TeamHomework5

Kyle Chang, Christian Warren, Dev Vadalia

April 5, 2021

Exercise 1

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

import numpy as np
import pandas as pd
import datetime as dt
import math
import yfinance as yf
import matplotlib.pyplot as plt
from IPython.display import display, Latex
from scipy import stats
```

[4]:
```
## --- Gets year by rolling quarters ---
frames = 0
yearStart = pd.to_datetime(dailyReturn.Date.min() , yearfirst= True) - pd.
 ↪tseries.offsets.DateOffset(days=1)
yearEnd = pd.to_datetime('2016-12-30', yearfirst= True)

rollingQuarterData = []

while frames < 15:
    mask = (pd.to_datetime(dailyReturn['Date'], yearfirst= True) >= (yearStart␣
 ↪)) & (pd.to_datetime(dailyReturn['Date'], yearfirst= True) <= yearEnd)
    currYear = pd.DataFrame(dailyReturn.loc[mask])
    rollingQuarterData = rollingQuarterData + [currYear]
    yearStart = yearStart + pd.tseries.offsets.QuarterEnd()
    yearEnd = yearEnd + pd.tseries.offsets.QuarterEnd()
    frames += 1
```

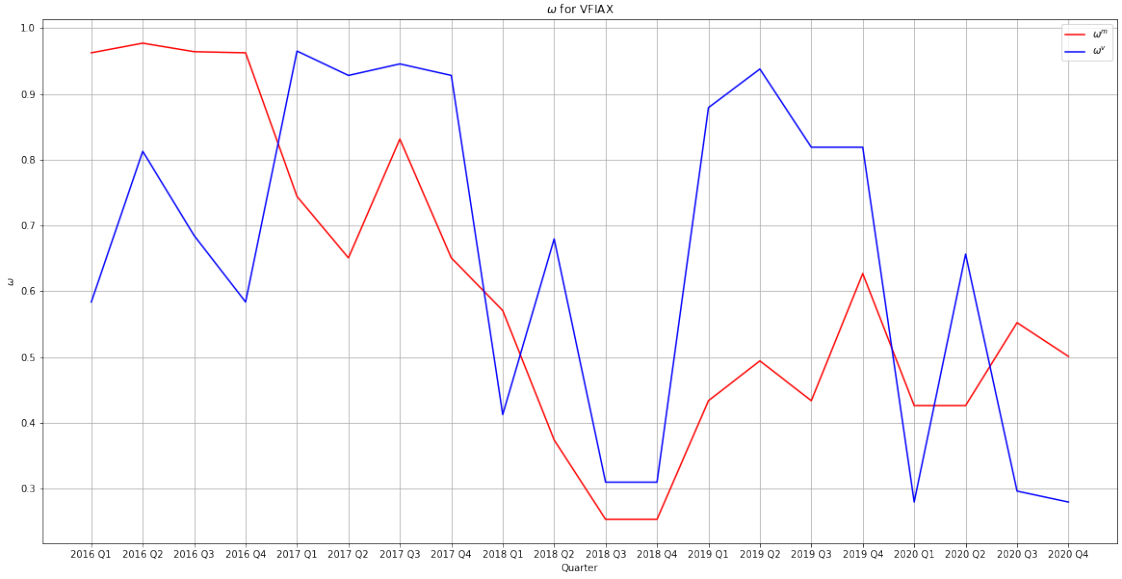
[5]:
```
def splitYearToQuarters(df):
    frames = 0
    quarterStart = pd.to_datetime(dailyReturn.Date.min() , yearfirst= True) - pd.
 ↪tseries.offsets.DateOffset(days=1)
    quarterEnd = quarterStart + pd.tseries.offsets.QuarterEnd()

    quarters = []

    while frames < 4:
        mask = (pd.to_datetime(df['Date'], yearfirst= True) >= (quarterStart ))␣
 ↪& (pd.to_datetime(df['Date'], yearfirst= True) <= quarterEnd)
        currQuarter = pd.DataFrame(df.loc[mask])
        quarters = quarters + [currQuarter]
        quarterStart = quarterStart + pd.tseries.offsets.QuarterEnd()
        quarterEnd = quarterEnd + pd.tseries.offsets.QuarterEnd()
        frames += 1
    return quarters
```

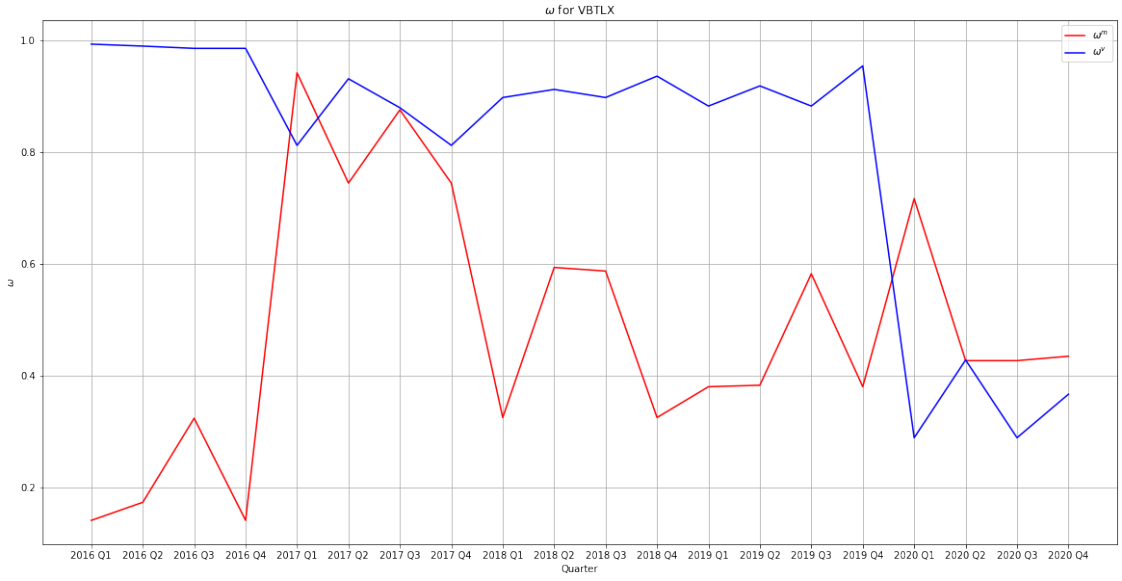
[6]:
```

quarters = [('-01-01', '-03-31'), ('-04-01', '-06-30'), ('-07-01', '-09-30'),␣
 ↪('-10-01', '-12-31')]
years = [
    ('2016-01-01','2016-12-31'),
    ('2017-01-01','2017-12-31'),
    ('2018-01-01','2018-12-31'),
    ('2019-01-01','2019-12-31'),
    ('2020-01-01','2020-12-31')
]

assets = ['VFIAX', 'VBTLX', 'VGSLX', 'VIMAX', 'VSMAX', 'VGHCX', 'AMZN', 'WMT',␣
 ↪'CVS']
df = yf.download(assets, '2015-12-31', '2021-01-05')['Adj Close'].
 ↪reindex(columns=assets)
returns = (df - df.shift(1))/df.shift(1)
```

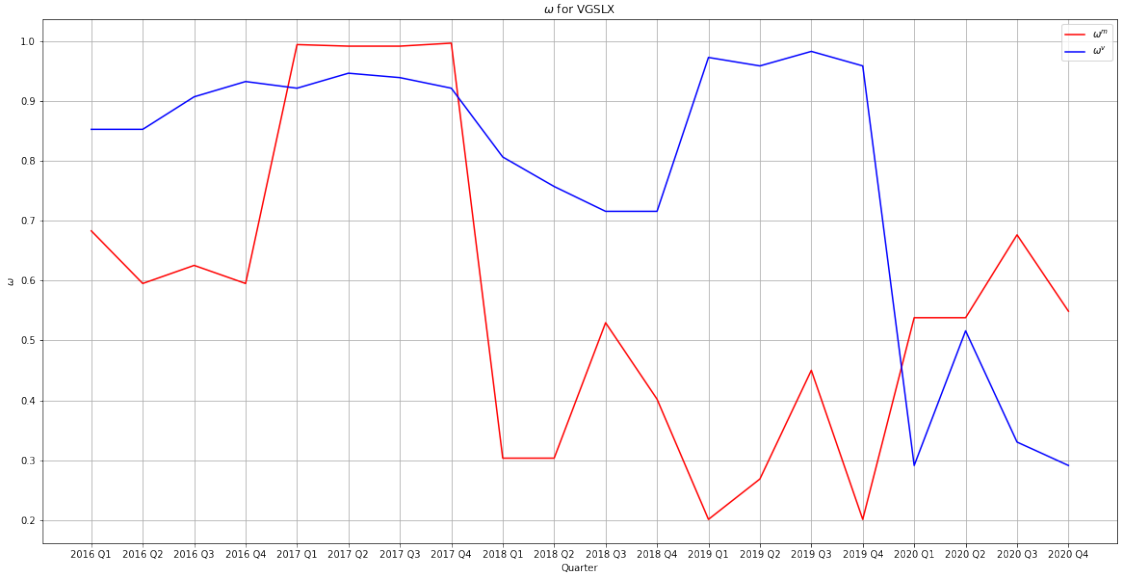
[7]:
```
a = np.matrix(dailyReturn.drop(['Date'], axis=1).values.tolist())
b = np.matrix(returns['2016-01-04': '2021-01-01'].values.tolist())
(a - b).max()
```

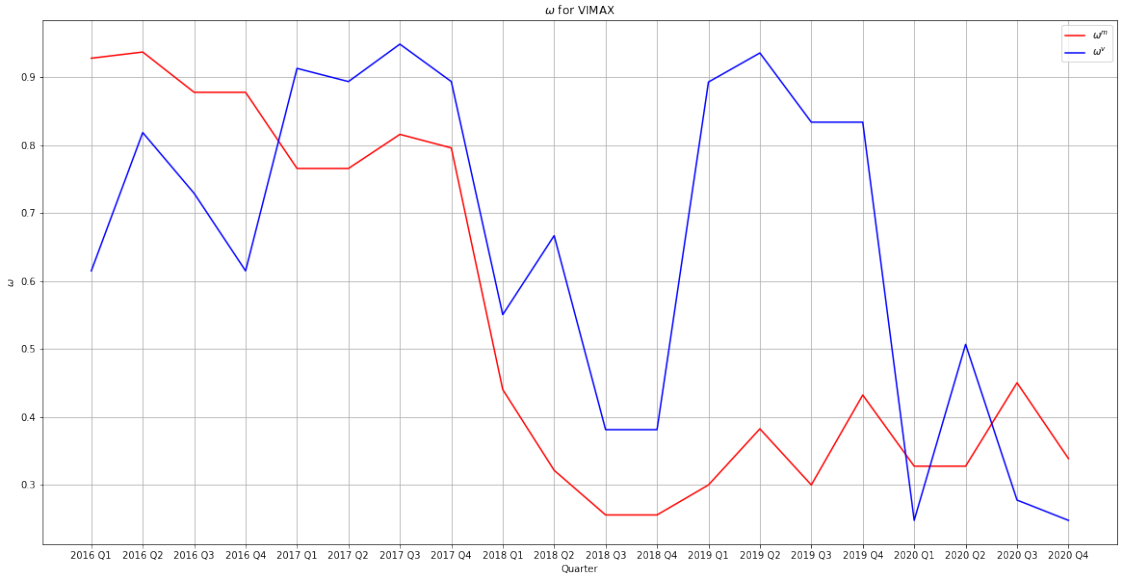
[7]:
```
1.0085114319612759e-07
```

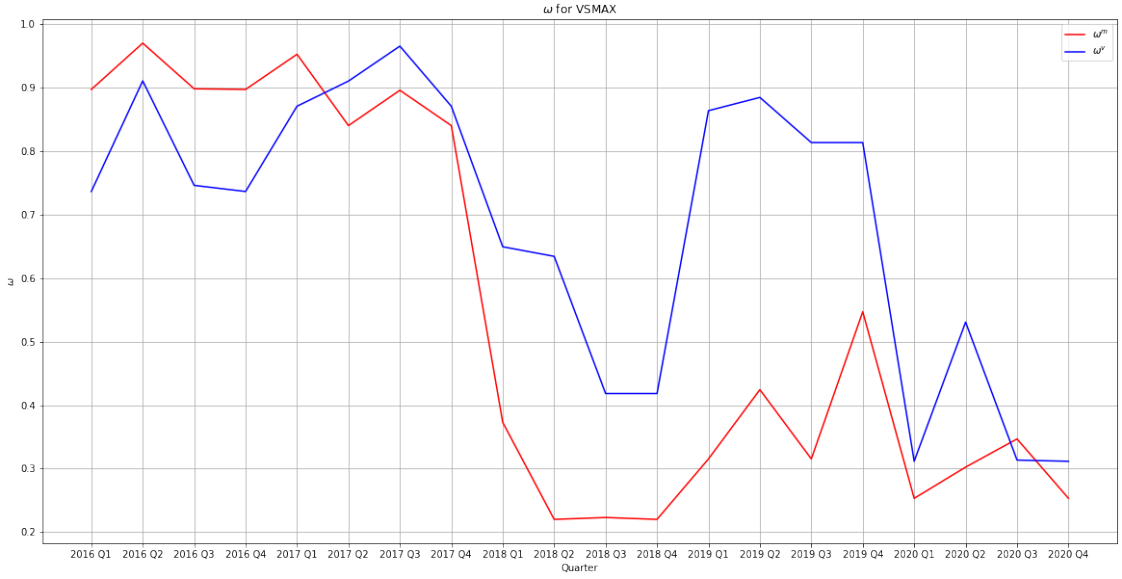
[8]:
```
for asset_num, asset in enumerate(returns.columns):
    cur_asset = returns[asset]
    plt.figure(num=asset_num, figsize=(20,10))

    x_labels = []
    omega_m_values = []
    omega_v_values = []

    for (year_start, year_end) in years:
        cur_year = year_start[:4]
        cur_year_returns = cur_asset[year_start:year_end]
        quarter_stats = []
        for quarter_index, (q_start, q_end) in enumerate(quarters):
            cur_quarter_returns = cur_asset[cur_year + q_start:cur_year + q_end]
            m1 = cur_quarter_returns.mean()
            v1 = cur_quarter_returns.var()
            D1 = len(cur_quarter_returns)

            quarter_stats.append((m1,v1,D1))
        for quarter_index, (m1,v1,D1) in enumerate(quarter_stats):

            other_quarters = [quarter_stats[j] for j in range(len(quarters)) if␣
 ↪j != quarter_index]

            omega_m = 1
            omega_v = 1

            for (m2, v2, D2) in other_quarters:
                omega_m = min(omega_m, 1/(1 + (m1 - m2)**2/(v1/D1 + v2/D2)))
                omega_v = min(omega_v, (4*v1*v2)/(v1+v2)**2)

            x_labels.append(cur_year + ' Q' + str(quarter_index+1))
            omega_m_values.append(omega_m)
            omega_v_values.append(omega_v)

    plt.plot(x_labels, omega_m_values, 'r', label='$\omega^m$')
    plt.plot(x_labels, omega_v_values, 'b', label='$\omega^v$')
    plt.title('$\omega$ for ' + asset)
    plt.ylabel('$\omega$')
    plt.xlabel('Quarter')
    plt.legend()
    plt.grid()
    plt.show()
```

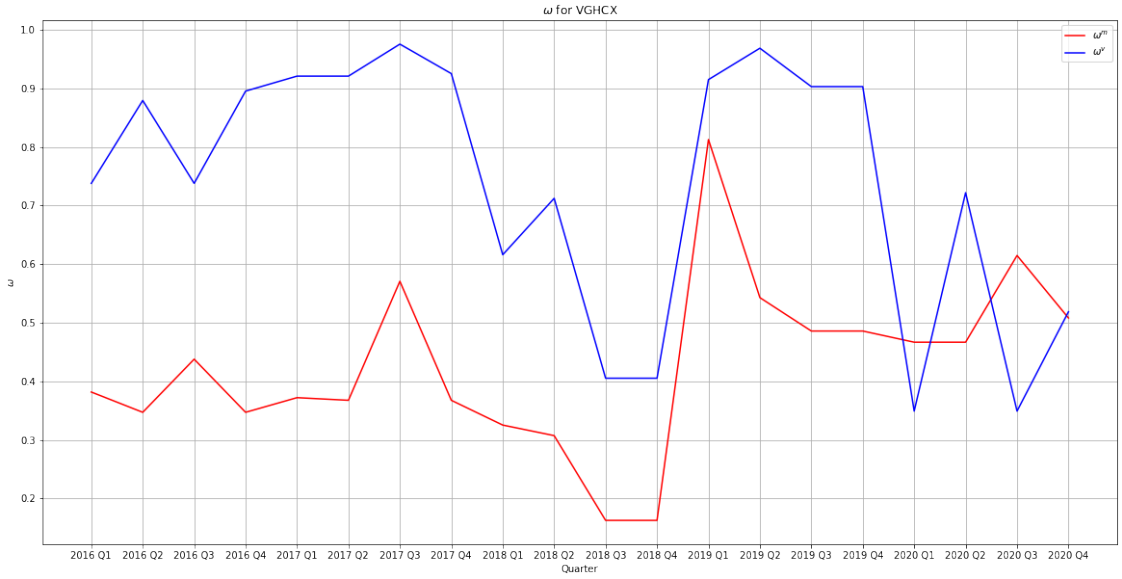

ω for VFIAX
ω
Quarter
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4

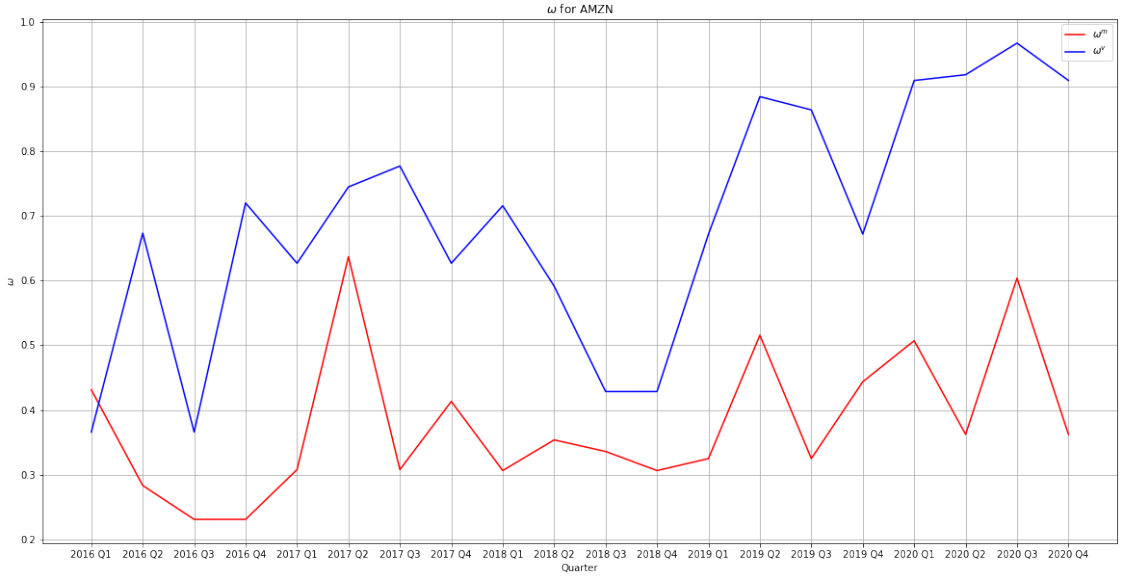

ω for VBTLX
ω
Quarter
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4

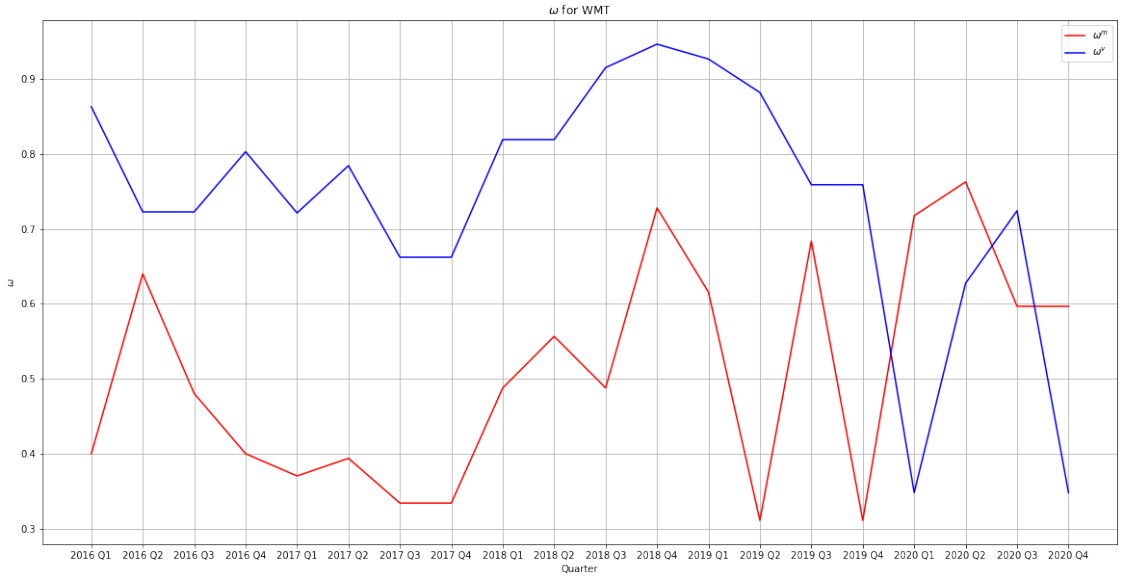

ω for VGSLX
ω
Quarter
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4

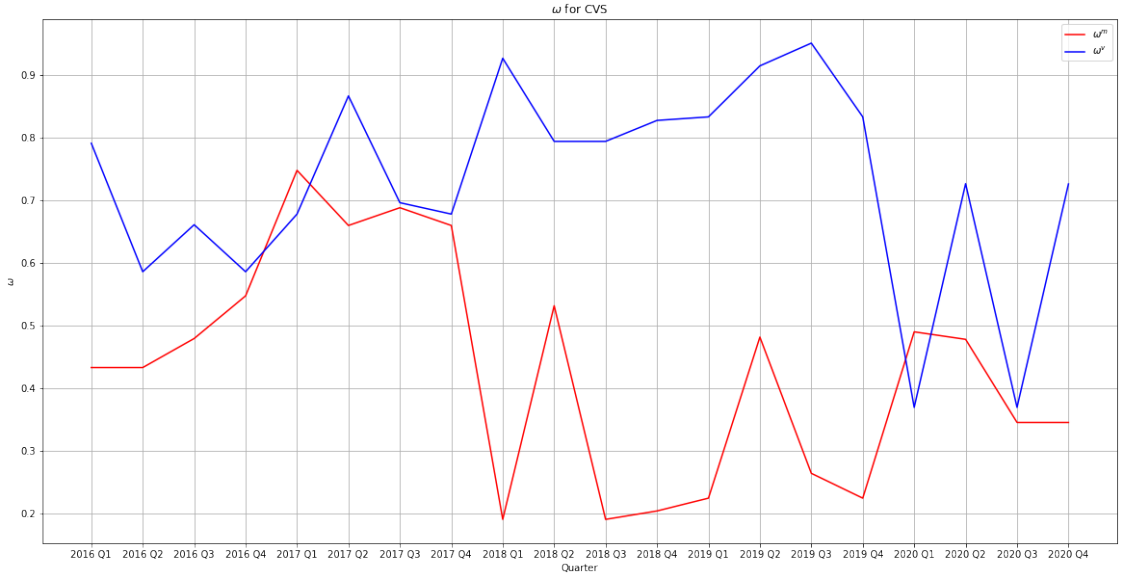

ω for VIMAX
ω
Quarter
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4


ω for VSMAX
ω
Quarter
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4


ω for VGHCX
ω
Quarter
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4


ω for AMZN
ω
Quarter
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4

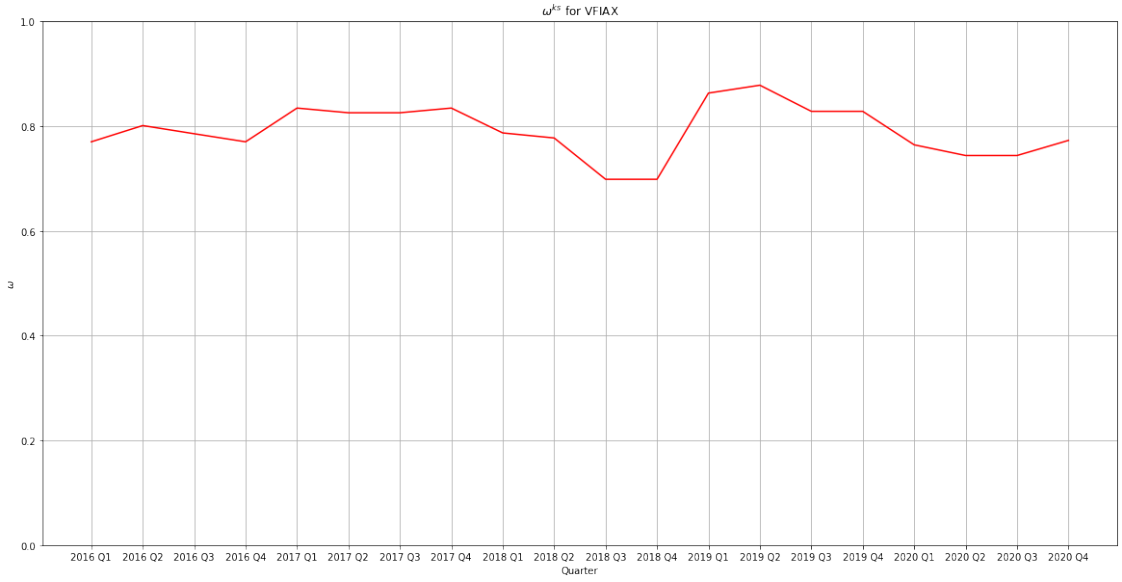

ω for WMT
ω
Quarter
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4

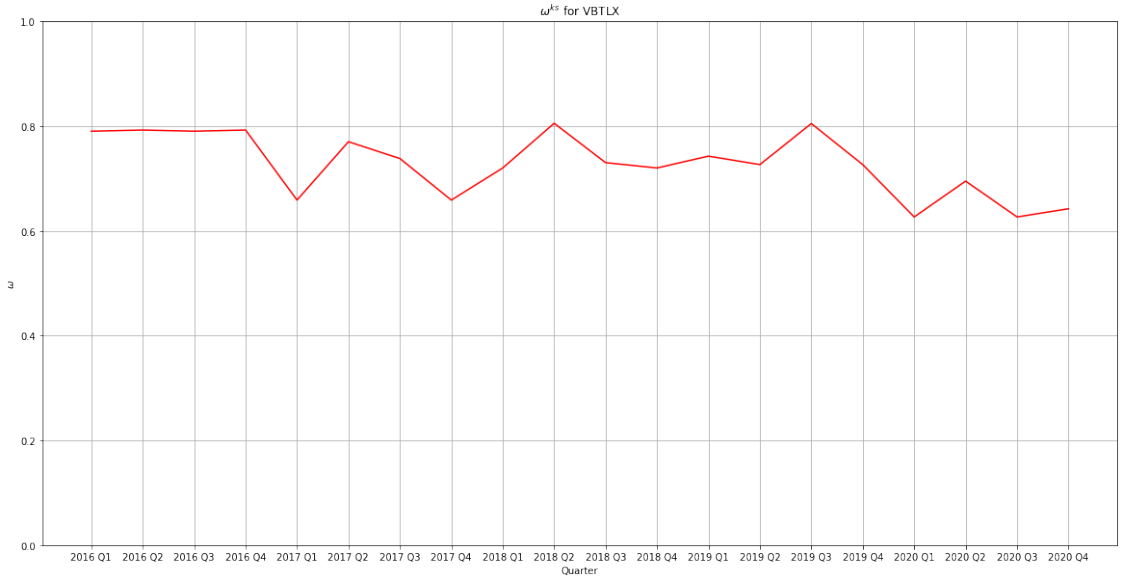

ω for CVS
ω^m
ω^κ
ω
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4
Quarter

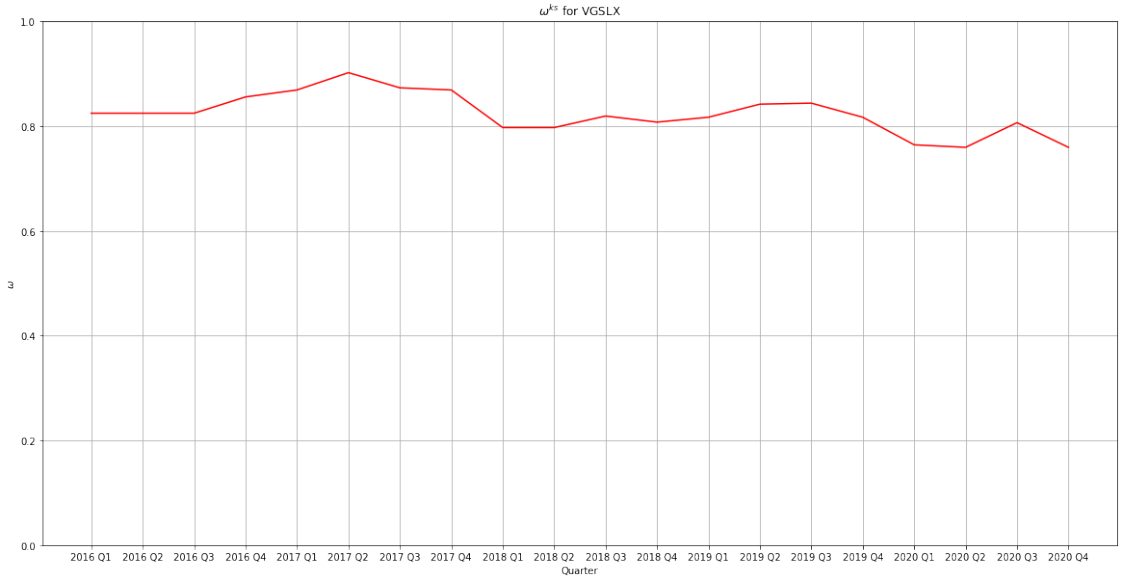
Exercise 2

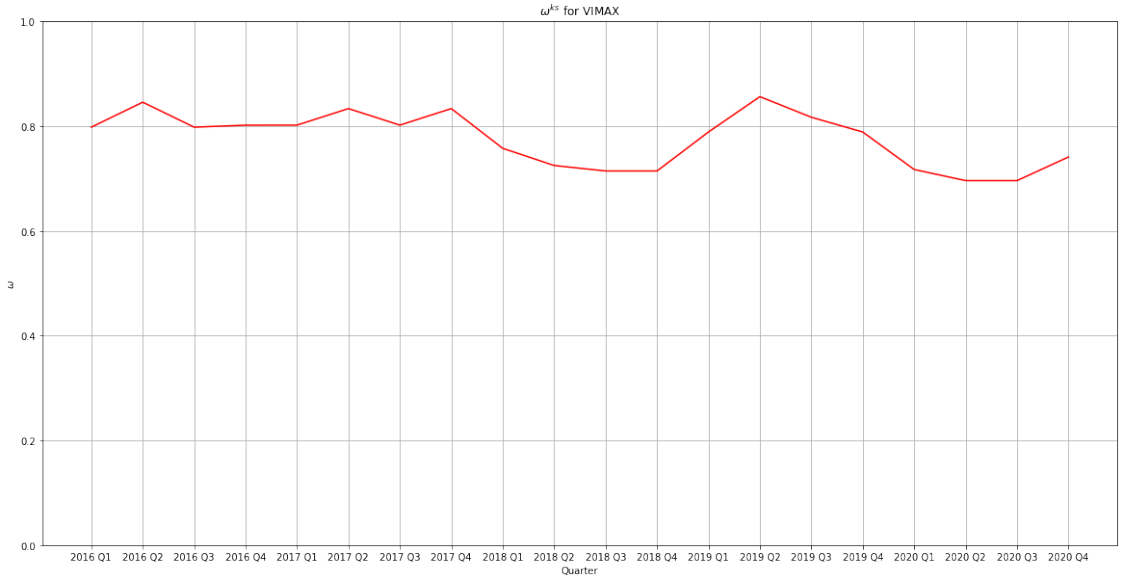
[9]:

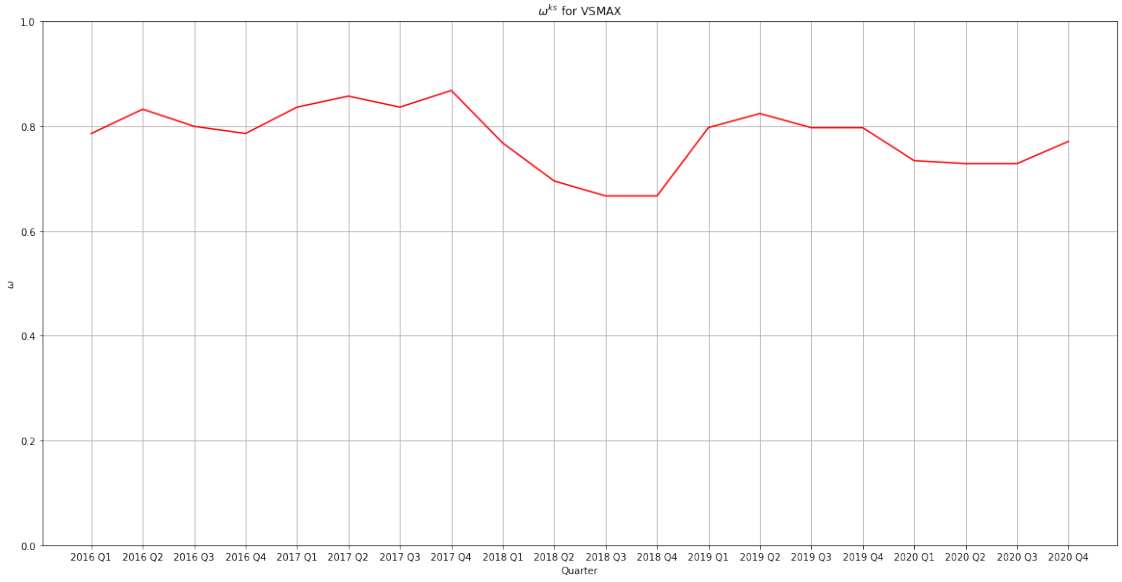
```
for asset_num, asset in enumerate(returns.columns):
    cur_asset = returns[asset]
    plt.figure(num=asset_num, figsize=(20,10))

    x_labels = []
    ks_values = []

    for (year_start, year_end) in years:
        cur_year = year_start[:4]
        cur_year_returns = cur_asset[year_start:year_end]

        for quarter_index, (q_start, q_end) in enumerate(quarters):

            cur_quarter_returns = cur_asset[cur_year + q_start:cur_year + q_end]

            other_quarters = [q for j, q in enumerate(quarters) if j !=␣
↪quarter_index]

            ks_val = 0

            for other_q_start, other_q_end in other_quarters:
                other_quarter_returns = cur_asset[cur_year + other_q_start:
↪cur_year + other_q_end]
                ks_val = max(ks_val, stats.ks_2samp(cur_quarter_returns,␣
↪other_quarter_returns)[0])

            x_labels.append(cur_year + ' Q' + str(quarter_index+1))
            ks_values.append(1-ks_val)

    plt.plot(x_labels, ks_values, 'r')
    plt.title('$\omega^{ks}$ for ' + asset)
    plt.ylabel('$\omega$')
    plt.xlabel('Quarter')
    plt.ylim((0,1))
    plt.grid()
    plt.show()
```

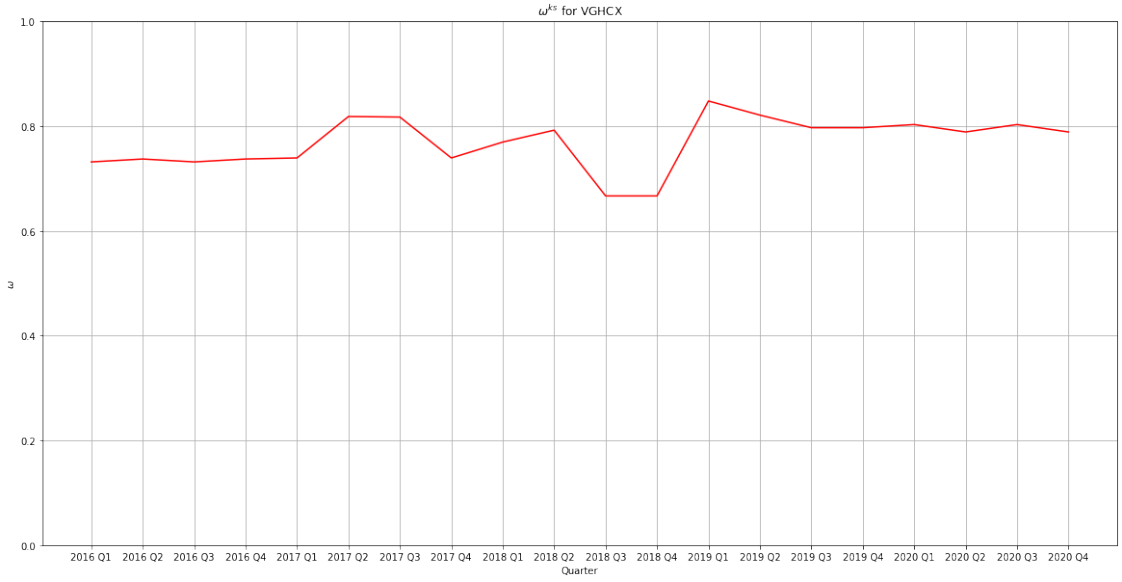

ω^{ks} for VFIAX
1.0
0.8
0.6
0.4
0.2
0.0
ω
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4
Quarter

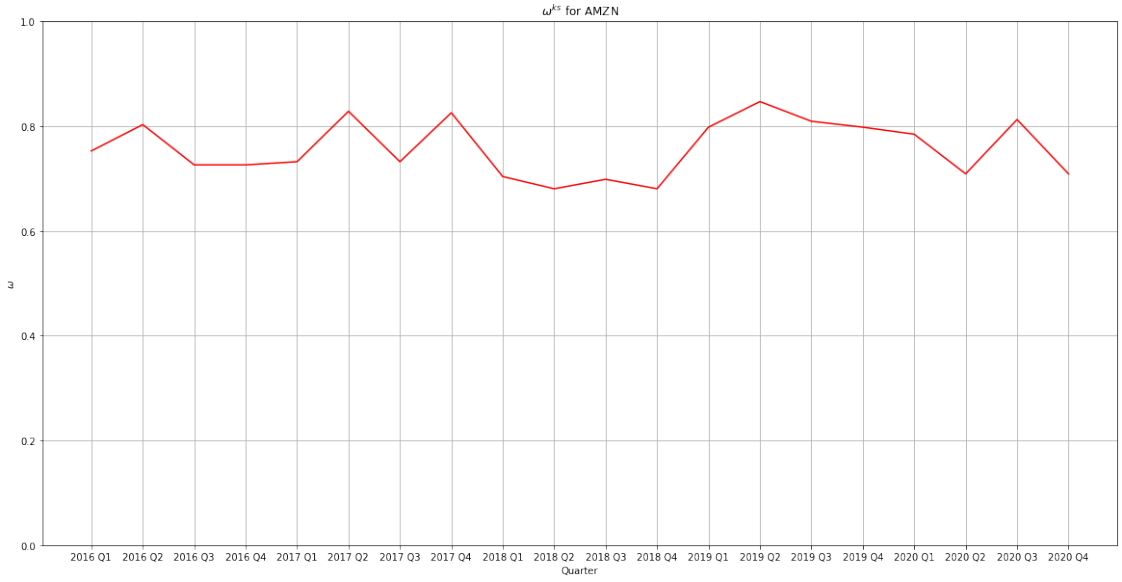

ω^{ks} for VBTLX
1.0
0.8
0.6
0.4
0.2
0.0
ω
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4
Quarter

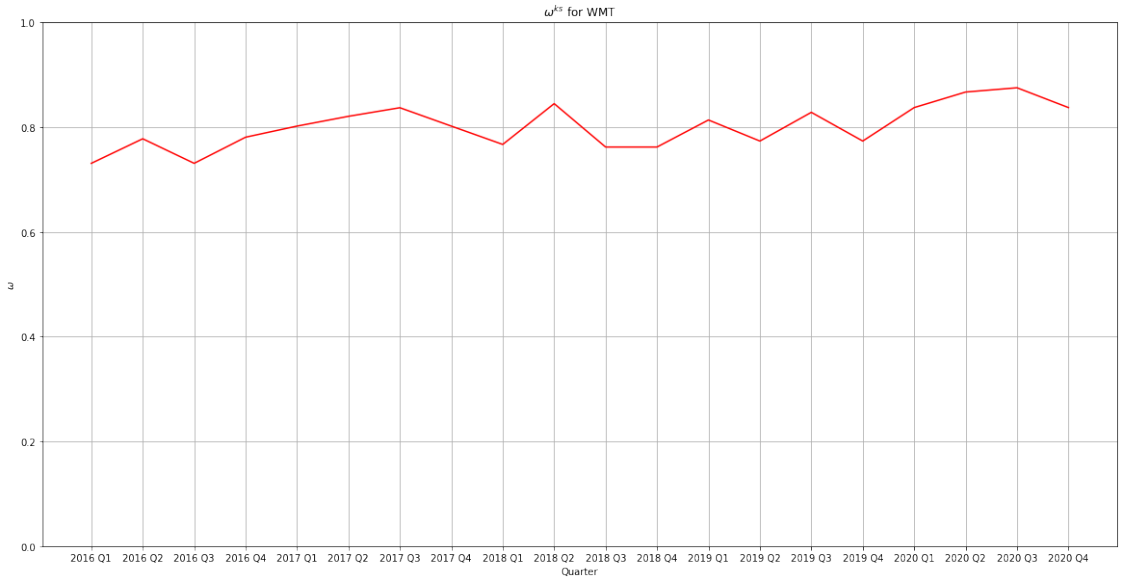

ω^{ks} for VGSLX
1.0
0.8
0.6
0.4
0.2
0.0
ω
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4
Quarter

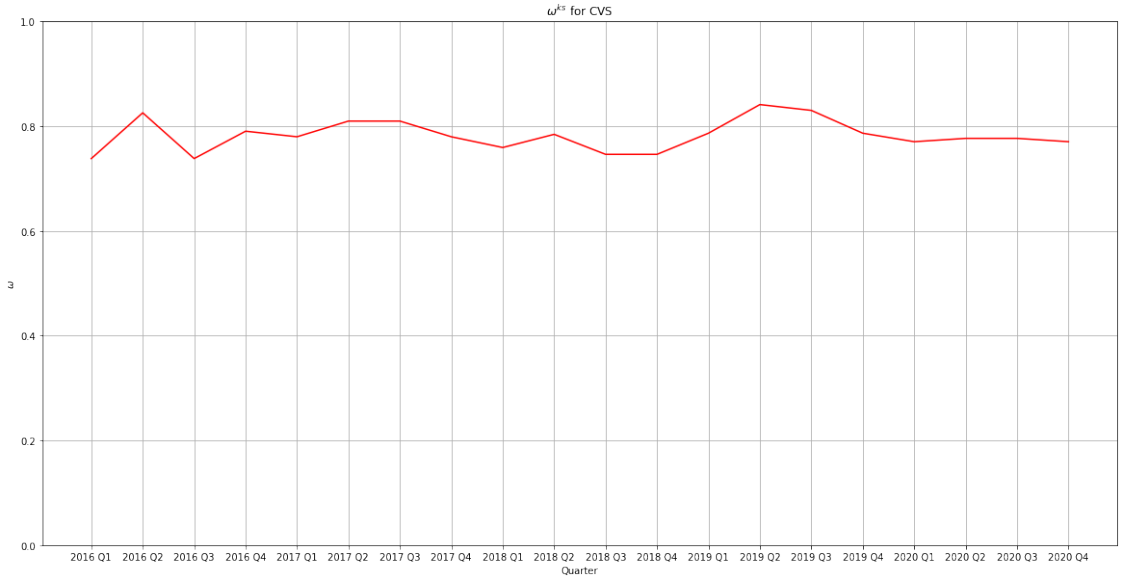

ω^{ks} for VIMAX
1.0
0.8
0.6
0.4
0.2
0.0
ω
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4
Quarter


ω^{ks} for VSMAX
1.0
0.8
0.6
0.4
0.2
0.0
ω
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4
Quarter


ω^{ks} for VGHCX
1.0
0.8
0.6
0.4
0.2
0.0
ω
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4
Quarter


ω^{ks} for AMZN
1.0
0.8
0.6
0.4
0.2
0.0
ω
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4
Quarter

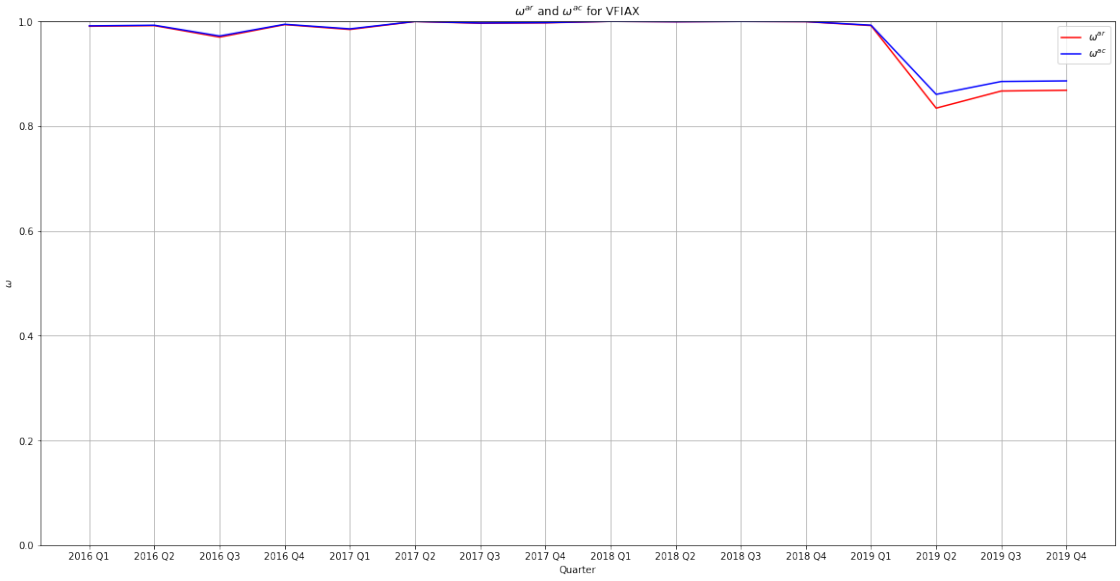

ω^{ks} for WMT
1.0
0.8
0.6
0.4
0.2
0.0
ω
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4
Quarter

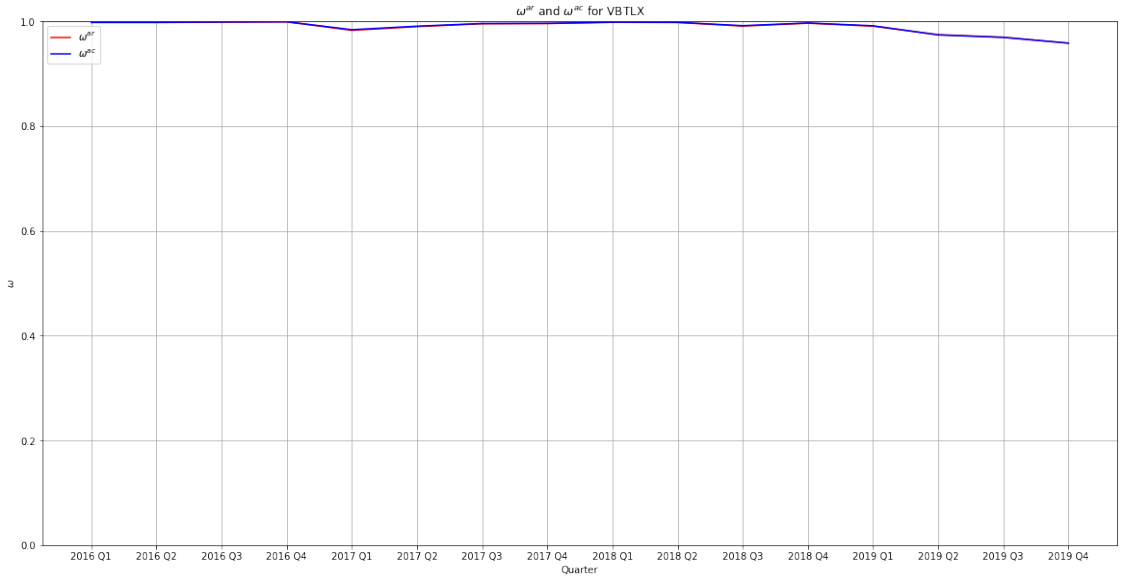

ω^{ks} for CVS
1.0
0.8
0.6
0.4
0.2
0.0
ω
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
2020 Q2
2020 Q3
2020 Q4
Quarter

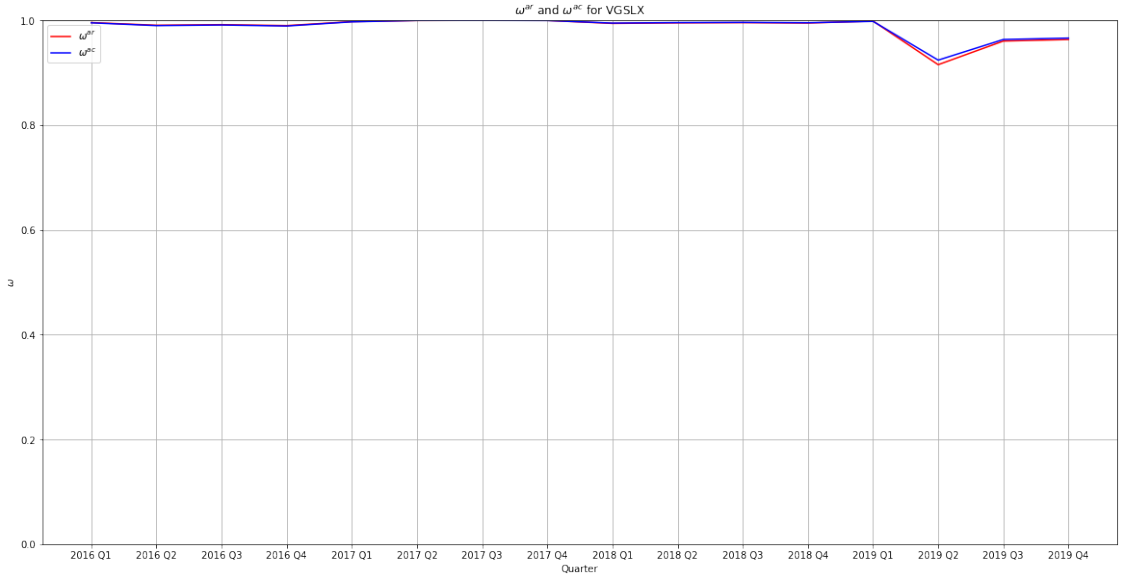
Exercise 3

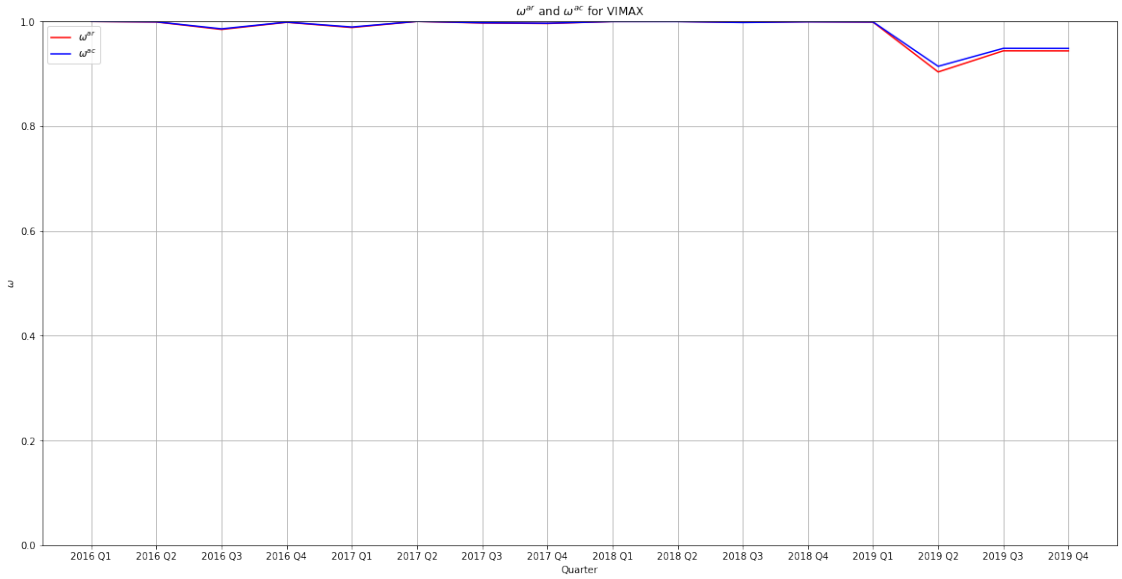
[11]:
```
for asset_num, asset in enumerate(returns.columns):
    cur_asset = returns[asset]
    plt.figure(num=asset_num, figsize=(20,10))

    x_labels = []
    ar_values = []
    ac_values = []

    for (year_start, year_end) in years[:-1]:
        cur_year = year_start[:4]
        for quarter_index, (q_start, q_end) in enumerate(quarters):
            year_diff = 1

            if quarter_index == 0:
                year_diff = 0

            period_start = cur_year + q_start
            period_end = str(int(cur_year) + year_diff) +␣
↪quarters[quarter_index-1][1]

            cur_period = cur_asset[period_start:period_end]

            m0 = cur_period.iloc[1:].mean()
            m1 = cur_period.iloc[:-1].mean()

            v00 = 0
            v10 = 0
            v11 = 0

            for day_return in cur_period.iloc[1:]:
                v00 += (day_return - m0)**2

            for day_return in cur_period.iloc[:-1]:
                v11 += (day_return - m1)**2

            for day in range(1, len(cur_period)):
                v10 += (cur_period.iloc[day-1] - m0)*(cur_period.iloc[day] - m1)


            v11 = v11/len(cur_period)
            v10 = v10/len(cur_period)
            v00 = v00/len(cur_period)
            W_mat = np.matrix([[1 - 1/len(cur_period), -(len(cur_period)- 1)/␣
↪(len(cur_period) * len(cur_period))], [-(len(cur_period)- 1)/ (len(cur_period)␣
↪* len(cur_period)), 1 - 1/len(cur_period)]])
```

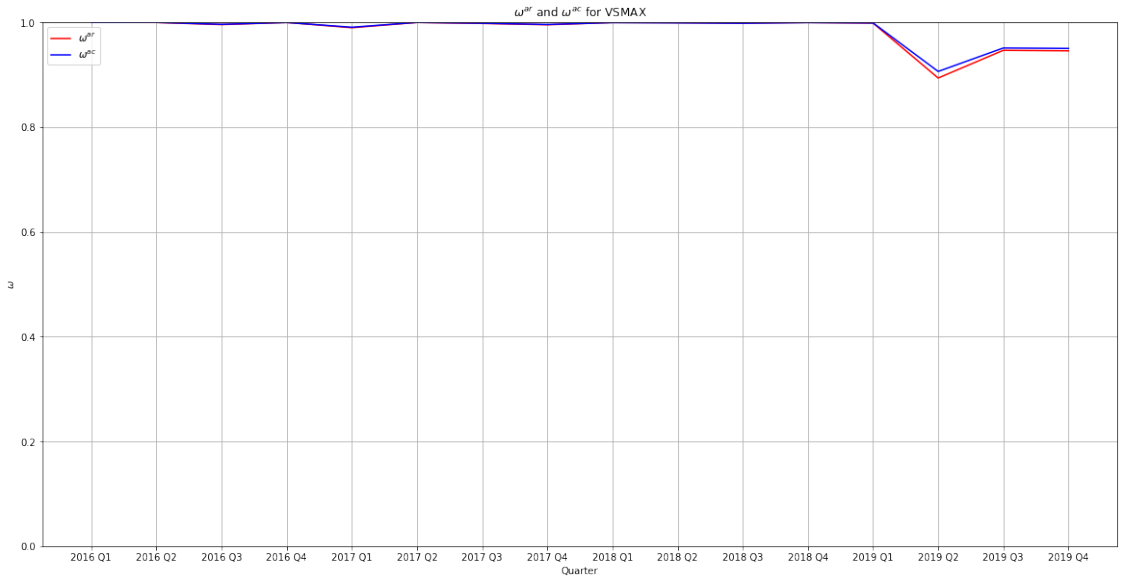
```
            V_mat = np.matrix([[v00, v10], [v10, v11 ]])

            omega_ar = 1 - (v10**2)/(v00*v11)
            omega_ac = np.square(np.trace(np.matmul(W_mat, V_mat)))/ (np.
 ↪trace(np.matmul(V_mat, V_mat)) * np.trace(np.matmul(W_mat, W_mat)))
            x_labels.append(cur_year + ' Q' + str(quarter_index+1))
            ar_values.append(omega_ar)
            ac_values.append(omega_ac)

    plt.plot(x_labels, ar_values, 'r', label='$\omega^{ar}$')
    plt.plot(x_labels, ac_values, 'b', label='$\omega^{ac}$')

    plt.title('$\omega^{ar}$ and $\omega^{ac}$ for ' + asset)

    plt.ylabel('$\omega$')
    plt.xlabel('Quarter')
    plt.ylim((0,1))
    plt.legend()
    plt.grid()
    plt.show()

### The data is plotted so the x value is equal to the year and quarter that␣
 ↪starts the year the data was taken from (2017 Q1 means 2017 Q1-Q4 used to␣
 ↪calculate value)
```

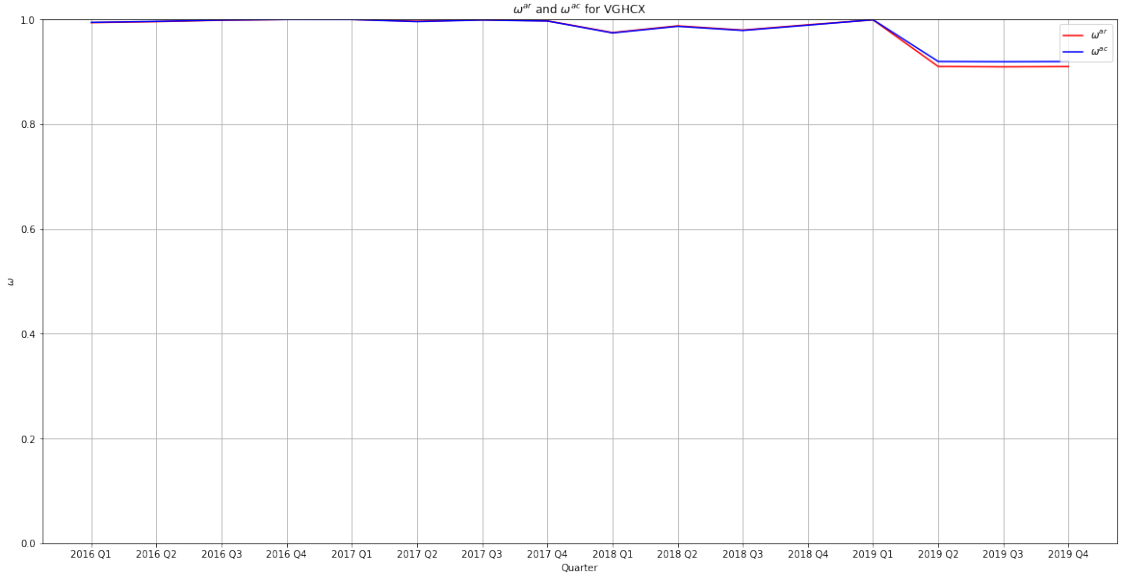


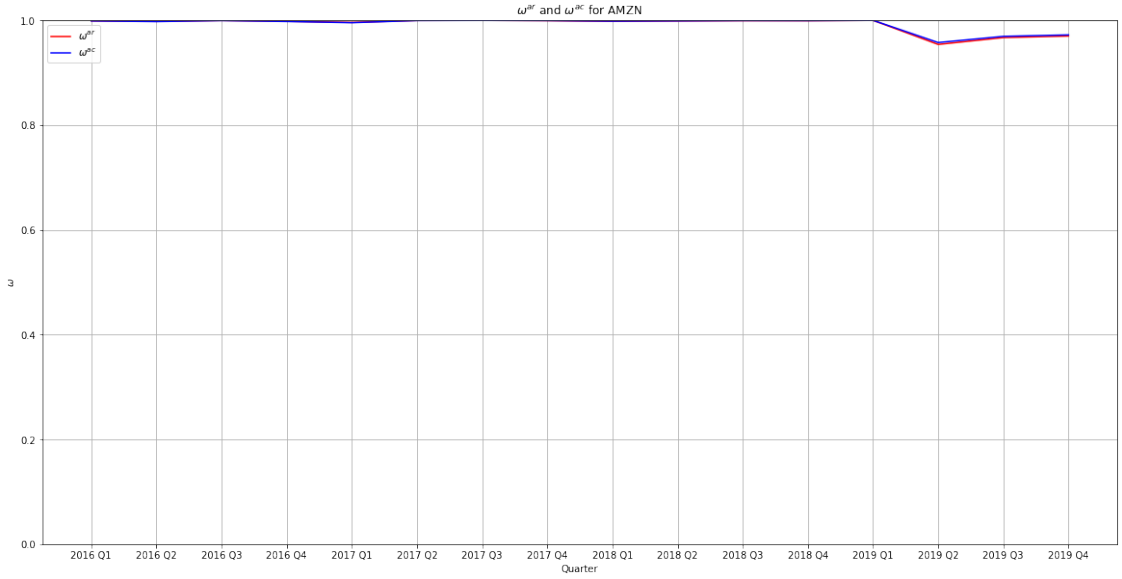

ω^{ar} and ω^{ac} for VBTLX
ω^{ar}
ω^{ac}
1.0
0.8
0.6
0.4
0.2
0.0
ω
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
Quarter

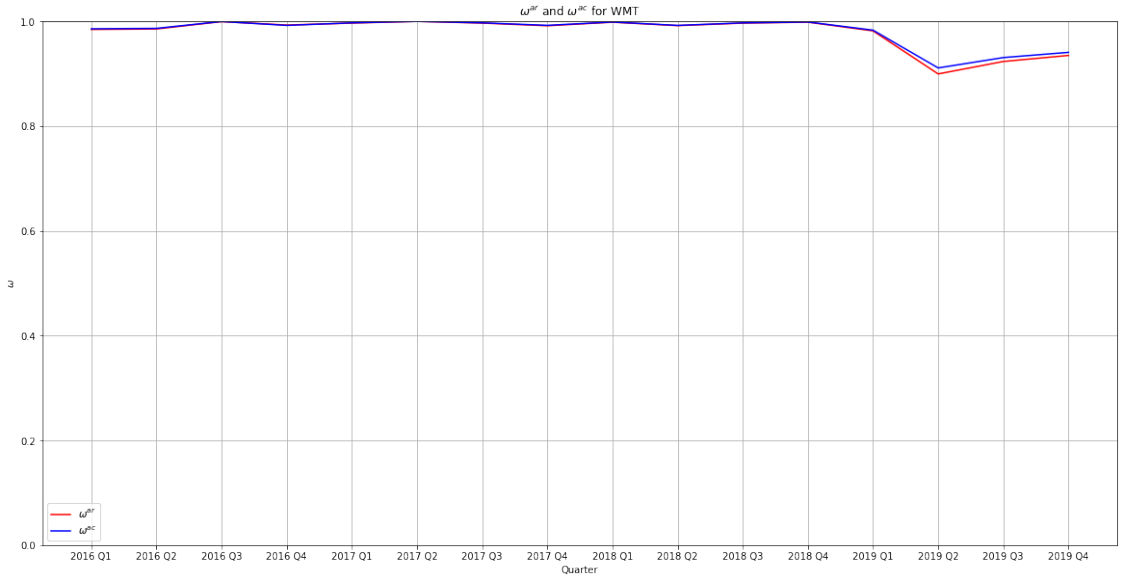

ω^{ar} and ω^{ac} for VGSLX
ω^{ar}
ω^{ac}
1.0
0.8
0.6
0.4
0.2
0.0
ω
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
Quarter

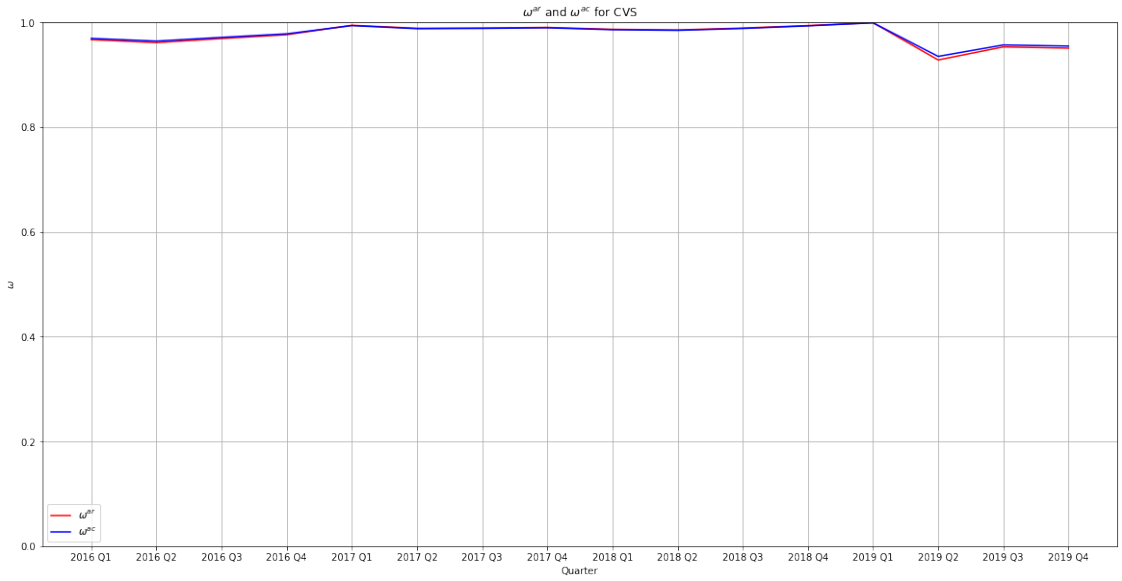

ω^{ar} and ω^{ac} for VIMAX
ω^{ar}
ω^{ac}
1.0
0.8
0.6
0.4
0.2
0.0
ω
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
Quarter


ω^{ar} and ω^{ac} for VSMAX
ω^{ar}
ω^{ac}
1.0
0.8
0.6
0.4
0.2
0.0
ω
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
Quarter


ω^{ar} and ω^{ac} for VGHCX
ω^{ar}
ω^{ac}
1.0
0.8
0.6
0.4
0.2
0.0
ω
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
Quarter


ω^{ar} and ω^{ac} for AMZN
ω^{ar}
ω^{ac}
1.0
0.8
0.6
0.4
0.2
0.0
ω
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
Quarter


ω^{ar} and ω^{ac} for WMT
1.0
0.8
0.6
0.4
0.2
0.0
ω
ω^{ar}
ω^{ac}
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
Quarter


ω^{ar} and ω^{ac} for CVS
1.0
0.8
0.6
0.4
0.2
0.0
ω
ω^{ar}
ω^{ac}
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
Quarter

Exercise 4

From exercise 2 - 3 we found that ω^{ks}, ω^{ar}, and ω^{ac} are all relatively near 1, which would indicate that IID models are a good fit for the data. We observed more variation from ω^{m} and ω^{v}. However it should be noted that ω^{m} and ω^{v} are possibly the weakest metric to assessing how well an IID model describes the data. With respect to one another ω^{v} is better than ω^{m} because variance captures more information than mean. We listed the major difference that we saw below.

2016 - VIMAX and VSMAX both had the highest ω^{m} and ω^{v} for that year, with (0.8, 0.95) and (0.9, 0.95) at their peaks in q2 respectively. As noted above they also had relatively high values for ω^{ks}, ω^{ar}, and ω^{ac} with (0.8, 1, 1) for both metrics for 2016. Together it would indicate that VIMAX and VSMAX are better described by IID models for 2016.

2017 - VGSLX had a ω^{m} sitting very close to 1 for all 4 quarters and the ω^{m} was similarly high, having values above 0.9 throughout the year. The ω^{ks} actually reached its peak of near 0.9 in 2017 Q2 and had values near this peak for the other 3 quarters. From the graph, the values for ω^{ar} and ω^{ac} are nearly indistinguishable from 1. All metrics lead us to believe that VGSLX is best described as an IID model for the year.

2018 - Due to the discrepancy between the trade war in quarter 4 with the rising markets in all other quarters, most of the assets see a drop in ω^{m}, ω^{v}, and ω^{ks} from their 2017 levels. During this time, the two assets that made the best case for an IID model were WMT and VGSLX. For Walmart, we see a rising ω^{v}, leading to the peak of the period in 2018 Q4 (near 0.95). Though the ω^{m} did not fare as well, it was still hovering around and above 0.5 and even reached 0.7 in Q4. The ω^{ks} hovers very close to 0.8, sitting slightly above or below for all 4 quarters and we see a ω^{ar} / ω^{ac} hugging 1 once again for the year. For VBTLX, there was a strong ω^{v} throughout the year, 0.9 for all 4 quarters. The ω^{m} was more volatile jumping from values near 0.35 in Q1 and Q4 to 0.6 in Q2 and Q3. The ω^{ks} was sitting above 0.7 for Q1, Q3 and Q4 while it hit one of its peaks of the period of above 0.8 in Q2. As before, the ω^{ar} and ω^{ac} are sitting nearly at 1 for the year. All these values for both assets in comparison to the others for the year make a strong case for them being best described as IID for the year out of the others.

2019 - Many of the assets make strong cases for being IID in 2019. All assets saw their ω^{v} sitting quite high, above 0.8 in almost all cases, however saw substantially lower values for ω^{m} with very few cases of the values breaking 0.6 in any quarter. ω^{ks} sits near 0.8 for all assets throughout the year and ω^{ar}/ω^{ac} sit almost at 1 for the year. Of all the assets, VGHCX sees some of the highest ω^{v} (above 0.9 for each quarter) and ω^{m} (peaks above 0.8 in Q1 then sits near 0.5). Similarly, its ω^{ks} is at its highest values at or above 0.8. Though this one stands out slightly, all assets seem to be strongly IID for 2019.

2020 - During 2020 most of our assets had fairly low metrics value for ω^{m}, and ω^{v}. The one exception to this was AMZN which had consistent ω^{v} values above 0.9. The ω^{m} values for siad year were fairly low. However mean is not a good indicator of how similar two distributions are to one another. Whereas, variance provides a better descriptor of two distributions, thus makes it a better indication of the similarities of two distributions. In terms of ω^{ks}, ω^{ar}, and ω^{ac} for 2020 their values sat around 0.75, 1, 1 respectively for each metric. Therefore we find that for 2020 AMZN was better described by IID models for 2020. Also it should be noted that with respect to all our assets AMZN most benefited from the coronavirus pandemic.