assignment1

March 22, 2023

0.1 MET CS677 Data Science with Python - Assignment 1

0.1.1 Alan Szmyt

Built with Python 3.10.6 In this assignment, you will use Python to analyze the distribution of returns and a number of trading strategies. This assignment has to be done in Python only (no NumPy or Pandas) except for generating daily returns data and saving it to a file. As part of the preliminary assignment, you have generated a daily stock file for your stock and a daily file "SPY.csv" for the S&P-500 (using the symbol "SPY"). For both files, you have data for **2015 - 2019** (5 years). In this assignment, you will investigate 2 sets of questions concerning daily returns. In this assignment, you will investigate 2 sets of questions concerning daily returns: 1. If you buy a stock for just one day, what is the best day of the week to do so? Specifically, you will analyze the daily returns for each day of the week. A "Monday" return is the percent change in (adjusted) closing price from the previous Friday to Monday. A "Tuesday" return is the percent change in price from Monday to Tuesday and so on. 2. Comparison of returns if you have an "oracle" that can predict the future behavior of your stock price.

We start with introducing some notation. Let $R = r1, ..., rn$ denote your daily returns for your stock for n days. The mean of the daily returns.

$$\mu(R) = \frac{r_1 + ... + r_n}{n} \tag{1}$$

To compute the standard deviation $\sigma(R)$ we can use:

$$\sigma^2(R) = \frac{r_1^2 + ... + r_n^2}{n} - \mu^2(R) \tag{2}$$

Let us split the daily returns R into two sets. 1. R^-: all negative returns 2. R^+: all non-negative returns

Finally, let $|R^-|$ denote the number of days with negative returns and let $|R^+|$ denote the number of days with non-negative returns.

Current Python Version:

[1]:
```
!python --version
```

```
Python 3.10.6
```

[2]:
```
import os
from pathlib import Path
from tabulate import tabulate
```

```
from IPython.display import display, HTML, Markdown, Latex
from collections import Counter
from assignment1 import Assignment1Error, read_ticker_file, StockDataEntry,␣
 ↪StockData, mean, StockDataTable, float_to_currency, difference,␣
 ↪percent_change, Oracle, LossGain, RevengeScenarioA, RevengeScenarioB,␣
 ↪RevengeScenarioC

# Directory path to the 'resources' folder.
resources: Path = Path(os.path.abspath('')).joinpath("resources").resolve()
```

[3]:
```
# Sony stock ticker file path and ticker abbreviation.
sony_ticker: str = "SONY"
sony_ticker_file: Path = resources.joinpath("SONY.csv")
```

[4]:
```
# Parse Sony stock data from csv file to a StockData instance.
sony_stock_data: StockData = StockData.from_ticker_file(
    sony_ticker_file, sony_ticker
)
```

```
Opening file /home/alan/src/bu/cs677/assignment1/assignment1/resources/SONY.csv
for ticker: SONY
```

[5]:
```
# S&P-500 stock ticker file path and ticker abbreviation.
spy_ticker: str = "SPY"
spy_ticker_file: Path = resources.joinpath("SPY.csv")
```

[6]:
```
# Parse S&P-500 stock data from csv file to a StockData instance.
spy_stock_data: StockData = StockData.from_ticker_file(
    spy_ticker_file, spy_ticker
)
```

```
Opening file /home/alan/src/bu/cs677/assignment1/assignment1/resources/SPY.csv
for ticker: SPY
```

0.1.2 Question #1

1. For each of the 5 years, compute the mean and standard deviation for the sets R, R− and R+ of daily returns for your stock for each day of the week.

[7]:
```
# Print the parsed Sony stock data with pretty format.
# sony_stock_data.pretty_print()
```

2. Summarize your results in the table as shown below (5 tables total).

[8]:
```
# Get stock data for each year in the main dataset.
sony_stock_data_per_year: list[StockData] = sony_stock_data.
 ↪stock_data_for_years()
```

[9]:
```
# Display each year as a table.
for data in sony_stock_data_per_year:
    table: StockDataTable = data.as_table()
    display(Markdown(f"{table.title}<br>{tabulate(table.table_data, table.
 ↪latex_headers, tablefmt='latex_raw')}"))
```

Data Table for SONY in 2016

Day	$\mu(R)$	$\sigma(R)$	|R− |	$\mu(R-)$	$\sigma(R-)$	|R+ |	$\mu(R+)$	$\sigma(R+)$
Monday	0.00024875	0.0161409	22	-0.0131527	0.00989557	24	0.0125334	0.00989557
Tuesday	0.00360813	0.0205048	22	-0.0145336	0.0148977	30	0.0169121	0.0148977
Wednesday	-0.000211196	0.021931	27	-0.0154648	0.0146296	25	0.0162627	0.0146296
Thursday	-0.00236061	0.0182095	26	-0.0152625	0.0107077	25	0.0110574	0.0107077
Friday	0.00239639	0.0330681	29	-0.014616	0.0383568	22	0.0248218	0.0383568

Data Table for SONY in 2017

Day	$\mu(R)$	$\sigma(R)$	|R− |	$\mu(R-)$	$\sigma(R-)$	|R+ |	$\mu(R+)$	$\sigma(R+)$
Monday	0.00133615	0.0121869	23	-0.00734599	0.00895061	23	0.0100183	0.00895061
Tuesday	0.00200117	0.0187979	27	-0.00711846	0.0229975	24	0.0122607	0.0229975
Wednesday	0.00158795	0.0118637	23	-0.00847699	0.00855503	29	0.00957049	0.00855503
Thursday	0.000887444	0.0152878	25	-0.0101982	0.0117666	26	0.0115467	0.0117666
Friday	0.00404438	0.0112905	15	-0.00824009	0.00797247	36	0.00916291	0.00797247

Data Table for SONY in 2018

Day	$\mu(R)$	$\sigma(R)$	|R− |	$\mu(R-)$	$\sigma(R-)$	|R+ |	$\mu(R+)$	$\sigma(R+)$
Monday	-0.00161914	0.0198012	26	-0.0149583	0.0107943	22	0.0141453	0.0107943
Tuesday	0.000735536	0.0175744	24	-0.0133905	0.0119715	27	0.013292	0.0119715
Wednesday	0.00495346	0.0165974	17	-0.0136767	0.00968424	33	0.0145508	0.00968424
Thursday	-0.00171743	0.0169815	24	-0.0153328	0.00800603	27	0.0103851	0.00800603
Friday	-6.99568e-05	0.0218275	25	-0.0153183	0.01258	26	0.0145919	0.01258

Data Table for SONY in 2019

Day	$\mu(R)$	$\sigma(R)$	|R− |	$\mu(R-)$	$\sigma(R-)$	|R+ |	$\mu(R+)$	$\sigma(R+)$
Monday	0.00340684	0.0163097	20	-0.00815489	0.015561	28	0.0116652	0.015561
Tuesday	9.70401e-06	0.0154279	26	-0.0112311	0.00978953	26	0.0112506	0.00978953
Wednesday	0.00133888	0.0164509	22	-0.0126673	0.00819523	29	0.0119643	0.00819523
Thursday	-0.000572659	0.0156412	21	-0.0145989	0.00908976	29	0.00958427	0.00908976
Friday	0.00345786	0.0211362	21	-0.0124346	0.0155949	30	0.0145826	0.0155949

Data Table for SONY in 2020

Day	$\mu(R)$	$\sigma(R)$	|R− |	$\mu(R-)$	$\sigma(R-)$	|R+ |	$\mu(R+)$	$\sigma(R+)$
Monday	0.00169048	0.0234772	16	-0.0215908	0.0127312	32	0.0133311	0.0127312
Tuesday	0.00352212	0.0210218	26	-0.0119665	0.0186082	26	0.0190107	0.0186082
Wednesday	0.00299416	0.0197988	18	-0.0181206	0.0119293	34	0.0141725	0.0119293
Thursday	-0.000298297	0.0224588	27	-0.0149837	0.0126194	24	0.0162228	0.0126194
Friday	0.00093815	0.0189705	24	-0.0136499	0.0148501	25	0.0149427	0.0148501

3. Are there more days with negative or non-negative returns?

[10]:
```
# Separate lists of all negative and non-negative daily returns entries.
negative_days: list[StockDataEntry] = sony_stock_data.negative_daily_returns
non_negative_days: list[StockDataEntry] = sony_stock_data.
 ↪non_negative_daily_returns

overall_return_status: str = sony_stock_data.overall_return_status
print(overall_return_status)
```

```
Total Negative Days: 576
Total Non-Negative Days: 682
There are more non-negative days for this dataset!
```

4. Does your stock lose more on a "down" day than it gains on an "up" day?

[11]:
```
# Compare the overall loss/gain means for the entire dataset.
loss_gain_comparison: LossGain = sony_stock_data.loss_gain_comparison()
print(loss_gain_comparison.comparison)
```

```
You gain more on a 'up' day.
```

5. Are these results the same across days of the week?

[12]:
```
# Get the loss/gain mean for each day of the week in the dataset.
loss_gain_per_day: dict[str, LossGain] = sony_stock_data.loss_gain_per_day()
for day, loss_gain in loss_gain_per_day.items():
    print(f"Loss/gain for {day}: {loss_gain.comparison}")
```

```
Loss/gain for Thursday: You gain more on a 'up' day.
Loss/gain for Tuesday: You gain more on a 'up' day.
Loss/gain for Friday: You gain more on a 'up' day.
Loss/gain for Wednesday: You gain more on a 'up' day.
Loss/gain for Monday: You gain more on a 'up' day.
```

0.1.3 Question #2

1. Are there any patterns across days of the week?

When analyzing the data table for all five years, I see that there is a pattern where Thursday seems to be negative overall, whereas every other day is positive. This can be shown below in part 3 as well. Interestingly though, it seems that Friday has the highest standard deviation, which indicates that there are more swings in the stock price moving from Thursday to Friday. So, there could be a potential for earnings or loss on Friday because of the increased risk. It also means that there could be some common occurrence that happens on Thursdays to Friday that investors should look into to take advantage of that change.

2. Are there any patterns across different years for the same day of the week?

[13]:
```
# Get a count of how many times each day was the best over the years for SONY␣
 ↪dataset.
```

```
best_sony_days_count: Counter = Counter([entry.best_return_day for entry in␣
 ↪sony_stock_data_per_year])
print(best_sony_days_count)
```

```
Counter({'Tuesday': 2, 'Friday': 2, 'Wednesday': 1})
```

[14]:
```
# Get a count of how many times each day was the best over the years for␣
 ↪S&P-500 dataset.
best_spy_days_count: Counter = Counter([entry.best_return_day for entry in␣
 ↪spy_stock_data.stock_data_for_years()])
print(best_spy_days_count)
```

```
Counter({'Tuesday': 2, 'Friday': 2, 'Wednesday': 1})
```

One pattern that I noticed looking across the weeks is that Tuesday seems to be a general positive day, so above I printed the counts for best days over the years. Interestingly, both the S&P-500 and SONY dataset have the same counts and Tuesday and Friday are the best days. I don't know why this pattern occurs, but this could be an area for investors to look into.

3. What are the best and worst days of the week to be invested for each year?

[15]:
```
# For each year, decide the best day and worst day based upon on average daily␣
 ↪return for each day.
for data in sony_stock_data_per_year:
    print(f"Best day for {data.years_str}: {data.best_return_day}")
    print(f"Worst day for {data.years_str}: {data.worst_return_day}")
```

```
Best day for 2016: Tuesday
Worst day for 2016: Thursday
Best day for 2017: Friday
Worst day for 2017: Thursday
Best day for 2018: Wednesday
Worst day for 2018: Thursday
Best day for 2019: Friday
Worst day for 2019: Thursday
Best day for 2020: Tuesday
Worst day for 2020: Thursday
```

4. Do these days change from year to year for your stock?

[16]:
```
# Best days across all years.
best_days_across_years: set[str] = {entry.best_return_day for entry in␣
 ↪sony_stock_data_per_year}
print(best_days_across_years)
```

```
{'Friday', 'Tuesday', 'Wednesday'}
```

[17]:
```
# Worst days across all years.
```

```
worst_days_across_years: set[str] = {entry.worst_return_day for entry in␣
 ↪sony_stock_data_per_year}
print(worst_days_across_years)
```

```
{'Thursday'}
```

The best day for SONY fluctuates each year between Tuesday, Friday, and Wednesday. The worst day is Thursday and is consistent across all years.

0.1.4 Question #3

Compute the aggregate table across all 5 years, one table for your stock and one table for S&P-500 (using data for "SPY").

[18]:
```
# Display the SONY stock data in a table.
sony_stock_table: StockDataTable = sony_stock_data.as_table()
display(Markdown(f"{sony_stock_table.title}<br>{tabulate(sony_stock_table.
 ↪table_data, sony_stock_table.latex_headers, tablefmt='latex_raw')}"))
```

Data Table for SONY in 2016 - 2020

Day	$\mu(R)$	$\sigma(R)$	|R− |	$\mu(R-)$	$\sigma(R-)$	|R+ |	$\mu(R+)$	$\sigma(R+)$
Monday	0.00101635	0.0181242	107	-0.0126709	0.0121245	129	0.0123693	0.0121245
Tuesday	0.00198004	0.0188358	125	-0.0114916	0.0164181	133	0.0146413	0.0164181
Wednesday	0.00211379	0.0177596	107	-0.0135502	0.0109867	150	0.0132874	0.0109867
Thursday	-0.000813254	0.017948	123	-0.0140724	0.0107337	131	0.0116362	0.0107337
Friday	0.00216297	0.0224562	114	-0.0133259	0.0198801	139	0.014866	0.0198801

[19]:
```
# Display the SPY stock data in a table.
spy_stock_table: StockDataTable = spy_stock_data.as_table()
display(Markdown(f"{spy_stock_table.title}<br>{tabulate(spy_stock_table.
 ↪table_data, spy_stock_table.latex_headers, tablefmt='latex_raw')}"))
```

Data Table for SPY in 2016 - 2020

Day	$\mu(R)$	$\sigma(R)$	|R− |	$\mu(R-)$	$\sigma(R-)$	|R+ |	$\mu(R+)$	$\sigma(R+)$
Monday	0.000508602	0.0139131	99	-0.00861271	0.00845957	137	0.00709992	0.00845957
Tuesday	0.00121495	0.0114646	112	-0.00670917	0.0105001	146	0.00729372	0.0105001
Wednesday	0.000906286	0.0111153	106	-0.00736878	0.00751546	151	0.00671528	0.00751546
Thursday	-2.13937e-05	0.0116764	117	-0.00713607	0.00636175	137	0.00605464	0.00636175
Friday	0.000555782	0.0111108	111	-0.00725718	0.00889529	142	0.0066631	0.00889529

1. What is the best and worst days of the week for each?

[20]:
```
# Calculate the best day of week for the SONY dataset by taking the mean for␣
 ↪all the entries for each day.
best_sony_return_day: str = sony_stock_data.best_return_day
print(f"The best day of the week for the SONY dataset is {best_sony_return_day}.
 ↪")
```

```
The best day of the week for the SONY dataset is Friday.
```

[21]:
```
# Calculate the worst day of week for the SONY dataset by taking the mean for␣
 ↪all the entries for each day.
worst_sony_return_day: str = sony_stock_data.worst_return_day
print(f"The worst day of the week for the SONY dataset is␣
 ↪{worst_sony_return_day}.")
```

```
The worst day of the week for the SONY dataset is Thursday.
```

2. Are these days the same for your stock as they are for S&P-500?

[22]:
```
# Calculate the best day of week for the S&P-500 dataset by taking the mean for␣
 ↪all the entries for each day.
best_spy_return_day: str = spy_stock_data.best_return_day
print(f"The best day of the week for the S&P-500 dataset is␣
 ↪{best_spy_return_day}.")
```

```
The best day of the week for the S&P-500 dataset is Tuesday.
```

[23]:
```
# Calculate the worst day of week for the S&P-500 dataset by taking the mean␣
 ↪for all the entries for each day.
worst_spy_return_day: str = spy_stock_data.worst_return_day
print(f"The worst day of the week for the S&P-500 dataset is␣
 ↪{worst_spy_return_day}.")
```

```
The worst day of the week for the S&P-500 dataset is Thursday.
```

For the next questions, suppose that you have an **"oracle"**. On any day, this oracle could tell you whether price of any stock would increase or decrease the next day. Assume that you have no transaction costs. You start with $$100 on the first trading day of 2016 to trade your stock and another $$100 to trade "SPY".

0.1.5 Question #4

You listen to the oracle and follow its advice. How much money will you have on the last trading day of **2020**:

[24]:
```
# An Oracle instance to listen to.
oracle: Oracle = Oracle()
```

1. SONY

[25]:
```
# Listen to the oracle for the SONY stock.
sony_oracle_return: float = oracle.make_predictions(stocks=sony_stock_data,␣
 ↪initial_investment=100.0)
print(f"Current investment of SONY after listening to oracle:␣
 ↪{float_to_currency(sony_oracle_return)}")
```

```
Current investment of SONY after listening to oracle: $809,108.93
```

2. S&P-500

[26]:
```
# Listen to the oracle for the SPY stock.
spy_oracle_return: float = oracle.make_predictions(stocks=spy_stock_data,␣
 ↪initial_investment=100.0)
print(f"Current investment after of S&P-500 listening to oracle:␣
 ↪{float_to_currency(spy_oracle_return)}")
```

```
Current investment after of S&P-500 listening to oracle: $11,985.06
```

0.1.6 Question #5

Consider the "buy-and-hold" strategy: you buy on the first trading day and sell on the last day. So, you do not listen to your oracle at all. As before, assume that you start with $$100 for both your stock and "SPY".

1. How much money will you have on the last trading day of **2020**?

[27]:
```
# Ignore the oracle and hold the stock throughout the duration of the dataset␣
 ↪for SONY.
sony_buy_and_hold_return: float = sony_stock_data.
 ↪buy_and_hold(initial_investment=100.0)
print(f"Current investment of SONY after ignoring the oracle:␣
 ↪{float_to_currency(sony_buy_and_hold_return)}")
```

```
Current investment of SONY after ignoring the oracle: $407.52
```

[28]:
```
# Ignore the oracle and hold the stock throughout the duration of the dataset␣
 ↪for S&P-500.
spy_buy_and_hold_return: float = spy_stock_data.
 ↪buy_and_hold(initial_investment=100.0)
print(f"Current investment of SPY after ignoring the oracle:␣
 ↪{float_to_currency(spy_buy_and_hold_return)}")
```

```
Current investment of SPY after ignoring the oracle: $203.81
```

2. How do these results compare with results obtained in question 4?

[29]:
```
# Difference between listening to the oracle and ignoring the oracle for SONY.
sony_return_difference: float = percent_change(sony_buy_and_hold_return,␣
 ↪sony_oracle_return)

if sony_oracle_return > sony_buy_and_hold_return:
    print(f"By not listening to the oracle, you have lost␣
 ↪{sony_return_difference:.2f}% of your potential gains for the SONY stock!")
else:
    print(f"You have overcome all odds and beat the oracle by {(100.0 -␣
 ↪sony_return_difference):.3f}% for the SONY stock!")
```

```
By not listening to the oracle, you have lost 99.95% of your potential gains for
the SONY stock!
```

[30]:
```
# Difference between listening to the oracle and ignoring the oracle for␣
 ↪S&P-500.
spy_return_difference: float = percent_change(spy_buy_and_hold_return,␣
 ↪spy_oracle_return)

if spy_oracle_return > spy_buy_and_hold_return:
    print(f"By not listening to the oracle, you have lost␣
 ↪{spy_return_difference:.2f}% of your potential gains for the S&P-500 stock!")
else:
    print(f"You have somehow overcome all odds and beat the oracle by {(100.0 -␣
 ↪spy_return_difference):.3f}% for the S&P-500 stock!")
```

```
By not listening to the oracle, you have lost 98.30% of your potential gains for
the S&P-500 stock!
```

The result of not listening to the oracle is a huge loss in total earnings because over the duration of the stock period, the stocks price fluctuates up and down and by using the oracle and taking advantage of only positive days, the more that the stock swings up, the more you will gain and when the stock price dips, you are spared all the losses.

0.1.7 Question #6

Your oracle got very upset that you did not follow its advice. It decided to take revenge by giving you wrong advice from time to time. Specifically, let us consider the following three scenarios:

(a) Oracle gave you the wrong results for the best 10 trading days. In other words, you missed the best 10 days, you missed the best 10 days and your overall profit will be lowered.

(b) Oracle gave you the wrong results for the worst 10 trading days. In other words, you realize the worst 10 days and your overall profit will be lowered.

(c) Oracle gave you the wrong results for the best 5 days and for the worst 5 days.

Please answer the following:

[31]:
```
# An angry oracle instance.
angry_oracle: Oracle = Oracle(RevengeScenarioA())
```

1. For each of the scenarios above (a, b, and c), compute the final amount that you will have for both your stock and "SPY".

[32]:
```
# Scenario A for SONY: Missing the top 10 best trading days.
revenge_sony_oracle_return_a: str = float_to_currency(
    angry_oracle.make_predictions(stocks=sony_stock_data)
)
print(revenge_sony_oracle_return_a)
```

```
$350,479.65
```

[33]:
```
# Scenario A for S&P-500: Missing the top 10 best trading days.
revenge_spy_oracle_return_a: str = float_to_currency(
    angry_oracle.make_predictions(stocks=spy_stock_data)
)
print(revenge_spy_oracle_return_a)
```

```
$6,849.91
```

[34]:
```
# Switch oracle's strategy to scenario B.
angry_oracle.strategy = RevengeScenarioB()
```

[35]:
```
# Scenario B for SONY: Realizing the top 10 worst trading days.
revenge_sony_oracle_return_b: str = float_to_currency(
    angry_oracle.make_predictions(stocks=sony_stock_data)
)
print(revenge_sony_oracle_return_b)
```

```
$384,000.45
```

[36]:
```
# Scenario B for S&P-500: Realizing the top 10 worst trading days.
revenge_spy_oracle_return_b: str = float_to_currency(
    angry_oracle.make_predictions(stocks=spy_stock_data)
)
print(revenge_spy_oracle_return_b)
```

```
$6,333.20
```

[37]:
```
# Switch oracle's strategy to scenario C.
angry_oracle.strategy = RevengeScenarioC()
```

[38]:
```
# Scenario C for SONY: Realizing the top 5 worst trading days and missing the␣
 ↪top 5
# best days.
revenge_sony_oracle_return_c: str = float_to_currency(
    angry_oracle.make_predictions(stocks=sony_stock_data)
)
print(revenge_sony_oracle_return_c)
```

```
$313,186.28
```

[39]:
```
# Scenario C for S&P-500: Realizing the top 5 worst trading days and missing the
# top 5 best days.
revenge_spy_oracle_return_c: str = float_to_currency(
    angry_oracle.make_predictions(stocks=spy_stock_data)
)
print(revenge_spy_oracle_return_c)
```

```
$5,648.88
```

2. Do you gain more by missing the worst days or by missing the best days?

[40]:
```
# Compare missing the best days versus missing the worst days for SONY.
if revenge_sony_oracle_return_a >= revenge_sony_oracle_return_b:
    print("You gained more by missing the best days for SONY.")
else:
    print("You gain more by missing the worst days for SONY.")

# Compare missing the best days versus missing the worst days for S&P-500.
if revenge_spy_oracle_return_a >= revenge_spy_oracle_return_b:
    print("You gained more by missing the best days for S&P-500.")
else:
    print("You gain more by missing the worst days for S&P-500.")
```

```
You gain more by missing the worst days for SONY.
You gained more by missing the best days for S&P-500.
```

3. Are the results in part (c) different from the results that you obtained in question 4.

[41]:
```
# Compare question 4 to scenario c.
revenge_versus_normal: str = float_to_currency(difference(
    angry_oracle.make_predictions(stocks=sony_stock_data), sony_oracle_return
))
print(f"The oracle's revenge in scenario c cost you {revenge_versus_normal} in␣
 ↪losses.")
```

```
The oracle's revenge in scenario c cost you $495,922.65 in losses.
```

Yes, the oracle's revenge in scenario c and all scenarios created a decent sized loss compared to when the oracle makes accurate predictions.