Methodology

This analysis was conducted by first using the yfinance library to pull a pandas data frame of stock prices for a set of seven stock tickers. 21 days' worth of closing prices were collected so that daily return could be computed.

Daily return was defined as: closing price from current day – closing price from previous day. Despite pulling 21 days' worth of closing price data, we define day 1 as the first daily return; hence, we have 20 entries of daily returns.

Once daily returns were collected into a matrix, broadcasting was utilized to mean center each of the columns of the daily returns matrix. Then, covariance was computed using

$$\text{Covariance (cov)} = X.T \text{ @ } X * 1/n\text{-}1$$

where n is the number of stocks in our basket and X is the matrix of daily returns.

Correlation was computed using data present in the covariance matrix. Since the positive diagonal (x_{ii}) is the variance of the data set, we can use those elements to compute the correlation matrix. The correlation matrix is defined as

$$\text{Correlation} = S \text{ @ cov @ } S$$

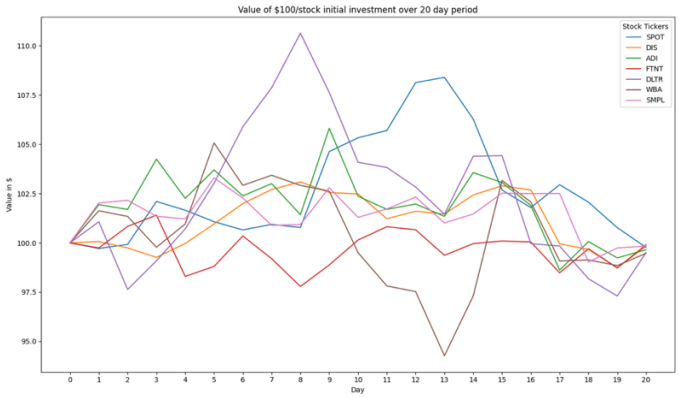
Where S is the identity matrix of the same dimensions as cov scaled to the reciprocal of the standard deviation of the dataset.

Finally, portfolio performance was analyzed by simulating the 20-day time period with an initial investment of $100 in each stock. On day 1, the percent change was applied to the initial investment. For the remaining days, the percent change from the initial data set was applied to the result of the previous day. The results and analysis are present below.

Results and Analysis

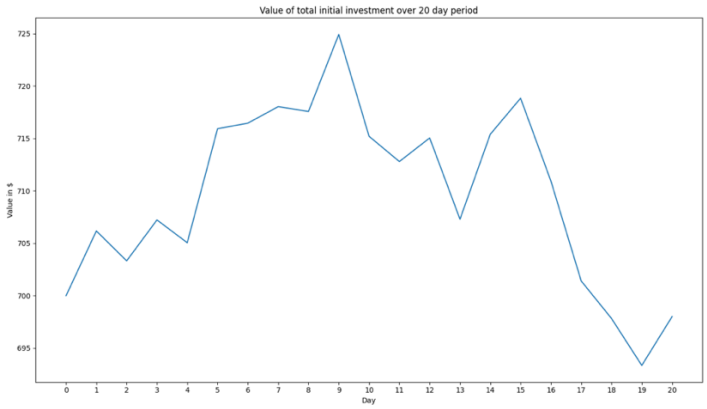
The following is a brief analysis on portfolio performance for the past 20 days of returns. We assumed that we would put $100 in for each stock at the beginning of the period. Then, we observed how each individual stock performed as well as how the total portfolio performed over the period. Each day, the total performance was calculated as the sum of the individual allocation performances. We start with the following data:

Figure 1:



We can see from this first figure, most stocks increased in value over the 20 day period while some decreased. WBA decreased in value. By day 12, the initial $100 was worth less than $95. Other stocks like DLTR and SPOT increased in value. DLTR peaking at about $110 and SPOT peaking at about $108. Although many of the stocks tended to move in the same direction, we see some stocks move against the trend. For example, SPOT rose around the same time WBA decreased. Now, let's look at total portfolio value over the period.

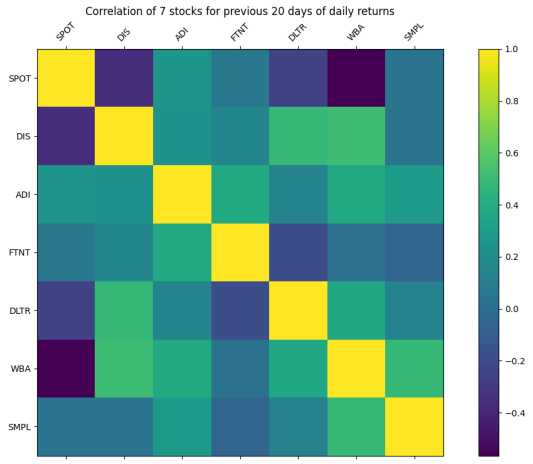
Figure 2:



Here we see a picture of our portfolio in aggregate over the same period. Despite the movement evident in the previous figure, we can see here that we ultimately lost money over the 20 period. Our initial investment of $700 ended at $698 despite peaking at around $725 halfway through the period. I think this is because many of the stocks were correlated

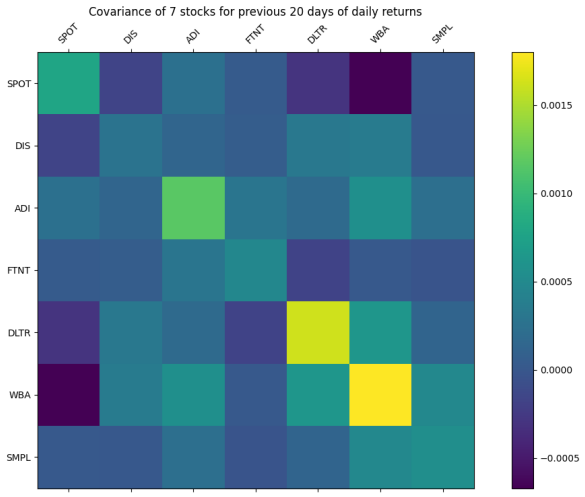
positively. Towards the end of the period (around day 13 Figure 1) we see a contraction in individual stock value across many of the holdings in our portfolio. So, the fact that we see a large decline at the tail end of our time period here is unsurprising.

Figure 3:



If we look at the columns in the correlation matrix above, we can see that very few of the stocks had negative or zero correlation with the others. Only, Spotify which was relative uncoupled and negatively coupled to the others stands out. Otherwise, most of the correlations are from 0.2-0.6 indicating slight to strong positive correlations. So, this portfolio was a little more susceptible to movements tending in the same direction.

Figure 4:



Our covariance matrix here tells us a similar story. Looking at the first column, we see that WBA and SPOT have a negative covariance suggesting that these two stocks will move in different directions. This may explain some of the behavior we see for the for these two

stocks moving against each other in the first plot. Overall, many of the stocks seem to have a small positive covariance suggesting they will move together over this period. This is reflected in the portfolio performance over this period.

Ultimately, a larger portfolio may mitigate the effects of the price movement homogeneity. By increasing the size of the basket, we may be able to reduce the chance that we have many stocks that move in the same directions. This is assuming, of course, that we want to decrease the average covariance/correlation of the portfolio. Another strategy that might achieve this goal is selecting stocks specifically to get to negative covariance. This may be more feasible for a smaller portfolio.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Oct  6 14:34:34 2024

@author: ramsonmunoz
"""
import numpy as np
import pandas as pd
import yfinance as yf
from datetime import datetime
import matplotlib.pyplot as plt


pd.set_option("display.max_columns",None)
# Define of list of strings that represent the tickers I want to look at:
#companies = ['AMZN','TSLA','MRNA','RGR','CALM','JPM','CENX']
#companies = ['JPM', 'VZ', 'XOM', 'DIS', 'PFE', 'CSCO', 'HD']
companies = ['SPOT', 'DIS', 'ADI', 'FTNT', 'DLTR', 'WBA', 'SMPL']

start = '2024-09-06'
end = datetime.now().strftime('%Y-%m-%d')
resolution = '1d'

companyDailyReturns = []

for company in companies:
    ticker = yf.Ticker(company) #creates an instance of the ticker for a given ticker in my list of companies
    tickerHistory = ticker.history(start=start, end=end, interval=resolution) # pulls the history for that ticker starting and ending at given points with the

    #print(tickerHistory.info())
    #priceOpen = tickerHistory[['Open']] # selects only open price data from all of the provided data. Useful if I want to compute intraday returns
    priceClose = tickerHistory[['Close']].to_numpy()  # selects only closing price data from all of the provided data. passes to numpy array

    dailyReturnVector = np.zeros(priceClose.size - 1)

    for i in range(1,priceClose.size):
        dailyReturnVector[i-1] = (priceClose[i].item() - priceClose[i -1].item()) / priceClose[i].item() # passes daily return for each company into a vector

    companyDailyReturns.append(dailyReturnVector)


companyDailyReturns = np.stack(companyDailyReturns).T #gets us a matrix of returns where each column is a stock and the rows are the daily returns for the 20 d

'''
Now we need to mean-center the columns so that we can compute the covariance. Here we take advantage of broadcasting.
'''

#print(companyDailyReturns)

for i in range(companyDailyReturns.shape[1]):
    companyDailyReturns[:,i] = companyDailyReturns[:,i] - np.mean(companyDailyReturns[:,i]) # we subtract each column by its mean

#print(companyDailyReturns)

'''
now we implement the covariace formula of X.T @ X which in our case returns a 7x7 matrix which, once scaled, should be covariance
'''

covariaceMatrix = (companyDailyReturns.T @ companyDailyReturns) * ((companyDailyReturns.shape[1] - 1) ** - 1) # we are doing 1/n-1 * X.T @ X here

#creating a figure to plot covariance matrix in matplotlib
tickers = companies
plt.figure(0)
cax = plt.matshow(covariaceMatrix)
cbar = plt.colorbar(cax)
# Title and labels
plt.xticks(ticks=np.arange(len(tickers)), labels=tickers, rotation=45)
plt.yticks(ticks=np.arange(len(tickers)), labels=tickers)

# Title and labels
plt.title('Covariance of 7 stocks for previous 20 days of daily returns')
#plt.tight_layout()
#plt.xlabel('Stocks')
#plt.ylabel('Stock Tickers')



'''
In order to generate the corrrelation matrix, we simply must generate R = SCS where C is the computed covariance matrix above, R is the correlation matrix and
S is a diagonal matrix where the diagonal elements are the reciprocals of the standard deviations of each respective stock.

Luckily, the diagonal elemenents of the covariance matrix are precisely the variances of each stock. So what we need to do is pull the diagonal elements. take
Then, we can mulitply using the formula above to get the correlation matrix. We should expect that the correlation matrix is
'''

reciprocalStdMatrix = np.diag(np.power(np.sqrt(np.diag(covariaceMatrix)),-1)) # generates S
#print(reciprocalStdMatrix)

correlationMatrix = (reciprocalStdMatrix @ covariaceMatrix) @ reciprocalStdMatrix

print("The covariance matrix for last 20 daily returns of our portfolio:\n")
print(covariaceMatrix,end='\n\n')

print("The correlation matrix for last 20 daily returns of our portfolio:\n")
print(correlationMatrix,end='\n\n')

#creating a figure to plot covariance matrix in matplotlib
tickers = companies
plt.figure(1)
cax = plt.matshow(correlationMatrix)
cbar = plt.colorbar(cax)
# Title and labels
plt.xticks(ticks=np.arange(len(tickers)), labels=tickers, rotation=45)
plt.yticks(ticks=np.arange(len(tickers)), labels=tickers)
```

```
# Title and labels
plt.title('Correlation of 7 stocks for previous 20 days of daily returns')
#plt.tight_layout()
#plt.xlabel('Stocks')
#plt.ylabel('Stock Tickers')

#plt.show()


#plt.show()


'''
To evaluate protfolio performace I am going to build a small simulation of the portfolio accross the time period of 20 days.

We will assume that we start with $70 distrbuted evenly accross all stocks.

We will assume no taxes, or fees for trading, and we are allowed to buy partial stocks.

We also assume that the portfolio is alive for only the 20 day period with the initial influx of cash on day 1.
'''

portfolio = np.zeros([21,7]) # we initialize an empty portfolio.
portfolio[0,:] += 100 # we purchace $100 worth of each stock at the end of the 0th day.
totalPortfolioValueDaily = np.zeros([21,1])
totalPortfolioValueDaily[0,0] = sum(portfolio[0,:])
#print(totalPortfolioValueDaily)
#print(portfolio)

for i in range(0,companyDailyReturns.shape[0]): # looping through the days
    portfolio[i+1,:] += (1+companyDailyReturns[i,:]) * portfolio[i,:] #allows us to track the performace of some initial ammount invested in each stock across
    totalPortfolioValueDaily[i+1, 0] = sum(portfolio[i+1, :])
#print(portfolio)
#print(totalPortfolioValueDaily)

indices = np.arange(len(totalPortfolioValueDaily))
#print(indices)

plt.figure(3)

plt.plot(indices,totalPortfolioValueDaily)
# Title and labels
plt.title('Value of total initial investment over 20 day period')
plt.tight_layout()
plt.xlabel('Day')
plt.ylabel('Value in $')
plt.xticks(indices)



plt.figure(4)

for i in range(len(tickers)):
    plt.plot(indices,portfolio[:,i],label=tickers[i])


# Title and labels
plt.title('Value of $100/stock initial investment over 20 day period')
plt.tight_layout()
plt.xlabel('Day')
plt.ylabel('Value in $')
plt.xticks(indices)
#legend
plt.legend(title='Stock Tickers')


#plt.show()
```

```
/usr/local/bin/python3.12 /Users/ramsonmunoz/Documents/College/FIU/FALL 2024/MAS 4107/Project 1/src/driver.py
The covariance matrix for last 20 daily returns of our portfolio:

[[ 7.81294272e-04 -1.66655339e-04  2.40727964e-04  3.64544055e-05
   -2.99000193e-04 -6.71540738e-04  2.35759222e-05]
 [-1.66655339e-04  2.69272675e-04  1.34456220e-04  5.56942941e-05
    3.14383845e-04  3.59006756e-04  1.29507714e-05]
 [ 2.40727964e-04  1.34456220e-04  1.16483457e-03  2.85813252e-04
    1.82676039e-04  5.60657460e-04  2.29562898e-04]
 [ 3.64544055e-05  5.56942941e-05  2.85813252e-04  4.77624470e-04
   -1.73999801e-04  2.02400537e-05 -2.94518170e-05]
 [-2.99000193e-04  3.14383845e-04  1.82676039e-04 -1.73999801e-04
    1.61960511e-03  6.28089828e-04  1.22909505e-04]
 [-6.71540738e-04  3.59006756e-04  5.60657460e-04  2.02400537e-05
    6.28089828e-04  1.80119733e-03  4.79532117e-04]
 [ 2.35759222e-05  1.29507714e-05  2.29562898e-04 -2.94518170e-05
    1.22909505e-04  4.79532117e-04  5.47515204e-04]]

The correlation matrix for last 20 daily returns of our portfolio:

[[ 1.         -0.36334214  0.25234058  0.05967596 -0.26580282 -0.56608858
    0.03604651]
 [-0.36334214  1.          0.24007815  0.15529994  0.47605761  0.51549663
    0.03372884]
 [ 0.25234058  0.24007815  1.          0.38318363  0.13299793  0.38706583
    0.28745607]
 [ 0.05967596  0.15529994  0.38318363  1.         -
```

```
27 0.19783398  0.02182166
28   -0.05759313]
29  [-0.26580282  0.47605761  0.13299793 -0.19783398
   1.          0.36773634
30    0.13052173]
31  [-0.56608858  0.51549663  0.38706583  0.02182166
   0.36773634  1.
32    0.48287971]
33  [ 0.03604651  0.03372884  0.28745607 -0.05759313
   0.13052173  0.48287971
34    1.         ]]
35
36
37 Process finished with exit code 0
38
```