Appendix

[Data Description]

msf.csv: monthly stock file for 100 stocks ranging from 1990 to 2021

msp500_risk_free.csv: monthly sp500 and the risk-free rate

Variables of msf.csv

Variable	Description
permno	permanent number for each stock (e.g., 10107: Microsoft)
ticker	Ticker (e.g., MSFT: Microsoft)
comnam	Company name (e.g., MICROSOFT CORP: Microsoft)
indname	Industry Name classified by Fama-French (please visit French's website for more information)
flag_sector	Dummy variable. 1: Money sector, 0: Well-diversified sector. Missing: none of them
mdate	monthly date, yyyy-mm-dd
ret	monthly holdings returns, including dividends or other cash adjustments i.e., ret = $\frac{Price_t + divident_t}{Price_{t-1}} - 1$, where t is the current month, and the price is adjusted for stock splits or other events. Provided by CRSP (Center for Research in Security Prices) (https://www.crsp.org/) *Please use it without adjusting.*
(Miscellaneous)	
me	market capitalization
me_l1m	market capitalization lagged by one month
prc	price
shrout	Shares outstanding

Variables of msp500_risk_free.csv

Variable	Description
mdate	monthly date, yyyy-mm-dd
spret	monthly returns of the S&P500 index provided by CRSP (Center for Research in Security Prices) (https://www.crsp.org/) Please use it without adjusting.
rf	risk-free rate obtained from the French' website.
(Miscellaneous)	
spindx	S&P500 index

[Python Guideline]

```
## Basic package to import
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize

## read csv files
msf = pd.read_csv('type in the directory of the saved csv file.')
# example
msp500 = pd.read_csv("msp500_risk_free.csv")

## date conversion
# example
msp500['mdate'] = pd.to_datetime(msp500 ['mdate'],format='%Y-%m-%d')

## calculating mean of a vector
a = df['ret'].mean()

## get inverse matrix of matrix A
A_inv = np.linalg.inv(A)

## matrix or vector multiplication of A and B
C = A.dot(B)

## hint for calculating covariance matrix
# use the pivoted matrix of the pandas DataFrame and cov().

## scipy optimization
# minimizing_vol: objective function
# x0: initialization of 100 weights for each stock
```

```
# method: 'SLSQP'
# cons: constraints for the optimization problem, such as summation of weights equal to 1
# bnds: boundary condition, such as weights should be greater than 1
solution = minimize(minimizing_vol, x0, method='SLSQP', constraints=cons, bounds=bnds)

## Figure
# In[Plotting]
msp500 = pd.read_csv("msp500_risk_free.csv")
msp500['mdate'] = pd.to_datetime(msp500['mdate'],format='%Y-%m-%d')

fig, ax = plt.subplots(1,1,figsize=(12,8))

msp500.set_index(['mdate'])['spret'].apply(lambda x:
np.log(1+x)).cumsum().plot(ax=ax)

ax.legend(['S&P500'],frameon=False,loc='upper left')

ax.set_ylabel('Log Cumulative Returns')
ax.set_xlabel('Date')
```