Quantitative Portfolio Management

Assignment #3

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.

Questions for Assignment 3 . . . I

Q3.1 Prepare the data for this assignment.

- Make sure you have already imported "pandas" and "yfinance" into Python.
- 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 (short names for all the companies).
- 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 company names that have more than 100 missing observations.

Questions for Assignment 3 . . . II

Q3.2 Compute the log returns for the companies in your dataset.

Q3.3 Compute the annualized mean return, volatility, and Sharpe ratios for these companies in your dataset.

Q3.4 Would it make sense to choose portfolio weights based only on the Sharpe ratios of the stocks in your dataset? Explain the reasons for your answer.

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

Discussion of Assignment 3: Initial setup

Load packages and initial definitions

```
# execute only once from the terminal/prompt
# pip install yfinance

import pandas as pd
import yfinance as yf
```

Discussion of Assignment: Q3.1

Q3.1 Prepare the data for this assignment, as per the instructions in the question.

Code for Q3.1

```
# get the table from Wikipedia that has the list of company names
table = pd.read_html('https://en.wikipedia.org/wiki/List_of_S%26
    P_500_companies')

# extract only the table
df = table[0]
df
```

- The output is shown on the next page.

Output for Q3.1

	Symbol	Security	GICS Sector	GICS Sub-Industry	Headquarters Location	Date added	CIK	Founded
0	MMM	3M	Industrials	Industrial Conglomerates	Saint Paul, Minnesota	1957 – 03–04	66740	1902
1	AOS	A. O. Smith	Industrials	Building Products	Milwaukee, Wisconsin	2017 – 07–26	91142	1916
2	ABT	Abbott	Health Care	Health Care Equipment	North Chicago, Illinois	1957 – 03–04	1800	1888
3	ABBV	AbbVie	Health Care	Pharmaceuticals	North Chicago, Illinois	2012 – 12 – 31	1551152	2013 (1888)
4	ACN	Accenture	Information Technology	IT Consulting & Other Services	Dublin, Ireland	2011 – 07–06	1467373	1989
...	...	...	...	...	...	...	...	...
498	YUM	Yum! Brands	Consumer Discretionary	Restaurants	Louisville, Kentucky	1997 – 10–06	1041061	1997
499	ZBRA	Zebra Technologies	Information Technology	Electronic Equipment & Instruments	Lincolnshire, Illinois	2019 – 12–23	877212	1969
500	ZBH	Zimmer Biomet	Health Care	Health Care Equipment	Warsaw, Indiana	2001 – 08–07	1136869	1927
501	ZION	Zions Bancorporation	Financials	Regional Banks	Salt Lake City, Utah	2001 – 06–22	109380	1873
502	ZTS	Zoetis	Health Care	Pharmaceuticals	Parsippany, New Jersey	2013 – 06–21	1555280	1952

503 rows × 8 columns

Discussion of Assignment: Q3.1 (contd.)

Code for Q3.1 (contd.)

```
# Now select only tickers
stockdata = df['Symbol'].to_list()

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

# Create an empty list to store the stock prices (improves speed)
dataframes = []

# Download from Yahoo the monthly prices
for ticker in stockdata:
    Stock_data = yf.download(ticker, start=start_date, end=end_date)
    dataframes.append(Stock_data["Adj Close"])

# Use concatenate to build the dataframe
SP500P = pd.concat(dataframes, axis=1)

# Get monthly data
SP500P.index = pd.to_datetime(SP500P.index)
SP500m = SP500P.resample('1M').last()
SP500m.columns = stockdata
SP500m
```

- The output is shown on the next page.

Output for Q3.1 (second part)

	MMM	AOS	ABT	ABBV	ACN	ADM	ADBE	ADP	AES	AFL	...	WTW
Date												
2000−01 − 31	24.575970	2.151636	8.490210	NaN	NaN	6.378316	13.668242	22.875603	28.590223	6.792754	...	NaN
2000−02 − 29	23.298599	1.879277	8.604282	NaN	NaN	5.485136	25.319601	21.006977	29.906002	5.728642	...	NaN
2000−03 − 31	23.397671	1.960985	9.174640	NaN	NaN	5.621411	27.638035	23.314587	28.099588	7.138769	...	NaN
2000−04−30	22.885803	2.254298	10.072313	NaN	NaN	5.416999	30.027847	26.002407	32.091526	7.647978	...	NaN
2000−05 − 31	22.801466	2.302261	10.661910	NaN	NaN	6.538914	27.948402	26.576204	31.132561	8.112567	...	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
2022−08 − 31	117.727753	55.108891	100.164154	127.937202	282.691559	86.067314	373.440002	238.238312	24.496204	58.022301	...	203.015488
2022−09−30	104.615334	47.425861	94.416794	127.699318	252.154648	78.781601	275.200012	221.431122	21.753012	54.878048	...	198.018875
2022−10 − 31	119.091064	53.789486	96.999535	140.715378	279.460022	94.968803	318.500000	236.614792	25.330849	63.578461	...	215.037827
2022−11 − 30	120.657944	59.641720	105.470078	154.921463	296.223663	95.884819	344.929993	258.582672	28.003372	70.649330	...	242.581497
2022−12 − 31	114.863068	56.205009	107.636734	155.334778	262.666809	91.311844	336.529999	234.967575	27.848444	70.659157	...	241.844833

276 rows × 503 columns

Discussion of Assignment: Q3.1 (contd.)

Code for Q3.1 (contd.)

```
# Drop columns that have only NaNs
SP500m1 = SP500m.dropna(axis=1, how='all')
SP500m1
```

- The output is shown below

	MMM	AOS	ABT	ABBV	ACN	ADM	ADBE	ADP	AES	AFL	...	WTW
Date												
2000–01–31	24.575970	2.151636	8.490210	NaN	NaN	6.378316	13.668242	22.875603	28.590223	6.792754	...	NaN
2000–02–29	23.298599	1.879277	8.604282	NaN	NaN	5.485136	25.319601	21.006977	29.906002	5.728642	...	NaN
2000–03–31	23.397671	1.960985	9.174640	NaN	NaN	5.621411	27.638035	23.314587	28.099588	7.138769	...	NaN
2000–04–30	22.885803	2.254298	10.072313	NaN	NaN	5.416999	30.027847	26.002407	32.091526	7.647978	...	NaN
2000–05–31	22.801466	2.302261	10.661910	NaN	NaN	6.538914	27.948402	26.576204	31.132561	8.112567	...	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
2022–08–31	117.727753	55.108891	100.164154	127.937202	282.691559	86.067314	373.440002	238.238312	24.496204	58.022301	...	203.015488
2022–09–30	104.615334	47.425861	94.416794	127.699318	252.154648	78.781601	275.200012	221.431122	21.753012	54.878048	...	198.018875
2022–10–31	119.091064	53.789486	96.999535	140.715378	279.460022	94.968803	318.500000	236.614792	25.330849	63.578461	...	215.037827
2022–11–30	120.657944	59.641720	105.470078	154.921463	296.223663	95.884819	344.929993	258.582672	28.003372	70.649330	...	242.581497
2022–12–31	114.863068	56.205009	107.636734	155.334778	262.666809	91.311844	336.529999	234.967575	27.848444	70.659157	...	241.844833

276 rows × 499 columns

Discussion of Assignment: Q3.2

Q3.2 Compute the log returns for the companies in your dataset.

Code for Q3.2

```
import numpy as np

# I decided to filter the columns with fewer than 100 observations
non_nan_counts = SP500m1.count()

SP500m1_filt = SP500m1.loc[:, non_nan_counts >= 100]
SP500m1_filt = SP500m1_filt.dropna()

# Compute the log-returns
log_return = np.log(SP500m1_filt / SP500m1_filt.shift(1)).dropna()
log_return
```

- The output is shown on the next page.

Output for Q3.2

	MMM	AOS	ABT	ABBV	ACN	ADM	ADBE	ADP	AES	AFL	...	WTw	GWW
Date													
2014−10−31	0.081887	0.123641	0.052267	0.101546	0.011019	-0.083637	0.013352	0.114379	-0.004052	0.025090	...	-0.021238	-0.019461
2014−11−30	0.045631	0.010813	0.020886	0.086591	0.062248	0.118715	0.049523	0.046120	-0.014317	0.006598	...	0.052391	-0.000273
2014−12−31	0.026080	0.044959	0.011393	-0.055867	0.033936	-0.012992	-0.013390	-0.021100	-0.007235	0.022513	...	0.054648	0.036802
2015−01−31	-0.012370	0.051816	-0.000512	-0.073493	-0.060940	-0.109000	-0.035991	-0.010127	-0.111348	-0.067894	...	-0.034279	-0.077678
2015−02−28	0.044580	0.062273	0.056674	0.002482	0.068969	0.032439	0.120362	0.073675	0.059565	0.093032	...	0.097198	0.009053
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2022−08−31	-0.131211	-0.114056	-0.058554	-0.065138	-0.059878	0.064675	-0.093692	0.013552	0.135723	0.042781	...	-0.000532	0.023946
2022−09−30	-0.118085	-0.150143	-0.059091	-0.001861	-0.114314	-0.088450	-0.305259	-0.073160	-0.118766	-0.055714	...	-0.024920	-0.126109
2022−10−31	0.129598	0.125910	0.026987	0.097061	0.102817	0.186869	0.146124	0.066322	0.152271	0.147161	...	0.082452	0.177749
2022−11−30	0.013071	0.103277	0.083721	0.096179	0.058256	0.009599	0.079719	0.088782	0.100302	0.105454	...	0.120524	0.034431
2022−12−31	-0.049219	-0.059349	0.020335	0.002664	-0.120228	-0.048867	-0.024654	-0.095768	-0.005548	0.000139	...	-0.003041	-0.080799

99 rows × 474 columns

Discussion of Assignment: Q3.3

Q3.3 Compute the annual mean return, volatility, and Sharpe ratios for these companies in your dataset.

- Note that for computing the Sharpe ratio you need to know the risk-free interest rate. There are several possibilities/alternatives.
- In theory, what you need is the rate that matches the maturity of your investment horizon, which in our case is one month.
- In practice, you could have done several things:
  - Fix r_f equal to some value (e.g., 0 or 0.10 per month, etc.) ... not a great approach because r_f is changing over time.
  - Download r_f from Kenneth R. French Data Library ... better choice
  - Obtain the risk-free rate from some other source.

Code for Q3.3

Code for Q3.3

```
import urllib.request
import zipfile

# set the URL
ff_url = "https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/ftp/F-
    F_Research_Data_Factors_daily_CSV.zip"

# Download the file and save it with the name fama_french.zip
urllib.request.urlretrieve(ff_url,'fama_french.zip')
zip_file = zipfile.ZipFile('fama_french.zip', 'r')

# Next we extact the file data and call it ff_factors.csv
zip_file.extractall()

# Make sure to close the file after extraction
zip_file.close()

# import data
ff_factors = pd.read_csv('F-F_Research_Data_Factors_daily.CSV', skiprows =
    3,index_col = 0)

# remove the last row
ff_factors = ff_factors.iloc[:-1]

# and here is the tail of the data
ff_factors.tail()
```

Output for Q3.3

Date	Mkt-RF	SMB	HML	RF
20230825	0.65	-0.07	-0.58	0.02
20230828	0.63	-0.01	0.41	0.02
20230829	1.50	0.01	-0.11	0.02
20230830	0.41	0.24	-0.45	0.02
20230831	-0.08	-0.22	0.21	0.02

Code for Q3.3 (contd.)

Code for Q3.3 (contd.)

```
# We now modify the index of the pd dataframe as a datetime index
ff_factors.index = pd.to_datetime(ff_factors.index, format= '%Y%m%d')

# and take the monthly data and print the tail
ff_factors = ff_factors.resample('1M').last()
ff_factors.tail()
```

2022-08-31	0.008
2022-09-30	0.009
2022-10-31	0.011
2022-11-30	0.014
2022-12-31	0.016

Freq: M, Name: RF, dtype: float 64

Code for Q3.3 (contd.)

Code for Q3.3: Define Sharpe ratio

```
# Define a function that, given as input the returns and the rf rate,
# (same length or rf constant), returns to us the following:
# annual mean, std, and Sharpe Ratio (in %).
# Set the default value for the rf rate is zero.

def SharpeRatio(ret,rf=0):
    mu = np.mean(ret-rf)*12
    std = ret.std()*np.sqrt(12)
    return mu, std, mu/std*100

# Apply the function to our dataframe
results = log_return.apply(SharpeRatio, rf=rf)
results = results.T

# Rename the columns
results.columns = ['Mean Return', 'Standard Deviation', 'Sharpe Ratio']
results
```

Output for Q3.3

Name	Mean Return	Standard Deviation	Sharpe Ratio
MMM	-0.029103	0.211114	-13.785432
AOS	0.083159	0.274018	30.347926
ABT	0.097458	0.202008	48.244432
ABBV	0.127465	0.263616	48.352684
ACN	0.122693	0.227667	53.891446
...			
YUM	0.089754	0.233236	38.482039
ZBRA	0.116548	0.379295	30.727526
ZBH	0.001142	0.260352	0.438800
ZION	0.044667	0.319243	13.991472
ZTS	0.134589	0.216714	62.104551

474 rows $\times$ 3 columns

Discussion of Assignment: Q3.4

Q3.4 Would it make sense to choose portfolio weights based only on the Sharpe ratios of the stocks in your dataset? Explain the reasons for your answer.

Answer

- No, it does not make sense to choose portfolio weights based only on the Sharpe ratio of the stocks.
- In addition to the expected return and volatility of a stock, we also care about its correlation with the other assets in the portfolio.
- It is the correlation (or covariance) that determines how a stock contributes to the riskiness of the portfolio.

End of questions