Python3对多股票的投资组合进行分析

原创 Asia-Lee 于 2019-04-21 10:45:27 发布 32039 收藏 535 版权

分类专栏： 量化交易 文章标签： 金融 股票 投资组合 数据分析 量化交易

目录

概述：

目前，金融市场总是变幻莫测，充满了不确定因素，是一个有许多投资风险的市场。这与其本身的市场规律和偶然性有关，金融危机、国家政策以及自然灾难等都会影响到金融市场，均会影响投资的收益情况。所以投资者总是希望能够找到应对的方法来减少投资的风险而增加收益。随着老百姓对合理的财富分配理论有着迫切的需求，学会优化投资理财，做到理性投资，是当前投资者最关心的问题。

投资优化的核心问题就是，投资者如何将现有的财富在可投资的风险资产中合理分配，以实现诸如既定风险下收益最大化或者累计收益率最大化等投资目标。现代金融学界的核心课题之一，也即投资组合优化问题，就是研究在不确定环境下如何理性购买并合理配置金融产品，从而实现收益率与风险间的均衡。投资组合理论通常也称为分散投资理论，其核心思想就是不把所有的鸡蛋放进同一篮子里面，即研究在资金有限和期望收益不确定的情况下，投资者该如何分配现有的资金，从而规避掉金融市场中的风险，实现收益最大化。投资的本质就是通过承担一定的风险来换取风险补偿（收益），

且总的来说，想要收益越大所要承受的风险越大。所以，投资者在做出投资决策时，必须根据自身情况接受风险和收益之间的折中。

要实现优化投资就要对投资进行量化。量化投资就是利用计算机科技并采用一定的模型去践行投资理念，实现投资策略的过程。本质上，量化投资就是通过观察市场的规律，试图寻找各个因素与未来股票收益之间的关系，并寻找较为成功，即较大概率成功的规律。

一、股票数据准备

1、股票选择

通过前面对股票数据的分析，我们根据股票的**年化收益率**和股票的**离散系数**，从20支股票中选取5支股票进行投资组合。

股票代码如下：'601318.SH', '601336.SH', '601398.SH', '601888.SH', '603993.SH'。

本文打算用以上5支股票构建投资组合，并用2016年3月1日—2017年12月31日的历史数据进行回溯测试。

首先，导入将要用到的Python包。

```
import matplotlib.pyplot as plt
from pandas import read_excel
import numpy as np
```

2、获取每支股票的收盘价

将股票的每日的收盘价存入数据框 StockPrices 变量中。

```
# 创建空的DataFrame变量，用于存储股票数据
StockPrices = pd.DataFrame()
market_value_list=[]  #存储每支股票的平均市值
# 创建股票代码的列表
ticker_list = ['601318.SH', '601336.SH', '601398.SH', '601888.SH', '603993.SH']
# 使用循环，挨个获取每只股票的数据，并存储每日收盘价
for ticker in ticker_list:
    stock_data = read_excel('stock_data/'+ticker+'.xlsx', parse_dates=['日期'], index_col='日期')
    stock_data=stock_data.loc['2016-03-01':'2017-12-29']
    StockPrices[ticker] = stock_data['收盘价']  #获取每支股票的收盘价
    #将每支股票的市值均值存入列表中
    market_value_list.append(stock_data['市值'].mean())
StockPrices.index.name = 'date'  # 日期为索引列
# 输出数据的前5行
print(StockPrices.head())
```

date	601318.SH	601336.SH	601398.SH	601888.SH	603993.SH
2016-03-01	29.26	34.86	4.03	42.19	3.33
2016-03-02	30.18	36.49	4.11	43.62	3.53
2016-03-03	30.38	36.77	4.13	42.93	3.56
2016-03-04	31.55	37.80	4.31	41.83	3.50
2016-03-07	31.05	37.75	4.29	43.03	3.70

3、计算股票的日收益率

计算股票每天的收益率，将数据存储在数据框 StockReturns 变量中。

```
# 计算每日收益率，并丢弃缺失值
StockReturns = StockPrices.pct_change().dropna()
# 打印前5行数据
```

```
# 打印前5行数据
print(StockReturns.head())
```

date	601318.SH	601336.SH	601398.SH	601888.SH	603993.SH
2016-03-02	0.031442	0.046758	0.019851	0.033894	0.060060
2016-03-03	0.006627	0.007673	0.004866	-0.015818	0.008499
2016-03-04	0.038512	0.028012	0.043584	-0.025623	-0.016854
2016-03-07	-0.015848	-0.001323	-0.004640	0.028688	0.057143
2016-03-08	0.008052	-0.009007	0.009324	-0.003254	-0.021622

至此，我们已经准备好了用于分析的数据 StockReturns， 它记录了5支股票2016年-2017年每天的收益率。

二、投资组合的收益计算

我们选了5支股票，可资金怎么分配呢？哪只买多些，哪只买少些？这就需要对它们设置相应的权重，下面我们采用三种权重分配的方案，来计算不同组合下的投资收益。

1、给定权重的投资组合

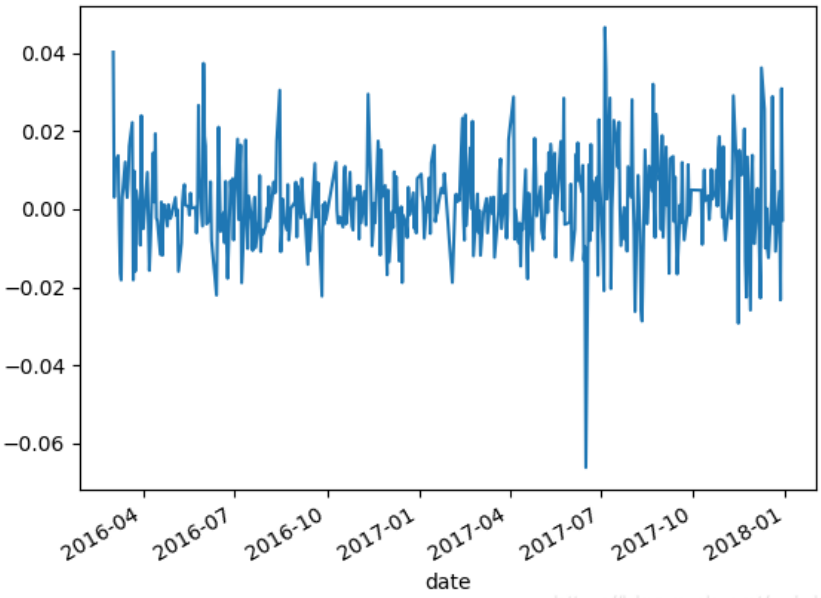
第一种方案是预先设置一组权重（所有股票权重的和为1）。这5支股票：'601318.SH', '601336.SH', '601398.SH', '601888.SH', '603993.SH'对应的权重如下：[0.32, 0.15, 0.10, 0.18, 0.25]。

我们将每支股票的收益，乘上其对应的权重，得到加权后的股票收益；再对所有股票加权后的收益求和，得到该组合投资的收益。

```
# 将收益率数据拷贝到新的变量 stock_return 中，这是为了后续调用的方便
stock_return = StockReturns.copy()

# 设置组合权重，存储为numpy数组类型
portfolio_weights = np.array([0.32, 0.15, 0.10, 0.18, 0.25])
# 计算加权的股票收益
WeightedReturns = stock_return.mul(portfolio_weights, axis=1)
# 计算投资组合的收益
StockReturns['Portfolio'] = WeightedReturns.sum(axis=1)
# 打印前5行数据
print(StockReturns.head())

# 绘制组合收益随时间变化的图
StockReturns.Portfolio.plot()
plt.show()
```

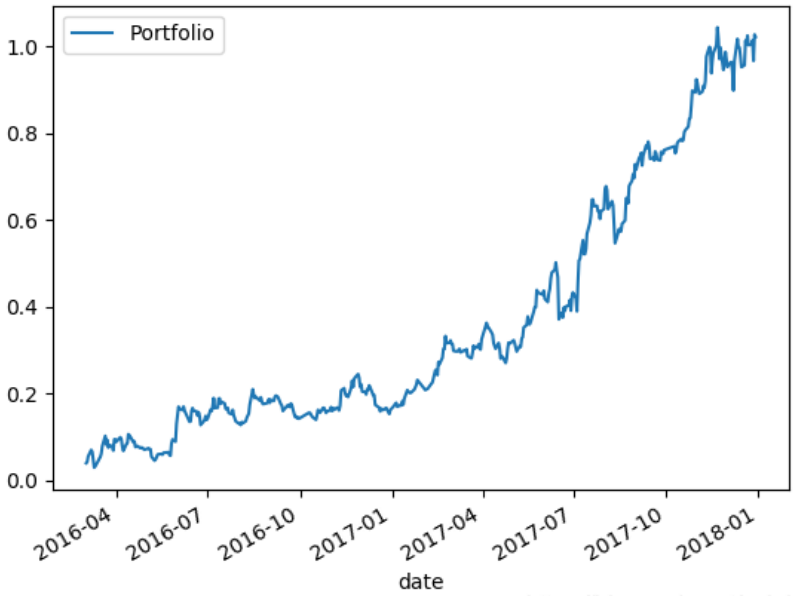
date	601318.SH	601336.SH	601398.SH	601888.SH	603993.SH	Portfolio
2016-03-02	0.031442	0.046758	0.019851	0.033894	0.060060	0.040176
2016-03-03	0.006627	0.007673	0.004866	-0.015818	0.008499	0.003036
2016-03-04	0.038512	0.028012	0.043584	-0.025623	-0.016854	0.012058
2016-03-07	-0.015848	-0.001323	-0.004640	0.028688	0.057143	0.013716
2016-03-08	0.008052	-0.009007	0.009324	-0.003254	-0.021622	-0.003833

绘制该组合投资收益随时间变化的图如下：



定义累积收益曲线绘制函数**cumulative_returns_plot()**，并绘制给定权重投资组合的累积收益曲线

```
# 定义累积收益曲线绘制函数
def cumulative_returns_plot(name_list):
    for name in name_list:
        CumulativeReturns = ((1+StockReturns[name]).cumprod()-1)
        CumulativeReturns.plot(label=name)
    plt.legend()
    plt.show()
# 计算累积的组合收益，并绘图
cumulative_returns_plot(['Portfolio'])
```

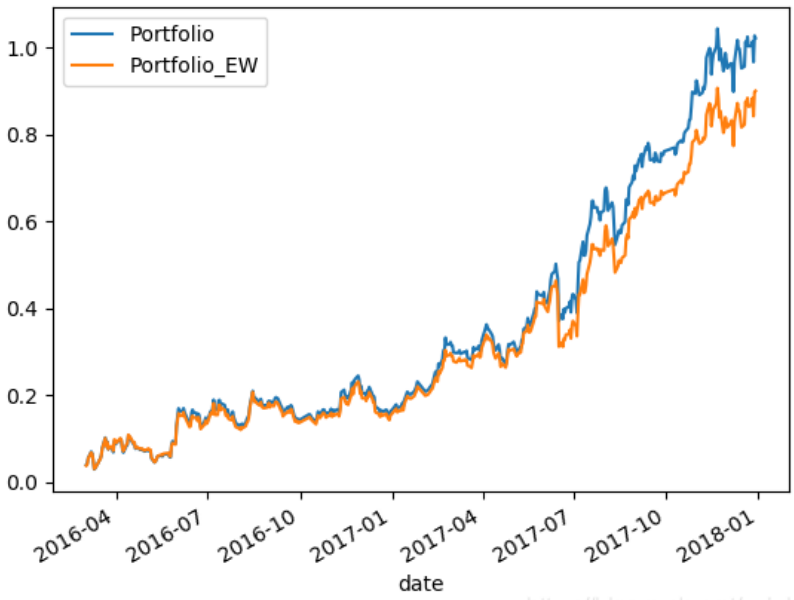


2、等权重的投资组合

第二种方案是**平均分配每支股票的权重**，使它们都相等。这是最简单的投资方法，可作为其他投资组合的参考基准。计算方法和上面一致，只需更改存储权重的数组。

```
# 设置投资组合中股票的数目
numstocks = 5
# 平均分配每一项的权重
portfolio_weights_ew = np.repeat(1/numstocks, numstocks)
# 计算等权重组合的收益
StockReturns['Portfolio_EW'] = stock_return.mul(portfolio_weights_ew, axis=1).sum(axis=1)
# 打印前5行数据
print(StockReturns.head())
# 绘制累积收益曲线
cumulative_returns_plot(['Portfolio', 'Portfolio_EW'])
```

date	601318.SH	601336.SH	601398.SH	601888.SH	603993.SH	Portfolio	Portfolio_EW
2016-03-02	0.031442	0.046758	0.019851	0.033894	0.060060	0.040176	0.038401
2016-03-03	0.006627	0.007673	0.004866	-0.015818	0.008499	0.003036	0.002369
2016-03-04	0.038512	0.028012	0.043584	-0.025623	-0.016854	0.012058	0.013526
2016-03-07	-0.015848	-0.001323	-0.004640	0.028688	0.057143	0.013716	0.012804
2016-03-08	0.008052	-0.009007	0.009324	-0.003254	-0.021622	-0.003833	-0.003301

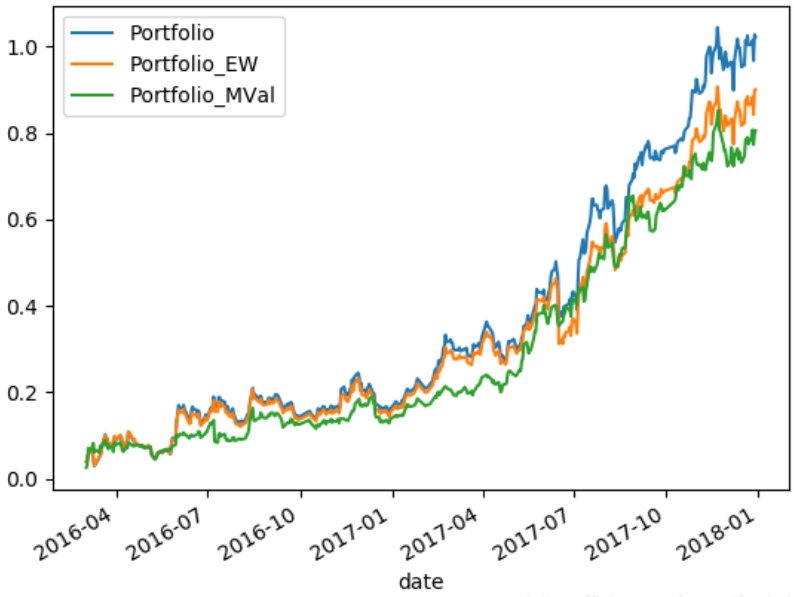


3、市值加权的投资组合

第三种法案是考虑了股票的市值，按市值的占比来分配权重。因此市值高的股票对应的权重就更大，当这些市值高的股票表现良好时，该投资组合的表现也更好。

```
#将上述获得的每支股票的平均市值转换为数组
market_values=np.array(market_value_list)
# 计算市值权重
market_weights = market_values / np.sum(market_values)
# 计算市值加权的组合收益
StockReturns['Portfolio_MVal'] = stock_return.mul(market_weights, axis=1).sum(axis=1)
# 打印前5行数据
print(StockReturns.head())
# 绘制累积收益曲线
cumulative_returns_plot(['Portfolio', 'Portfolio_EW', 'Portfolio_MVal'])
```

date	601318.SH	601336.SH	601398.SH	601888.SH	603993.SH	Portfolio	Portfolio_EW	Portfolio_MVal
2016-03-02	0.031442	0.046758	0.019851	0.033894	0.060060	0.040176	0.038401	0.025896
2016-03-03	0.006627	0.007673	0.004866	-0.015818	0.008499	0.003036	0.002369	0.005208
2016-03-04	0.038512	0.028012	0.043584	-0.025623	-0.016854	0.012058	0.013526	0.038255
2016-03-07	-0.015848	-0.001323	-0.004640	0.028688	0.057143	0.013716	0.012804	-0.005122
2016-03-08	0.008052	-0.009007	0.009324	-0.003254	-0.021622	-0.003833	-0.003301	0.006895



实际问题中，我们需要根据最终的投资累积收益来确定最优的投资组合。本实验中，可以看出给定权重的投资组合稍微优于其它两种，但是不太明显。所以针对不同的问题需要具体分析。

三、投资组合的相关性分析

1、投资组合的相关矩阵

相关矩阵用于估算多支股票收益之间的线性关系，可使用pandas数据框内建的 `.corr()` 方法来计算。

```
# 计算相关矩阵
correlation_matrix = stock_return.corr()
# 输出相关矩阵
print(correlation_matrix)
```

	601318.SH	601336.SH	601398.SH	601888.SH	603993.SH
601318.SH	1.000000	0.701984	0.330955	0.114626	0.000129
601336.SH	0.701984	1.000000	0.375629	0.098861	0.034417
601398.SH	0.330955	0.375629	1.000000	0.021648	-0.005072
601888.SH	0.114626	0.098861	0.021648	1.000000	-0.045553
603993.SH	0.000129	0.034417	-0.005072	-0.045553	1.000000

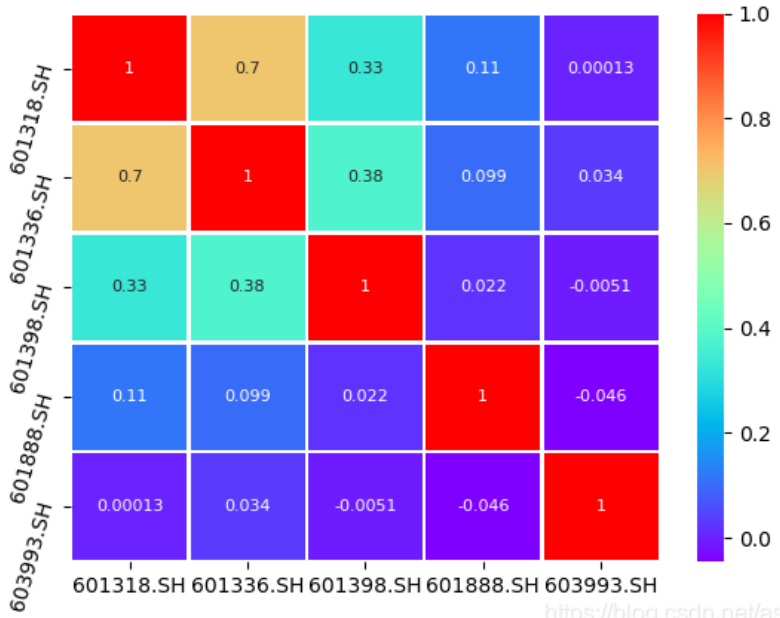
矩阵中每一个元素都是其对应股票的**相关系数**，**取值从-1到1，正数代表正相关，负数代表负相关。**

我们观察到矩阵的对角线永远是1，因为自己和自己当然是完全相关的。另外相关矩阵也是对称的，即上三角和下三角呈镜像对称。

为了便于观察，可以将**数值的相关矩阵**用**热图**的形式展现出来。以下采用了 seaborn 包来绘制热图。

```
import seaborn as sns
#创建热图
```

```
sns.heatmap(correlation_matrix,annot=True,cmap='rainbow',linewidths=1.0,annot_kws={'size':8})
plt.xticks(rotation=0)
plt.yticks(rotation=75)
plt.show()
```



2、投资组合的协方差矩阵

相关系数只反应了股票之间的线性关系，但并不能告诉我们股票的波动情况，而协方差矩阵则包含这一信息。可使用pandas数据框内建的 .cov() 方法来计算协方差矩阵。

```
# 计算协方差矩阵
cov_mat = stock_return.cov()
# 年化协方差矩阵
cov_mat_annual = cov_mat * 252
# 输出协方差矩阵
print(cov_mat_annual)
```

	601318.SH	601336.SH	601398.SH	601888.SH	603993.SH
601318.SH	0.049218	0.047267	0.011549	0.012058	0.000013
601336.SH	0.047267	0.092116	0.017932	0.014227	0.004686
601398.SH	0.011549	0.017932	0.024742	0.001615	-0.000358
601888.SH	0.012058	0.014227	0.001615	0.224837	-0.009689
603993.SH	0.000013	0.004686	-0.000358	-0.009689	0.201224

3、投资组合的标准差

投资组合的风险可以用标准差来衡量，只要知道组合权重和协方差矩阵，就可以通过以下公式进行计算。

$$\sigma = \sqrt{w_T \cdot \Sigma \cdot w}$$

- σ：投资组合的标准差
- Σ：收益的协方差矩阵
- w：投资组合的权重（ 是权重的转置）
- $\cdot$ 是点积运算

在NumPy中，使用.T属性对数组进行**转置**，np.dot()函数用于计算两个数组的点积。

```
# 计算投资组合的标准差
portfolio_volatility = np.sqrt(np.dot(portfolio_weights.T, np.dot(cov_mat_annual, portfolio_weights)))
print(portfolio_volatility)
```

```
0.18631959177724158
```

四、探索股票的最优投资组合

应该选择怎样的组合权重才是最好的呢？是让收益最大吗？还是风险最小？我们需要综合权衡风险和收益这两个因素。

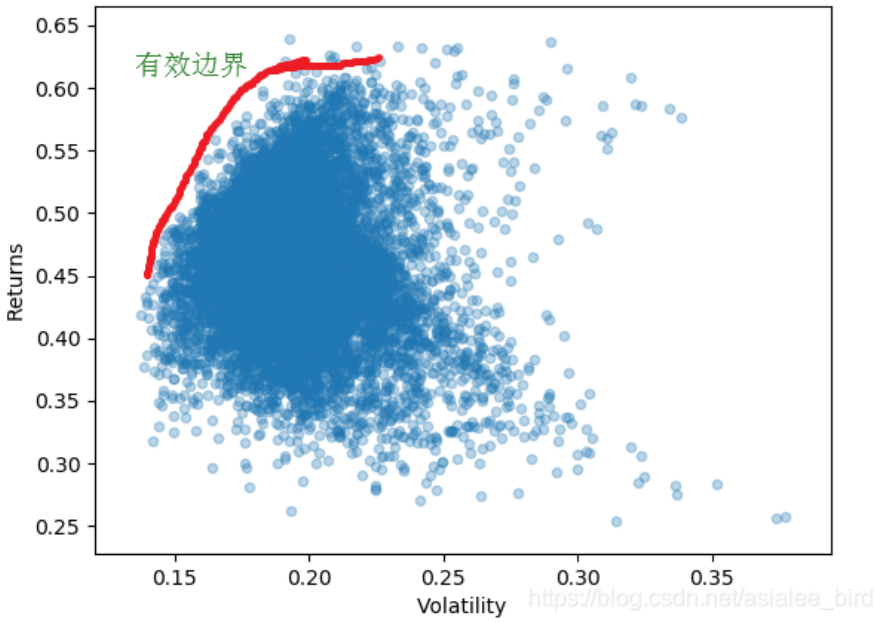
诺贝尔经济学奖得主马科维茨（Markowitz）提出的**投资组合理论**被广泛用于组合选择和资产配置中。该理论中的均值-方差分析法和有效边界模型可用于寻找最优的投资组合。

1、使用蒙特卡洛模拟Markowitz模型

采用蒙特卡洛模拟来进行分析，也就是随机生成一组权重，计算该组合下的收益和标准差，重复这一过程许多次（比如1万次），**将每一种组合的收益和标准差绘制成散点图。**

```
# 设置模拟的次数
number = 10000
# 设置空的numpy数组，用于存储每次模拟得到的权重、收益率和标准差
random_p = np.empty((number, 7))
# 设置随机数种子，这里是为了结果可重复
np.random.seed(7)

#循环模拟10000次随机的投资组合
for i in range(number):
    #生成5个随机数，并归一化，得到一组随机的权重数据
    random5=np.random.random(5)
    random_weight=random5/np.sum(random5)

    #计算年平均收益率
    mean_return=stock_return.mul(random_weight,axis=1).sum(axis=1).mean()
    annual_return=(1+mean_return)**252-1

    #计算年化标准差，也成为波动率
    random_volatility=np.sqrt(np.dot(random_weight.T,np.dot(cov_mat_annual,random_weight)))

    #将上面生成的权重，和计算得到的收益率、标准差存入数组random_p中
    random_p[i][:5]=random_weight
    random_p[i][5]=annual_return
    random_p[i][6]=random_volatility

#将Numpy数组转化为DataF数据框
```

```
RandomPortfolios=pd.DataFrame(random_p)  #设置数据框RandomPortfolios每一列的名称
RandomPortfolios.columns=[ticker +'_weight' for ticker in ticker_list]+['Returns','Volatility']

#绘制散点图
RandomPortfolios.plot('Volatility','Returns',kind='scatter',alpha=0.3)
plt.show()
```



投资的本质是在风险和收益之间做出选择，上图正是刻画了这两个要素。其中每一个点都代表着一种投资组合的情况，**横坐标是代表风险的标准差，纵坐标是收益率。**

Markowitz投资组合理论认为，理性的投资者总是在**给定风险水平下对期望收益进行最大化，或者是在给定收益水平下对期望风险做最小化。**反映在图中也就是红色曲线所示的**有效边界，只有在有效边界上的点才是最有效的投资组合。**

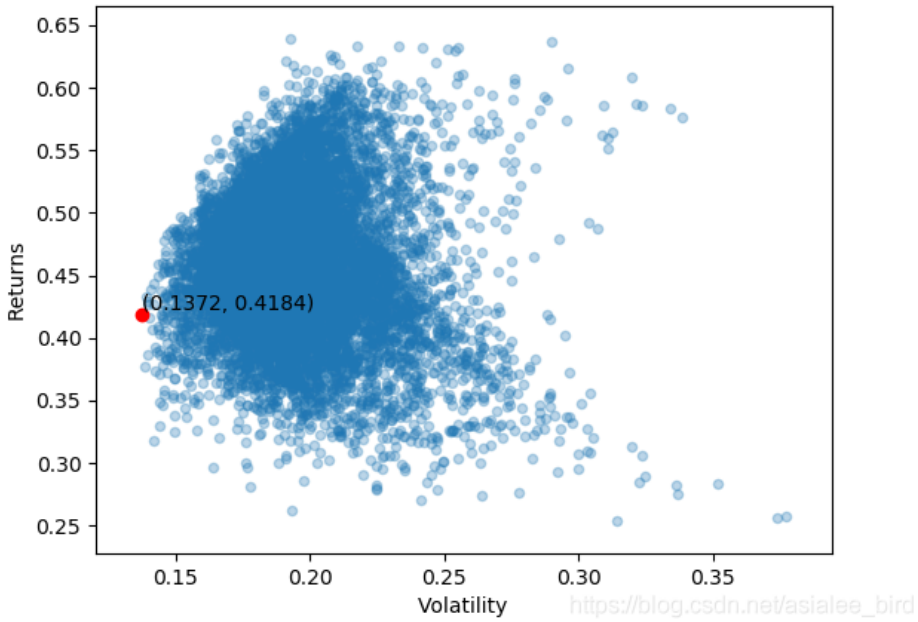
现在我们知道，理性的投资者都会选择有效边界上的投资组合。可具体选择哪个点呢？接着往下看。

2、投资风险最小组合

一种策略是选择最低的风险，且在该风险水平下收益最高的组合，称为最小风险组合（GMV portfolio）。

让我们找到风险最小的组合，并绘制在代表收益-风险的散点图中。

```
# 找到标准差最小数据的索引值
min_index = RandomPortfolios.Volatility.idxmin()

# 在收益-风险散点图中突出风险最小的点
RandomPortfolios.plot('Volatility', 'Returns', kind='scatter', alpha=0.3)
x = RandomPortfolios.loc[min_index,'Volatility']
y = RandomPortfolios.loc[min_index,'Returns']
plt.scatter(x, y, color='red')
#将该点坐标显示在图中并保留四位小数
plt.text(np.round(x,4),np.round(y,4),(np.round(x,4),np.round(y,4)),ha='left',va='bottom',fontsize=10)
plt.show()
```



获取风险最小组合的权重如下：

```
# 提取最小波动组合对应的权重，并转换成Numpy数组
GMV_weights = np.array(RandomPortfolios.iloc[min_index, 0:numstocks])
# 计算GMV投资组合收益
StockReturns['Portfolio_GMV'] = stock_return.mul(GMV_weights, axis=1).sum(axis=1)
#输出风险最小投资组合的权重
print(GMV_weights)
```

```
[0.25595595 0.01126822 0.5362288  0.06293017 0.13361687]
```

3、投资最优组合

（1）夏普比率

夏普比率(Sharpe Ratio)是由诺贝尔奖得主威廉·夏普提出的，用以帮助投资者比较投资的回报和风险。理性的投资者一般都是固定所能承受的风险，追求最大的回报；或者在固定预期回报，追去最小的风险。所以夏普比率计算的是，**每承受一单位的总风险所产生的超额回报**。计算公式如下：

$$夏普比率 = \frac{R_p - R_f}{\sigma_p}$$

其中 R_p 是预期回报率，R_f 是无风险利率，σ_p 是超额收益的标准差。

分子计算了差值，说的是将某项投资与代表整个投资类别的基准进行比较，得到**超额回报**。分母标准差代表**收益的波动率**，对应着风险，因为波动越大预示着风险越高。

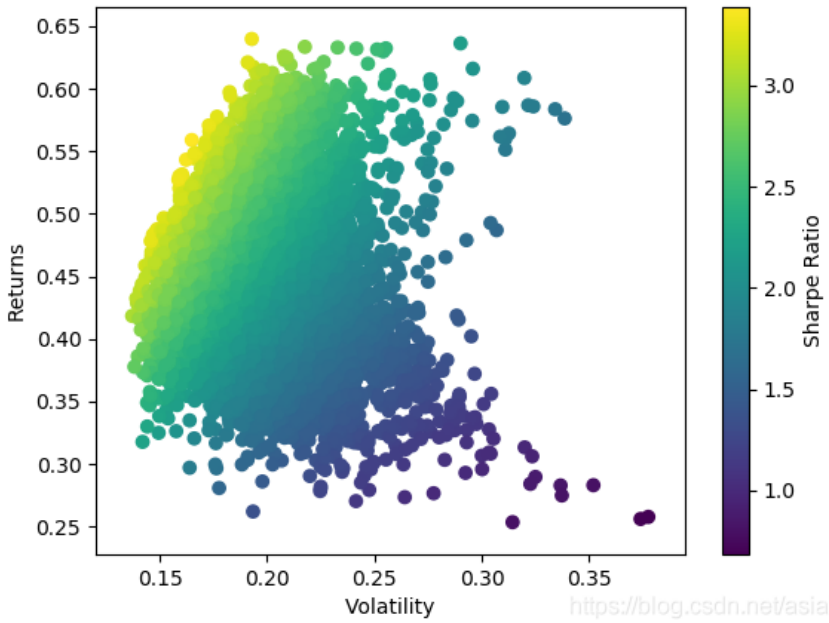
只要将超额回报的均值除以其标准差，即可得到衡量回报和风险的夏普比率。另外需再乘上sqrt(252)（一年有252个交易日），得到年化的夏普比率。

（2）夏普最优组合的选择

其实我们更想**在收益和风险之间找到平衡点**，**夏普比率**这个变量能帮我做出更好的决策，它计算的是每承受一单位的风险所产生的超额回报。

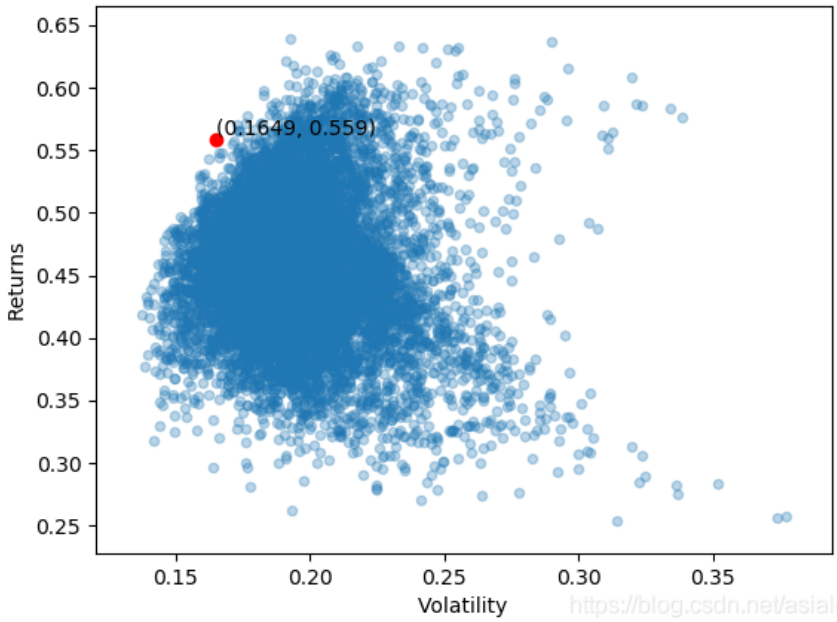
我们首先来计算上述蒙特卡洛模拟的组合所对应的夏普比率，并将之作为第三个变量绘制在收益-风险的散点图中，这里采用颜色这一视觉线索来表征夏普比率。

```
# 设置无风险回报率为0
risk_free = 0
# 计算每项资产的夏普比率
RandomPortfolios['Sharpe'] = (RandomPortfolios.Returns - risk_free) / RandomPortfolios.Volatility
# 绘制收益-标准差的散点图，并用颜色描绘夏普比率
plt.scatter(RandomPortfolios.Volatility, RandomPortfolios.Returns, c=RandomPortfolios.Sharpe)
plt.colorbar(label='Sharpe Ratio')
plt.show()
```



我们发现散点图上沿的组合具有较高的夏普比率。接着再找到夏普比率最大的组合，将其绘制在收益-风险的散点图中。

```
# 找到夏普比率最大数据对应的索引值
max_index = RandomPortfolios.Sharpe.idxmax()
# 在收益-风险散点图中突出夏普比率最大的点
RandomPortfolios.plot('Volatility', 'Returns', kind='scatter', alpha=0.3)
x = RandomPortfolios.loc[max_index,'Volatility']
y = RandomPortfolios.loc[max_index,'Returns']
plt.scatter(x, y, color='red')
#将该点坐标显示在图中并保留四位小数
plt.text(np.round(x,4),np.round(y,4),(np.round(x,4),np.round(y,4)),ha='left',va='bottom',fontsize=10)
plt.show()
```



获取夏普比率最大组合的权重如下：

```
# 提取最大夏普比率组合对应的权重，并转化为numpy数组
MSR_weights = np.array(RandomPortfolios.iloc[max_index, 0:numstocks])
# 计算MSR组合的收益
StockReturns['Portfolio_MSR'] = stock_return.mul(MSR_weights, axis=1).sum(axis=1)
#输出夏普比率最大的投资组合的权重
print(MSR_weights)
```

```
[0.5973892  0.00802452 0.2244098  0.02268103 0.14749544]
```

交流学习资料共享欢迎入QQ群：955817470