1.16/9

注意到

E(X)=\frac{1}{4}E(X|0)+\frac{3}{4}E(X|1)

E(X|1) = 2

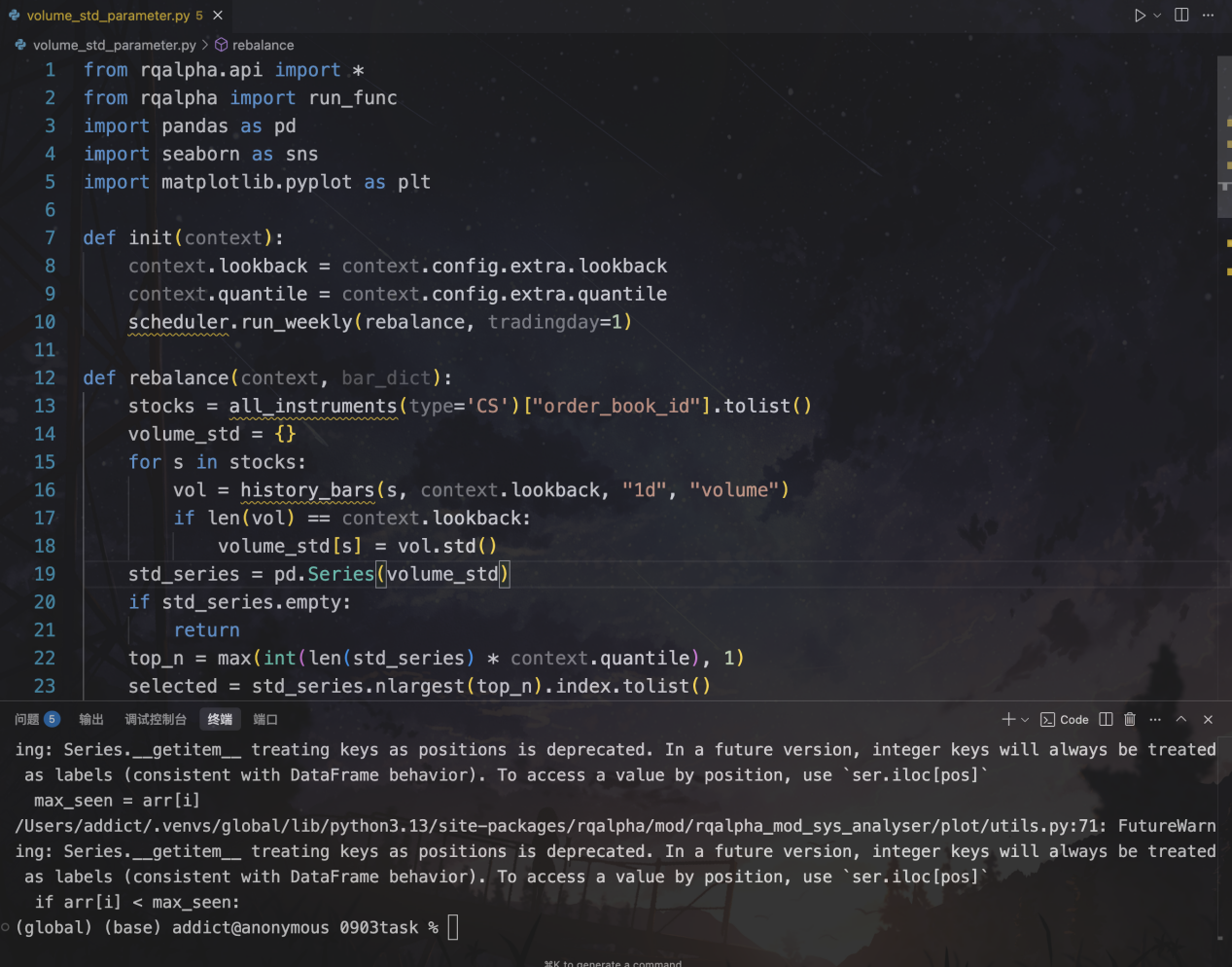
E(X|0) = \frac{1}{4}E(X|00) + \frac{3}{4}E(X|01)

E(X|00) = E(X|0) + 1

E(X|01) = E(X) + 2

解得E(X) = 25/9

去除前两次得16/9

2.

from rqalpha.api import \*

from rqalpha import run\_func

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

def init(context):

context.lookback = context.config.extra.lookback

context.quantile = context.config.extra.quantile

scheduler.run\_weekly(rebalance, tradingday=1)

def rebalance(context, bar\_dict):

stocks = all\_instruments(type='CS')["order\_book\_id"].tolist()

volume\_std = {}

for s in stocks:

vol = history\_bars(s, context.lookback, "1d", "volume")

if len(vol) == context.lookback:

volume\_std[s] = vol.std()

std\_series = pd.Series(volume\_std)

if std\_series.empty:

return

top\_n = max(int(len(std\_series) \* context.quantile), 1)

selected = std\_series.nlargest(top\_n).index.tolist()

for h in list(context.portfolio.positions.keys()):

if h not in selected:

order\_target\_percent(h, 0)

if selected:

w = 1.0 / len(selected)

for s in selected:

order\_target\_percent(s, w)

if \_\_name\_\_=='\_\_main\_\_':

lookback\_range = range(10, 41, 5) # 10,15,...,40

quantile\_range = [0.05, 0.1, 0.15, 0.2] # 5%~20%

records = []

for lb in lookback\_range:

for qt in quantile\_range:

config = {

"base": {

"start\_date": "2010-06-30",

"end\_date": "2019-06-30",

"benchmark": "000300.XSHG",

"accounts": {"stock": 100\_000}

},

"mod": {

"sys\_analyser": {"enabled": True, "plot": False}

},

"extra": {

"lookback": lb,

"quantile": qt

}

}

result = run\_func(init=init, config=config)

summary = result["sys\_analyser"]["summary"]

records.append({

"lookback": lb,

"quantile": qt,

"annual\_return": summary["annualized\_returns"]

})

df = pd.DataFrame(records)

pivot = df.pivot(index="lookback", columns="quantile", values="annual\_return")

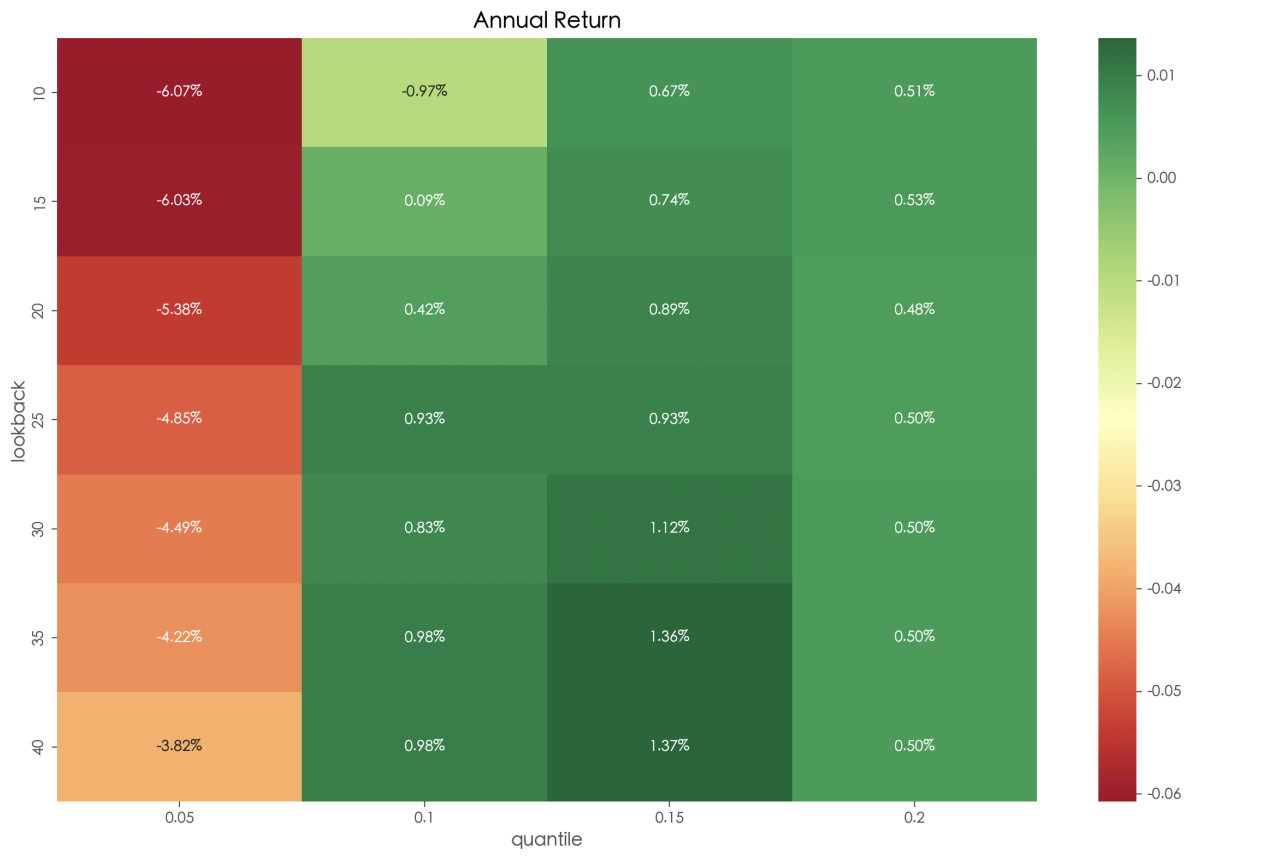
plt.figure(figsize=(12, 8))

sns.heatmap(pivot, annot=True, fmt=".2%", cmap="RdYlGn")

plt.title("Annual Return")

plt.tight\_layout()

plt.show()



回测时间为2010-01-01到2019-01-01

3.def find\_best\_opportunity(bids, asks):

# 寻找所有的套利机会，按利润排序，选利润最大的

opportunities = []

for bid\_key, bid\_data in bids.items():

bid\_price, bid\_quantity = bid\_data

bid\_exchange, bid\_fee\_str = bid\_key.split('-')

bid\_fee = float(bid\_fee\_str)

for ask\_key, ask\_data in asks.items():

ask\_price, ask\_quantity = ask\_data

ask\_exchange, ask\_fee\_str = ask\_key.split('-')

ask\_fee = float(ask\_fee\_str)

if bid\_exchange == ask\_exchange:

continue

buy\_total = bid\_price + bid\_fee

sell\_total = ask\_price - ask\_fee

if buy\_total < sell\_total:

profit = sell\_total - buy\_total

max\_quantity = min(bid\_quantity, ask\_quantity)

total\_profit = profit \* max\_quantity

opportunities.append([total\_profit,bid\_exchange + "-" + bid\_fee\_str,ask\_exchange + "-" + ask\_fee\_str,max\_quantity])

opportunities.sort(reverse=True)

if opportunities:

return opportunities[0]

else:

return None

def execute\_arbitrage(bids, asks, opp):

# 执行套利

asks[opp[2]][1]-=opp[3]

bids[opp[1]][1]-=opp[3]

if asks[opp[2]][1]==0:

del asks[opp[2]]

if bids[opp[1]][1]==0:

del bids[opp[1]]

return opp[0]

opp = find\_best\_opportunity(bids,asks)

overall\_profit = 0

count = 0

while opp:

overall\_profit+=execute\_arbitrage(bids,asks,opp)

count+=1

opp = find\_best\_opportunity(bids,asks)

print(f"共进行{count}次套利，总收益为{overall\_profit}")

print(asks)

print(bids)

4.

取一支固定股票（记i），我们研究与它相似的股票。r为给定参数

每月的指标作Z-score处理：即对于每个指标，对其dataframe操作，增加两列，一列是均值\mu\_t，一列是标准差\sigma\_t，将该指标转换为标准正态分布（即对原dataframe中的每个元素X\_t,变为(X\_t - \mu\_t) / (\sigma\_t).保存到新的dataframe里

再用处理后的指标计算每支股票与i的欧式距离，除了i以外，其他股票每个指标与i中的相应指标相减再平方，得到的值求和开根，得到每支股票与i的欧式距离，这成为一个dataframe，记作dataframe0.

在dataframe0中，每月取出距离i最近的r支股票，按市值加权求和前一个月的收益率得到SIM\_t(i)

所以，对i我们构建了一个因子叫SIM\_t(i)，再减去i过去一月的收益率r\_{t-1}(i)，就得到

\Delta ER\_i(t) = SIM\_t(i) - r\_t(i)

可能的改进：

1.在计算市值的SIM\_t(i)时研报中按市值加权求和，但这难以反映已经被挑选出的r支股票与i的远近关系，相当于把它们视为和i一样近，可能更好的方式是按距离的倒数加权求和。

2.在定义相似指标时可以考虑业务的相似性，我们可以用一个指标来定义两支股票业务的相似性，同行业的股票此指标就较高，不同行业则较低。这样的方法可以让同行业的股票更“近”，但缺点是在同行竞争时会出现此消彼长的状况，这与我们想要构建的“相似”不符。

3.定义距离不一定需要明确公式，可能更好的角度是learning的角度，用机器学习聚类的方式有希望提升对相似性的刻画程度