### [Mean-Variance Optimization](https://drive.google.com/file/d/1TSXMmj9OJXg7aEEZjvT2lv_GT4t4wlJv/view?usp=sharing)

# import needed modules

!pip install quandl

import quandl

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

#get adjusted closing prices of 5 selected companies with Quandl

quandl.ApiConfig.api\_key = 'YOUR API KEY' #copy and paste your api key here

selected = ['CNP', 'F', 'WMT', 'GE', 'TSLA'] # selected assets list

data = quandl.get\_table('WIKI/PRICES', ticker = selected,

qopts = { 'columns': ['date', 'ticker', 'adj\_close'] },

date = { 'gte': '2014-1-1', 'lte': '2018-3-15' }, paginate=True)

print(data.head())

print(data.tail())

date ticker adj\_close

None

0 2018-03-15 WMT 87.51

1 2018-03-14 WMT 87.67

2 2018-03-13 WMT 88.30

3 2018-03-12 WMT 88.07

4 2018-03-09 WMT 88.72

date ticker adj\_close

None

5280 2014-01-08 CNP 19.132992

5281 2014-01-07 CNP 19.334039

5282 2014-01-06 CNP 19.132992

5283 2014-01-03 CNP 19.107861

5284 2014-01-02 CNP 19.116238

# reorganise data pulled by setting date as index with

# columns of tickers and their corresponding adjusted prices

clean = data.set\_index('date')

table = clean.pivot(columns='ticker')

print(table.head())

print(table.tail())

adj\_close

ticker CNP F GE TSLA WMT

date

2014-01-02 19.116238 12.726742 24.266002 150.10 71.343743

2014-01-03 19.107861 12.784441 24.248354 149.56 71.108673

2014-01-06 19.132992 12.842140 24.054226 147.00 70.710863

2014-01-07 19.334039 12.677286 24.080698 149.36 70.927850

2014-01-08 19.132992 12.809169 24.010106 151.28 70.367299

adj\_close

ticker CNP F GE TSLA WMT

date

2018-03-09 27.13 10.73 14.94 327.17 88.72

2018-03-12 27.43 10.81 15.10 345.51 88.07

2018-03-13 27.31 10.78 14.43 341.84 88.30

2018-03-14 27.05 11.02 14.27 326.63 87.67

2018-03-15 26.92 11.07 14.36 325.60 87.51

# calculate daily and annual returns of the stocks

returns\_daily = table.pct\_change()

returns\_annual = returns\_daily.mean() \* 252 # use average instead of expected value

print(returns\_daily.tail())

print(returns\_annual)

adj\_close

ticker CNP F GE TSLA WMT

date

2018-03-09 0.001477 0.011310 0.028926 -0.005864 0.009099

2018-03-12 0.011058 0.007456 0.010710 0.056056 -0.007326

2018-03-13 -0.004375 -0.002775 -0.044371 -0.010622 0.002612

2018-03-14 -0.009520 0.022263 -0.011088 -0.044495 -0.007135

2018-03-15 -0.004806 0.004537 0.006307 -0.003153 -0.001825

ticker

adj\_close CNP 0.099449

F -0.007950

GE -0.105323

TSLA 0.266748

WMT 0.066674

dtype: float64

# get daily and covariance of returns of the stock

cov\_daily = returns\_daily.cov()

cov\_annual = cov\_daily \* 252

#returns\_daily.head()

print(cov\_annual)

adj\_close

ticker CNP F GE TSLA WMT

ticker

adj\_close CNP 0.035437 0.010728 0.011515 0.015068 0.008379

F 0.010728 0.050348 0.020566 0.023589 0.009375

GE 0.011515 0.020566 0.039715 0.018830 0.008987

TSLA 0.015068 0.023589 0.018830 0.164388 0.011725

WMT 0.008379 0.009375 0.008987 0.011725 0.035811

# empty lists to store returns, volatility and weights of imiginary portfolios

port\_returns = []

port\_volatility = []

stock\_weights = []

# set the number of combinations for imaginary portfolios

num\_assets = len(selected)

num\_portfolios = 50000 # of simulated portfolio

wmin = 0.05 # minimum portfolio weights per asset

# populate the empty lists with each portfolios returns,risk and weights

for single\_portfolio in range(num\_portfolios):

weights = np.random.random(num\_assets)

# Return random floats in the half-open interval [0.0, 1.0)

# weights = np.random.standard\_normal(num\_assets)

weights /= np.sum(weights) # weights = weights/np.sum(weights)

weights = weights\*(1-wmin\*num\_assets) + wmin

returns = np.dot(weights, returns\_annual)

volatility = np.sqrt(np.dot(weights.T, np.dot(cov\_annual, weights)))

port\_returns.append(returns)

port\_volatility.append(volatility)

stock\_weights.append(weights)

# a dictionary for Returns and Risk values of each portfolio

portfolio = {'Returns': port\_returns,

'Volatility': port\_volatility}

# extend original dictionary to accomodate each ticker and weight in the portfolio

for counter,symbol in enumerate(selected):

portfolio[symbol+' weight'] = [weight[counter] for weight in stock\_weights]

# make a nice dataframe of the extended dictionary

df = pd.DataFrame(portfolio)

# get better labels for desired arrangement of columns

column\_order = ['Returns', 'Volatility'] + [stock+' weight' for stock in selected]

# reorder dataframe columns

df = df[column\_order]

print(df.head())

print(df.tail())

Returns Volatility CNP weight F weight WMT weight GE weight \

0 0.045234 0.146936 0.169591 0.311494 0.159090 0.118003

1 0.087965 0.162763 0.148729 0.245774 0.068872 0.232928

2 0.084268 0.158226 0.247806 0.128795 0.156085 0.229563

3 0.048010 0.160265 0.292186 0.071342 0.365001 0.199241

4 0.100057 0.167774 0.314799 0.107896 0.132028 0.269029

TSLA weight

0 0.241822

1 0.303697

2 0.237751

3 0.072229

4 0.176248

Returns Volatility CNP weight F weight WMT weight GE weight \

49995 0.095538 0.153582 0.278273 0.065922 0.096561 0.206280

49996 0.046892 0.151928 0.105195 0.268423 0.202404 0.158012

49997 0.070083 0.183535 0.150007 0.077767 0.341300 0.314873

49998 0.063587 0.143532 0.505730 0.182581 0.110607 0.064907

49999 0.079069 0.182919 0.077102 0.131672 0.257421 0.319727

TSLA weight

49995 0.352964

49996 0.265966

49997 0.116054

49998 0.136175

49999 0.214078

# plot the efficient frontier with a scatter plot

plt.style.use('seaborn')

df.plot.scatter(x='Volatility', y='Returns', figsize=(10, 8), grid=True,

# color='turquoise',edgecolors='turquoise')

color='#81D8D0',edgecolors='turquoise')

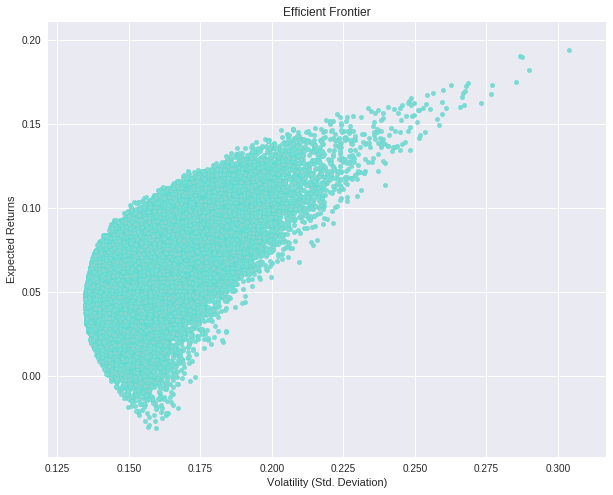
# color='red',edgecolors='turquoise')

plt.xlabel('Volatility (Std. Deviation)')

plt.ylabel('Expected Returns')

plt.title('Efficient Frontier')

plt.show()



port\_returns = []

port\_volatility = []

sharpe\_ratio = []

stock\_weights = []

# set the number of combinations for imaginary portfolios

num\_assets = len(selected)

num\_portfolios = 50000

#set random seed for reproduction's sake

np.random.seed(101)

# populate the empty lists with each portfolios returns,risk and weights

rf = 0.02 # risk free rate

for single\_portfolio in range(num\_portfolios):

weights = np.random.random(num\_assets)

weights /= np.sum(weights)

weights = weights\*(1-wmin\*num\_assets) + wmin

returns = np.dot(weights, returns\_annual)

volatility = np.sqrt(np.dot(weights.T, np.dot(cov\_annual, weights)))

sharpe = (returns - rf) / volatility

sharpe\_ratio.append(sharpe)

port\_returns.append(returns)

port\_volatility.append(volatility)

stock\_weights.append(weights)

# a dictionary for Returns and Risk values of each portfolio

portfolio = {'Returns': port\_returns,

'Volatility': port\_volatility,

'Sharpe Ratio': sharpe\_ratio}

# extend original dictionary to accomodate each ticker and weight in the portfolio

for counter,symbol in enumerate(selected):

portfolio[symbol+' Weight'] = [Weight[counter] for Weight in stock\_weights]

# make a nice dataframe of the extended dictionary

df = pd.DataFrame(portfolio)

# get better labels for desired arrangement of columns

column\_order = ['Returns', 'Volatility', 'Sharpe Ratio'] + [stock+' Weight' for stock in selected]

# reorder dataframe columns

df = df[column\_order]

print(df.head())

Returns Volatility Sharpe Ratio CNP Weight F Weight WMT Weight \

0 0.067415 0.143301 0.330876 0.246365 0.267002 0.060828

1 0.064735 0.164037 0.272711 0.262298 0.128149 0.277501

2 0.097097 0.164463 0.468779 0.196397 0.143014 0.098046

3 0.082906 0.186043 0.338129 0.140542 0.082562 0.285186

4 0.067441 0.147980 0.320592 0.376027 0.296370 0.073066

GE Weight TSLA Weight

0 0.115223 0.310583

1 0.233695 0.098356

2 0.257514 0.305029

3 0.334068 0.157643

4 0.115590 0.138948

# plot frontier, max sharpe & min Volatility values with a scatterplot

plt.style.use('seaborn-dark')

df.plot.scatter(x='Volatility', y='Returns', c='Sharpe Ratio',

cmap='coolwarm', edgecolors='grey', figsize=(10, 8), grid=True)

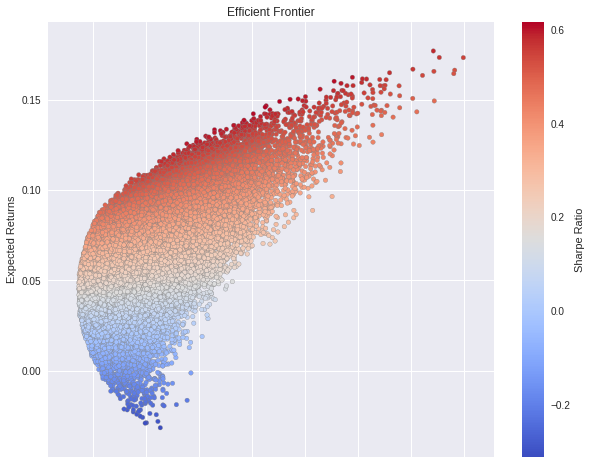
#cmap='RdYlGn'

plt.xlabel('Volatility (Std. Deviation)')

plt.ylabel('Expected Returns')

plt.title('Efficient Frontier')

plt.show()



# find min Volatility & max sharpe values in the dataframe (df)

min\_volatility = df['Volatility'].min()

max\_sharpe = df['Sharpe Ratio'].max()

# use the min, max values to locate and create the two special portfolios

sharpe\_portfolio = df.loc[df['Sharpe Ratio'] == max\_sharpe]

min\_variance\_port = df.loc[df['Volatility'] == min\_volatility]

# plot frontier, max sharpe & min Volatility values with a scatterplot

plt.style.use('seaborn-dark')

df.plot.scatter(x='Volatility', y='Returns', c='Sharpe Ratio',

cmap='hot', edgecolors='grey', figsize=(10, 8), grid=True)

plt.scatter(x=sharpe\_portfolio['Volatility'], y=sharpe\_portfolio['Returns'], c='teal', marker='D', s=200)

plt.scatter(x=min\_variance\_port['Volatility'], y=min\_variance\_port['Returns'], c='cyan', marker='D', s=200 )

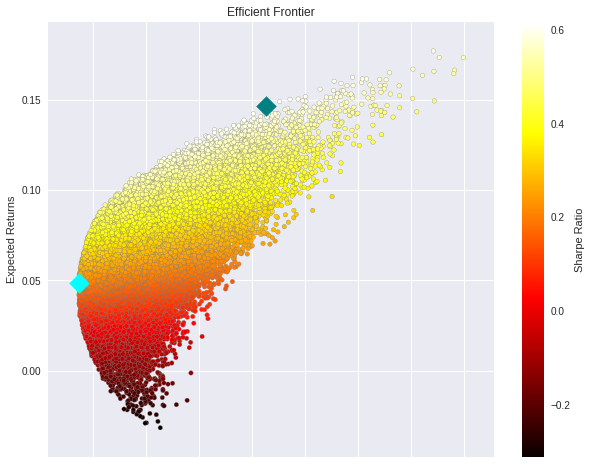
# https://matplotlib.org/api/markers\_api.html

plt.xlabel('Volatility (Std. Deviation)')

plt.ylabel('Expected Returns')

plt.title('Efficient Frontier')

plt.show()



# print the details of the 2 special portfolios

print(min\_variance\_port.T)

41218

Returns 0.048738

Volatility 0.134591

Sharpe Ratio 0.213517

CNP Weight 0.333920

F Weight 0.105344

WMT Weight 0.182750

GE Weight 0.052044

TSLA Weight 0.325942

print(sharpe\_portfolio.T)

453

Returns 0.146700

Volatility 0.205300

Sharpe Ratio 0.617145

CNP Weight 0.352502

F Weight 0.054546

WMT Weight 0.053129

GE Weight 0.408255

TSLA Weight 0.131569

# check sum to one

np.dot(min\_variance\_port,[0,0,0,1,1,1,1,1])[0]