第八周作业

Q1.

股票代码	股票名称
600004	白云机场
600015	华夏银行
600023	浙能电力
600033	福建高速
600183	生益科技

（1）按照投资组合策略，用 2016 年的股票收盘价，使用蒙特卡洛模拟，绘制投资收益与波动率关系图，并且在最大夏普比和最小方差的情况下，分别计算各项资产权重。

代码：

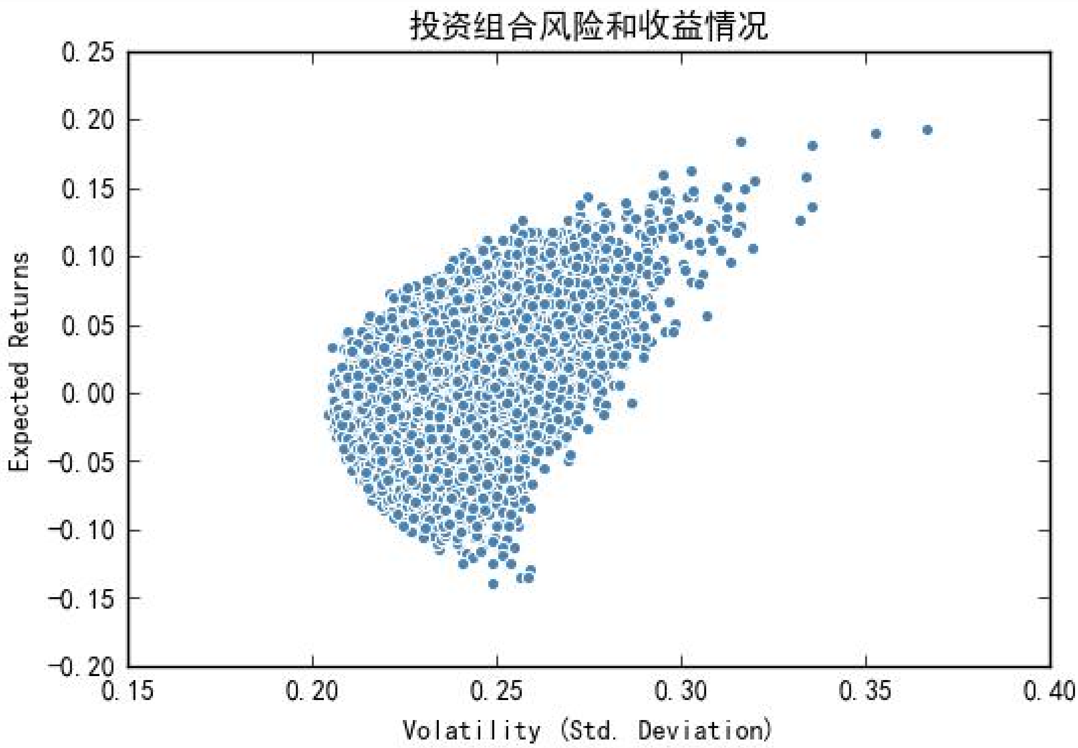
```
# 第八章代码
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt
import scipy as sp
import pandas as pd
# 支持中文显示
import seaborn as sns
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
pd.set_option('display.max_column', 8 )




####### 投资组合实际案例运用
# 通过真实股票数据案例
import tushare as ts
# 股票池
symbol = ['600004','600015','600023','600033','600183']
# 002697 红旗连锁, 600783 鲁信创投, 000413 东旭光电, 601588 北辰实业
data = ts.get_k_data('hs300',start='2016-01-01',end='2016-12-31')
data = data[['date','close']]
data.rename(columns={'close': 'hs300'},inplace=True)
```

```
# 分别为沪深300,北京银行，航天动力和上海能源
# data = pd.DataFrame()
for i in symbol:
    get_data = ts.get_k_data(i,start='2016-01-01',end='2016-12-31')
    get_data = get_data[['date','close']]
    get_data.rename(columns={'close': i + '_close'},inplace=True)
    data = pd.merge(data,get_data,left_on='date',right_on='date',how='left')
data.index = data['date']
del data['date']
del data['hs300']
data = data.dropna() #删除缺失值
data.index = pd.to_datetime(data.index)
(data/data.iloc[0]*100).plot(figsize=(8,4)) #量纲级处理

# 计算收益率
returns = np.log(data/data.shift(1))
returns = returns.dropna()
# 给不同资产分配权重
# 用蒙特卡洛法产生大量的模拟
port_returns = [] #投资组合收益率
port_volatility = [] #波动
stock_weights = []#权重
num_assets =5 #资产数量
num_portfolios = 10000 #产生10000次随机模拟

for single_portfolio in range(num_portfolios):
    weights = np.random.random(num_assets)
    weights /= np.sum(weights)
    port_returns.append(np.dot(weights, returns.mean()*252))#期望收益
    volatility = np.sqrt(np.dot(np.dot(weights,returns.cov()*252),weights.reshape(-1,1))[0])# 波动
    port_volatility.append(volatility)
    stock_weights.append(weights)

portfolio = {'Returns': port_returns, 'Volatility': port_volatility} #创建一个字典
# and weight in the portfolio  投资组合权重
for counter,stock in enumerate(symbol):
    portfolio[stock +'_weight'] = [weight[counter] for weight in stock_weights]
df = pd.DataFrame(portfolio)
#按顺序取数
column_order = ['Returns', 'Volatility'] + [stock+'_weight' for stock in symbol]
df = df[column_order]
df.head()
# 绘制图形
```

```
sns.scatterplot(x = 'Volatility',y = 'Returns',data = df,color="steelblue", marker='o', s=20)
plt.xlabel('Volatility (Std. Deviation)')
plt.ylabel('Expected Returns')
plt.title('投资组合风险和收益情况')
plt.show()
```

```
# 计算夏普比最大的投资组合
# 计算夏普比
# 假设无风险收益率每天为 0.04
df['sharp_ratio'] = (df['Returns'] - 0.04)/df['Volatility']
sharp_ratio = df.loc[df['sharp_ratio']==df['sharp_ratio'].max(),:]#计算夏普比例最大对应的值
min_vari = df.loc[df['Volatility']==df['Volatility'].min(),:]#计算方差最小对应的值
```

结果：



计算夏普比例最大对应的值：

```
      Returns  Volatility  600004_weight  600015_weight  600023_weight  \
1046  0.184358    0.315911       0.366203       0.013222       0.006803

      600033_weight  600183_weight  sharp_ratio
1046       0.002474       0.611297     0.456959
```

计算方差最小对应的值：

```
  Returns    Volatility    600004_weight    600015_weight    600023_weight    \
5749 -0.01566        0.204213            0.212345            0.501919            0.013677

          600033_weight    600183_weight    sharp_ratio
5749            0.267173            0.004885        -0.272558
```

(2) 绘制有效前沿图

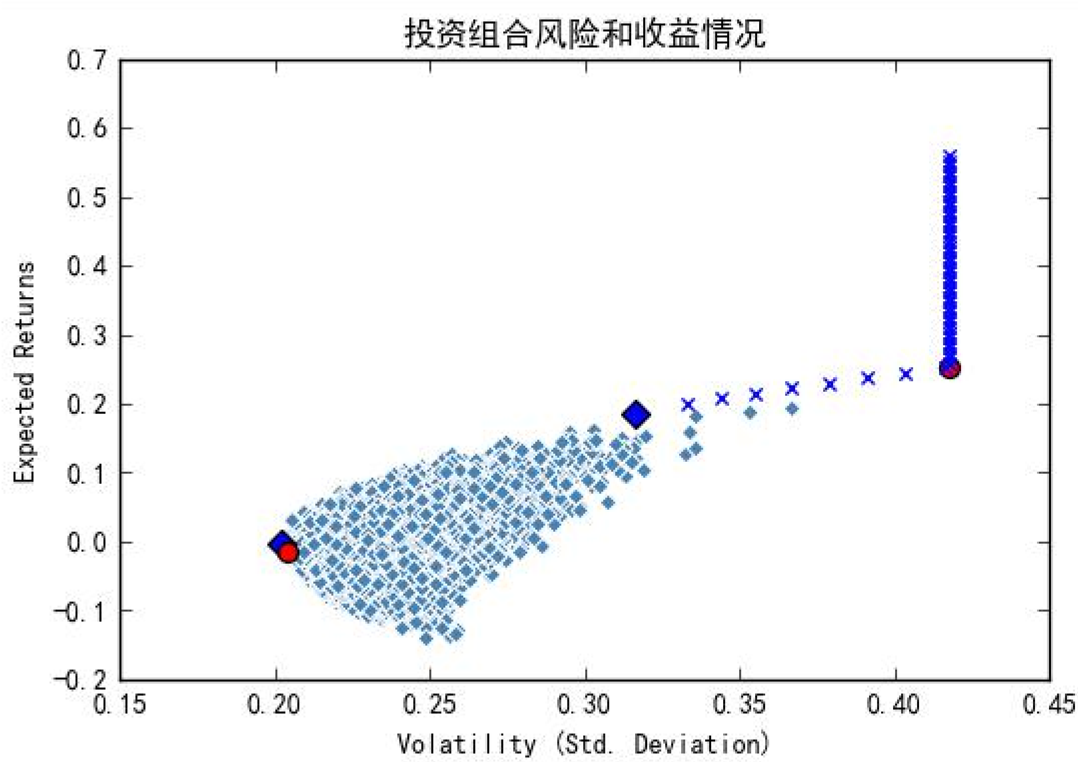
代码：

```
#  使用函数求解
num = 5 #投资组合资产个数
#  定义函数，返回投资组合预期收益,标准差和夏普比例
def statistics(weights):
        weights = np.array(weights)
        port_returns = np.dot(weights.reshape(1,-1),returns.mean()*252)
        port_variance = np.sqrt(np.dot(np.dot(weights, returns.cov()*252), weights.reshape(-1, 1)))
        return np.array([port_returns, port_variance, (port_returns - 0.04)/port_variance])

#最优化投资组合的推导是一个约束最优化问题
import scipy.optimize as sco
#最小化夏普指数的负值
def min_sharpe(weights):
        return -statistics(weights)[2]
#  约束是所有参数(权重)的总和为 1。这可以用 minimize 函数的约定表达如下
cons=({'type':'eq', 'fun':lambda x: np.sum(x)-1})
#我们还将参数值(权重)限制在 0 和 1 之间。这些值以多个元组组成的一个元组形式提供给最小化函数
bnds = tuple((0,1) for x in range(num))
#优化函数调用中忽略的唯一输入是起始参数列表(对权重的初始猜测)。我们简单的使用平均分布。
opts = sco.minimize(min_sharpe, num*[1./num,], method = 'SLSQP', bounds = bnds, constraints = cons)
opts #结算结果
opts['x'].round(3) #权重
statistics(opts['x']) #  得到投资组合，分别为收益率，方差和夏普比例

# # # # # #  方差最小
def min_variance(weights):
```

```
        return statistics(weights)[1]

optv = sco.minimize(min_variance, num*[1.0/num,], method='SLSQP',bounds=bnds,
                                        constraints=cons)
optv['x'] #权重
# 得到方差最小的投资组合
statistics(optv['x'])   # 得到投资组合，分别为收益率，方差和夏普比例

### 投资组合有效边界
def min_variance(weights):
        return statistics(weights)[1]

#在不同目标收益率水平（target_returns）循环时，最小化的一个约束条件会变化。
target_returns = np.linspace(0.2,0.56,50)
target_variance = []
for tar in target_returns:
        cons = ({'type':'eq','fun':lambda x:statistics(x)[0]-tar},{'type':'eq','fun':lambda x:np.sum(x)-1})
        res = sco.minimize(min_variance, num*[1./num,],method = 'SLSQP', bounds = bnds,
constraints = cons)
        target_variance.append(res['fun'])
target_variance = np.array(target_variance)

# 绘制波动最小和夏普比例最高在图形上
sharpe_portfolio =   statistics(opts['x'])   #计算夏普比例最大对应的值
min_variance_port = statistics(optv['x']) ##计算方差最大对应的值
sns.scatterplot(x = 'Volatility',y = 'Returns',color='steelblue',data = df,
                    marker='D', s= 20)
plt.scatter(x= sharpe_portfolio[1], y=sharpe_portfolio[0], c='red', marker='o', s=50)
plt.scatter(x= min_variance_port[1], y=min_variance_port[0], c='blue', marker='D', s=50 )
plt.scatter(x=min_vari.Volatility, y=min_vari.Returns, c='red', marker='o', s=50)
plt.scatter(x=sharp_ratio.Volatility, y=sharp_ratio.Returns, c='blue', marker='D', s=50 )
# 有效边界
#叉号：有效前沿
plt.scatter(target_variance,target_returns, marker = 'x')
plt.xlabel('Volatility (Std. Deviation)')
plt.ylabel('Expected Returns')
plt.title('投资组合风险和收益情况')
plt.show()
```

结果：

叉号：有效前沿



Q2 假设现在有一个投资组合，由股票 A 和股票 B 构成，绘制不同的相关系数下，投资组合方差的变化情况

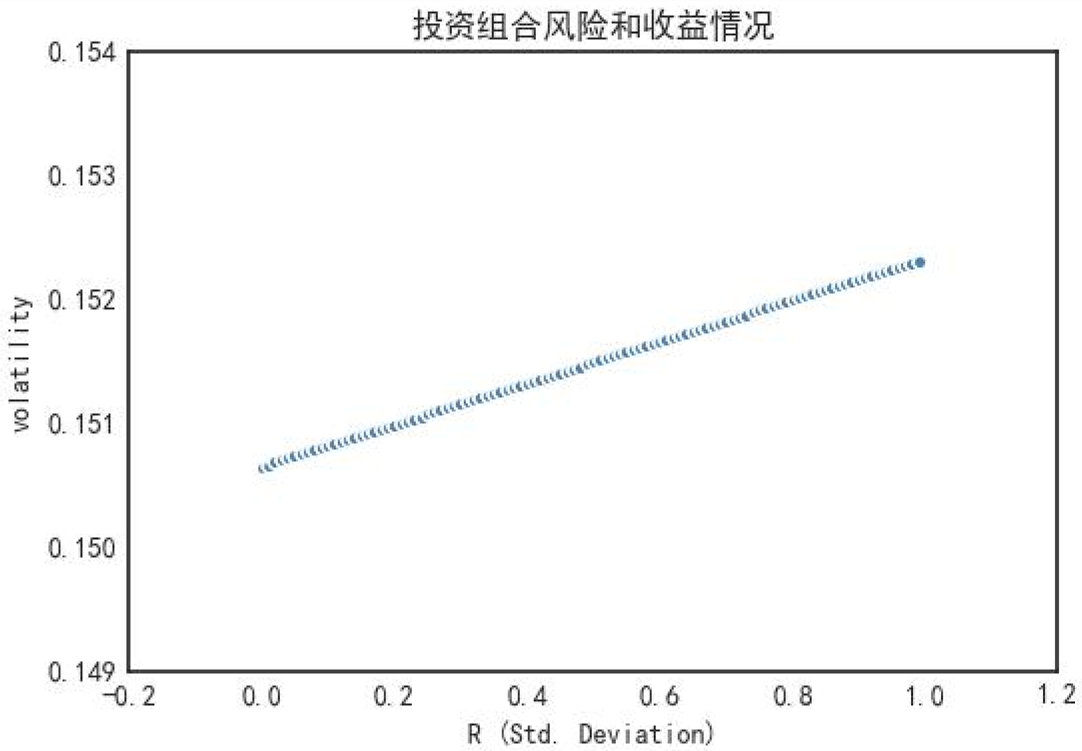
假设股票 A 收益率为 15%，股票 B 收益率为 12%，投资股票 A 的权重为 80%，股票 B 的权重为 20%。

代码：

```
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt
import scipy as sp
import pandas as pd
# 支持中文显示
import seaborn as sns
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
pd.set_option('display.max_column', 8 )
####### 基于股票的历史收益来进行投资组合优化
# 产生三个数据，基于模拟法，模拟三只股票的收益率数据
stock1 = npr.normal(0.15,0.03,100)# 第一只股票收益率
stock2 = npr.normal(0.12,0.05,100)# 第二只股票收益率
stock_data = pd.DataFrame({'A': stock1,'B':stock2})
selected = ['A','B']
# 用蒙特卡洛法产生大量的模拟
port_returns = [] #投资组合收益率
```

```
port_volatility = [] #波动
stock_weights = [] #权重
num_assets = 2 #资产数量
R=[]      #相关系数
num_portfolios = 100    #产生 10000 次随机模拟
for r in range(100):
    r=r/100
    for single_portfolio in range(num_portfolios):
        weights=np.array([0.8,0.2])
        returns = np.dot(weights, stock_data.mean()) #期望收益
        #cov=np.array([[][]])
        volatility = np.sqrt(np.dot(np.dot(weights,

np.asarray([[stock_data['A'].std(),r*stock_data['A'].std()*stock_data['B'].std()],

[r*stock_data['A'].std()*stock_data['B'].std(),stock_data['B'].std()]])),
                                    weights.reshape(-1,1))[0])#波动
        port_returns.append(returns)
        port_volatility.append(volatility)
        stock_weights.append(weights)
        R.append(r)
#创建一个字典，存储相关数据
portfolio = {'volatility': port_volatility, 'R': R}

# and weight in the portfolio  投资组合权重
for counter,symbol in enumerate(selected):
    portfolio[symbol+'_weight'] = [weight[counter] for weight in stock_weights]
df = pd.DataFrame(portfolio) #转换为 dataframe

column_order = ['volatility', 'R'] + [stock+'_weight' for stock in selected]
df = df[column_order]
df.head()

#  绘制图形
plt.style.use('seaborn-white')
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
sns.scatterplot(x = 'R',y = 'volatility',data = df,color="steelblue", marker='o', s=20)
plt.xlabel('R (Std. Deviation)')
plt.ylabel('volatility')
plt.title('投资组合风险和收益情况')
plt.show()
```

结果：


投资组合风险和收益情况
0.154
0.153
0.152
0.151
0.150
0.149
volatility
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
R (Std. Deviation)