Portfolio Team Homework 7

Kyle Chang, Christian Warren, Dev Vadalia

April 19, 2021

[15]:
```
import yfinance as yf
import pandas as pd
import numpy as np
import matlab.engine
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
from matplotlib.lines import Line2D
import matplotlib.pyplot as plt
from cvxopt import matrix, solvers
import math
solvers.options['show_progress'] = False

eng = matlab.engine.start_matlab()
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')
]
mu_si = np.array([0.22, 0.33, 0.30, 0.33, 0.54, 0.63, 0.98, 1.02, 1.44, 1.63, 1.
 ↪91, 2.13, 2.42, 2.44, 2.35, 1.98, 1.54, 1.12, 0.14, 0.12])/100
mu_cl = np.add(mu_si, [.03]*20)
mu_si = mu_si/252
mu_cl = mu_cl/252

mu_safe = {}
mu_credit = {}

for year_index, year in enumerate(years):
    for quarter_index in range(4):
        mu_safe[year[0][:4] + ' Q' + str(quarter_index + 1)] =␣
 ↪mu_si[year_index*4 + quarter_index]
        mu_credit[year[0][:4] + ' Q' + str(quarter_index + 1)] =␣
 ↪mu_cl[year_index*4 + quarter_index]
```

```
mu_si = mu_safe
mu_cl = mu_credit

# 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)
```

[16]:
```
# Tested and returns same as matlab
def compute_y(var):
    ones = eng.transpose(matlab.double([1]*len(var)))
    return eng.mldivide(matlab.double(var.values.tolist()), matlab.double(ones)).
 ↪_data.tolist()

# Tested and returns same as matlab
def compute_z(mean, variance):
    return eng.mldivide(matlab.double(variance.values.tolist()), eng.
 ↪transpose(matlab.double(mean.values.tolist())))._data.tolist()

# np.dot values return the same as matlab for test data
def compute_abc(mean, variance):
    y = compute_y(variance)
    z = compute_z(mean, variance)
    ones = [1]*len(mean)
    # a = sum(eng.times(ones, matlab.double(y))._data)
    # b = sum(eng.times(ones, matlab.double(z))._data)
    # c = sum(eng.times(ones, matlab.double(mean.values.tolist()))._data)
    return np.sum(np.array(ones).conj()*y, axis=0), np.sum(np.array(ones).
 ↪conj()*z, axis=0), np.sum(np.array(mean.values.tolist()).conj()*z, axis=0)
    # return a,b,c

# returns same values as matlab
def compute_mv_vars(mean, variance):
    a,b,c = compute_abc(mean, variance)
    sigma_mv = 1/(a**0.5)
    mu_mv = b/a
    nu_as = ((a*c - b**2)/a)**0.5
    return sigma_mv, mu_mv, nu_as


def is_solvent(portfolio, returns):
    return not any(y <= 0 for y in [1 + np.dot(portfolio, returns.iloc[day]) for␣
 ↪day in range(len(returns))])
```

```
def is_long(portfolio):
    return not any(y < 0 for y in portfolio.tolist())
```

[17]:
```
class Portfolio:
    def __init__(self, assets, name, start_date='2015-12-28',␣
↪end_date='2021-01-05'):
        self.data = yf.download(assets, start_date, end_date)['Adj Close']
        self.data = self.data.reindex(columns=assets)
        self.returns = (self.data - self.data.shift(1))/self.data.shift(1)
        self.start_date = start_date
        self.end_date = end_date
        self.portfolio_name = name

    # Returns mean and variance of portfolio for period
    def get_mean_and_variance(self, period_start = None, period_end = None):

        # Can add checks to make sure dates are valid with data that is contained
        if period_start is not None and period_end is not None:
            return self.returns[period_start:period_end].mean(), self.
↪returns[period_start:period_end].cov()

        return self.returns.mean(), self.returns.cov()

    # Returns minimum volatility portfolio allocation and parameters
    def get_minimum_volatility_portfolio_parameters(self, period_start = None,␣
↪period_end = None):
        if period_start is None and period_end is None:
            period_start = self.start_date
            period_end = self.end_date

        m, V = Portfolio.get_mean_and_variance(self, period_start, period_end)

        sigma_mv, mu_mv, nu_as = compute_mv_vars(m, V)
        y = compute_y(V)

        f_mv = (sigma_mv**2)*np.array(y)

        return pd.Series(data=f_mv, index=self.data.columns), sigma_mv, mu_mv,␣
↪nu_as

    # Returns safe tangent portfolio allocation if it exists and relevant␣
↪parameters
    def get_safe_tangent_portfolio_parameters(self, mu_si, period_start = None,␣
↪period_end = None):
        if period_start is None and period_end is None:
            period_start = self.start_date
```

```
            period_end = self.end_date

        m, V = Portfolio.get_mean_and_variance(self, period_start, period_end)

        f_mv, sigma_mv, mu_mv, nu_as = Portfolio.
↪get_minimum_volatility_portfolio_parameters(self, period_start, period_end)

        if mu_mv < mu_si:
            return None

        sigma_st = sigma_mv*(1 + ((nu_as*sigma_mv)/(mu_mv-mu_si))**2)**0.5
        mu_st = mu_mv + (nu_as**2)*(sigma_mv**2)/(mu_mv-mu_si)
        nu_st = nu_as*(1 + ((mu_mv-mu_si)/(nu_as*sigma_mv))**2)**0.5

        y = compute_y(V)
        z = compute_z(m, V)

        f_st = (sigma_mv**2/(mu_mv-mu_si))*(z - mu_si*np.array(y)).
↪reshape((len(m)))

        return pd.Series(data=f_st, index=self.data.columns), sigma_st, mu_st,␣
↪nu_st

    def get_credit_tangent_portfolio_parameters(self, mu_cl, period_start =␣
↪None, period_end = None):
        if period_start is None and period_end is None:
            period_start = self.start_date
            period_end = self.end_date

        m, V = Portfolio.get_mean_and_variance(self, period_start, period_end)

        f_mv, sigma_mv, mu_mv, nu_as = Portfolio.
↪get_minimum_volatility_portfolio_parameters(self, period_start, period_end)

        if mu_mv < mu_cl:
            return None

        sigma_ct = sigma_mv*(1 + ((nu_as*sigma_mv)/(mu_mv-mu_cl))**2)**0.5
        mu_ct = mu_mv + (nu_as**2)*(sigma_mv**2)/(mu_mv-mu_cl)
        nu_ct = nu_as*(1 + ((mu_mv-mu_cl)/(nu_as*sigma_mv))**2)**0.5

        y = compute_y(V)
        z = compute_z(m, V)

        f_ct = (sigma_mv**2/(mu_mv-mu_cl))*(z - mu_cl*np.array(y)).
↪reshape((len(m)))
```

```
        return pd.Series(data=f_ct, index=self.data.columns), sigma_ct, mu_ct,␣
↪nu_ct

    def get_long_tangent_portfolio(self, mu_si, period_start=None,␣
↪period_end=None):
        if period_start is None and period_end is None:
            period_start = self.start_date
            period_end = self.end_date

        m, V = Portfolio.get_mean_and_variance(self, period_start, period_end)

        mu_start = min(m)
        mu_end = max(m)
        step_size = (mu_end - mu_start)/300

        long_mus = np.arange(mu_start, mu_end + step_size, step_size)

        # Long Frontier
        Q = matrix(V.values, tc='d')
        z = matrix(np.zeros((len(V))).tolist(), tc='d')
        I = matrix((-1*np.identity(len(V))), tc='d')
        A = matrix(np.array([np.ones((len(V))).tolist(), m.values.tolist()]),␣
↪tc='d')

        nu_ca = (None, None, None, -100)

        # Will need to adjust for time periods longer than a year
        year = period_start[:4]

        for cur_mu in long_mus:
            deq = matrix(np.array([1, cur_mu]), tc='d')
            sol = solvers.qp(Q, z, I, z, A, deq)
            current_long_allocation = np.reshape(np.array(sol['x']), (len(m)))

            sigma_lf = (np.matmul(np.matmul(current_long_allocation, V.values),␣
↪np.reshape(current_long_allocation, (len(current_long_allocation),1)))[0])**0.5

            # Finding capital allocation line with greatest slope
            curNu_ca = (cur_mu - mu_si)/sigma_lf

            if curNu_ca > nu_ca[-1]:
                nu_ca = (current_long_allocation, sigma_lf, cur_mu, curNu_ca)

        return nu_ca
```

[29]:
```
def calculateSignalToNoiseForEstimators(expectedValue, stdDev, gamma):
    return np.abs((expectedValue - gamma)/ stdDev)

def getVarianceEstimators(f_vector, mean, Var, expectedValue, stdDev):

    estimators = pd.DataFrame(columns = ['estimator name','estimator value',␣
 ↪'signal to Noise'])
    gamma_q = np.transpose(mean) @ f_vector - (np.transpose(f_vector)@ (mean @␣
 ↪np.transpose(mean) + Var) @ f_vector) * (1/2)
    gamma_p = np.transpose(mean) @ f_vector - (np.transpose(f_vector)@ (Var) @␣
 ↪f_vector) * (1/2)
    gamma_t = np.log(1 + np.transpose(mean) @ f_vector) - (np.
 ↪transpose(f_vector)@ (Var) @ f_vector) * (1/(2* np.square(1 + np.
 ↪transpose(mean) @ f_vector)))
    gamma_r = np.log(1 + np.transpose(mean) @ f_vector) - (np.
 ↪transpose(f_vector)@ (Var) @ f_vector) * (1/2)
    gamma_s = np.log(1 + np.transpose(mean) @ f_vector) - (np.
 ↪transpose(f_vector)@ (Var) @ f_vector) * (1/(2* (1 + 2 * np.transpose(mean) @␣
 ↪f_vector)))

    estimators.loc[len(estimators.index)] = ['gamma q', gamma_q,␣
 ↪calculateSignalToNoiseForEstimators(expectedValue, stdDev, gamma_q)]
    estimators.loc[len(estimators.index)] = ['gamma p', gamma_p,␣
 ↪calculateSignalToNoiseForEstimators(expectedValue, stdDev, gamma_p)]
    estimators.loc[len(estimators.index)] = ['gamma t', gamma_t,␣
 ↪calculateSignalToNoiseForEstimators(expectedValue, stdDev, gamma_t)]
    estimators.loc[len(estimators.index)] = ['gamma r', gamma_r,␣
 ↪calculateSignalToNoiseForEstimators(expectedValue, stdDev, gamma_r)]
    estimators.loc[len(estimators.index)] = ['gamma s', gamma_s,␣
 ↪calculateSignalToNoiseForEstimators(expectedValue, stdDev, gamma_s)]
    return estimators
```

[30]:
```
groupA = Portfolio(['VFIAX', 'VBTLX', 'VGSLX'], 'Group A')
groupAB = Portfolio(['VFIAX', 'VBTLX', 'VGSLX', 'VIMAX', 'VSMAX', 'VGHCX'],␣
 ↪'Group AB')
groupABC = Portfolio(['VFIAX', 'VBTLX', 'VGSLX', 'VIMAX', 'VSMAX', 'VGHCX',␣
 ↪'AMZN', 'WMT', 'CVS'], 'Group ABC')
```

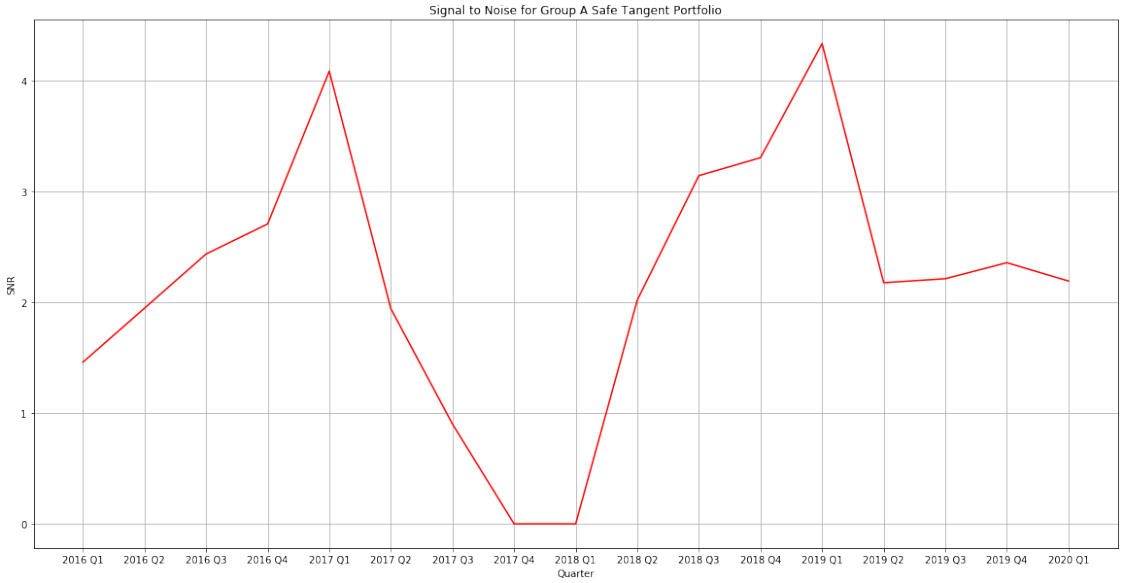
[31]:
```
lgd_groupA = mpatches.Patch(color='blue', label='Group A')
lgd_groupAB = mpatches.Patch(color='red', label='Group AB (And Assets Not␣
 ↪Included in A)')
lgd_groupABC = mpatches.Patch(color='green', label='Group ABC (And Assets Not␣
 ↪Included in AB)')
assets = Line2D([0], [0], marker='o', color='w', markerfacecolor='k',␣
 ↪label='Assets', markersize=12)
```

```
rf = Line2D([0], [0], marker='D', color='w', markerfacecolor='k', label='Risk␣
 ↪Free Rates', markersize=12)
ctg = Line2D([0], [0], marker='*', color='w', markerfacecolor='k', label='Credit␣
 ↪Tangent Portfolio', markersize=12)
stg = Line2D([0], [0], marker='^', color='w', markerfacecolor='k', label='Safe␣
 ↪Tangent Portfolio', markersize=12)
ltg = Line2D([0], [0], marker='X', color='w', markerfacecolor='k', label='Long␣
 ↪Tangent Portfolio', markersize=12)
equi = Line2D([0], [0], marker='P', color='w', markerfacecolor='k',␣
 ↪label='Equidistributed Portfolio', markersize=12)
mv_portfolio = Line2D([0], [0], marker='s', color='w', markerfacecolor='k',␣
 ↪label='Minimum Variance Portfolio', markersize=12)
mv_frontier = Line2D([0], [0], linewidth=1., color='k', label='Minimum Variance␣
 ↪Frontier', markersize=12)
efficient_frontier = Line2D([0], [0], linestyle='--', linewidth=1., color='k',␣
 ↪label='Efficient Frontier', markersize=12)
efficient_frontier.set_linestyle('--')
long_frontier = Line2D([0], [0], linestyle=':', linewidth=1., color='k',␣
 ↪label='Long Frontier', markersize=12)
long_frontier.set_linestyle(':')
lmtd_frontier_1 = Line2D([0], [0], linestyle='-.', linewidth=1., color='m',␣
 ↪label='Limited Leverage Frontier (l = 1)', markersize=12)
lmtd_frontier_1.set_linestyle('-.')
lmtd_frontier_5 = Line2D([0], [0], linestyle='-.', linewidth=1., color='c',␣
 ↪label='Limited Leverage Frontier (l = 5)', markersize=12)
lmtd_frontier_5.set_linestyle('-.')
```

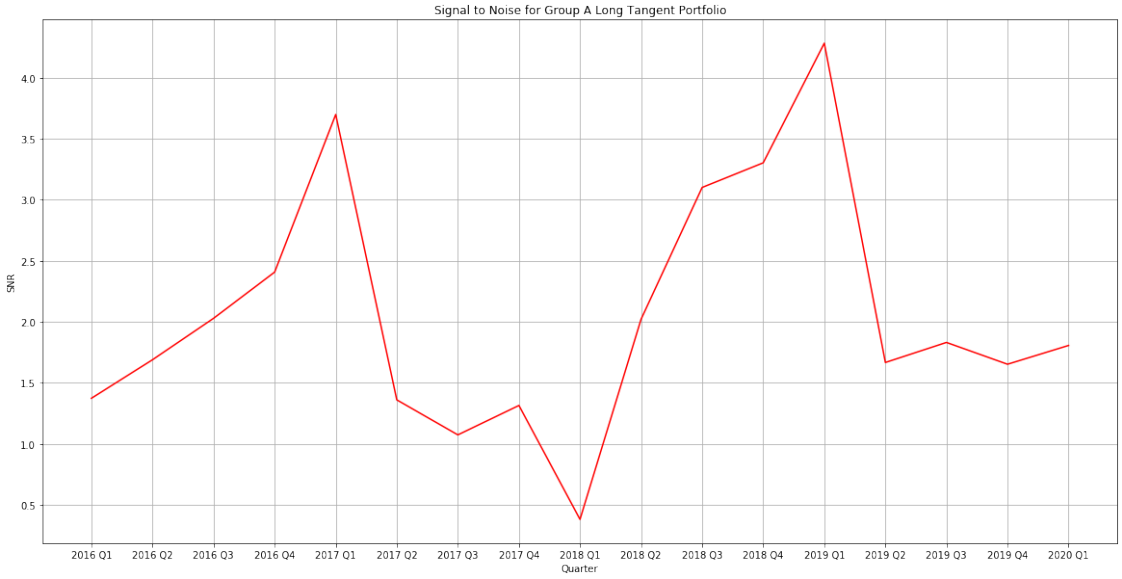
[32]:
```
for group_number, portfolio in enumerate([groupA, groupAB, groupABC]):
    results = [[], [], []]
    gamma_table = [pd.DataFrame(columns = ['Gamma Q', 'Gamma P', 'Gamma T',␣
 ↪'Gamma R', 'Gamma S']), pd.DataFrame(columns = ['Gamma Q', 'Gamma P', 'Gamma␣
 ↪T', 'Gamma R', 'Gamma S']), pd.DataFrame(columns = ['Gamma Q', 'Gamma P',␣
 ↪'Gamma T', 'Gamma R', 'Gamma S'])]
    x_axis = []

    for year_index, (y_start, y_end) in enumerate(years):
        cur_year = y_start[:4]
        for quarter_index, (q_start, q_end) in enumerate(quarters):

            cur_year_and_quarter = cur_year + ' Q' + str(quarter_index + 1)

            if cur_year != '2020' or (cur_year == '2020' and quarter_index == 0):

                period_start = y_start[:4]+ q_start
```

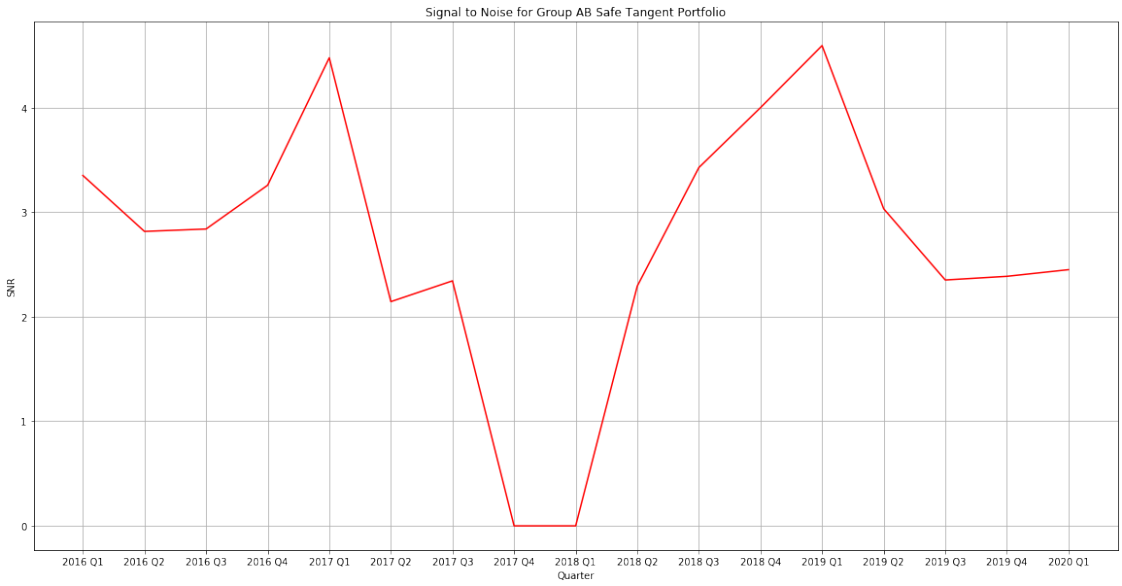
```
            period_end = str(int(cur_year) + 1) +␣
↪quarters[quarter_index-1][1] if quarter_index != 0 else cur_year +␣
↪quarters[quarter_index-1][1]

            period_returns = portfolio.returns[period_start:period_end]

            st_params = portfolio.
↪get_safe_tangent_portfolio_parameters(mu_si[cur_year_and_quarter],␣
↪period_start=period_start, period_end=period_end)
            ct_params = portfolio.
↪get_credit_tangent_portfolio_parameters(mu_cl[cur_year_and_quarter],␣
↪period_start=period_start, period_end=period_end)
            lt_params = portfolio.
↪get_long_tangent_portfolio(mu_si[cur_year_and_quarter],␣
↪period_start=period_start, period_end=period_end)

            for tangent_index, params in enumerate([st_params, ct_params,␣
↪lt_params]):
                if params is not None:
                    f, sigma, mu, _ = params

                    log_returns = (period_returns@f).apply(math.log1p)

                    uniformWeights = 1/log_returns.shape[0]

                    wBar = np.sum(np.square(np.ones(log_returns.shape[0]) *␣
↪uniformWeights))

                    mean = np.sum(log_returns, axis=0) * uniformWeights
                    difference = log_returns.subtract(mean)
                    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)[0][0]

                    results[tangent_index].append(signalToNoise)

                    m, V = portfolio.
↪get_mean_and_variance(period_start=period_start, period_end=period_end)

                    gammas = getVarianceEstimators(f, m, V, mean,␣
↪StdOfExpectedValue)
```

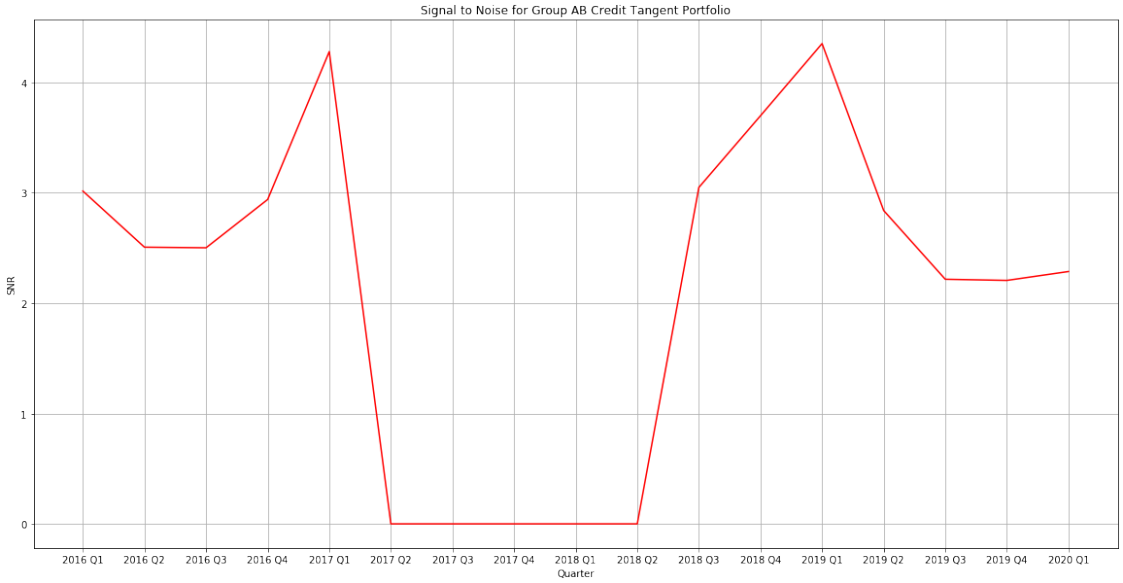
```
                        toAdd = []
                        for g in gammas.values[:,2]:
                            toAdd.append(g[0][0])
                        gamma_table[tangent_index].loc[cur_year_and_quarter] =␣
↪toAdd

                    else:
                        results[tangent_index].append(0)

                    if tangent_index == 0:
                        x_axis.append(cur_year_and_quarter)



    plt.figure(num=group_number*3, figsize=(20,10))
    plt.plot(x_axis, results[0], 'r')
    plt.title('Signal to Noise for ' + portfolio.portfolio_name + ' Safe Tangent␣
↪Portfolio')
    plt.ylabel('SNR')
    plt.xlabel('Quarter')
    plt.grid()
    plt.show()

    plt.figure(num=group_number*3 + 1, figsize=(20,10))
    plt.plot(x_axis, results[1], 'r')
    plt.title('Signal to Noise for ' + portfolio.portfolio_name + ' Credit␣
↪Tangent Portfolio')
    plt.ylabel('SNR')
    plt.xlabel('Quarter')
    plt.grid()
    plt.show()

    plt.figure(num=group_number*3 + 2, figsize=(20,10))
    plt.plot(x_axis, results[2], 'r')
    plt.title('Signal to Noise for ' + portfolio.portfolio_name + ' Long Tangent␣
↪Portfolio')
    plt.ylabel('SNR')
    plt.xlabel('Quarter')
    plt.grid()
    plt.show()

    display('Safe Tangent ' + portfolio.portfolio_name, gamma_table[0].style,␣
↪'Credit Tangent ' + portfolio.portfolio_name, gamma_table[1].style, 'Long␣
↪Tangent ' + portfolio.portfolio_name, gamma_table[2].style)
```

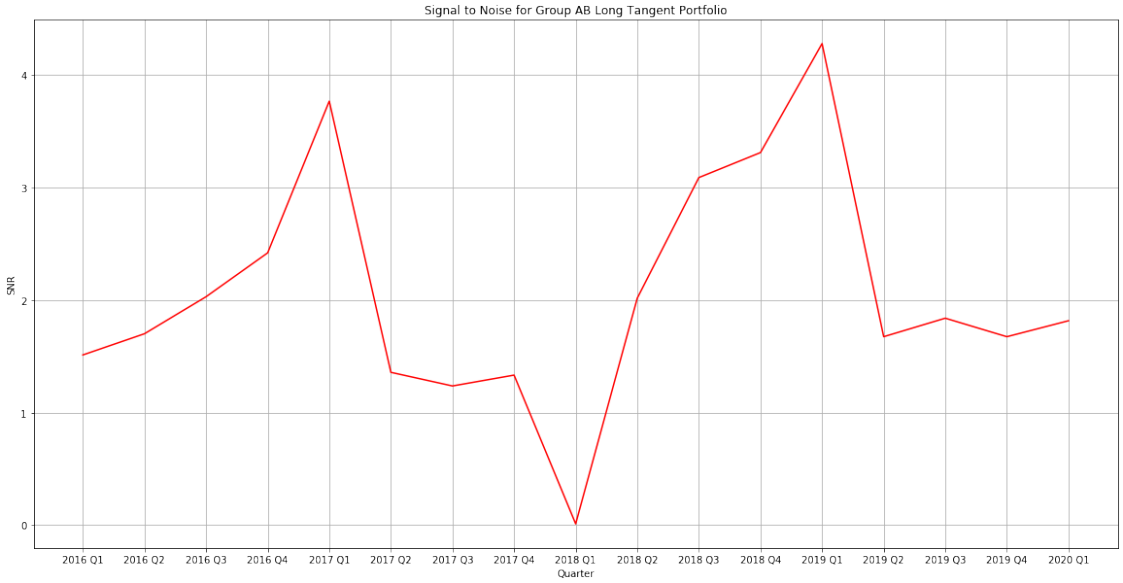

Signal to Noise for Group A Safe Tangent Portfolio
SNR
Quarter
0
1
2
3
4
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1


Signal to Noise for Group A Credit Tangent Portfolio
SNR
Quarter
0
1
2
3
4
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

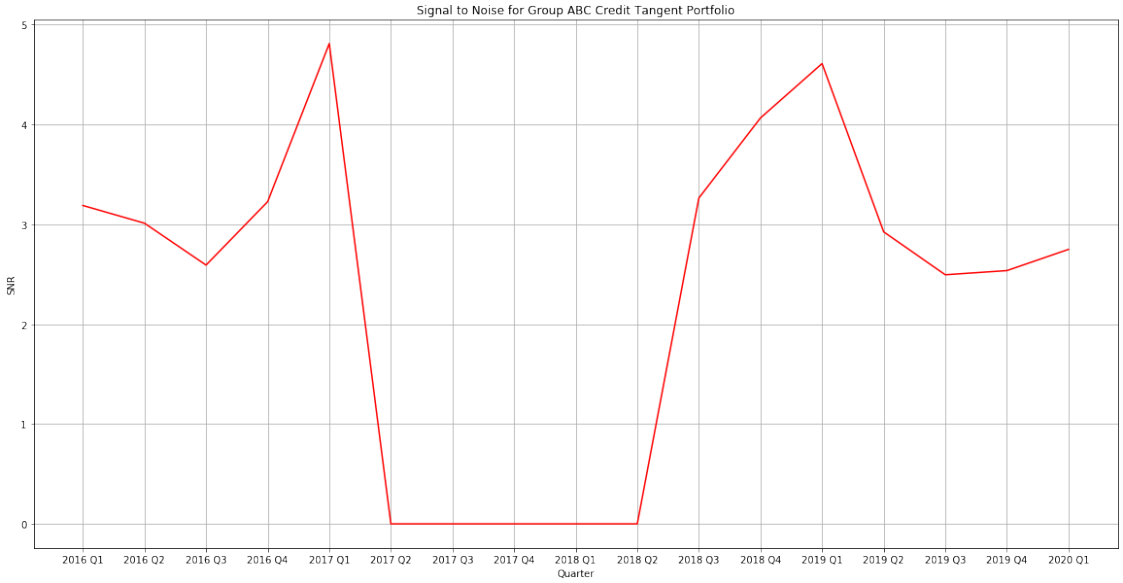

Signal to Noise for Group A Long Tangent Portfolio
SNR
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
2019 Q1
2019 Q2
2019 Q3
2019 Q4
2020 Q1
Quarter


Signal to Noise for Group AB Safe Tangent Portfolio
SNR
4
3
2
1
0
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
Quarter


Signal to Noise for Group AB Credit Tangent Portfolio
SNR
Quarter
0
1
2
3
4
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


Signal to Noise for Group AB Long Tangent Portfolio
SNR
Quarter
0
1
2
3
4
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


Signal to Noise for Group ABC Safe Tangent Portfolio
SNR
Quarter
0
1
2
3
4
5
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


Signal to Noise for Group ABC Credit Tangent Portfolio
SNR
Quarter
0
1
2
3
4
5
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

The ratios of the portfolios produce higher values for signal to noise on average, typically sitting above 1 for the majority of the quarters and hitting values above 4 for all 9 portfolios. In contrast, only 2 of the 5 years had a majority of individual assets above 1. The highest SNR that an individual asset reached in a year was 2.95 by VFIAX in 2017 but in the preceding and following year, VFIAX had an SNR of below 1. This shows that individual assets were much more variable when it came to SNR while the portfolios as a whole were more stable as well as having higher values.

The portfolio that we are most certain of the expected value of its return is group ABC. The safe and credit tangent portfolios were always significantly above 2 when they existed and the long tangent portfolio only saw a SNR value below 2 in 2 of the 17 quarters. Values for SNR peaked above 5 for the safe tangent portfolio and above 4 for both the credit and long tangent portfolios. These high values for SNR show that the growth rate had relatively low variance of the expected value. Because of these consistently high values, the portfolio for group ABC, and the safe tangent portfolio specifically, gives the most reason to be certain of the expected value of its returns

Safe Tangent Group A

	Gamma Q	Gamma P	Gamma T	Gamma R	Gamma S
2016 Q1	0.001449	-0.000064	0.000069	0.000075	0.000069
2016 Q2	0.001084	-0.000241	0.000068	0.000082	0.000068
2016 Q3	0.000682	-0.000465	0.000097	0.000121	0.000097
2016 Q4	0.001085	-0.000478	0.000073	0.000089	0.000073
2017 Q1	0.002479	-0.000823	0.000048	0.000059	0.000048
2017 Q2	0.000691	-0.000242	0.000080	0.000095	0.000080
2017 Q3	-0.004826	-0.004904	-0.004756	-0.003432	-0.004755
2018 Q2	0.002718	-0.000207	0.000062	0.000072	0.000062
2018 Q3	0.002273	-0.000436	0.000052	0.000060	0.000052
2018 Q4	0.002675	-0.000600	0.000055	0.000067	0.000055
2019 Q1	0.009193	-0.001048	0.000059	0.000074	0.000059
2019 Q2	0.000626	-0.000322	0.000115	0.000135	0.000115
2019 Q3	0.000567	-0.000400	0.000068	0.000090	0.000068
2019 Q4	0.000963	-0.000544	0.000096	0.000130	0.000096
2020 Q1	0.001708	-0.000432	0.000074	0.000100	0.000074

Credit Tangent Group A

	Gamma Q	Gamma P	Gamma T	Gamma R	Gamma S
2016 Q1	0.000427	-0.000045	0.000116	0.000158	0.000116
2016 Q2	-0.001488	-0.001675	-0.000806	-0.000266	-0.000806
2016 Q3	-0.000744	-0.000976	0.000733	0.001233	0.000734
2016 Q4	-0.000355	-0.000903	0.000226	0.000340	0.000226
2017 Q1	0.001678	-0.000943	0.000057	0.000074	0.000057
2018 Q3	0.002194	-0.000435	0.000053	0.000061	0.000053
2018 Q4	0.002249	-0.000615	0.000059	0.000075	0.000059
2019 Q1	0.007889	-0.001097	0.000065	0.000084	0.000065
2019 Q2	0.000568	-0.000239	0.000210	0.000236	0.000210
2019 Q3	0.000487	-0.000383	0.000096	0.000122	0.000096
2019 Q4	0.000485	-0.000607	0.000137	0.000198	0.000137
2020 Q1	0.001235	-0.000449	0.000113	0.000153	0.000113

Long Tangent Group A

	Gamma Q	Gamma P	Gamma T	Gamma R	Gamma S
2016 Q1	0.001313	-0.000059	7.039065e-05	0.000077	7.039134e-05
2016 Q2	0.000773	-0.000235	6.836096e-05	0.000089	6.836492e-05
2016 Q3	0.000378	-0.000435	1.079494e-04	0.000147	1.079603e-04
2016 Q4	0.000842	-0.000438	9.112001e-05	0.000112	9.112482e-05
2017 Q1	0.002288	-0.000729	5.146291e-05	0.000064	5.146555e-05
2017 Q2	0.000782	-0.000128	3.230976e-05	0.000043	3.231114e-05
2017 Q3	-0.000088	-0.000424	-1.735776e-04	-0.000104	-1.735580e-04
2017 Q4	-0.000068	-0.000554	-1.670098e-04	-0.000082	-1.669808e-04
2018 Q1	0.000038	-0.000003	5.758507e-05	0.000026	5.758811e-05
2018 Q2	0.002741	-0.000207	6.204953e-05	0.000071	6.205078e-05
2018 Q3	0.002215	-0.000434	5.274969e-05	0.000061	5.275098e-05
2018 Q4	0.002679	-0.000600	5.452455e-05	0.000067	5.452701e-05
2019 Q1	0.008858	-0.001058	5.939101e-05	0.000076	5.939532e-05
2019 Q2	0.000664	-0.000231	3.794832e-05	0.000055	3.795122e-05
2019 Q3	0.000677	-0.000307	8.297219e-06	0.000026	8.300336e-06
2019 Q4	0.001592	-0.000256	-2.195008e-07	0.000016	-2.169451e-07
2020 Q1	0.002142	-0.000300	1.057528e-06	0.000018	1.060415e-06

Safe Tangent Group AB

	Gamma Q	Gamma P	Gamma T	Gamma R	Gamma S
2016 Q1	0.001145	-0.001359	0.000278	0.000355	0.000278
2016 Q2	0.003827	-0.000578	0.000097	0.000118	0.000097
2016 Q3	0.003024	-0.000661	0.000101	0.000128	0.000101
2016 Q4	0.003319	-0.000751	0.000078	0.000098	0.000078
2017 Q1	0.007321	-0.001037	0.000049	0.000062	0.000049
2017 Q2	0.001854	-0.000302	0.000117	0.000136	0.000117
2017 Q3	0.001398	0.001070	0.004411	0.005892	0.004414
2018 Q2	0.001987	-0.000377	0.000134	0.000157	0.000134
2018 Q3	0.002231	-0.000631	0.000066	0.000079	0.000066
2018 Q4	0.002471	-0.001182	0.000105	0.000132	0.000105
2019 Q1	0.017319	-0.001298	0.000068	0.000087	0.000068
2019 Q2	0.001467	-0.001091	0.000101	0.000156	0.000101
2019 Q3	0.001101	-0.000489	0.000086	0.000113	0.000086
2019 Q4	0.002845	-0.000561	0.000097	0.000132	0.000097
2020 Q1	0.004972	-0.000656	0.000100	0.000142	0.000100

Credit Tangent Group AB

	Gamma Q	Gamma P	Gamma T	Gamma R	Gamma S
2016 Q1	-0.001036	-0.001465	0.005113	0.007674	0.005120
2016 Q2	0.000711	-0.000997	0.000340	0.000470	0.000341
2016 Q3	-0.000403	-0.001465	0.000477	0.000770	0.000477
2016 Q4	0.000274	-0.001394	0.000217	0.000329	0.000217
2017 Q1	0.005144	-0.001235	0.000063	0.000085	0.000063
2018 Q3	0.001308	-0.000680	0.000110	0.000133	0.000110
2018 Q4	0.001077	-0.001403	0.000207	0.000261	0.000207
2019 Q1	0.013647	-0.001426	0.000086	0.000114	0.000086
2019 Q2	0.000491	-0.001308	0.000160	0.000264	0.000160
2019 Q3	0.000824	-0.000494	0.000118	0.000155	0.000118
2019 Q4	0.001834	-0.000631	0.000136	0.000197	0.000136
2020 Q1	0.003167	-0.000716	0.000223	0.000306	0.000223

Long Tangent Group AB

	Gamma Q	Gamma P	Gamma T	Gamma R	Gamma S
2016 Q1	0.004760	-0.000093	0.000078	0.000087	0.000078
2016 Q2	0.003317	-0.000233	0.000067	0.000087	0.000067
2016 Q3	0.002166	-0.000434	0.000108	0.000147	0.000108
2016 Q4	0.003146	-0.000428	0.000088	0.000108	0.000088
2017 Q1	0.007498	-0.000729	0.000050	0.000061	0.000050
2017 Q2	0.002087	-0.000130	0.000032	0.000043	0.000032
2017 Q3	0.000594	-0.000409	-0.000073	0.000005	-0.000073
2017 Q4	0.000817	-0.000529	-0.000144	-0.000064	-0.000144
2018 Q1	0.001429	0.000060	0.000060	0.000060	0.000060
2018 Q2	0.003266	-0.000207	0.000062	0.000072	0.000062
2018 Q3	0.002913	-0.000431	0.000053	0.000061	0.000053
2018 Q4	0.004522	-0.000578	0.000054	0.000065	0.000054
2019 Q1	0.018800	-0.001051	0.000060	0.000076	0.000060
2019 Q2	0.003177	-0.000233	0.000037	0.000053	0.000037
2019 Q3	0.001453	-0.000306	0.000010	0.000027	0.000010
2019 Q4	0.003972	-0.000258	0.000002	0.000017	0.000002
2020 Q1	0.007447	-0.000300	0.000002	0.000018	0.000002

Safe Tangent Group ABC

	Gamma Q	Gamma P	Gamma T	Gamma R	Gamma S
2016 Q1	0.002618	-0.001676	0.000288	0.000381	0.000288
2016 Q2	0.009812	-0.001212	0.000138	0.000189	0.000138
2016 Q3	0.006210	-0.000955	0.000119	0.000162	0.000119
2016 Q4	0.005397	-0.000865	0.000079	0.000100	0.000079
2017 Q1	0.031074	-0.001447	0.000059	0.000077	0.000059
2017 Q2	0.009714	-0.001163	0.000257	0.000331	0.000257
2018 Q2	0.005264	-0.000425	0.000142	0.000170	0.000142
2018 Q3	0.009389	-0.000816	0.000072	0.000088	0.000072
2018 Q4	0.007356	-0.001409	0.000098	0.000126	0.000098
2019 Q1	0.027837	-0.001420	0.000068	0.000088	0.000068
2019 Q2	0.004351	-0.001172	0.000077	0.000131	0.000077
2019 Q3	0.009252	-0.000749	0.000133	0.000177	0.000133
2019 Q4	0.014726	-0.000897	0.000180	0.000242	0.000180
2020 Q1	0.015649	-0.001146	0.000136	0.000208	0.000136

Credit Tangent Group ABC

	Gamma Q	Gamma P	Gamma T	Gamma R	Gamma S
2016 Q1	-0.001869	-0.002529	0.005992	0.009883	0.006005
2016 Q2	-0.001224	-0.003445	0.001415	0.002590	0.001417
2016 Q3	-0.002493	-0.003422	0.001436	0.003786	0.001442
2016 Q4	0.001333	-0.001507	0.000210	0.000304	0.000210
2017 Q1	0.021277	-0.001843	0.000094	0.000128	0.000094
2018 Q3	0.005288	-0.000985	0.000134	0.000172	0.000134
2018 Q4	0.004342	-0.001712	0.000175	0.000230	0.000175
2019 Q1	0.022448	-0.001556	0.000085	0.000113	0.000085
2019 Q2	0.002412	-0.001446	0.000100	0.000205	0.000100
2019 Q3	0.006386	-0.000826	0.000220	0.000298	0.000220
2019 Q4	0.008607	-0.001054	0.000406	0.000557	0.000406
2020 Q1	0.009040	-0.001478	0.000301	0.000475	0.000301

Long Tangent Group ABC

	Gamma Q	Gamma P	Gamma T	Gamma R	Gamma S
2016 Q1	0.008336	-0.000124	0.000080	0.000091	0.000080
2016 Q2	0.006696	-0.000594	0.000200	0.000274	0.000200
2016 Q3	0.004395	-0.000623	0.000134	0.000197	0.000134
2016 Q4	0.004828	-0.000439	0.000089	0.000110	0.000089
2017 Q1	0.029994	-0.001060	0.000071	0.000088	0.000071
2017 Q2	0.008873	-0.000599	0.000352	0.000428	0.000352
2017 Q3	0.003799	-0.000485	0.001422	0.001880	0.001423
2017 Q4	0.004592	-0.001969	0.001313	0.001905	0.001314
2018 Q1	0.000678	0.000014	0.000111	0.000562	0.000111
2018 Q2	0.008291	-0.000228	0.000069	0.000080	0.000069
2018 Q3	0.010687	-0.000561	0.000064	0.000076	0.000064
2018 Q4	0.009860	-0.000765	0.000065	0.000081	0.000065
2019 Q1	0.029671	-0.001102	0.000061	0.000077	0.000061
2019 Q2	0.007363	-0.000404	0.000019	0.000038	0.000019
2019 Q3	0.010803	-0.000570	0.000058	0.000088	0.000058
2019 Q4	0.016391	-0.000690	0.000106	0.000152	0.000106
2020 Q1	0.021030	-0.000591	0.000073	0.000108	0.000073

The values of the mean variance approximation error with the signal to noise ratio that we calculated in exercise 1 appear to be significantly closer to zero than the signal to noise ratio values. Thus it would imply that our approximations are generally good, as their distance from our growth rate estimator is very small especially in relation to the standard deviation of the expected value. All of our approximations appear to be within 10^{-5} of Ex[log(1 + R)]. In regards to the ratio, we did not see any values above 0.03 again signifying that the approximations for all the estimators used were close to the expected value.

With regards to which estimator seems the most reasonable, we believe that the taylor series approximation is the most reasonable. First because it generally has the lowest error among all of our approximations. The highest value in a quarter we saw was 0.006 and the vast majority of other quarters were significantly lower than this. Second the taylor series approximation is a fairly natural way of performing approximations, and generally introduces minimal assumptions. The taylor approximation used introduces an error of $O(r^3)$ and since we are dealing with small daily return values, this error is quite small in practice.