TeamHomework4

Kyle Chang, Christian Warren, Dev Vadalia

March 29, 2021

[1]:
```
#!/usr/bin/env python

import numpy as np
import pandas as pd
import datetime as dt
import math
```

[2]:
```
def splitInToYears(df):
    cols = df.columns

    df2016 = pd.DataFrame(columns = cols)
    df2017 = pd.DataFrame(columns = cols)
    df2018 = pd.DataFrame(columns = cols)
    df2019 = pd.DataFrame(columns = cols)
    df2020 = pd.DataFrame(columns = cols)

    for index, row in df.iterrows():
        if dt.datetime.strptime(df['Date'][index],'%Y-%m-%d').year == 2016:
            df2016 = df2016.append(row,ignore_index=True)
        if dt.datetime.strptime(df['Date'][index],'%Y-%m-%d').year == 2017:
            df2017 = df2017.append(row,ignore_index=True)
        if dt.datetime.strptime(df['Date'][index],'%Y-%m-%d').year == 2018:
            df2018 = df2018.append(row,ignore_index=True)
        if dt.datetime.strptime(df['Date'][index],'%Y-%m-%d').year == 2019:
            df2019 = df2019.append(row,ignore_index=True)
        if dt.datetime.strptime(df['Date'][index],'%Y-%m-%d').year == 2020:
            df2020 = df2020.append(row,ignore_index=True)
    return [df2016, df2017,df2018, df2019,df2020]

## --- Estimator dataframes ---
# returns array of dataframes split by year in accending order
def calculateEstimators(dataframes):
    estimators = []
    assets = ['VFIAX', 'VBTLX', 'VGSLX', 'VIMAX', 'VSMAX', 'VGHCX', 'AMZN',␣
↪'WMT', 'CVS']
    for df in dataframes:
        data = df[df.columns[1:]]
```

```
        uniformWeights = 1/data.shape[0]

        wBar = np.sum(np.square(np.ones(data.shape[0]) * uniformWeights))

        mean = np.array(np.sum(data, axis=0) * uniformWeights).reshape(-1,1)
        difference = np.subtract(data , np.matmul(mean,np.ones((data.
 ↪shape[0],1)).transpose()).transpose())
        variance =  np.array(1/(1 - wBar) * np.sum(uniformWeights * np.
 ↪square(difference), axis = 0)).reshape(-1,1)
        StdOfExpectedValue = np.array(np.sqrt(wBar) * np.sqrt(variance)).
 ↪reshape(-1,1)

        signalToNoise = np.absolute(np.array(mean/StdOfExpectedValue)).
 ↪reshape(-1,1)
        estimator = pd.DataFrame({
                    'Expected Return': mean.reshape(-1,),
                    'Variance Estimator': variance.reshape(-1,),
                    'Std Dev Expected Return': StdOfExpectedValue.reshape(-1,),
                    'Signal to Noise': signalToNoise.reshape(-1,)},
                    index=assets)
        estimators.append(estimator)
    return estimators
# print(ExpectedReturn)
```

[3]:
```
## --- Data Wrangling ---

# Group A
VFIAX = pd.read_csv("Data/VFIAX.csv")
VFIAX.columns = ['Date','Open','High','Low','Close','VFIAX Close','Volume']
VBTLX = pd.read_csv("Data/VBTLX.csv")
VBTLX.columns = ['Date','Open','High','Low','Close','VBTLX Close','Volume']
VGSLX = pd.read_csv("Data/VGSLX.csv")
VGSLX.columns = ['Date','Open','High','Low','Close','VGSLX Close','Volume']

# Group B
VIMAX = pd.read_csv("Data/VIMAX.csv")
VIMAX.columns = ['Date','Open','High','Low','Close','VIMAX Close','Volume']
VSMAX = pd.read_csv("Data/VSMAX.csv")
VSMAX.columns = ['Date','Open','High','Low','Close','VSMAX Close','Volume']
VGHCX = pd.read_csv("Data/VGHCX.csv")
VGHCX.columns = ['Date','Open','High','Low','Close','VGHCX Close','Volume']

# Group C
AMZN = pd.read_csv("Data/AMZN.csv")
AMZN.columns = ['Date','Open','High','Low','Close','AMZN Close','Volume']
WMT = pd.read_csv("Data/WMT.csv")
WMT.columns = ['Date','Open','High','Low','Close','WMT Close','Volume']
```

```
CVS = pd.read_csv("Data/CVS.csv")
CVS.columns = ['Date','Open','High','Low','Close','CVS Close','Volume']
```

[4]:

```
## --- Assemble -- code into a dataframe for Close of Day ---

close = pd.concat([VFIAX['Date'], VFIAX['VFIAX Close'], VBTLX['VBTLX Close'],␣
↪VGSLX['VGSLX Close'], VIMAX['VIMAX Close'], VSMAX['VSMAX Close'], VGHCX['VGHCX␣
↪Close'], AMZN['AMZN Close'], WMT['WMT Close'], CVS['CVS Close'] ], axis=1)
#print(close)

## --- generate mean daily return ---

dailyReturn = pd.DataFrame(columns = ['Date', 'VFIAX Daily Return','VBTLX Daily␣
↪Return','VGSLX Daily Return', 'VIMAX Daily Return', 'VSMAX Daily Return',␣
↪'VGHCX Daily Return','AMZN Daily Return', 'WMT Daily Return','CVS Daily␣
↪Return'])
for index, row in close.iterrows():
    if index == 0: continue
    #print((close['VFIAX Close'][index] - close['VFIAX Close'][index-1])/␣
↪(close['VFIAX Close'][index-1]))
    dailyReturn = dailyReturn.append({'Date': close['Date'][index],
                'VFIAX Daily Return': ((close['VFIAX Close'][index] -␣
↪close['VFIAX Close'][index-1])/(close['VFIAX Close'][index-1])),
                'VBTLX Daily Return': ((close['VBTLX Close'][index] -␣
↪close['VBTLX Close'][index-1])/(close['VBTLX Close'][index-1])),
                'VGSLX Daily Return': ((close['VGSLX Close'][index] -␣
↪close['VGSLX Close'][index-1])/(close['VGSLX Close'][index-1])),
                'VIMAX Daily Return': ((close['VIMAX Close'][index] -␣
↪close['VIMAX Close'][index-1])/(close['VIMAX Close'][index-1])),
                'VSMAX Daily Return': ((close['VSMAX Close'][index] -␣
↪close['VSMAX Close'][index-1])/(close['VSMAX Close'][index-1])),
                'VGHCX Daily Return': ((close['VGHCX Close'][index] -␣
↪close['VGHCX Close'][index-1])/(close['VGHCX Close'][index-1])),
                'AMZN Daily Return': ((close['AMZN Close'][index] - close['AMZN␣
↪Close'][index-1])/(close['AMZN Close'][index-1])),
                'WMT Daily Return': ((close['WMT Close'][index] - close['WMT␣
↪Close'][index-1])/(close['WMT Close'][index-1])),
                'CVS Daily Return': ((close['CVS Close'][index] - close['CVS␣
↪Close'][index-1])/(close['CVS Close'][index-1]))},ignore_index=True)
```

Exercise 1

2016

	Expected Return	Variance Estimator	Std Dev Expected Return	Signal to Noise
VSMAX	0.000723	0.000112	0.000666	1.085166
VFIAX	0.000482	0.000068	0.000520	0.926836
VGHCX	-0.000606	0.000111	0.000664	0.912577
WMT	0.000663	0.000147	0.000764	0.866886
CVS	-0.000682	0.000190	0.000867	0.786628
VIMAX	0.000468	0.000092	0.000605	0.773408
VBTLX	0.000103	0.000005	0.000142	0.726289
VGSLX	0.000380	0.000115	0.000676	0.562157
AMZN	0.000586	0.000350	0.001179	0.497363

2017

	Expected Return	Variance Estimator	Std Dev Expected Return	Signal to Noise
VFIAX	0.000794	0.000018	0.000266	2.990815
VIMAX	0.000714	0.000024	0.000311	2.298355
AMZN	0.001856	0.000174	0.000834	2.226893
WMT	0.001588	0.000132	0.000724	2.193117
VSMAX	0.000620	0.000041	0.000404	1.534995
VGHCX	0.000484	0.000042	0.000409	1.183258
VBTLX	0.000139	0.000003	0.000118	1.180622
VGSLX	0.000211	0.000041	0.000403	0.524879
CVS	-0.000142	0.000188	0.000865	0.164112

2018

	Expected Return	Variance Estimator	Std Dev Expected Return	Signal to Noise
AMZN	1.255538e-03	0.000517	0.001436	0.874591
VIMAX	-3.338606e-04	0.000105	0.000646	0.516644
VSMAX	-3.331022e-04	0.000114	0.000673	0.494679
VGSLX	-1.941401e-04	0.000104	0.000643	0.302116
VGHCX	-2.011099e-04	0.000112	0.000669	0.300567
VFIAX	-1.236112e-04	0.000116	0.000679	0.182170
CVS	-1.216537e-04	0.000337	0.001159	0.104945
WMT	-2.482504e-05	0.000228	0.000954	0.026029
VBTLX	3.018343e-07	0.000003	0.000115	0.002617

2019

	Expected Return	Variance Estimator	Std Dev Expected Return	Signal to Noise
VBTLX	0.000335	0.000005	0.000141	2.373610
VFIAX	0.001117	0.000062	0.000495	2.257628
VIMAX	0.001105	0.000065	0.000506	2.183200
VGSLX	0.001037	0.000059	0.000484	2.142270
WMT	0.001087	0.000082	0.000570	1.906907
VSMAX	0.001003	0.000085	0.000581	1.725868
AMZN	0.000926	0.000208	0.000909	1.019067
CVS	0.000762	0.000261	0.001018	0.748512
VGHCX	0.000443	0.000092	0.000605	0.731791

2020

	Expected Return	Variance Estimator	Std Dev Expected Return	Signal to Noise
AMZN	0.002535	0.000589	0.001525	1.661972
VBTLX	0.000293	0.000010	0.000195	1.502099
WMT	0.001022	0.000394	0.001247	0.819413
VFIAX	0.000902	0.000470	0.001364	0.661107
VIMAX	0.000925	0.000524	0.001439	0.642574
VSMAX	0.001010	0.000631	0.001579	0.639748
VGHCX	0.000334	0.000320	0.001124	0.296806
VGSLX	0.000173	0.000714	0.001680	0.102886
CVS	0.000101	0.000619	0.001564	0.064797

2016:

VSMAX: After dipping from 60 to 46 in the second half of 2015, VSMAX saw consistent growth throughout 2016 growing back to 60. The general trend was upwards and constant, leading to the highest expected return and relatively small volatility resulting in a higher signal to noise ratio.

VFIAX: After seeing a sharp dip to start the year, the fund steadily grew throughout the year, climbing 30 dollars in a steady manner. This resulted in a similar situation to VSMAX, however the volatility of expected return was slightly larger than the expected return.

VGSLX: Saw rapid growth for the first half of the year, climbing nearly 30% and declined over the second half of the year. The expected return was the lowest in magnitude of all the funds besides the bonds and it had the highest volatility of all the funds, giving it the second smallest SNR.

AMZN: Trended downward in closing prices in the first half of 2016 then recovered in the latter half of 2016. As a result it had the highest variance in returns, thus lowested signal to noise.

The rest of the assets behaved relatively similarly to VFIAX, seeing a SNR of roughly between 0.7-0.9. CVS and VGHCX were the two assets with negative expected returns.

2017:

VFIAX: Even steadier growth was seen continuing from 2016 leading to an even smaller value for volatility and a larger expected return.

AMZN/WMT: Saw very high expected returns that 'overpowered' the high volatility seen over the year. Both companies saw their stock price increase nearly 50% on the year, associated with sharp rises that explained the higher volatility.

CVS: Consistently saw the stock price increase and dip by large quantities each month and overall ended the year down. The volatility was the highest of all assets and it saw one of the smallest expected returns leading to the lowest SNR.

VGSLX: Similarly saw consistent price fluctuations however in a smaller range (114-121) and ended the year near where it started, meaning low ER and relatively high volatility.

As the market did exceptionally well, the rest of the funds similarly saw large growth and low volatility and the rest of the funds had SNR of >1

2018:

The trade war affected all the stocks, leading to a negative return for the year in many cases and high volatility when compared to the other years. The expected return was small in magnitude due to the gains made in the first half of the year being erased by the dip at the end of the year.

AMZN: Their sharp rise in the first half of the year meant that even though they were harshly affected by the trade war as well, they ended the year positive. This higher expected return allowed them to have the highest SNR even with the highest volatility

2019:

VGHCX/CVS: Sharp rises as well as declines lead to a small expected return and a relatively high volatility = small SNR. Healthcare related saw a dip Q1 2019

Amazon was an outlier for higher growth during much of the recovery and corrected to the rest of the group leading to a different spot in the order.

Rest of companies having steady recovery from trade war, all within a very tight grouping for the majority of the year.

2020:

AMZN, VBTLX, WMT were not hit very hard by the 2020 Coronavirus crash, with AMZN skyrocketing because of the crash, VBTLX staying flat for the duration, and Walmart staying in a steady uptrend.

VFIAX, VIMAX, VSMAX were hit hard during the corona crash but quickly recovered along with the rest of the overall stock market, but then continued upwards after the crash to higher than the pre crash high.

VGHCX, VGSLX, CVS these companies were hit by the corona crash but unlike the companies above they did not grow past their prior peaks

Exercise 2

[17]:
```
logReturns = dailyReturn[dailyReturn.columns[1:]].applymap(math.log1p)
logReturns.insert(0,'Date',dailyReturn[dailyReturn.columns[0]])

logReturnsOverTimeSpan = splitInToYears(logReturns)

estimators = calculateEstimators(logReturnsOverTimeSpan)

for years_after_2016, estimator in enumerate(estimators):
    display(2016 + years_after_2016, estimator.sort_values(by='Signal to Noise',␣
 ↪axis=0, ascending=False).style)
```

2016

	Expected Return	Variance Estimator	Std Dev Expected Return	Signal to Noise
VSMAX	0.000667	0.000112	0.000667	0.999493
VGHCX	-0.000661	0.000111	0.000664	0.994955
CVS	-0.000779	0.000197	0.000884	0.881306
VFIAX	0.000448	0.000068	0.000520	0.860266
WMT	0.000590	0.000145	0.000758	0.778840
VBTLX	0.000100	0.000005	0.000142	0.708404
VIMAX	0.000422	0.000093	0.000607	0.695569
VGSLX	0.000323	0.000116	0.000677	0.476212
AMZN	0.000412	0.000349	0.001177	0.350242

2017

	Expected Return	Variance Estimator	Std Dev Expected Return	Signal to Noise
VFIAX	0.000785	0.000018	0.000266	2.956056
VIMAX	0.000702	0.000024	0.000311	2.258311
AMZN	0.001771	0.000167	0.000815	2.173279
WMT	0.001523	0.000127	0.000711	2.142005
VSMAX	0.000599	0.000041	0.000404	1.483660
VBTLX	0.000137	0.000003	0.000118	1.166026
VGHCX	0.000463	0.000042	0.000410	1.127987
VGSLX	0.000191	0.000041	0.000403	0.474195
CVS	-0.000236	0.000190	0.000869	0.271606

2018

	Expected Return	Variance Estimator	Std Dev Expected Return	Signal to Noise
AMZN	0.000997	0.000519	0.001438	0.693449
VIMAX	-0.000386	0.000105	0.000648	0.596259
VSMAX	-0.000390	0.000114	0.000675	0.577869
VGHCX	-0.000258	0.000114	0.000673	0.382527
VGSLX	-0.000246	0.000104	0.000644	0.381743
VFIAX	-0.000181	0.000116	0.000680	0.266585
CVS	-0.000291	0.000340	0.001165	0.249585
WMT	-0.000139	0.000230	0.000958	0.145263
VBTLX	-0.000001	0.000003	0.000115	0.011798

2019

	Expected Return	Variance Estimator	Std Dev Expected Return	Signal to Noise
VBTLX	0.000332	0.000005	0.000141	2.356403
VFIAX	0.001085	0.000062	0.000495	2.191343
VIMAX	0.001072	0.000065	0.000507	2.115788
VGSLX	0.001007	0.000059	0.000485	2.078679
WMT	0.001046	0.000081	0.000568	1.841993
VSMAX	0.000960	0.000085	0.000582	1.650334
AMZN	0.000822	0.000208	0.000909	0.904661
VGHCX	0.000396	0.000094	0.000610	0.649815
CVS	0.000632	0.000261	0.001018	0.620537

2020

	Expected Return	Variance Estimator	Std Dev Expected Return	Signal to Noise
VBTLX	0.000288	0.000010	0.000195	1.475302
AMZN	0.002240	0.000586	0.001521	1.472461
WMT	0.000829	0.000385	0.001234	0.671497
VFIAX	0.000665	0.000477	0.001374	0.483936
VIMAX	0.000660	0.000535	0.001454	0.454088
VSMAX	0.000690	0.000650	0.001602	0.430377
VGHCX	0.000173	0.000323	0.001130	0.153230
CVS	-0.000209	0.000625	0.001571	0.132763
VGSLX	-0.000192	0.000744	0.001715	0.112109

2016:

AMZN and VSMAX: Remained in the same position as in the previous exercise.

CVS and VGHCX: These have higher confidence because all of our assets had the following relationshipE(log(1 + R)) were less than or equal to the expected returns. However CVS and VGHCX in particular had negative average returns. So this influenced the calculation for the other estimators which factor into the signal to noise calculation. Additionally we took the absolute value of signal to noise so that we could perform a better analysis of the data.

2017:

VGHCX and VBTLX: These two assets switch positions in terms of ordering for the signal to noise. The predominant reason behind this is that Log(1 + R) lowered the expected returns for VBTLX, while not impacting the variance or the std dev, thus lowering the confidence in the estimator. Furthermore the estimators were particularly small for VBTLX which is why the difference changed the ordering.

All other assets maintained the same position as the previous exercise.

2018:

VGHCX and VGSLX: These two assets switch positions in terms of ordering for the signal to noise. The difference between these two assets is small in terms of signal to noise, so small differences in expected returns, variance, and std dev can change the ordering.

All other assets maintained the same position as the previous exercise.

2019:

VGHCX and CVS: These two assets switch positions in terms of ordering for the signal to noise.

All other assets maintained the same position as the previous exercise.

2020:

AMZN VBTLX These two assets switch positions in terms of ordering for the signal to noise.

VGSLX CVS These two assets switch positions in terms of ordering for the signal to noise.

WMT VFIAX VIMAX VSMAX VGHCX all remained the same

Exercise 3

For the context of this problem R is the daily returns.

We see that the $E[R]$ is greater than $E[log(1+R)]$. However when we calculate $log(1+E[R])$ we see that they are less than $E[log(1+R)]$. This relationship is due to Jensen's inequality which states that given a convex/ (concave) function g it satisfies the following inequality: $g(E[x]) \leq E[g(x)]$