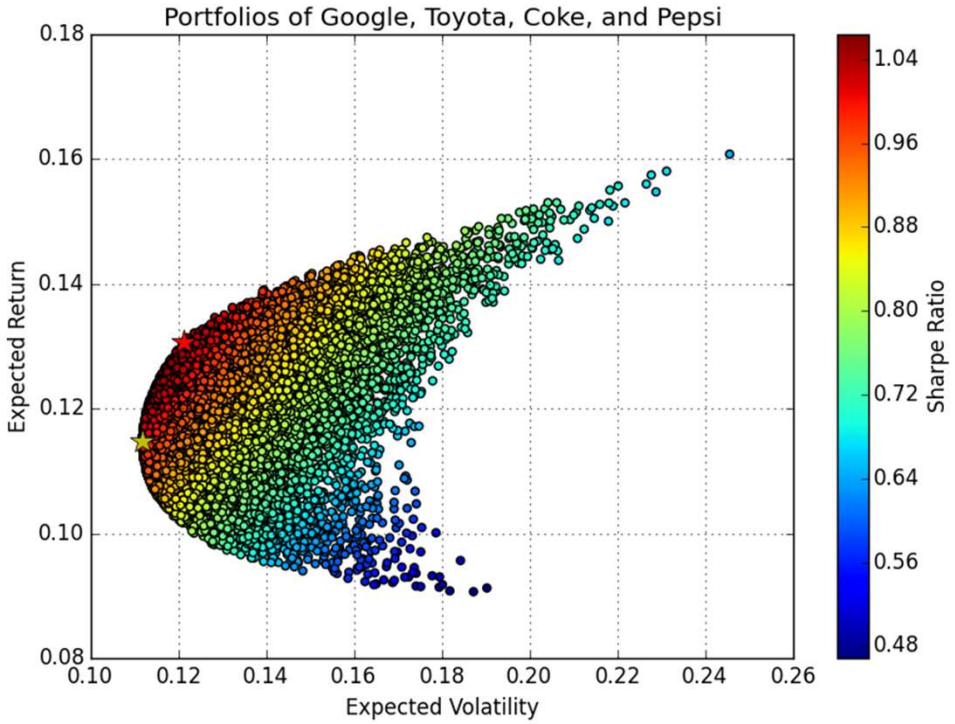


Financial Portfolio Optimization

ปัญหาการจัดพอร์ตทางการเงินที่เหมาะที่สุด

CSS 341 Introduction to

Data Science

Chukiat Worasucheep

Learning objectives for this session

- ❑ เข้าใจว่าปัญหาการจัดพอร์ตทางการเงินที่เหมาะที่สุด (portfolio optimization) *คืออะไร*และมี*ความสำคัญ*อย่างไร
- ❑ เข้าใจ*ลักษณะ*และ*แนวทาง*การแก้ปัญหา portfolio optimization
- ❑ เรียนรู้วิธีการ*ดึงข้อมูล*ซื้อขายหุ้นรายวันย้อนหลังจากแหล่งข้อมูลออนไลน์
- ❑ ทดลองใช้ differential evolution ในการทำ portfolio optimization หุ้น 8 ตัว
- ❑ เข้าใจการ*ตีความ*และนำผลลัพธ์ portfolio optimization ที่ได้ไปใช้งาน
- ❑ เข้าใจความหมายและสามารถใช้งาน *Sharpe ratio* ได้ในการวัดความเสี่ยงเทียบกับผลตอบแทนการลงทุน

Python library to use today

pip install *pandas_datareader*

Another possible library:

pip install *yfinance*

Contents

- ❑What is portfolio optimization?
- ❑Steps of portfolio selection
- Python code for portfolio selection
- Portfolio optimization using differential evolution algorithm
  - ✓ Maximize return
  - ✓ Minimize risk
  - ✓ Maximize return/risk ratio

Warning

- การลงทุนมีความเสี่ยง ผู้ลงทุนต้องศึกษาข้อมูลให้รอบคอบก่อนการตัดสินใจ
- เนื้อหาในวันนี้มิได้เป็นการเสนอแนะหรือชี้นำการลงทุนใดๆ ทั้งสิ้น
- ผู้เรียนควรพิจารณาหลักการ เหตุผล และความเหมาะสมอย่างรอบคอบ

Financial portfolio optimization problem

"Don't put all your eggs in one basket."

- a.k.a. *portfolio selection problem*
- *"Financial portfolio optimization problem is the selecting the best portfolio (asset distribution), out of the set of all portfolios being considered, according to some objective(s)."* (Wikipedia)
- The objectives typically maximize expected *return* and minimize costs like financial *risk*.

Markowitz's mean-variance (MV) model

- Harry Markowitz's *Modern Portfolio Theory (MPT)*
- Investment is based solely on *expected return* and *risk*, reflecting in his mean-variance (MV) model.
- The *expected return* is a probability expressing the estimated return of the investment in the security during a time period.
- The *risk* is expressed as the *variance* of expected returns of the security on a daily or weekly basis.
- Used to find the biggest reward at a given level of risk or the smallest risk at a given level of return.

Definitions

- *Portfolio* (พอร์ตการลงทุน)– a group of assets, such as stocks or bonds, held by an investor.
- *Portfolio weight* – the percentage of a portfolio's total value invested in a particular asset.
- *Expected return* (ผลตอบแทนที่คาดหวัง) – average return on a risky asset expected in the future.
- *Volatility* (or *risk*) – variance of expected returns of an asset.
  - "ความแปรปรวนของผลตอบแทน"

Expected portfolio return

To calculate expected return of a portfolio, multiply the *portfolio weight* of respectively security by its *expected return* and sum the products.

$$E(r)_p = wiE(r)_i + wjE(r)_j$$

Where:

$E(r)_p$ = expected return of portfolio

w_i = weight of asset i.

$E(r)_i$ = expected return of asset i.

Example #1 – expected portfolio return

- Calculate the expected return of a portfolio with three stocks.
- Stocks A, B, and C each has an expected return of 3%, 1%, and 9% and makes up 25% 50%, and 25%, respectively, of the portfolio.

- (0.25 x 3%) + (0.5 x 1%) + (0.25 x 9%) = 3.5%

Import stock data from Yahoo! Finance

```
from pandas_datareader import data as web
import datetime as dt
import pandas as pd
import numpy as np

ticker = ['MSFT']
start = dt.datetime(2017, 1, 1)
end   = dt.datetime(2021, 12, 30)
data  = web.DataReader(ticker, 'yahoo', start, end)
stocks = pd.DataFrame(data)
stocks[ticker] = data['Adj Close'].pct_change()
```

Daily return

$$\text{Percentage Change Formula} = \frac{\text{Old Number} - \text{New Number}}{\text{Old Number}} \times 100$$

	A	B	C	D	E	F	G
1	Date	High	Low	Open	Close	Volume	Adj Close
2	4/1/2017	38.1	37.3	37.4	38.0	75879000	30.1962
3	5/1/2017	38.8	38.2	38.2	38.7	75282000	30.7524
4	6/1/2017	38.9	38.5	38.7	38.9	45129000	30.9114
5	9/1/2017	39	38.2	38.9	38.3	40455000	30.4346
6	10/1/2017	38.9	38.2	38.3	38.8	43224000	30.8319
7	11/1/2017	38.9	38.5	38.5	38.5	38177000	30.5935
8	12/1/2017	38.9	38.3	38.9	38.3	35313000	30.4346
9	16/1/2017	39.1	38.5	38.6	38.9	42809000	30.9114
10	17/1/2017	39	38.6	38.8	38.7	32141000	30.7524
11	18/1/2017	38.8	38.5	38.7	38.5	23665000	30.5935
12	19/1/2017	38.6	38.3	38.5	38.5	32121000	30.5935
13	20/1/2017	38.8	38.5	38.7	38.6	33027000	30.6730
14	23/1/2017	38.9	38.5	38.7	38.9	24963000	30.9114
15	24/1/2017	40.8	38.9	38.9	40.8	128983000	32.4212
16	25/1/2017	41.2	40.2	40.8	41.0	74923000	32.5801
17	26/1/2017	43	41.4	42	42.0	121160000	33.3747
18	27/1/2017	42.2	41.4	42	41.8	43895000	33.2158
19	30/1/2017	42	41.4	41.6	41.8	28986000	33.2158
20	31/1/2017	41.6	40.4	41.6	40.4	70486000	32.1033
21	1/2/2017	41	40	40.4	40.4	80764000	32.1033
22	2/2/2017	40.6	40.2	40.2	40.4	45951000	32.1033
23	3/2/2017	41.6	40.2	40.4	41.0	69129000	32.5801

Contents

- What is portfolio optimization?
- Steps of portfolio selection
- Python code for portfolio selection
- Portfolio optimization using differential evolution algorithm

Steps of portfolio selection

1. คำนวณ annualized weighted mean of *portfolio returns* (ผลตอบแทนเฉลี่ยของพอร์ต).

2. คำนวณ *portfolio risk* (ความเสี่ยงของพอร์ต).

   2.1 คำนวณ annualized *covariance matrix of returns* (cov_mat)

   2.2 คำนวณ *portfolio variance* (ความแปรปรวนของผลตอบแทนของพอร์ต) from covariance matrix (cov_mat)

   2.3 คำนวณ $Portfolio\ risk = \sqrt{Portfolio\ variance}$

3. คำนวณ *Sharpe ratio*.

Sharpe ratio เป็นอัตราส่วนที่ใช้วัดผลตอบแทนเฉลี่ยที่ได้รับเพิ่มขึ้นจากการลงทุน
ในสินทรัพย์ (กลุ่ม) หนึ่งเทียบกับความเสี่ยงที่ได้ตามมา (William Sharpe, 1994).

1. คำนวณ annualized weighted mean of portfolio returns

Date	AMZN	GOOG	MSFT	TSLA	BAC	CPALL.BK	PTT.BK
2017-05-31	-0.002087	-0.011292	-0.008095	0.017637	-0.018612	-0.003984	0.002551
2017-06-01	0.001337	0.002166	0.003723	-0.001877	0.009817	-0.004000	-0.007634
2017-06-02	0.010824	0.008946	0.023681	-0.001528	-0.007954	0.004016	0.002564
2017-06-05	0.004579	0.008282	0.007246	0.021980	-0.001782	-0.008000	-0.020460
2017-06-06	-0.008247	-0.007228	0.003321	0.015922	-0.008032	0.004032	0.005222
2017-06-07	0.007049	0.004475	-0.001679	0.016644	0.016644	-0.008032	0.000000
2017-06-08	0.000198	0.002518	-0.006078	0.028778	0.016372	0.004049	-0.007792
2017-06-09	-0.031635	-0.034146	-0.022655	-0.034270	0.030474	0.000000	0.002618
2017-06-12	-0.013697	-0.007296	-0.007679	0.004730	0.004647	-0.004032	-0.005222
2017-06-13	0.016457	0.011136	0.012468	0.047185	-0.000421	0.004049	0.010499

returns

AMZN	GOOG	MSFT	TSLA	BAC	CPALL.BK	PTT.BK
-0.001522	-0.002244	0.000414	0.011783	0.004115	-0.00119	-0.001765

Mean of returns

$$\text{Annualized Return} = \left((1+r_1)\times(1+r_2)\times(1+r_3)\times\cdots\times(1+r_n)\right)^{\frac{1}{n}} - 1$$

$$\begin{aligned}\text{Annualized Return} &= \left((1+.03)\times(1+.07)\times(1+.05)\times(1+.12)\times(1+.01)\right)^{\frac{1}{5}} - 1\\ &= 1.309^{0.20} - 1\\ &= 1.0553 - 1\\ &= .0553, \text{or } 5.53\%\end{aligned}$$

weight

```
[0.243, 0.176, 0.18 , 0.098, 0.087, 0.156, 0.06 ]
```

Annualized return (Rp) = sum(returns.mean() * weights) * 252

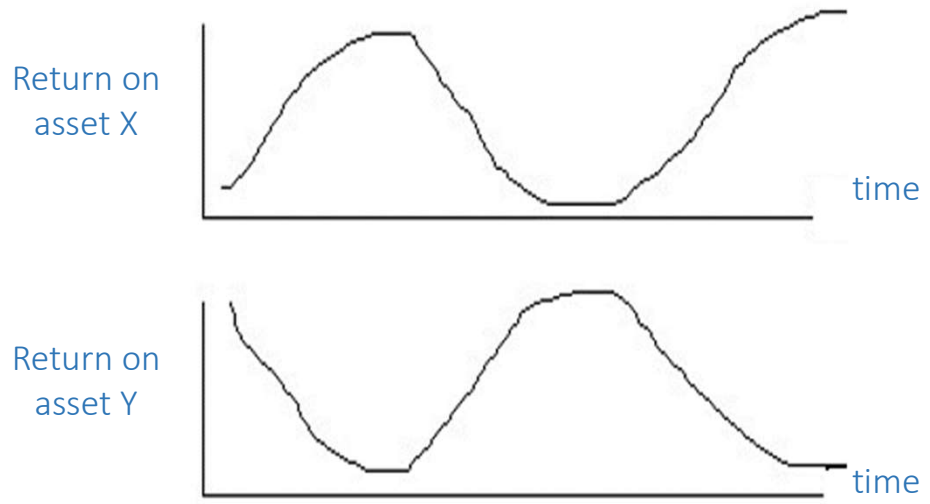
2. คำนวณ portfolio risk (=$\sqrt{portfolio\ variance}$)

- *Math/statistic foundations*

- *Variance(x)* ความแปรปรวน

$$S^2 = \frac{\sum(x_i - \bar{x})^2}{n-1}$$

S^2 = sample variance
x_i = the value of the one observation
$\bar{x}$ = the mean value of all observations
n = the number of observations



- *Covariance(x, y)* ความแปรปรวนร่วมเกี่ยว
  - a systematic relationship between two random variables in which a change in the other reflects a change in one variable.
  - -∞ to +∞

$$cov_{x,y} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{N-1}$$

$cov_{x,y}$ = covariance between variable x and y
x_i = data value of x
y_i = data value of y
$\bar{x}$ = mean of x
$\bar{y}$ = mean of y
N = number of data values

2. คำนวณ *portfolio risk* ($=\sqrt{portfolio\ variance}$)

- Portfolio variance (ความแปรปรวนของผลตอบแทน) of two stocks:
- $\sigma_p^2 = w_1^2\sigma_1^2 + w_2^2\sigma_2^2 + 2w_1w_2\sigma_{12}$

where

w_1 is the portfolio weight of the asset 1.
w_2 is the portfolio weight of the asset 2.
σ_1^2 is the variance of the asset 1.
σ_2^2 is the variance of the asset 2.
σ_{12} is the covariance of the two assets.

- Portfolio variance of three stocks:
- $\sigma_p^2 = w_1^2\sigma_1^2 + w_2^2\sigma_2^2 + w_3^2\sigma_3^2 + 2w_1w_2\sigma_{12} + 2w_1w_3\sigma_{13} + 2w_2w_3\sigma_{23}$

- Portfolio variance of n stocks:
  (ความแปรปรวนของผลตอบแทนของพอร์ต)
  - Multiply the squared weight of each security by its corresponding variance and adding twice the weighted average weight multiplied by the covariance of all individual security pairs.

2. คำนวณ *portfolio risk* ($=\sqrt{portfolio\ variance}$)

2.1 คำนวณ annualized covariance matrix of returns

cov_mat = returns.cov() * 252

Date	AMZN	GOOG	MSFT	TSLA	BAC	CPALL.BK	PTT.BK
2017-05-31	-0.002087	-0.011292	-0.008095	0.017637	-0.018612	-0.003984	0.002551
2017-06-01	0.001337	0.002166	0.003723	-0.001877	0.009817	-0.004000	-0.007634
2017-06-02	0.010824	0.008946	0.023681	-0.001528	-0.007954	0.004016	0.002564
2017-06-05	0.004579	0.008282	0.007248	0.021990	-0.001782	-0.008000	-0.020460
2017-06-06	-0.008247	-0.007228	0.003321	0.015922	-0.008032	0.004032	0.005222
2017-06-07	0.007049	0.004475	-0.001793	0.019272	0.016644	-0.008032	0.000000
2017-06-08	0.000198	0.002518	-0.006078	0.028778	0.016372	0.004049	-0.007792
2017-06-09	-0.031635	-0.034146	-0.022655	-0.034270	0.030474	0.000000	0.002618
2017-06-12	-0.013697	-0.007296	-0.007679	0.004730	0.004647	-0.004032	-0.005222
2017-06-13	0.016457	0.011136	0.012468	0.047185	-0.000421	0.004049	0.010499

returns

	AMZN	GOOG	MSFT	TSLA	BAC	CPALL.BK	PTT.BK
AMZN	1.886746e-04	1.757927e-04	0.000145	0.000226	-0.000092	2.719678e-06	9.492902e-07
GOOG	1.757927e-04	1.835886e-04	0.000145	0.000215	-0.000074	-1.226698e-07	-2.480224e-05
MSFT	1.448335e-04	1.446370e-04	0.000163	0.000122	-0.000102	1.600540e-05	5.233003e-06
TSLA	2.255894e-04	2.149254e-04	0.000122	0.000476	-0.000137	1.037768e-05	5.951993e-06
BAC	-9.194234e-05	-7.417163e-05	-0.000102	-0.000137	0.000211	-6.005904e-06	-1.690269e-05
CPALL.BK	2.719678e-06	-1.226698e-07	0.000016	0.000010	-0.000006	2.529460e-05	2.515647e-05
PTT.BK	9.492902e-07	-2.480224e-05	0.000005	0.000006	-0.000017	2.515647e-05	7.679362e-05

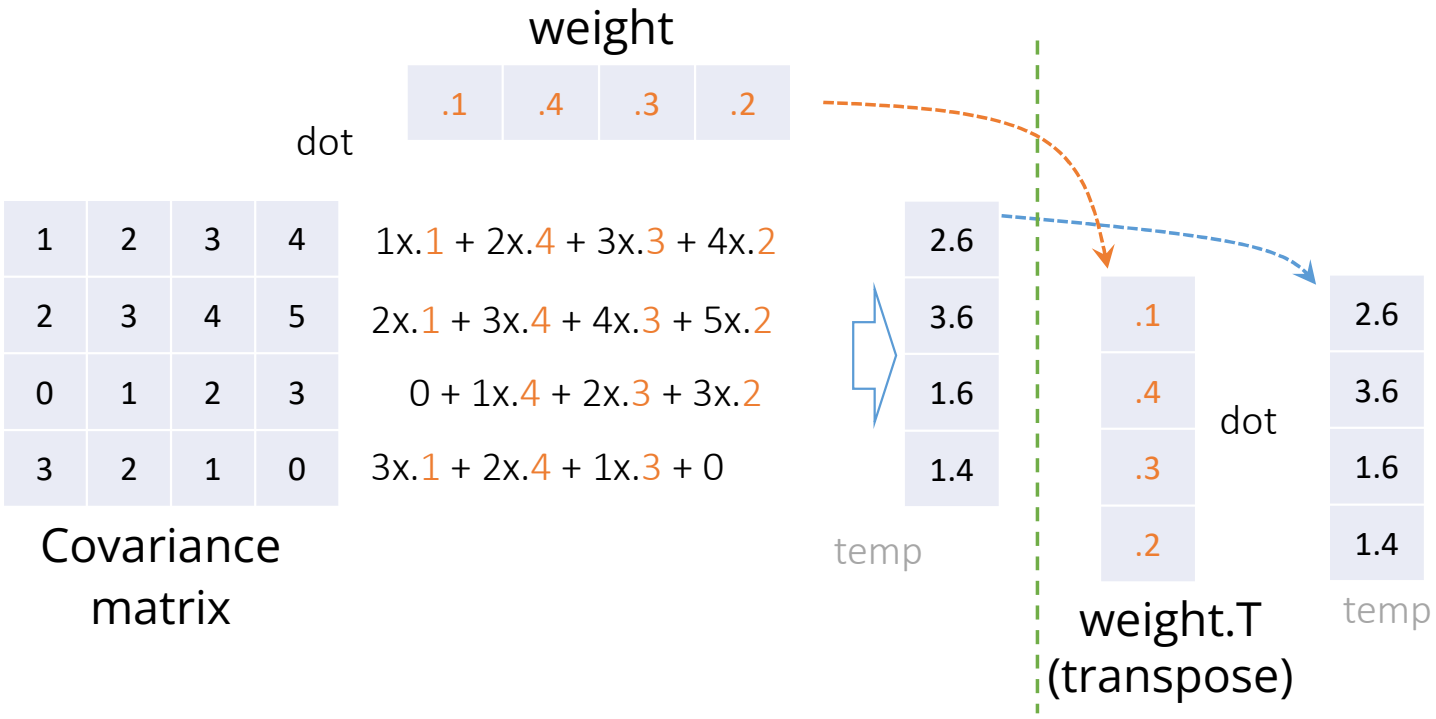
Covariance matrix

2. คำนวณ *portfolio risk* ($=\sqrt{portfolio\ variance}$)

- Portfolio variance of n stocks: (ความแปรปรวนของผลตอบแทนของพอร์ต)
  - Multiply the squared weight of each security by its corresponding variance and adding twice the weighted average weight multiplied by the covariance of all individual security pairs.

2.2 คำนวณ portfolio variance from covariance matrix (cov_mat)

```
port_var  = np.dot(weights.T, np.dot(cov_mat, weights))
```



Portfolio variance (risk^2)

$=.1x2.6 + .4x3.6 + .3x1.6 + .2x1.4$

$= 2.46$

$$2.3\ risk, \sigma_p = \sqrt{variance}$$

Note: ตัวเลขทั้งหมดในหน้านี้เป็นเลขสมมติเพื่อความเข้าใจในการทำงานของ np.dot.

3. คำนวณ Sharpe ratio

- *Sharpe ratio* is the average return earned in excess of the risk-free rate per unit of risk, represented by the standard deviation σ_p. (William Sharpe, 1994).

$$Shape\ ratio = \frac{R_p - Rf}{\sigma_p}$$

- R_p is return rate from investment → annualized return rate (from step 1.)
- R_f or risk-free rate is the return of an investment with risk-free asset, i.e. short-term government treasury bills.
- σ_p is portfolio risk (from step 3.)

- What is a Good Sharpe ratio? "> 1.0"

Contents

- What is portfolio optimization?
- Steps of portfolio selection
- Python code for portfolio selection
- Portfolio optimization using differential evolution algorithm

Python code

In [1]:
```
# import necessary Python libraries
#
from pandas_datareader import data as web
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
import datetime as dt

plt.style.use('fivethirtyeight')
```

In [3]:
```
#
# Parameter settings for this program!
#

RF = 0.01         # risk-free interest, for calculating Sharpe ratio
n_ports = 200     # number of trial (randomized) portfolio to be created
tickers = ['TSLA', 'AAPL', 'AMZN', 'GOOG', 'K', 'SJM', 'CPALL.BK', 'PTT.BK']

start = dt.datetime(2017, 1, 1)
end   = dt.datetime(2021, 12, 30)
```

Download data from Yahoo Finance and save to local disk

In [4]:
```
# Download stock prices from Yahoo Finance using pandas_datareader library
#

# check if directory (or folder) not exist, then make that directory
folder = 'data'
if not os.path.isdir(folder):
    os.mkdir(folder)

# download and save price data of each stock into that directory
for nm in tickers:
    data = web.DataReader(nm, 'yahoo', start, end)
    filename = folder + '\\' + nm + '.csv'
    data.to_csv(filename)
    print('Downloading and saving', len(data), 'rows to', filename, flush=True)
```

```
Downloading and saving 1258 rows to data\TSLA.csv
Downloading and saving 1258 rows to data\AAPL.csv
Downloading and saving 1258 rows to data\AMZN.csv
Downloading and saving 1258 rows to data\GOOG.csv
Downloading and saving 1258 rows to data\K.csv
Downloading and saving 1258 rows to data\SJM.csv
Downloading and saving 1215 rows to data\CPALL.BK.csv
Downloading and saving 1215 rows to data\PTT.BK.csv
```

Read stock price data from local disk

In [5]:
```
# Read stock price returns data directly from csv files in data folder
#

# check if directory (or folder) exist?
folder = 'data'
if not os.path.isdir(folder):
    sys.exit("Folder 'data' does not exist for reading stock prices. Program terminately abnormally.")

returns = pd.DataFrame()

for ticker in tickers:
    filename = folder + '\\' + ticker + '.csv'
    data = pd.read_csv(filename, index_col='Date')
    data = pd.DataFrame(data)
    print('Reading', len(data), 'rows from', filename, flush=True)

    data[ticker] = data['Adj Close'].pct_change()

    if returns.empty:    # 1st time
        returns = data[[ticker]]
    else:
        returns = returns.join(data[[ticker]], how='outer')

returns = returns.dropna()
returns
```

```
Reading 1258 rows from data\TSLA.csv
Reading 1258 rows from data\AAPL.csv
Reading 1258 rows from data\AMZN.csv
Reading 1258 rows from data\GOOG.csv
Reading 1258 rows from data\K.csv
Reading 1258 rows from data\SJM.csv
Reading 1215 rows from data\CPALL.BK.csv
Reading 1215 rows from data\PTT.BK.csv
```

See the daily returns of the stocks

Out[5]:

Date	TSLA	AAPL	AMZN	GOOG	K	SJM	CPALL.BK	PTT.BK
2017-01-05	-0.001057	0.005085	0.030732	0.009048	-0.000957	0.005057	-0.003846	0.018421
2017-01-06	0.009967	0.011148	0.019912	0.015277	0.000274	-0.001084	-0.015444	0.005168
2017-01-09	0.009912	0.009160	0.001168	0.000620	-0.013674	-0.007982	-0.027451	-0.015424
2017-01-10	-0.006097	0.001009	-0.001280	-0.002306	-0.006516	0.005703	0.000000	0.013055
2017-01-11	-0.000609	0.005373	0.003920	0.003877	0.002931	0.014836	-0.008065	-0.007732
...	...	...	...	...	...	...	...	...
2021-12-23	0.057619	0.003644	0.000184	0.001317	0.001434	-0.001127	0.004274	0.013333
2021-12-27	0.025248	0.022975	-0.008178	0.006263	0.006203	0.006467	-0.004292	-0.006579
2021-12-28	-0.005000	-0.005767	0.005844	-0.010914	0.010907	0.010086	0.008621	0.006623
2021-12-29	-0.002095	0.000502	-0.008555	0.000386	-0.000938	-0.000740	0.004274	0.006579
2021-12-30	-0.014592	-0.006578	-0.003289	-0.003427	0.000626	0.001776	0.004255	-0.006536

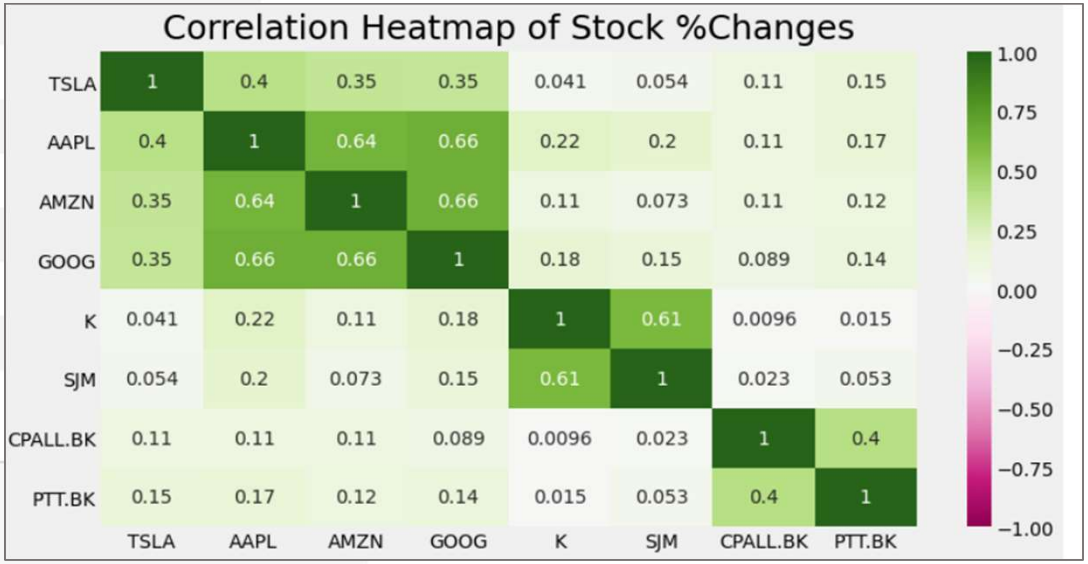
1176 rows × 8 columns

View correlation of stock returns

In [7]:
```
correlation = returns.corr()
correlation
```

Out[7]:

	TSLA	AAPL	AMZN	GOOG	K	SJM	CPALL.BK	PTT.BK
TSLA	1.000000	0.398206	0.353293	0.350614	0.040557	0.053861		
AAPL	0.398206	1.000000	0.640037	0.662079	0.220105	0.202763		
AMZN	0.353293	0.640037	1.000000	0.662878	0.111829	0.073259		
GOOG	0.350614	0.662079	0.662878	1.000000	0.179043	0.149020		
K	0.040557	0.220105	0.111829	0.179043	1.000000	0.614903		
SJM	0.053861	0.202763	0.073259	0.149020	0.614903	1.000000		
CPALL.BK	0.106719	0.109065	0.108593	0.089129	0.009577	0.023377		
PTT.BK	0.148973	0.165171	0.123186	0.137077	0.014869	0.052623		



In [8]:
```
# plotting correlation heatmap

plt.figure(figsize = (12,6))
p = sb.heatmap(correlation, vmin=-1, vmax=1, cmap="PiYG", annot=True)
p.set_title('Correlation Heatmap of Stock %Changes', fontsize=25, pad=10)
plt.show()
```

Perform portfolio selection with 'randomized' weights

In [10]:
```
portfolio_weigths = []
portfolio_returns = []
portfolio_risks = []
sharpe_ratios = []

np.random.seed(123)

for portfolio in range(n_ports):

    # generate random portfolio weights
    weights = np.random.random_sample(len(tickers))
    weights = np.round(weights / np.sum(weights), 3)
    portfolio_weigths.append(weights)

    # calculate annualized return
    annualized_return = np.sum(returns.mean() * weights) * 252
    portfolio_returns.append(annualized_return)

    # calculate annualized covariance matrix & portfolio risk
    cov_mat = returns.cov() * 252

    # calculate portfolio risk = SQRT (WT * Covariance_Matrix * W)
    port_var  = np.dot(weights.T, np.dot(cov_mat, weights))
    port_risk = np.sqrt(port_var)

    #print('port_risk =', port_risk)
    portfolio_risks.append(port_risk)

    # Sharpe ratio
    sr = (annualized_return - RF) / port_risk
    sharpe_ratios.append(sr)
```

In [11]:
```
portfolio_weigths[:5]
```

Out[11]:
```
[array([0.152, 0.063, 0.05 , 0.121, 0.157, 0.093, 0.215, 0.15 ]),
 array([0.134, 0.11 , 0.096, 0.204, 0.123, 0.017, 0.111, 0.206]),
 array([0.043, 0.041, 0.125, 0.125, 0.15 , 0.2  , 0.171, 0.144]),
 array([0.234, 0.105, 0.117, 0.074, 0.095, 0.204, 0.03 , 0.141]),
 array([0.097, 0.111, 0.096, 0.071, 0.096, 0.202, 0.213, 0.113])]
```

In [12]:
```
print('First 3 portfolio_returns =', portfolio_returns[:3])
print('First 3 portfolio_risks   =', portfolio_risks[:3])
print('First 3 sharpe_ratios     =', sharpe_ratios[:3])
```

```
First 3 portfolio_returns = [0.20578790562865654, 0.2550263726680254, 0.156
First 3 portfolio_risks   = [0.1741602590540481, 0.19139926055156697, 0.150
First 3 sharpe_ratios     = [1.1241824437565666, 1.2801845313399742, 0.9794
```

In [13]:
```
print('Max portfolio_returns =', max(portfolio_returns))
print('Min portfolio_risks   =', min(portfolio_risks))
print('Max sharpe_ratios     =', max(sharpe_ratios))
```

```
Max portfolio_returns = 0.40827909177308436
Min portfolio_risks   = 0.14739848811813533
Max sharpe_ratios     = 1.445721654037839
```

Combine results to see them all with input weights

[14]:
```
# convert to numpy array before pack all of them into a dataframe
#
portfolio_returns = np.array(portfolio_returns)
portfolio_risks   = np.array(portfolio_risks)
sharpe_ratios     = np.array(sharpe_ratios)
```

[15]:
```
portfolio_metrics = [portfolio_returns, portfolio_risks, sharpe_ratios, portfolio_weigths]
```

[16]:
```
port_df = pd.DataFrame(portfolio_metrics).T
port_df = port_df.round(4)
port_df.columns = ['Return', 'Risk', 'Sharpe', 'Weights']
port_df
```

t[16]:

	Return	Risk	Sharpe	Weights
0	0.205788	0.17416	1.124182	[0.152, 0.063, 0.05, 0.121, 0.157, 0.093, 0.21...
1	0.255026	0.191399	1.280184	[0.134, 0.11, 0.096, 0.204, 0.123, 0.017, 0.11...
2	0.156929	0.150018	0.979409	[0.043, 0.041, 0.125, 0.125, 0.15, 0.2, 0.171,...
3	0.285659	0.217892	1.265114	[0.234, 0.105, 0.117, 0.074, 0.095, 0.204, 0.0...
4	0.188859	0.161164	1.109794	[0.097, 0.111, 0.096, 0.071, 0.096, 0.202, 0.2...
...	...	...	...	...
195	0.24494	0.198577	1.183117	[0.221, 0.066, 0.095, 0.058, 0.106, 0.205, 0.1...
196	0.267257	0.191668	1.3422	[0.112, 0.162, 0.173, 0.149, 0.046, 0.204, 0.0...
197	0.297758	0.208397	1.380818	[0.133, 0.204, 0.154, 0.167, 0.051, 0.084, 0.0...
198	0.26823	0.205789	1.254833	[0.2, 0.155, 0.026, 0.146, 0.091, 0.019, 0.21,...
199	0.272732	0.198407	1.324206	[0.146, 0.205, 0.076, 0.157, 0.11, 0.029, 0.12...

See top 5 portfolios with the highest returns

+40.8%

```
[17]: # show ports with the highest return
      #
      highest_return = port_df.sort_values('Return', ascending=False).head(5)
      print(highest_return)
      print(tickers)
```

```
       Return      Risk    Sharpe  \
147  0.408279  0.283036  1.407167
116  0.383038  0.288936  1.291075
28   0.375578  0.273075  1.338744
145   0.37039  0.251463  1.433173
37   0.357568  0.254509   1.36564

                                                 Weights
147  [0.289, 0.393, 0.119, 0.047, 0.028, 0.083, 0.0...
116  [0.378, 0.145, 0.146, 0.02, 0.164, 0.006, 0.01...
28   [0.343, 0.034, 0.173, 0.188, 0.117, 0.018, 0.0...
145  [0.226, 0.25, 0.138, 0.21, 0.094, 0.014, 0.068...
37   [0.258, 0.262, 0.038, 0.185, 0.005, 0.223, 0.0...
['TSLA', 'AAPL', 'AMZN', 'GOOG', 'K', 'SJM', 'CPALL.BK', 'PTT.BK']
```

See top 5 portfolios with the lowest risks

[18]:
```
# show ports with the lowest risk
#
lowest_risk = port_df.sort_values('Risk').head(5)
print(lowest_risk)
print(tickers)
```

```
       Return      Risk   Sharpe  \
129  0.151452  0.147398  0.95966
2    0.156929  0.150018 0.979409
131  0.143882  0.150362 0.890396
132   0.13075  0.151214 0.798541
43   0.139514  0.152388 0.849895

                                                 Weights
129  [0.015, 0.057, 0.108, 0.192, 0.222, 0.014, 0.3...
2    [0.043, 0.041, 0.125, 0.125, 0.15, 0.2, 0.171,...
131  [0.025, 0.052, 0.242, 0.0, 0.285, 0.123, 0.258...
132  [0.058, 0.02, 0.05, 0.107, 0.137, 0.255, 0.175...
43   [0.061, 0.046, 0.074, 0.072, 0.195, 0.182, 0.1...
['TSLA', 'AAPL', 'AMZN', 'GOOG', 'K', 'SJM', 'CPALL.BK', 'PTT.BK']
```

See top 5 portfolios with the highest Sharpe ratios

[19]:
```
# show ports with the highest Sharpe ratio
#
highest_sharpe = port_df.sort_values('Sharpe', ascending=False).head(5)
print(highest_sharpe)
print(tickers)
```

```
       Return      Risk    Sharpe  \
169  0.345514  0.232074  1.445722
154  0.332911  0.224166  1.440503
145   0.37039  0.251463  1.433173
111  0.320384  0.218123  1.422979
193   0.31302  0.214289  1.414069

                                                 Weights
169  [0.184, 0.179, 0.223, 0.19, 0.063, 0.054, 0.05...
154  [0.152, 0.208, 0.196, 0.208, 0.008, 0.131, 0.0...
145  [0.226, 0.25, 0.138, 0.21, 0.094, 0.014, 0.068...
111  [0.134, 0.23, 0.208, 0.175, 0.11, 0.061, 0.058...
193  [0.132, 0.157, 0.353, 0.062, 0.028, 0.174, 0.0...
['TSLA', 'AAPL', 'AMZN', 'GOOG', 'K', 'SJM', 'CPALL.BK', 'PTT.BK']
```

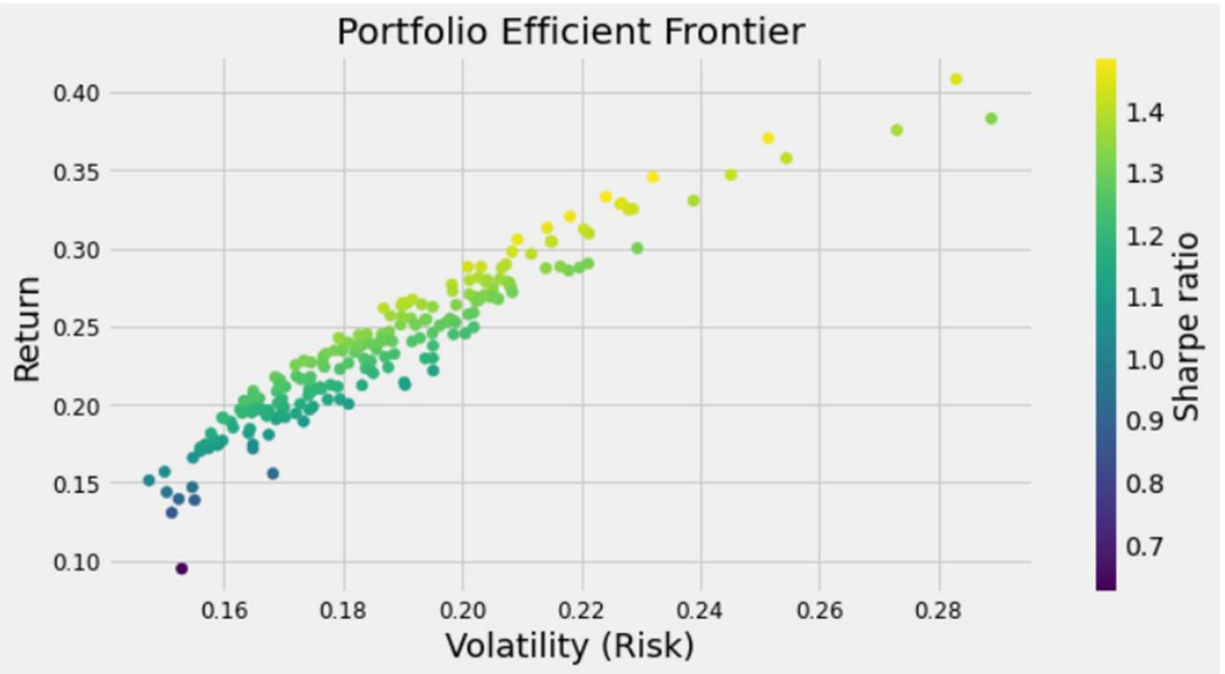
View portfolio efficient frontier from 100 random portfolios

```
[20]: # visualization
      #
      plt.figure(figsize=(10,5))
      plt.scatter(portfolio_risks, portfolio_returns, c = portfolio_returns / portfolio_risks)

      plt.title ('Portfolio Efficient Frontier', fontsize = 20)
      plt.xlabel('Volatility (Risk)', fontsize = 18)
      plt.ylabel('Return', fontsize = 18)
      plt.xticks(fontsize = 12)
      plt.yticks(fontsize = 12)
      plt.colorbar(label = 'Sharpe ratio')
```

n_port = 200

```
[20]: <matplotlib.colorbar.Colorbar at 0x1f85cf102b0>
```



Highest return, lowest risk, or highest Sharpe ratio ที่ได้มานี่ดีที่สุดแล้วไหม?

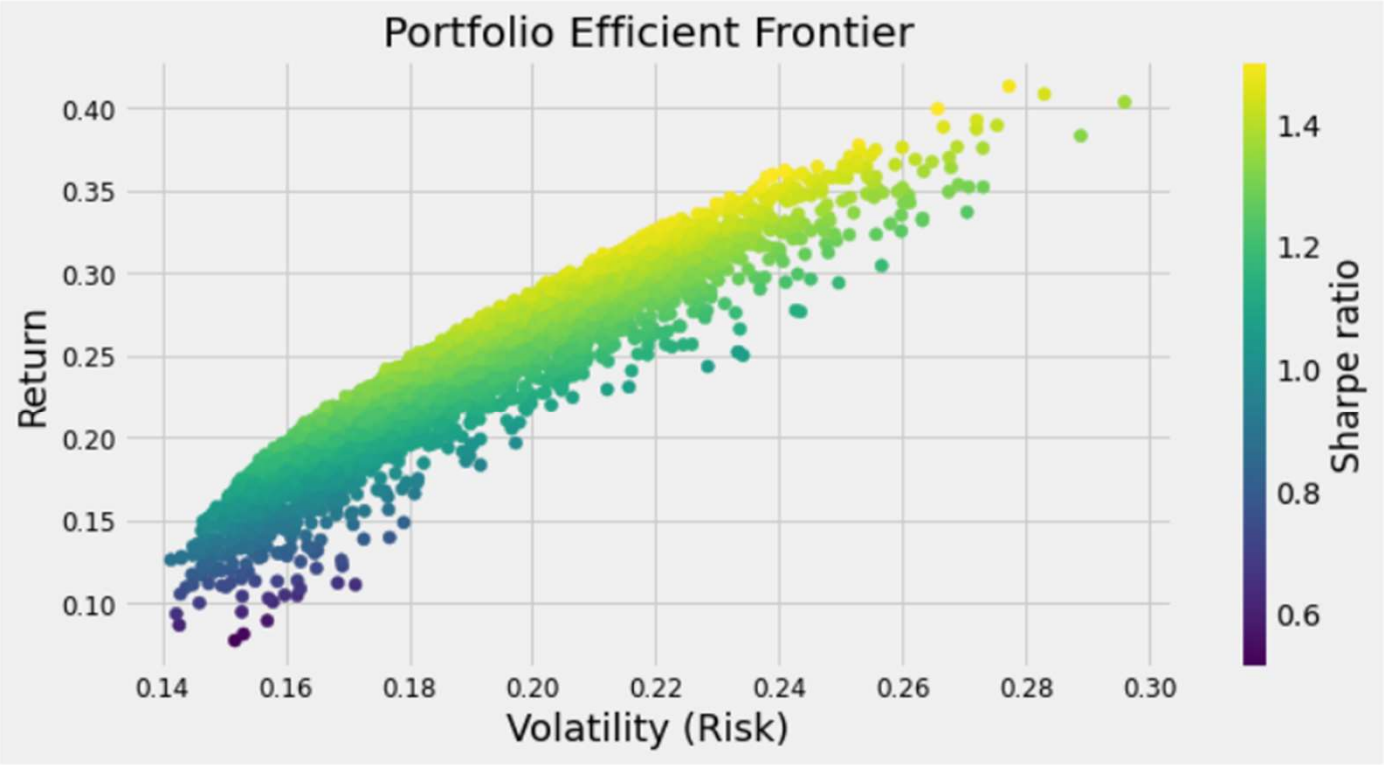
Portfolio efficient frontier of 5000 randomized portfolios

[3]:
```
#
# Parameter settings for this program!
#

RF = 0.01         # risk-free interest, for calculating Sharpe ratio
n_ports = 5000    # number of trial (randomized) portfolio to be created
tickers = ['TSLA', 'AAPL', 'AMZN', 'GOOG', 'K', 'SJM', 'CPALL.BK', 'PTT.BK']

start = dt.datetime(2017, 1, 1)
end   = dt.datetime(2021, 12, 30)
```

n_ports = 5000



Contents

- What is portfolio optimization?
- Steps of portfolio selection
- Python code for portfolio selection
- Portfolio optimization using differential evolution algorithm

Let’s formulate the portfolio optimization problem

- Let’s try the same 8 stocks.
- What is the objective function?
- What are decision variables?
- How to calculate objective function?

Optimize the portfolio using Differential Evolution (DE)

In [22]:
```
from scipy.optimize import differential_evolution
import numpy as np
```

In [23]:
```
def objective_function(weights):

    # make sure they're summed to 100% (or 1.00)
    weights = np.round(weights / np.sum(weights), 5)

    # calculate annualized return
    annualized_return = np.sum(returns.mean() * weights) * 252

    # calculate annualized covariance matrix & portfolio risk
    cov_mat = returns.cov() * 252

    # calculate portfolio risk = SQRT (WT * Covariance_Matrix * W)
    port_var  = np.dot(weights.T, np.dot(cov_mat, weights))
    port_risk = np.sqrt(port_var)

    # Sharpe ratio
    sr = (annualized_return - RF) / port_risk

    return -annualized_return   # to maximize annualized_return
    #return port_risk           # to minimize port_risk
    #return -sr                 # to maximize sr (Sharpe ratio)
```

In [23]:
```
bounds = []
for i in range(len(tickers)):
    bounds.append([0, 1.0])
bounds
```

Out[23]:
```
[[0, 1.0],
 [0, 1.0],
 [0, 1.0],
 [0, 1.0],
 [0, 1.0],
 [0, 1.0],
 [0, 1.0],
 [0, 1.0]]
```

Run the DE to optimize portfolio. First, to maximize return

[24]:
```
# Perform optimization using DE with some parameter settings
# https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html
#

result = differential_evolution(objective_function, bounds, seed=123)
print(result)

# Reproportionate the weights to make their summation equal to 1.0
#
w = result.x
w = np.round(w / np.sum(w), 5)

# round to 4 decimal digits
weights = [round(e, 4) for e in w]

# combine weight and ticker into a final result dataframe
res = pd.DataFrame(tickers, columns=['ticker'])
res['weight'] = weights
res
```

```
     fun: -0.6620013062013055
 message: 'Optimization terminated successfully.'
    nfev: 11529
     nit: 95
 success: True
       x: array([9.55401933e-01, 6.45534225e-03, 2.93890717e-03, 1.24655372e-04,
       4.10338842e-04, 1.75510686e-03, 5.12693547e-04, 1.33427899e-03])
```

[24]:

	ticker	weight
0	TSLA	0.9860
1	AAPL	0.0067
2	AMZN	0.0030
3	GOOG	0.0001
4	K	0.0004
5	SJM	0.0018
6	CPALL.BK	0.0005
7	PTT.BK	0.0014

Create a function to recompute 3 important metrices

[25]:
```python
# Recompute and report portfolio return, portfolio risk, and Sharpe ratio
#
def report_return_risk_sharpe(w):
    weights = np.round(w / np.sum(w), 5)

    # calculate annualized return
    annualized_return = np.sum(returns.mean() * weights) * 252
    print('Return =\t', round(annualized_return, 4))

    # calculate annualized covariance matrix & portfolio risk
    cov_mat = returns.cov() * 252

    # calculate portfolio risk = SQRT (WT * Covariance_Matrix * W)
    port_var  = np.dot(weights.T, np.dot(cov_mat, weights))
    port_risk = np.sqrt(port_var)

    print('Port risk =\t', round(port_risk, 4))

    # Sharpe ratio
    sr = (annualized_return - RF) / port_risk
    print('Sharpe ratio =\t', round(sr, 4))
```

[24]:

	ticker	weight
0	TSLA	0.9860
1	AAPL	0.0067
2	AMZN	0.0030
3	GOOG	0.0001
4	K	0.0004
5	SJM	0.0018
6	CPALL.BK	0.0005
7	PTT.BK	0.0014

[26]:
```python
report_return_risk_sharpe(weights)
```

```
Return =          0.662
Port risk =       0.5943
Sharpe ratio =    1.0972
```

Now, let's try tuning parameters of DE for better results

In [27]:
```
#
# Warning: this will take several seconds or even a few minutes to complete running!
#
result = differential_evolution(objective_function, bounds, seed=123,
                                maxiter=1500, popsize=30, recombination=0.85)
print(result)

# Reproportionate the weights to make their summation equal to 1.0
#
w = result.x
w = np.round(w / np.sum(w), 5)

# round to 4 decimal digits
weights = [round(e, 4) for e in w]
print('Now the weights are', weights, ', totalling', np.sum(weights))

# combine weight and ticker into a final result dataframe
res = pd.DataFrame(tickers, columns=['ticker'])
res['weight'] = weights
res
```

Warning ใช้เวลารันนาน!!!!

```
     fun: -0.6645048943590429
 message: 'Optimization terminated successfully.'
    nfev: 26409
     nit: 109
 success: True
       x: array([9.60558879e-01, 1.67272662e-04, 1.41748222e-03, 7.67433153e-04,
       4.39232596e-04, 1.32698653e-03, 1.04528886e-03, 4.45401494e-04])
Now the weights are [0.9942, 0.0002, 0.0015, 0.0008, 0.0004, 0.0014, 0.0011, 0.0005]
```

DE's default parameter values

maxiter = max. iteration (1000)

popsize = # of DE vectors (15)

recombination = CR (0.7)

Out[27]:

	ticker	weight
0	TSLA	0.9942
1	AAPL	0.0002
2	AMZN	0.0015
3	GOOG	0.0008
4	K	0.0004
5	SJM	0.0014
6	CPALL.BK	0.0011
7	PTT.BK	0.0005

In [28]:
```
report_return_risk_sharpe(weights)
```

```
Return =          0.6645
Port risk =       0.5982
Sharpe ratio =    1.0941
```

Now try to minimize port risk using DE

[29]:
```python
def minimize_port_risk(weights):

    # make sure they're summed to 100% (or 1.00)
    weights = np.round(weights / np.sum(weights), 5)

    # calculate annualized return
    annualized_return = np.sum(returns.mean() * weights) * 252

    # calculate annualized covariance matrix & portfolio risk
    cov_mat = returns.cov() * 252

    # calculate portfolio risk = SQRT (WT * Covariance_Matrix * W)
    port_var  = np.dot(weights.T, np.dot(cov_mat, weights))
    port_risk = np.sqrt(port_var)

    # Sharpe ratio
    sr = (annualized_return - RF) / port_risk

    #return -annualized_return  # to maximize annualized_return
    return port_risk            # to minimize port_risk
    #return -sr                 # to maximize sr (Sharpe ratio)
```

Run DE to minimize portfolio risk

In [30]:
```
result = differential_evolution(minimize_port_risk, bounds, seed=123,
                                maxiter=1500, popsize=30, recombination=0.85)
print(result)

# Reproportionate the weights to make their summation equal to 1.0
#
w = result.x
w = np.round(w / np.sum(w), 5)

# round to 4 decimal digits
weights = [round(e, 4) for e in w]
print('Now the weights are', weights, ', totalling', np.sum(weights))

# combine weight and ticker into a final result dataframe
res = pd.DataFrame(tickers, columns=['ticker'])
res['weight'] = weights
res
```

Warning ใช้เวลารันนาน!!!!

```
     fun: 0.1375749715357068
 message: 'Optimization terminated successfully.'
    nfev: 3849
     nit: 15
 success: True
       x: array([0.00422721, 0.03062257, 0.21973874, 0.27394392, 0.59669227,
       0.41465754, 0.99426137, 0.18130946])
Now the weights are [0.0016, 0.0113, 0.0809, 0.1009, 0.2197, 0.1527, 0.3661, 0.0668] , totalling 0.999999
```

Out[30]:

	ticker	weight
0	TSLA	0.0016
1	AAPL	0.0113
2	AMZN	0.0809
3	GOOG	0.1009
4	K	0.2197
5	SJM	0.1527
6	CPALL.BK	0.3661
7	PTT.BK	0.0668

In [31]:
```
report_return_risk_sharpe(weights)
```

```
Return =          0.0905
Port risk =       0.1376
Sharpe ratio =    0.5851
```

Now try to maximize Sharpe ratio using DE

[32]:
```python
def maximize_sharpe_ratio(weights):

    # make sure they're summed to 100% (or 1.00)
    weights = np.round(weights / np.sum(weights), 5)

    # calculate annualized return
    annualized_return = np.sum(returns.mean() * weights) * 252

    # calculate annualized covariance matrix & portfolio risk
    cov_mat = returns.cov() * 252

    # calculate portfolio risk = SQRT (WT * Covariance_Matrix * W)
    port_var  = np.dot(weights.T, np.dot(cov_mat, weights))
    port_risk = np.sqrt(port_var)

    # Sharpe ratio
    sr = (annualized_return - RF) / port_risk

    #return -annualized_return  # to maximize annualized_return
    #return port_risk           # to minimize port_risk
    return -sr                 # to maximize sr (Sharpe ratio)
```

Run DE to maximize Sharpe ratio

```
In [33]: result = differential_evolution(maximize_sharpe_ratio, bounds, seed=123,
                                         maxiter=1500, popsize=30, recombination=0.85)
         print(result)

         # Reproportionate the weights to make their summation equal to 1.0
         #
         w = result.x
         w = np.round(w / np.sum(w), 5)

         # round to 4 decimal digits
         weights = [round(e, 4) for e in w]
         print('Now the weights are', weights, ', totalling', np.sum(weights))

         # combine weight and ticker into a final result dataframe
         res = pd.DataFrame(tickers, columns=['ticker'])
         res['weight'] = weights
         res
```

Warning ใช้เวลารันนาน!!!!

```
     fun: -1.4915647501397697
 message: 'Optimization terminated successfully.'
    nfev: 3369
     nit: 13
 success: True
       x: array([0.57115736, 0.75293385, 0.99950842, 0.53144731, 0.00843299,
       0.05007659, 0.04339691, 0.00930592])
Now the weights are [0.1926, 0.2538, 0.337, 0.1792, 0.0028, 0.0169, 0.0146, 0.0031] , totalling 1.0
```

Out[33]:

	ticker	weight
0	TSLA	0.1926
1	AAPL	0.2538
2	AMZN	0.3370
3	GOOG	0.1792
4	K	0.0028
5	SJM	0.0169
6	CPALL.BK	0.0146
7	PTT.BK	0.0031

```
In [34]: report_return_risk_sharpe(weights)

         Return =          0.4127
         Port risk =       0.27
         Sharpe ratio =    1.4916
```

Summary of results from different ways

Randomized (n = 200 ports)

	Return	Risk	Sharpe
147	0.408279	0.283036	1.407167
116	0.383038	0.288936	1.291075
28	0.375578	0.273075	1.338744

	Return	Risk	Sharpe
129	0.151452	0.147398	0.95966
2	0.156929	0.150018	0.979409
131	0.143882	0.150362	0.890396

	Return	Risk	Sharpe
169	0.345514	0.232074	1.445722
154	0.332911	0.224166	1.440503
145	0.37039	0.251463	1.433173

Optimize with differential evolution

Out[27]:

	ticker	weight
0	TSLA	0.9942
1	AAPL	0.0002
2	AMZN	0.0015
3	GOOG	0.0008
4	K	0.0004
5	SJM	0.0014
6	CPALL.BK	0.0011
7	PTT.BK	0.0005

```
In [28]: report_return_risk_sharpe(weights)
```

```
Return =          0.6645
Port risk =       0.5982
Sharpe ratio =    1.0941
```

Out[30]:

	ticker	weight
0	TSLA	0.0016
1	AAPL	0.0113
2	AMZN	0.0809
3	GOOG	0.1009
4	K	0.2197
5	SJM	0.1527
6	CPALL.BK	0.3661
7	PTT.BK	0.0668

```
In [31]: report_return_risk_sharpe(weights)
```

```
Return =          0.0905
Port risk =       0.1376
Sharpe ratio =    0.5851
```

Out[33]:

	ticker	weight
0	TSLA	0.1926
1	AAPL	0.2538
2	AMZN	0.3370
3	GOOG	0.1792
4	K	0.0028
5	SJM	0.0169
6	CPALL.BK	0.0146
7	PTT.BK	0.0031

```
In [34]: report_return_risk_sharpe(weights)
```

```
Return =          0.4127
Port risk =       0.27
Sharpe ratio =    1.4916
```

Summary

- Limitations of using Sharpe ratio
  - The Sharpe ratio uses the standard deviation of returns in the denominator as its proxy of total portfolio risk, which assumes that returns are normally distributed.
  - Risk is calculated from estimated return during the computed period.

  - Sortino ratio, like Sharpe ratio, but takes only the downside. (ref.)

- Key Takeaways of Mean-Variance model and Sharpe ratio
  - Mean-variance analysis is a tool used by investors to weigh investment decisions.
  - The analysis helps investors determine the biggest reward at a given level of risk or the least risk at a given level of return.
  - Sharpe ratio must be > 1.0 to be risk-worth investing.

Summary

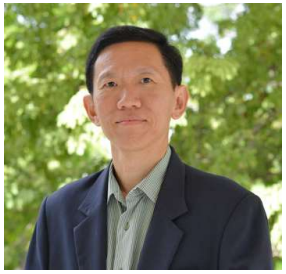
- What is portfolio optimization?
- Steps of computing for portfolio selection (return, risk, Sharpe ratio)
- Python code for portfolio selection with randomized weights
- Portfolio optimization using differential evolution algorithm
  - ✓ Maximize return
  - ✓ Minimize risk
  - ✓ Maximize Sharpe ratio

Warning

- การลงทุนมีความเสี่ยง ผู้ลงทุนต้องศึกษาข้อมูลให้รอบคอบก่อนการตัดสินใจ
- เนื้อหาในวันนี้มิได้เป็นการเสนอแนะหรือชี้นำการลงทุนใดๆ ทั้งสิ้น
- ผู้เรียนควรพิจารณาหลักการ เหตุผล และความเหมาะสมอย่างรอบคอบ

About me

- Assoc. Prof. Chukiat Worasucheep
- Research Areas:
  - Computational Intelligence for financial and engineering applications
  - Data science for financial industry
  - Automatic trading systems
  - Financial prediction



- Teaching:
  - Data science
  - Computational Intelligence
  - Operating Systems
  - Exploring Computer Science

- Education:
  - Master of Science in Computer Science
    - Oregon State University, USA
  - Master of Business Administration (MBA)
    - Thammasat University
  - Bachelor of Engineering (Computer Engineering)
    - Chulalongkorn University