tutor_week4

import random
random.sample(sequence, k)

作用： 从给定的sequence中随机选择k个不重复的元素，并以列表形式返回这些元素.

sequence：待抽样的序列，可以是列表、元组、字符串或集合等。
k：抽取元素的数量。

```python
# sequence is list
import random
list1 = [1,2,3,4,5]
random_list_num = random.sample(list1,2)
random_list_num
```

✓ 0.0s

```
[5, 1]
```

```python
# sequence is tuple
import random
tuple1 = (1,2,3,4,5)
random_tuple_num = random.sample(tuple1,2)
random_tuple_num
```

✓ 0.0s

```
[4, 5]
```

import random

random.sample(sequence, k)

作用： 从给定的sequence中随机选择k个不重复的元素，并以列表形式返回这些元素.

sequence：待抽样的序列，可以是列表、元组、字符串或集合等。

k：抽取元素的数量。

```python
# sequence is str
import random
str1 = '12345'
random_str_num = random.sample(str1,2)
random_str_num
```

✓ 0.0s

```
['3', '2']
```

```python
# sequence is set
import random
set1 = {3, "apple", 4.5}
random_set_num = random.sample(set1,2)
random_set_num
```

✓ 0.0s

dict.fromkeys(keys, values)

作用：创建一个新字典

参数：

keys：键的列表

values: 或有，可以不传入，默认为None.

字典（dict）：键值对集合，可变、键不可重复、无序。例如：{"name": "John", "age": 30}。

```python
keys = ['a','b','c']
new_dict = dict.fromkeys(keys)
new_dict
```

✓ 0.0s

```
{'a': None, 'b': None, 'c': None}
```

```python
keys = ['a','b','a']
new_dict = dict.fromkeys(keys)
new_dict
```

✓ 0.0s

```
{'a': None, 'b': None}
```

共享值的问题与解决

注意：

当使用dict.fromkeys()为所有键指定一个可变对象（如列表）作为值时，需要注意所有的键将共享这个可变对象的同一实例。

```python
keys = ['a', 'b', 'c', 'd']
value = []
new_dict = dict.fromkeys(keys, value)
print(new_dict)

# 向这个共享列表添加一个元素
new_dict['a'].append(1)
print(new_dict)
new_dict['b'] = [4]
print(new_dict)
```

✓ 0.0s

```
{'a': [], 'b': [], 'c': [], 'd': []}
{'a': [1], 'b': [1], 'c': [1], 'd': [1]}
{'a': [1], 'b': [4], 'c': [1], 'd': [1]}
```

value_cross_sec_momentum(horizon,frequency,shift=0) 函数细节详解：

```python
def value_cross_sec_momentum(horizon,frequency,shift=0):
    # Compute dates at which portfolio is rebalanced
    rebalancing_dates = [dates[shift+(i+1)*frequency] for i in range(n_dates)]
```

```python
def value_cross_sec_momentum(horizon,frequency,shift=0):
    rebalancing_dates = []
    for i in range(n_dates):
        the_rebal_date = dates[shift+(i+1)*frequency]
        rebalancing_dates.append(the_rebal_date)
    rebalancing_dates
```

列表解析式与for循环的对应关系

value_cross_sec_momentum(horizon,frequency,shift=0) 函数细节详解：

```python
def value_cross_sec_momentum(horizon,frequency,shift=0):
    # Initialize "old units" to 0
    units_old = np.zeros([len(tickers)])
```

np.zeros(shape, dtype=float, order='C')

作用：返回一个给定形状和类型的用0填充的数组；

shape：返回的形状
dtype：数据类型，默认numpy.float64。 dtype=int，返回整数0
order：可选参数，c代表与c语言类似，行优先；F代表列优先

```python
import numpy as np
units_old = np.zeros(8,dtype = int)
units_old
```

✓ 0.0s

```
array([0, 0, 0, 0, 0, 0, 0, 0])
```

```python
import numpy as np
units_old = np.zeros((2,2),dtype = float)
units_old
```

✓ 0.0s

```
array([[0., 0.],
       [0., 0.]])
```

value_cross_sec_momentum(horizon,frequency,shift=0) 函数细节详解：

```python
def value_cross_sec_momentum(horizon,frequency,shift=0):
    for i in range(len(rebalancing_dates)):
        ## We don't trade, if we don't have enough data about past performance available yet
        if (i+1)*frequency<horizon:
            continue
        # Initialize units to 0
        units = np.zeros_like(rel_ma)
```

np.zeros_like(array, dtype = None, order = 'K', subok = True， shape = None)

作用：返回一个给定形状和类型的用0填充的数组

array：一定形状和数据类型的数组，如果没有传入shape，全0数组依据array形状和数据类型创建。

dtype：数据类型。如果没有指定，返回数组的数据类型会和数组a相同。dtype=int，返回整数0

order：可选参数，{'C', 'F', 'A', or 'K'}，指定数组在内存中的存储顺序。默认是'K'，保持与数组array相同的存储顺序。

subok：bool类型，默认为True。 subok = False, 表示返回一个基础类型数组（即，总是返回一个基础的、非子类化的数组）。

shape：指定返回的形状

```python
import numpy as np
# 创建一个3x3的整数数组
a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
# 使用np.zeros_like创建一个形状相同的全零数组
b = np.zeros_like(a)
b
```

✓ 0.0s

```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

```python
import numpy as np
# 创建一个3x3的整数数组
a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
# 使用np.zeros_like创建一个形状相同的全零数组
b = np.zeros_like(a, dtype = float,shape = (2,2))
print(b)
```

✓ 0.0s

```
[[0. 0.]
 [0. 0.]]
```

value_cross_sec_momentum(horizon,frequency,shift=0) 函数细节详解：

```
def value_cross_sec_momentum(horizon,frequency,shift=0):
    for i in range(len(rebalancing_dates)):
        ## We don't trade, if we don't have enough data about past performance available yet
        if (i+1)*frequency<horizon:
            continue
        # Sort stocks according to past performance
        sort_ind = np.argsort(rel_ma)
```

```
x = np.array([3, 1, 2])
sort_index = np.argsort(x)
print('sort_index', sort_index)
sort_result = x[sort_index]
print('sort_result', sort_result)
```

✓ 0.0s

```
sort_index [1 2 0]
sort_result [1 2 3]
```

```
x = np.array([3, 1, 2])
desc_sort_index = np.argsort(-x) #按降序排列
print('desc_sort_index', desc_sort_index)
desc_sort_index = x[desc_sort_index]
print('desc_sort_index', desc_sort_index)
```

✓ 0.0s

```
desc_sort_index [0 2 1]
desc_sort_index [3 2 1]
```

value_cross_sec_momentum(horizon,frequency,shift=0) 函数细节详解：

```python
def value_cross_sec_momentum(horizon,frequency,shift=0):
    # Compute dates at which portfolio is rebalanced
    rebalancing_dates = [dates[shift+(i+1)*frequency] for i in range(n_dates)]

    for i in range(len(rebalancing_dates)):
        ## We don't trade, if we don't have enough data about past performance available yet
        if (i+1)*frequency<horizon:
            continue # 退出本次循环
```

```python
# break 跳出整个循环
number = 0
for number in range(5):
    if number == 3:
        break
    print("number is",number)
print("end loop")
```

✓ 0.0s

```
number is 0
number is 1
number is 2
end loop
```

```python
# continue 跳出本次循环
number = 0
for number in range(5):
    if number == 3:
        continue
    print("number is",number)
print("end loop")
```

✓ 0.0s

```
number is 0
number is 1
number is 2
number is 4
end loop
```

```python
# pass 不做任何事情，一般用做占位语句。
# 目前还不清楚具体是什么，提醒开发者这里将来需要添加代码。
number = 0
for number in range(5):
    if number == 3:
        pass
    print("number is",number)
print("end loop")
```

✓ 0.0s

```
number is 0
number is 1
number is 2
number is 3
number is 4
end loop
```

homework_week4

1. Suppose R has a bi-variate normal distribution with mean vector (1,2) and covariance matrix with diagonal entries 1 and off-diagonal entries 0. Consider two portfolios: portfolio A with weights *x=(-1.,2.)* and portfolio B with weights *x=(0.5,0.5)*. Which of the following statements is correct?

   a) Portfolio A dominates Portfolio B
   b) Portfolio B dominates Portfolio A
   c) Neither of the two portfolios dominates each other.

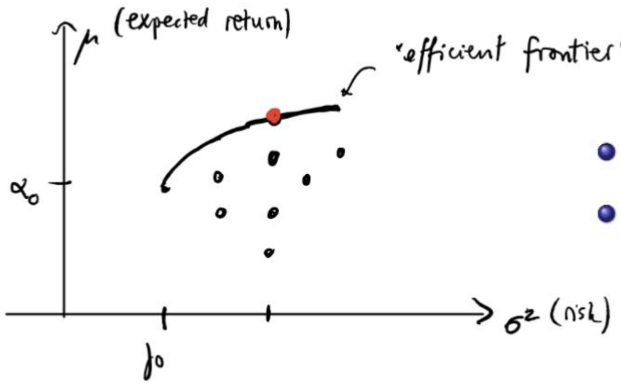
$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- **Notice:**
  - $\mathbb{E}[R_p(x)] = \sum_{i=1}^{n} x_i \mu_i = \mu \cdot x$
  -

$$\sigma_p^2(x) = \text{Var}(R_p(x)) = \text{Cov}(R_p(x), R_p(x)) = \text{Cov}\left(\sum_{i=1}^{n} x_i R_i, \sum_{j=1}^{n} x_j R_j\right)$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \underbrace{\text{Cov}(R_i, R_j)}_{=C_{i,j}}$$

$$= x^\top C x$$



- $\text{Var}(R) < \text{Var}(\bar{R})$ and $\mathbb{E}[R] \geq \mathbb{E}[\bar{R}]$
- $\mathbb{E}[R] > \mathbb{E}[\bar{R}]$ and $\text{Var}(R) \leq \text{Var}(\bar{R})$

Figure: Efficient frontier (illustration)

Expected Return

The expected return for a portfolio can be calculated as:

$$E[R_x] = x \cdot \mu^T$$

- For portfolio A: $E[R_A] = (-1, 2) \cdot (1, 2)^T = -1 * 1 + 2 * 2 = 3$
- For portfolio B: $E[R_B] = (0.5, 0.5) \cdot (1, 2)^T = 0.5 * 1 + 0.5 * 2 = 1.5$

Risk (Standard Deviation)

The risk (variance) for a portfolio is given by:

$$\text{Var}(R_x) = x \Sigma x^T$$

The square root of the variance gives us the standard deviation, which is the risk.

- For portfolio A: $\text{Var}(R_A) = (-1, 2)\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} -1 \\ 2 \end{bmatrix} = (-1, 2) \cdot (-1, 2)^T = 1 * 1 + 2 * 2 = 5$, so the risk (standard deviation) is $\sqrt{5}$.
- For portfolio B: $\text{Var}(R_B) = (0.5, 0.5)\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = (0.5, 0.5) \cdot (0.5, 0.5)^T = 0.5 * 0.5 + 0.5 * 0.5 = 0.5$, so the risk (standard deviation) is $\sqrt{0.5}$.

2. Suppose R has a bi-variate normal distribution with mean vector (1,1) and covariance matrix with diagonal entries 1 and off-diagonal entries -0.5. Consider two portfolios: portfolio A with weights *x=(-1.,2.)* and portfolio B with weights *x=(0.5,0.5)*. Which of the following statements is correct?

a) Portfolio A dominates Portfolio B

b) Portfolio B dominates Portfolio A

c) Neither of the two portfolios dominates each other.

$$\Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$$

Expected Return

Since the mean vector is now $\mu = (1,1)$, the expected returns calculation does not change based on the mean vector but rather on the weights of the portfolios. Let's recalculate with the given weights.

- For portfolio A: $E[R_A] = (-1,2) \cdot (1,1)^T = -1 + 2 = 1$
- For portfolio B: $E[R_B] = (0.5,0.5) \cdot (1,1)^T = 0.5 + 0.5 = 1$

Risk (Standard Deviation)

We calculate the variance for each portfolio using the covariance matrix to find their risks.

- For portfolio A:

$$\mathrm{Var}(R_A) = (-1,2)\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}\begin{bmatrix} -1 \\ 2 \end{bmatrix} = (-1,2) \cdot (-1,-1)^T = 1 + (-2) + (-2) +$$

So the risk (standard deviation) for A is $\sqrt{3}$.

- For portfolio B:

$$\mathrm{Var}(R_B) = (0.5,0.5)\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = (0.5,0.5) \cdot (0.25,0.25)^T = 0.5 * 0.25 +$$

So the risk (standard deviation) for B is $\sqrt{0.25} = 0.5$.

- $\mathrm{Var}(R) < \mathrm{Var}(\bar{R})$ and $\mathbb{E}[R] \geq \mathbb{E}[\bar{R}]$
- $\mathbb{E}[R] > \mathbb{E}[\bar{R}]$ and $\mathrm{Var}(R) \leq \mathrm{Var}(\bar{R})$

3. Suppose you want to modify the cross-sectional momentum strategy to only buy the 5% best-performing stocks, but not short-sell the 5% worst-performing stocks. Which adjustments do you have to make in the function “value_cross_sec_momentum”?

   a) Change line 28 to value.append(value[-1]+(np.sum(units_old*prices_date)))
   b) Delete line 25
   c) Delete line 25 and change line 28 as in a)
   d) Delete line 25 and change line 24 to units[long_ind] = total_cap/(n_titles)

```
9       set_cap = False  # Set to False at the beginning
10      for i in range(len(rebalancing_dates)):
11          ## We don't trade, if we don't have enough data about past performance available yet
12          if (i+1)*frequency<horizon:
13              continue # 退出本次循环
14          date = rebalancing_dates[i] # Current date
15          rel_ma = moving_averages.loc[date,tickers]  # Access current moving averages
16          prices_date = prices.loc[date] # Access current prices
17          sort_ind = np.argsort(rel_ma) # Sort stocks according to past performance
18          long_ind = sort_ind[len(tickers)-n_titles:] # Indices of those stocks that performed best
19          short_ind = sort_ind[:n_titles] # Indices of those stocks that performed worst
20          units = np.zeros_like(rel_ma) # Initialize units to 0
21          ## In the first iteration this will be false; we start with initial capital. Afterwards this is how much our portfolio is
22          if set_cap is True:
23              total_cap = value[-1]+np.sum(units_old*prices_date) # Previous value + gains you make from selling stocks (or buying b
24          units[long_ind] = total_cap/(2*n_titles) # Set equal weights for stocks that you buy
25          units[short_ind] = -total_cap/(2*n_titles) # Set equal weights for stocks that you shortsell
26          units = units/prices_date # Convert from proportion of wealth to actual units
27          ## Update value: liquidate previous position, build current one.
28          value.append(value[-1]+(np.sum(units_old*prices_date)-np.sum(units*prices_date)))
29          ## Set variables for next iteration
30          units_old = units
31          set_cap = True
32      ## At terminal time we liquidate the full position:
33      value.append(value[-1]+(np.sum(units*prices.loc[dates[-1]])))
```

4. Consider the time-series momentum strategy as used in the lecture. Suppose you use initial capital c_2 = 2000000 instead of c_1= 1000000. Denote by S_2 the Sharpe ratio associated to c_2 and by S_1 the Sharpe ratio associate to c_1. How do S_1 and S_2 compare? (You can test this by evaluating the Sharpe ratio with the same parameters as in the lecture at shift=0 with two different initial capitals.)

   a) $S\_1 > S\_2$

   b) $S\_1 = S\_2$

   c) $S\_1 < S\_2$

```python
sharpe=[]
out_of_sample_ind = y_train.shape[0]
for shift in range(9):
    values = np.array(value_strategy_predictor(windowsize,frequency,linear_predictor_mean,shift+out_of_sample_ind))
    returns_strat = (values[1:] - values[:-1])/values[:-1]
    mu_hat = np.mean(returns_strat)
    sigma_hat = np.std(returns_strat)
    ## returns and std are estimated based on a frequency 20/252 -> annualized sharpe ratio has factor np.sqrt(252/20)
    sharpe.append(mu_hat/sigma_hat*np.sqrt(252/frequency))
print(np.mean(sharpe))
print(sharpe)
```

0.5s Python

```
0.179542374853537
[0.48441308784024656, 0.49010630745663225, 0.34320162166603346, 0.22877958085095113, 0.04163870918643329, 0.09394423556635452, -0.1480368830301871, -0.09069058293840025,
```

c_1 = 1000000

```python
sharpe=[]
out_of_sample_ind = y_train.shape[0]
for shift in range(9):
    values = np.array(value_strategy_predictor(windowsize,frequency,linear_predictor_mean,shift+out_of_sample_ind))
    returns_strat = (values[1:] - values[:-1])/values[:-1]
    mu_hat = np.mean(returns_strat)
    sigma_hat = np.std(returns_strat)
    ## returns and std are estimated based on a frequency 20/252 -> annualized sharpe ratio has factor np.sqrt(252/20)
    sharpe.append(mu_hat/sigma_hat*np.sqrt(252/frequency))
print(np.mean(sharpe))
print(sharpe)
```

0.4s Python

```
0.179542374853537
[0.48441308784024656, 0.49010630745663225, 0.34320162166603346, 0.22877958085095113, 0.04163870918643329, 0.09394423556635452, -0.1480368830301871, -0.09069058293840025,
```

c_2 = 2000000

5. Suppose R has a bi-variate normal distribution with mean vector (1,2) and covariance matrix with diagonal entries 1 and off-diagonal entries 0. Using the results from the lecture, what is the maximal expected return that can be achieved by an efficient portfolio with risk equal to 1?

a) 1
b) 1.5
c) 2
d) 2.5

$$\Sigma = \begin{bmatrix} 1 & \boxed{0} \\ 0 & 1 \end{bmatrix}$$

corr(Ra,Rb) = 0

$$\sigma^2 = w_1^2\sigma_1^2 + w_2^2\sigma_2^2$$

with $\sigma_1^2 = \sigma_2^2 = 1$ (the variances of both assets, given by the diagonal of the covariance matrix), and the risk level squared (σ^2) set to 1, we find:

$$1 = w_1^2 + w_2^2$$

w1+w2 =1
w1^2+w2^2 = 1

w2 = 1
w1 = 0
E(R) = 1*2+0*1 = 2

- Note that we must have $\sum_{i=1}^{n} x_i = 1$ for any portfolio.