strategy has the higher Sharpe ratio?

Helpful hints

- Helpful links for information on downloading S&P 500 ticker symbols.
  - from Danny Groves
  - from GitHub
- Finally, please save the data you have downloaded and created for these ten companies because we will be using it again.

Discussion of Assignment 5: Initial setup

- We start by loading the libraries we will need.

Code to load required libraries

```
import pandas as pd
import yfinance as yf
import numpy as np
import pandas_datareader as pdr

# only additional package, relative to Assignment 4
from scipy.optimize import minimize
```

Code to download the data

Code to download the data

```
# List of tickets for which we will download the data
tickers=["MMM","AOS","ABT","ADM","ADBE","ADP","AES","AFL","A","AKAM"]

# Set the start and end dates
start_date = "2000-01-01"
end_date = "2022-12-31"

# Create an empty dataframe
stock_prices = pd.DataFrame()

# Download the data
for ticker in tickers:
    price = yf.download(ticker,start=start_date,end=end_date)
    stock_prices[ticker] = price["Adj Close"]

# Change the index column to be the date
stock_prices.index = pd.to_datetime(stock_prices.index)
```

Code to construct monthly returns

Code to construct monthly returns

```
# Extract the stock price at the end of each month
month_stock_prices = stock_prices.resample("1M").last()
month_stock_prices.index = month_stock_prices.index.date

# Compute returns
month_return=np.log(month_stock_prices/month_stock_prices.shift(1))
month_return=month_return.dropna() # delete the first row - without data
```

Discussion of Assignment: Q5.1

Q5.1 Choose the estimation window to be $T^{\text{est}} = 60$ months of monthly returns. Call this the estimation sample. Use the estimation sample to compute the following two portfolio strategies:

a. mean-variance portfolio with nonnegativity constraints on the weights (when a risk-free rate is available, and set this rate to 0); we will refer to this portfolio as "MVP-C."

b. global minimum variance (GMV) portfolio with nonnegativity constraints; we will refer to this portfolio as "GMV-C".

- For each of the two portfolios, rescale the weights in the risky assets so that they sum to 1; that is, you are "fully invested" in just the risky assets.

Code for Q5.1: MVP-C

Code to construct constrained MVP portfolio

```
# Step 1 of 2: Define objective to be minimized (portfolio variance)
def objective_MVP(weights, mu, V):
    portfolio_return = weights.T @ mu
    portfolio_variance = weights.T @ V @ weights
    sharpe = portfolio_return/np.sqrt(portfolio_variance)
    return -sharpe
```

Code for Q5.1: MVP-C (continued)

Code to construct constrained MVP portfolio

```
# Step 2 of 2: Calculate the optimal weight of MVP-C portfolio:
def MVP_C_w(returns):
    mu = returns.mean()
    V = returns.cov()

    # Define the constraint that the weights sum to 1
    constraints = ({'type': 'eq', 'fun': lambda weights: np.sum(weights)
      - 1})
    # Define the bounds for each weight (0 <= weight <= 999)
    bounds = tuple((0, 999) for _ in range(len(V)))
    # Define the initial guess for weights
    initial_weights = np.ones(len(V)) / len(V)

    # Use scipy.optimize.minimize to find the optimal weights
    result = minimize(objective_MVP, initial_weights, args = (mu,V,),
     method = 'SLSQP',bounds = bounds, constraints = constraints)

    # Extract the optimal weights
    optimal_weights = result.x

    return optimal_weights
```

Code for Q5.1: GMV-C

- Having defined the mean-variance portfolio with short-sale constraints (MVP-C);
- We now define the global minimum variance portfolio with short-sale constraints (GMV-C).

Code to construct constrained GMV portfolio

```
# Step 1 of 2: Define the objective to be minimized (portfolio variance)
def objective_GMV(weights, V):
    portfolio_variance = weights.T @ V @ weights
    return portfolio_variance
```

Code for Q5.1: GMV-C (continued)

Code to construct constrained GMV portfolio

```
# Step 2 of 2: Calculate the optimal weight of GMV-C portfolio:
def GMV_C_w(returns):
    V = returns.cov()

    # Define the constraint that the weights sum to 1
    constraints = ({'type': 'eq', 'fun': lambda weights: np.sum(weights)
      - 1})
    # Define the bounds for each weight (0 <= weight <= 999)
    bounds = tuple((0, 999) for _ in range(len(V)))
    # Define the initial guess for the weights
    initial_weights = np.ones(len(V)) / len(V)

    # Use scipy.optimize.minimize to find the optimal weights
    result = minimize(objective_GMV, initial_weights, args=(V,), method=
     'SLSQP',bounds=bounds, constraints=constraints)

    # Extract the optimal weights
    optimal_weights = result.x

    return optimal_weights
```

Discussion of Assignment: Q5.2

Q5.2 Now use a rolling window of $T^{\text{est}} = 60$ months to estimate the portfolio weights for the two strategies listed above for each of the $T - T^{\text{est}}$ months. That is, repeat the calculations of the previous question for all the dates *after* the first 60 months.

Code for Q5.2: MVP-C

Code for rolling-window analysis of MVP-C

```
MVP_C_weights = pd.DataFrame(index=month_return.index, columns=
     month_return.columns)

for i in month_return.index[60:]:            #start from the 61th month
    start_date = i - pd.DateOffset(months=60)
    end_date = i-pd.DateOffset(months=1)
    start_date = pd.to_datetime(start_date).date()
    end_date=pd.to_datetime(end_date).date()

    df = month_return.loc[start_date:end_date]

    # calculate MVP-C portfolio for each rolling window
    MVP_C_weights.loc[i] = MVP_C_w(df)

MVP_C_weights = MVP_C_weights.dropna()
```

Code for Q5.2: GMV-C

Code for rolling-window analysis of GMV-C

```
GMV_C_weights = pd.DataFrame(index = month_return.index, columns =
     month_return.columns)

for i in month_return.index[60:]:#start from the 61th month
    start_date = i - pd.DateOffset(months = 60)
    end_date = i-pd.DateOffset(months = 1)
    start_date = pd.to_datetime(start_date).date()
    end_date = pd.to_datetime(end_date).date()

    df = month_return.loc[start_date:end_date]

    # calculate GMV-C portfolio for each rolling window
    GMV_C_weights.loc[i] = GMV_C_w(df)

GMV_C_weights=GMV_C_weights.dropna()
```

Discussion of Assignment: Q5.3

Q5.3 Use the time-series of portfolios weights for each of the two portfolio strategies, to compute the out-of-sample portfolio returns. That is, for each of the two portfolio strategies that you estimate at each date t, compute its out-of-sample return in month $t+1$.

Code for computing portfolio return at date $t+1$

```
# Because Rf is assumed to be 0,
#  the portfolio return equals the return on the risky-asset portfolio

# Return on MVP-C portfolio with non-negative constraints:
MVP_C_return = (MVP_C_weights * month_return.iloc[60:]).sum(axis = 1)

# Return on GMV-C portfolio with non-negative constraints:
GMV_C_return = (GMV_C_weights * month_return.iloc[60:]).sum(axis = 1)
```

Discussion of Assignment: Q5.4

Q5.4 Now, compute the Sharpe ratio of the out-of-sample returns for the two portfolio strategies. Which strategy has the higher Sharpe ratio?

Code to compute the Sharpe ratio

```
def sharpe_ratio(m_return):                # input is monthly return
    Rf=0
    m_mean_return = m_return.mean()        # monthly return mean
    m_vol = m_return.std()                 # monthly return volatility

    y_mean_return = m_mean_return * 12     # transform to yearly mean
    y_vol = m_vol * np.sqrt(12)            # transform to yearly volatility

    SR = (y_mean_return-Rf)/y_vol

    return SR
```

Discussion of Assignment: Q5.4 (continued)

Code for printing the output

```
# Sharpe ratio for MVP-C portfolio with constraints:
print(f'Sharpe Ratio of MVP-C portfolio with non-negative constraints is
      {sharpe_ratio(MVP_C_return)}.')

# Sharpe ratio for GMV-C portfolio with constraints:
print(f'Sharpe Ratio of GMV-C portfolio with non-negative constraints is
      {sharpe_ratio(GMV_C_return)}.')
```

```
Sharpe Ratio of MVP-C portfolio is 0.6314017700840544.
Sharpe Ratio of GMV-C portfolio is 0.6790106272989513.
```

- From the above result, we conclude that, with short-sale constraints, the mean-variance portfolio performs almost as well as the global minimum-variance portfolio – at least for the companies and sample period considered.

End of assignment