基于历史模拟法计算投资组合 VaR

1 数据准备

1.1 标的与初始配置

投资标的：黄金 ETF、红利低波 ETF、纳斯达克 ETF。
初始权重分别为 30%、40%、30%。
初始资金：1,000,000 元。

1.2 数据来源与预处理

从 `data.xlsx` 读取三个标的从起始日至 2025-04-18 的每日收盘价数据。
计算每日收益率：

$$r_{j,t} = \frac{P_{j,t}}{P_{j,t-1}} - 1$$

其中：

- $r_{j,t}$ 表示第 j 个资产在第 t 天的收益率；
- $P_{j,t}$ 是第 j 个资产第 t 天的收盘价。

1.3 样本选择与回测区间

历史样本选择：2023-12-29 含当天，向前 500 个交易日。
回测区间：2024-01-01 至 2025-04-18。

2 分析一：三种历史模拟方法计算 VaR 和 Kupiec 检验

2.1 组合的损失推导

计算每个资产在组合中的份额：

$$\text{shares}_j = \frac{w_j V_n}{P_{j,n}}$$

其中：

- w_j 是第 j 个资产的权重；
- V_n 是组合的总市值，初始为 100 万；
- $P_{j,n}$ 是第 j 个资产在日期 n 的价格；
- shares_j 是第 j 个资产的持仓份额。

当价格变动时的市值计算公式是：

$$V_n = \sum_{j=1}^{3} \text{shares}_j P_{j,n}$$

这个公式表达了组合市值 V_n 是组合中每个资产持仓份额和当前价格的加权和。

组合日回报为：

$$R_i = \sum_{j=1}^{3} w_j r_{j,i}$$

其中：

- R_i 是第 i 天组合的收益率；
- $r_{j,i}$ 是第 j 个资产第 i 天的收益率。

组合模拟情景下第 i 天的市值记为：

$$V_i = V_n(1 + R_i)$$

其中：

- V_i 是第 i 天的组合市值；
- V_n 是基准日期的组合市值（如 2023-12-29 日）；
- R_i 是第 i 天的组合收益率。

组合的损失定义为市值的下降，表达为：

$$L_i = V_n - V_i = -V_n R_i$$

即模拟情景第 i 天组合市值的亏损金额。

2.2 传统历史模拟法

方法步骤：

1. 收集过去 $n = 500$ 天的组合损失数据 L_i，将其从大到小排序；
2. 每个历史损失的权重均为 $\frac{1}{n}$；
3. 计算累计权重，找到累计权重大于或等于在置信水平等于 0.99 下 $\alpha = 0.01$ 的第一个损失值作为 VaR。

数学表达为：

$$\text{VaR}_{\text{HS}} = L_{\lceil n \times \alpha \rceil}$$

其中 $\lceil \cdot \rceil$ 表示向上取整。

2.3 时间加权历史模拟法

更重视近期数据，引入时间加权，定义权重为：

$$w_i = \frac{\lambda^{n-i}(1-\lambda)}{1-\lambda^n}, \quad i = 1, 2, \ldots, n$$

其中：

- $\lambda = 0.99$ 是衰减因子，越接近 1，历史数据权重衰减越慢；
- w_i 是第 i 天损失对应的权重，保证权重和为 1。

然后将损失按权重排序，找到累计权重首次达到 $\alpha = 0.01$ 的对应损失作为 VaR：

$$\text{VaR}_{\text{TW}} = L_{(w_i)}$$

2.4 EWMA 波动率加权历史模拟法

考虑收益的波动率变化，步骤如下：

1. 递归计算波动率估计：
$$s_i^2 = \lambda_2 s_{i-1}^2 + (1-\lambda_2) r_i^2$$
其中：
   - s_i 是第 i 天的波动率估计；
   - r_i 是组合第 i 天的收益率；
   - $\lambda_2 = 0.95$ 是波动率衰减因子。
2. 计算标准差：
$$\sigma_i = \sqrt{s_i^2}.$$
3. 对回报进行波动率调整：
$$R_i' = R_i \times \frac{\sigma_{n+1}}{\sigma_i}$$
其中 σ_{n+1} 是第 $n+1$ 天的波动率估计，用于调整波动率的时变性。
4. 使用调整后的损失计算时间加权 VaR，权重同时间加权方法。

最终 VaR 表达为：
$$\mathrm{VaR}_{\mathrm{EWMA}} = L'_{(w_i)}$$
即加权排序后的调整损失的 α 分位点。

2.5 Kupiec 非违约率检验

定义违约事件指示函数：
$$\mathrm{exceptions}_i = \mathbf{I}(L_i > \mathrm{VaR})$$
其中：

- $\mathbf{I}(\cdot)$ 是指示函数，当括号内条件成立时值为 1，否则为 0；
- L_i 是第 i 天的组合损失；
- VaR 是对应置信水平下的风险值。

统计违约事件总数：
$$x = \sum_{i=1}^{n} \mathrm{exceptions}_i$$
其中 n 是回测期间总交易日数，即 2024 年 1 月 1 日至 2025 年 4 月 18 日的交易日数量。

计算似然比统计量：
$$LR = -2\ln(L_0) + 2\ln(L_1)$$
其中
$$L_0 = (1-p)^{n-x} p^x, \quad L_1 = (1-\frac{x}{n})^{n-x} \left(\frac{x}{n}\right)^x$$
参数含义：

- $p = \alpha$ 是设定的 VaR 置信水平（这里是 0.01，即 99% 置信水平 VaR 的违约概率）；
- x/n 是实际观测到的违约率；

- L_0 是假设模型正确时的似然函数；
- L_1 是最大似然估计下的似然函数。

通过 χ^2 分布计算 p 值，用于检验模型违约率是否与理论置信水平一致。

3 分析—结果分析

3.1 VaR 结果

执行 Python 代码 a.py，代码见附录。针对三种历史模拟法：传统历史模拟法、时间加权历史模拟法（$\lambda = 0.99$）、以及 EWMA 波动率加权历史模拟法（$\lambda = 0.95$）进行了 VaR 的计算。计算结果如下所示：

- **传统历史模拟法：** VaR(99%) = 20641.04 元
- **时间加权历史模拟法（$\lambda = 0.99$）：** VaR(99%) = 13553.79 元
- **EWMA 波动率加权历史模拟法（$\lambda = 0.95$）：** VaR(99%) = 10677.19 元

从计算结果中可以看出，传统历史模拟法的 VaR 值最大，这意味着基于过去的 500 个数据点计算的 VaR 预测较为保守，可能高估了风险。相对而言，EWMA 波动率加权法和时间加权历史模拟法通过不同的加权方式使得计算的 VaR 较小，反映出在考虑波动率和时间加权的情况下，风险较为缓和，这可能与历史数据前期波动率大有关。

以下是三种方法的损失排序和权重图示（仅展示部分结果）：

	A	B
1	loss	weight
2	34834.37	0.002
3	26441.25	0.002
4	22268.93	0.002
5	20756.74	0.002
6	20641.04	0.002
7	18249.86	0.002
8	17720.47	0.002
9	16530.42	0.002
10	16076.98	0.002
11	15882.67	0.002
12	15875.83	0.002
13	15620.15	0.002
14	13728.57	0.002
15	13690.43	0.002
16	13587.88	0.002
17	13553.79	0.002
18	13367.64	0.002
19	13278.74	0.002
20	13057.41	0.002
21	12611.73	0.002
22	12602.05	0.002
23	12433.24	0.002
24	11520.32	0.002

Figure 1: 传统历史模拟法的损失排序和对应的权重

loss	weight
34834.37	0.000123
26441.25	0.000162
22268.93	0.000172
20756.74	0.000429
20641.04	0.000438
18249.86	0.000108
17720.47	8.50E-05
16530.42	0.000221
16076.98	9.31E-05
15882.67	0.000117
15875.83	9.12E-05
15620.15	0.000563
13728.57	0.002781
13690.43	0.000194
13587.88	0.004327
13553.79	0.000143
13367.64	8.94E-05
13278.74	0.001376
13057.41	0.000465
12611.73	0.000122
12602.05	0.0084
12433.24	0.006152
11520.32	0.000384

Figure 2: 时间加权历史模拟法的损失排序和对应的权重

loss	weight
13044.39	8.75E-12
11102.01	5.06E-09
10876.72	1.31E-12
10827.64	3.49E-11
10677.19	0.019861
9857.312	5.61E-09
9753.448	7.04E-05
9508.108	1.94E-06
9260.761	0.000673
9100.258	1.88E-12
8933.739	0.00405
8841.33	2.02E-08
8717.566	4.49E-12
8638.328	2.08E-12
8566.732	4.75E-11
8351.892	1.69E-12
8308.492	0.003655
8146.003	1.71E-10
7753.695	9.00E-07
7621.16	6.77E-12
7405.786	0.001782
7274.374	2.88E-09
6959.884	0.000281

Figure 3: EWMA 波动率加权历史模拟法的损失排序和对应的权重

3.2 组合价值

用固定的权重计算出的 2024 年 1 月 1 日至 2025 年 4 月 18 日的组合价值如下图所示（单位为百万元），表明市场逐步上升。

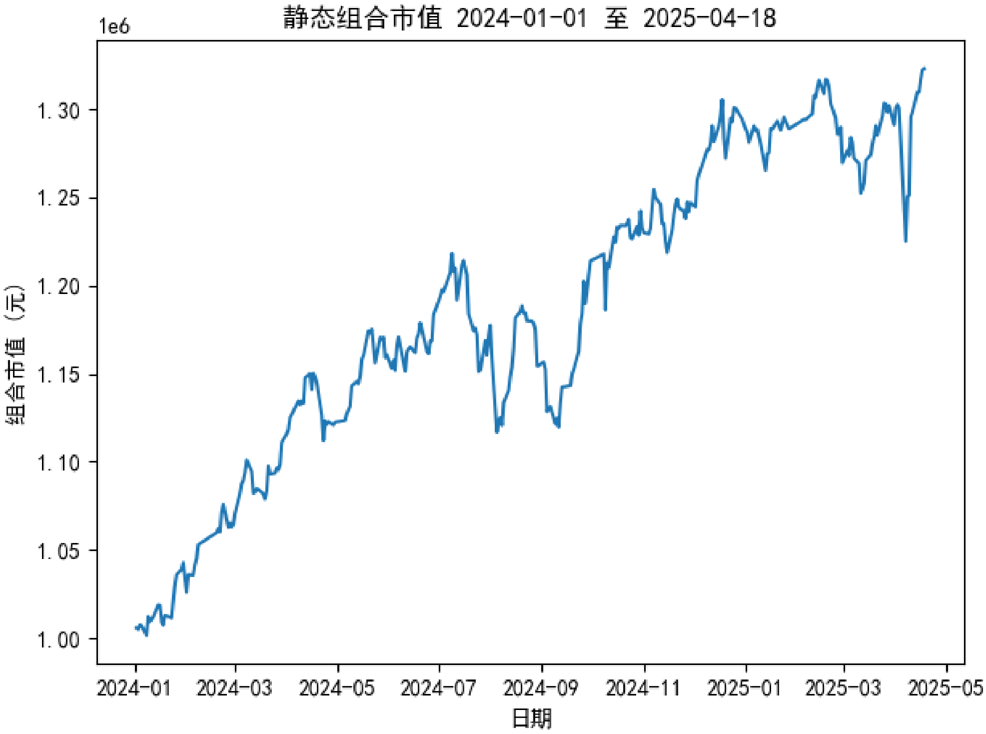

Figure 4: 2024 年 1 月 1 日至 2025 年 4 月 18 日的组合价值

3.3 Kupiec 检验结果分析

根据实验结果，我们还得到了三种方法的 LR 和 p-value 值，具体如下：

- **传统历史模拟法：**

  $$\text{LR} = 0.24, \quad \text{p-value} = 0.6271$$

  由于 p-value 大于 0.05，说明我们无法拒绝零假设，即传统历史模拟法的风险预测与实际情况相符，模型有效。

- **时间加权历史模拟法（ =0.99）：**

  $$\text{LR} = 17.73, \quad \text{p-value} = 0.0000$$

  由于 p-value 小于 0.05，说明我们拒绝零假设，表明时间加权历史模拟法的风险预测存在显著偏差，模型无效。

- **EWMA 波动率加权法（ =0.95）：**

  $$\text{LR} = 34.16, \quad \text{p-value} = 0.0000$$

  由于 p-value 小于 0.05，说明我们拒绝零假设，表明 EWMA 波动率加权法的风险预测也存在显著偏差，模型无效。

根据 Kupiec 非违约率检验的结果，**传统历史模拟法**的 VaR 预测结果符合实际数据，而**时间加权历史模拟法**和 **EWMA 波动率加权法**则表现出显著的偏差，表明这两种方法的风险预测不准确，下面分析原因。

3.4 加权后模型不理想的原因分析

用历史数据计算滚动 30 日波动率，代码见附录 b.py，如图可以观察到，三种资产的波动率存在明显的波动周期，尤其是 2022 年初至 2023 年期间，波动率水平较高。

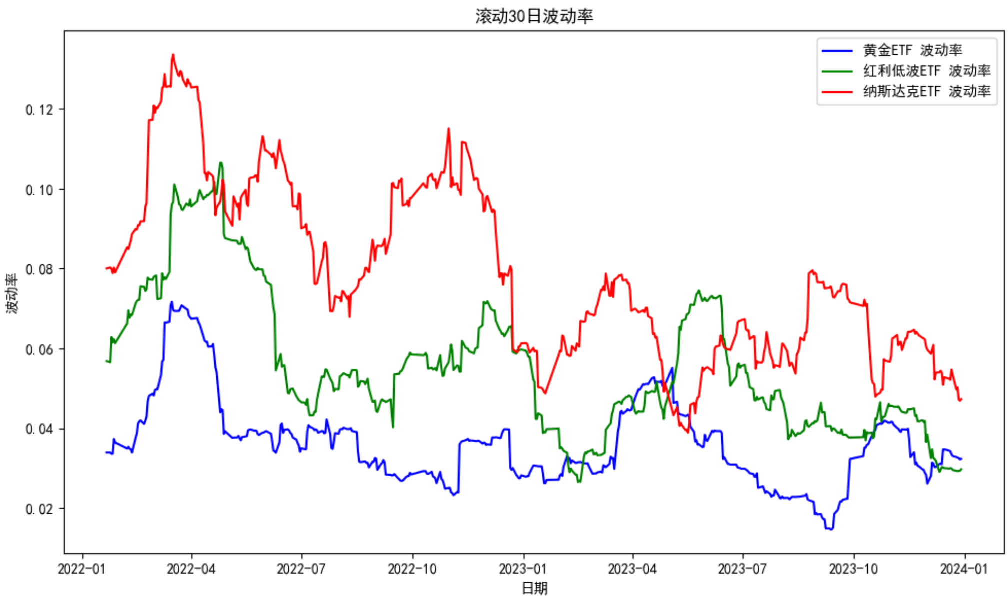

Figure 5: 滚动 30 日波动率

时间加权历史模拟法和 EWMA 波动率加权法采用指数衰减的权重分配方式，具体如下：

- **时间加权历史模拟法**中，每个历史损失的权重按照时间指数衰减，权重为：

$$w_i = \frac{\lambda^{n-i}(1-\lambda)}{1-\lambda^n}$$

  其中，λ 为衰减因子（本模型中取 0.99），n 为样本容量，i 为样本索引。较早的历史数据权重快速减小。

- **EWMA 波动率加权法**中，波动率通过指数加权移动平均计算：

$$\sigma_i^2 = \lambda\sigma_{i-1}^2 + (1-\lambda)r_i^2$$

  其中，λ 为波动率衰减因子（本模型中取 0.95），r_i 为第 i 日组合收益率。较早的高波动率数据对当前波动率估计影响逐渐减弱。

由于上述衰减机制：

1. **前期高波动率事件权重被弱化**，导致模型对早期极端风险记忆不足；
2. 这使得计算得到的 VaR 在极端风险事件发生后可能偏低，低估了风险暴露；
3. Kupiec 非违约率检验反映出时间加权和 EWMA 方法未能准确捕捉风险，导致拒绝假设（p-value 极低）。

综上，时间加权和 EWMA 方法的风险估计结果不佳，主要原因是历史极端风险事件的“遗忘效应”。为了改善风险估计，可以考虑调整 λ 参数，降低衰减速度，或者引入更好的风险建模方法。

4 分析二：网格搜索最小化组合 VaR 的动态调整

4.1 重平衡框架

每隔 5 个交易日（即交易日序列的第 1、6、11 ... 天）进行一次动态权重调整。

4.2 网格搜索优化

在每个重平衡日 d_k，使用该日及之前最近 500 个交易日的收益数据，遍历权重空间：

$$w_1, w_2 \in [-2, 2], \quad w_3 = 1 - w_1 - w_2$$

筛选满足 $w_3 \in [-2, 2]$ 的权重组合（允许做空）。计算对应组合历史收益，使用传统历史模拟法计算组合 VaR，寻找使组合 VaR 最小的权重 $w^* = (w_1^*, w_2^*, w_3^*)$。

4.3 动态组合市值更新

对于重平衡日 d_k，组合在重平衡前的市值计算为：

$$V_{d_k}^- = \sum_{j=1}^{3} \text{shares}_{j,d_k^-} \times P_j(d_k)$$

其中：

- shares_{j,d_k^-} 是第 j 个资产在重平衡日前持有的份额；
- $P_j(d_k)$ 是第 j 个资产在重平衡日的价格；
- $V_{d_k}^-$ 是重平衡日开始时的组合市值。

重平衡后份额重新分配为：

$$\text{shares}_{j,d_k} = \frac{w_{j,k}^* \times V_{d_k}^-}{P_j(d_k)}$$

其中：

- $w_{j,k}^*$ 是在重平衡日 d_k 选定的第 j 个资产的新权重；
- 通过该公式保证重平衡当天组合总市值保持不变。

非重平衡日 t 的组合市值按持仓份额和市价计算：

$$V_t = \sum_{j=1}^{3} \text{shares}_{j,d_k} \times P_j(t), \quad t \in (d_k, d_{k+1})$$

其中：

- shares_{j,d_k} 是最后一次重平衡时的持仓份额；
- $P_j(t)$ 是第 j 个资产在日期 t 的价格；
- V_t 是该日期组合的市值。

该计算方式保证了持仓份额不变，组合价值随市场价格变动动态变化。

4.4 结果分析

动态调整采用每 5 个交易日通过网格搜索使组合 VaR 最小化的权重进行重平衡，调整后的权重如图所示。这里只展示部分结果：

```
2024-01-02 重平衡最小VaR权重：黄金ETF:0.6052, 红利低波ETF:0.1643, 纳斯达克ETF:0.2305
2024-01-09 重平衡最小VaR权重：黄金ETF:0.6052, 红利低波ETF:0.1643, 纳斯达克ETF:0.2305
2024-01-16 重平衡最小VaR权重：黄金ETF:0.6052, 红利低波ETF:0.1643, 纳斯达克ETF:0.2305
2024-01-23 重平衡最小VaR权重：黄金ETF:0.6052, 红利低波ETF:0.1643, 纳斯达克ETF:0.2305
2024-01-30 重平衡最小VaR权重：黄金ETF:0.6052, 红利低波ETF:0.1643, 纳斯达克ETF:0.2305
2024-02-06 重平衡最小VaR权重：黄金ETF:0.6052, 红利低波ETF:0.1643, 纳斯达克ETF:0.2305
2024-02-21 重平衡最小VaR权重：黄金ETF:0.6052, 红利低波ETF:0.1643, 纳斯达克ETF:0.2305
2024-02-28 重平衡最小VaR权重：黄金ETF:0.7094, 红利低波ETF:0.1884, 纳斯达克ETF:0.1022
2024-03-06 重平衡最小VaR权重：黄金ETF:0.7094, 红利低波ETF:0.1884, 纳斯达克ETF:0.1022
2024-03-13 重平衡最小VaR权重：黄金ETF:0.7094, 红利低波ETF:0.1884, 纳斯达克ETF:0.1022
2024-03-20 重平衡最小VaR权重：黄金ETF:0.7094, 红利低波ETF:0.1884, 纳斯达克ETF:0.1022
2024-03-27 重平衡最小VaR权重：黄金ETF:0.7094, 红利低波ETF:0.1884, 纳斯达克ETF:0.1022
2024-04-03 重平衡最小VaR权重：黄金ETF:0.6854, 红利低波ETF:0.1483, 纳斯达克ETF:0.1663
2024-04-12 重平衡最小VaR权重：黄金ETF:0.6613, 红利低波ETF:0.2605, 纳斯达克ETF:0.0782
2024-04-19 重平衡最小VaR权重：黄金ETF:0.6613, 红利低波ETF:0.2605, 纳斯达克ETF:0.0782
2024-04-26 重平衡最小VaR权重：黄金ETF:0.7094, 红利低波ETF:0.1884, 纳斯达克ETF:0.1022
2024-05-08 重平衡最小VaR权重：黄金ETF:0.7094, 红利低波ETF:0.1884, 纳斯达克ETF:0.1022
2024-05-15 重平衡最小VaR权重：黄金ETF:0.6934, 红利低波ETF:0.2124, 纳斯达克ETF:0.0942
2024-05-22 重平衡最小VaR权重：黄金ETF:0.6613, 红利低波ETF:0.2605, 纳斯达克ETF:0.0782
```

Figure 6: 动态组合每 5 交易日的重平衡权重

动态调整和静态固定权重的组合市值对比图如下：

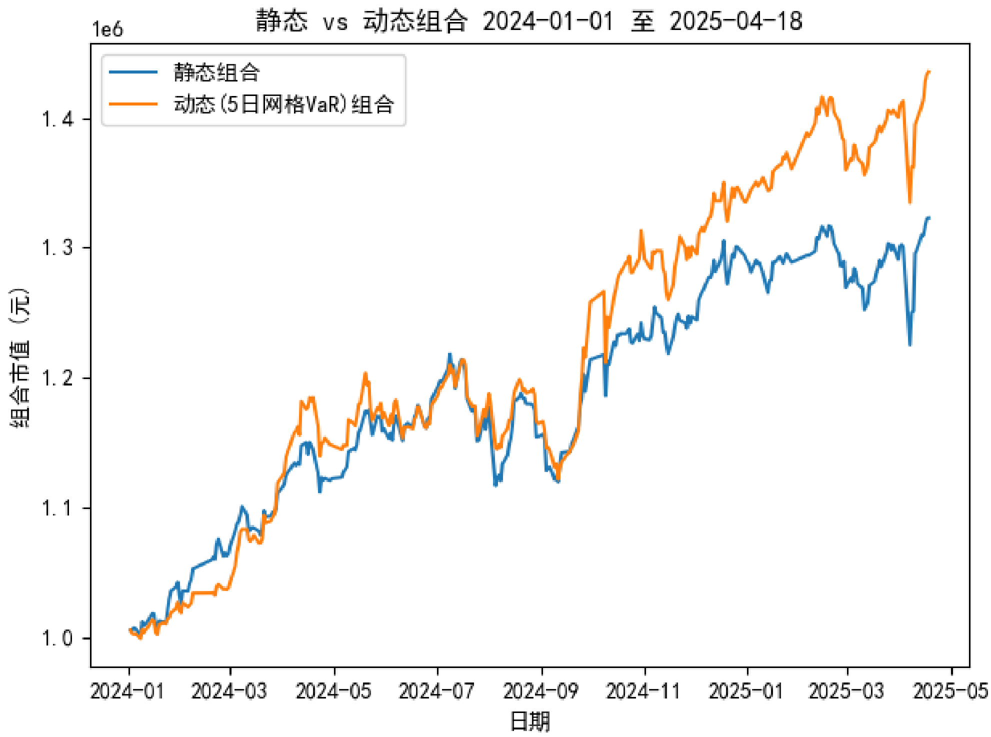

Figure 7: 静态组合与动态 (5 日网格搜索 VaR) 组合市值对比

由图 6 可见，动态组合权重存在明显的调整，这说明网格搜索方法在追踪最小组合 VaR

时，会根据市场波动和历史风险动态调整权重配置，以降低潜在的风险暴露。图 7 展示了动态组合与静态组合的市值变化趋势。整体来看，动态组合的市值波动更小且呈现出更优的增长趋势，最终市值明显高于静态组合。这说明动态调整权重基于 VaR 最小化的策略在风险控制和收益提升方面具有优势。

动态调整的成功原因主要包括：

- **风险动态捕捉：** 网格搜索 VaR 最小化使组合权重根据最新日期的前 500 个交易日的历史数据进行调整，及时反映市场波动变化。
- **权重灵活分配：** 允许权重在 [-2,2] 范围内变化，支持做空操作，提高策略的灵活性。
- **降低风险暴露：** 动态降低高风险资产权重，增加低波动资产权重，有效降低 VaR，防范尾部风险。

综上所述，动态网格搜索 VaR 最小化策略通过灵活的权重调整，能更有效管理风险，实现组合市值的稳健增长，优于传统静态固定配置方案。

A 附录

Listing 1: a.py

```
import math
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chi2
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
# 1. 读取数据
data = pd.read_excel('data.xlsx', parse_dates=['日期'])
data.set_index('日期', inplace=True)
assets = ['黄金ETF', '红利低波ETF', '纳斯达克ETF']
prices = data[assets]
returns_full = prices.pct_change().dropna()
# 2. 初始配置
weights = np.array([0.3, 0.4, 0.3])    # 初始权重
init_capital = 1_000_000               # 初始资金（元）
# 3. VaR 样本窗口：2023-12-29 含当天，往前 500 日
hist_end = '2023-12-29'
hist_returns = returns_full.loc[:hist_end].iloc[-500:]
port_r = hist_returns.dot(weights)    # 组合回报
n = len(port_r)
alpha = 0.01                          # 99% VaR
# 计算当日组合市值 V_n 及份额
price_n = prices.loc[hist_end]
shares = weights * init_capital / price_n
V_n = shares.dot(price_n)

# --- 4.1 传统历史模拟 VaR ---
loss_hist = -V_n * port_r.values
df_hist = pd.DataFrame({
```

```
    'loss': loss_hist,
    'weight': np.ones(n)/n
}).sort_values('loss', ascending=False)
cumw = df_hist['weight'].cumsum()
VaR_hist = df_hist.loc[cumw >= alpha, 'loss'].iloc[0]
df_hist.to_csv('historical_loss_weights.csv', index=False)
print(f"传统历史模拟 VaR(99%): {VaR_hist:.2f} 元")

# --- 4.2 时间加权历史模拟 VaR ( =0.99) ---
lam = 0.99
exps = np.arange(n-1, -1, -1)
w_tw = lam**exps * (1-lam) / (1-lam**n)
w_tw /= w_tw.sum()
df_tw = pd.DataFrame({
    'loss': loss_hist,
    'weight': w_tw
}).sort_values('loss', ascending=False)
cumw2 = df_tw['weight'].cumsum()
VaR_tw = df_tw.loc[cumw2 >= alpha, 'loss'].iloc[0]
df_tw.to_csv('time_weighted_loss_weights.csv', index=False)
print(f"时间加权历史模拟 VaR(99%,  =0.99): {VaR_tw:.2f} 元")

# --- 4.3 EWMA 波动率加权 VaR ( =0.95) ---
lam2 = 0.95
s2 = port_r.var()
sigma = []
for r in port_r:
    s2 = lam2*s2 + (1-lam2)*r*r
    sigma.append(math.sqrt(s2))
sigma = np.array(sigma)
sigma_n1 = sigma[-1]
adj_r = port_r.values * (sigma_n1/sigma)
loss_ewma = -V_n * adj_r
w_ewma = lam2**exps * (1-lam2) / (1-lam2**n)
w_ewma /= w_ewma.sum()
df_ewma = pd.DataFrame({
    'loss': loss_ewma,
    'weight': w_ewma
}).sort_values('loss', ascending=False)
cumw3 = df_ewma['weight'].cumsum()
VaR_ewma = df_ewma.loc[cumw3 >= alpha, 'loss'].iloc[0]
df_ewma.to_csv('ewma_loss_weights.csv', index=False)
print(f"EWMA 波动率加权 VaR(99%,  =0.95): {VaR_ewma:.2f} 元")

# --- 5. 静态组合回测 2024-01-01 至 2025-04-18 ---
start, end = '2024-01-01', '2025-04-18'
prices_period = prices.loc[start:end]
dates = prices_period.index
static_values = prices_period.dot(shares)
pd.DataFrame({
```

```
    'date': dates,
    'portfolio_value': static_values.values
}).to_csv('portfolio_value.csv', index=False)
plt.figure()
plt.plot(dates, static_values, label='静态组合')
plt.xlabel('日期'); plt.ylabel('组合市值␣(元)')
plt.title(f'静态组合市值␣{start}␣至␣{end}')
plt.tight_layout()
plt.savefig('static_portfolio_value.png')
plt.close()

# --- 6. Kupiec 非违约率检验 ---
def kupiec_test(returns, VaR, alpha):
    losses = -V_n * returns
    x = (losses > VaR).sum()
    n = len(losses)
    p = alpha
    L0 = (1-p)**(n-x) * p**x
    L1 = (1-x/n)**(n-x) * (x/n)**x if x>0 and x<n else 1e-10
    lr = -2*(math.log(L0) - math.log(L1))
    return lr, 1-chi2.cdf(lr, df=1)
rets = static_values.pct_change().dropna()
for name, va in [('传统', VaR_hist), ('时间加权', VaR_tw), ('EWMA', VaR_ewm
    lr, pv = kupiec_test(rets, va, alpha)
    print(f"{name}␣Kupiec␣LR:␣{lr:.2f},␣p-value:␣{pv:.4f}")

# --- 7. 分析二: 每 5 交易日网格搜索动态重平衡 VaR 最小化 ---
grid_points = 500   # 搜索精度
weight_range = np.linspace(-2, 2, grid_points)
rebalance_dates = dates[::5]  # 更新间隔为5天
port_dyn = pd.Series(index=dates, dtype=float)
current_shares = shares.copy()
for d in rebalance_dates:
    # 1) 计算旧市值
    V_old = prices.loc[d].dot(current_shares)
    # 2) 网格搜索最小 VaR 权重
    hist = returns_full.loc[:d].iloc[-500:]
    best_var, best_w = np.inf, None
    for w1 in weight_range:
        for w2 in weight_range:
            w3 = 1 - w1 - w2
            if -2 <= w3 <= 2:
                w = np.array([w1, w2, w3])
                var_i = -np.quantile(hist.dot(w), alpha)
                if var_i < best_var:
                    best_var, best_w = var_i, w
    # 3) 打印并重分配份额 (市值不变)
    print(
        f"{d.strftime('%Y-%m-%d')}␣重平衡最小VaR权重: "
        + ",␣".join(f"{asset}:{w:.4f}" for asset, w in zip(assets, best_w))
```

```
    )
    current_shares = best_w * V_old / prices.loc[d]
    port_dyn.loc[d] = V_old
    # 4) 填充下一个重平衡日前的市值
    next_idx = np.where(dates==d)[0][0] + 1
    next_d = rebalance_dates[list(rebalance_dates).index(d)+1] \
             if d != rebalance_dates[-1] else None
    if next_d is not None:
        for t in dates[next_idx:dates.get_loc(next_d)]:
            port_dyn.loc[t] = prices.loc[t].dot(current_shares)
# 最后填充剩余
for t in dates:
    if pd.isna(port_dyn.loc[t]):
        port_dyn.loc[t] = prices.loc[t].dot(current_shares)
# 保存 & 绘图对比
pd.DataFrame({
    'date': dates,
    'dynamic_value': port_dyn.values
}).to_csv('dynamic_portfolio_value.csv', index=False)
plt.figure()
plt.plot(dates, static_values, label='静态组合')
plt.plot(dates, port_dyn,      label='动态(5日网格VaR)组合')
plt.xlabel('日期'); plt.ylabel('组合市值 (元)')
plt.title(f'静态 vs 动态组合 {start} 至 {end}')
plt.legend(); plt.tight_layout()
plt.savefig('static_vs_dynamic_grid.png')
plt.close()
```

Listing 2: b.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.switch_backend('Agg')
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
# 1. 读取数据
data = pd.read_excel('data.xlsx', parse_dates=['日期'])
data.set_index('日期', inplace=True)
assets = ['黄金ETF', '红利低波ETF', '纳斯达克ETF']
prices = data[assets]
returns_full = prices.pct_change().dropna()
# 选择 2023-12-29 之前的 500 个交易日的数据
hist_end = '2023-12-29'
hist_returns = returns_full.loc[:hist_end].iloc[-500:]  # 获取2023-12-29之前
# 2. 波动性分析：计算滚动窗口波动率（30日窗口）
window = 30
volatility = hist_returns.rolling(window).std() * np.sqrt(window)
plt.figure(figsize=(10, 6))
plt.plot(volatility['黄金ETF'], label='黄金ETF 波动率', color='blue')
plt.plot(volatility['红利低波ETF'], label='红利低波ETF 波动率', color='green
```

```
plt.plot(volatility['纳斯达克ETF'], label='纳斯达克ETF␣波动率', color='red'
plt.title('滚动30日波动率')
plt.xlabel('日期')
plt.ylabel('波动率')
plt.legend()
plt.tight_layout()
plt.savefig('滚动30日波动率.png')  # 保存图像
plt.close()
```