《双 ETF 轮动策略的实证研究与分析》

——以中泰证券视角的量化实证分析（实习报告）

2025 年 10 月 25 日

一. 研究背景

近年来，ETF（交易型开放式指数基金）在中国市场快速发展，成为资产配置的重要工具。对券商而言，开发低成本、易执行、客户易理解的策略产品，有助于提升客户黏性和资产留存率。

本研究聚焦“双 ETF 动量轮动”——一种基于动量择优（Momentum Selection）的策略。目标是：在提升长期收益的同时，有效控制回撤，并探索其在智能投顾与公募产品中的应用前景。

二. 方法选择与理由

2.1 动量择优简介

- 逻辑：基于“强者恒强”假设，对过去 N 个月的累计收益进行比较，择优满仓持有。
- 规则：统一在月末评比，次月初调仓；赢家通吃（100%/0%），并计入交易成本。
- 特点：规则化、透明化，能有效捕捉趋势，但在震荡市可能出现频繁切换。

2.2 为什么选动量而不是其他轮动方法？

- 相比估值轮动、波动率轮动、风险平价：

a) 更简单透明，便于客户理解；
b) 更契合股－金组合的趋势特性；
c) 不依赖主观判断或复杂计算，更适合落地智能投顾产品。

三． 标的选择与理由

本研究选取沪深 300ETF（510300）与黄金 ETF（518880）作为轮动标的：

1）信息纯度

-在月频动量下，黄金（gold）与沪深 300（equity）的阶段性趋势更鲜明；

-引入国债 ETF（bond）后，三选一的名次差经常很小，噪音切换变多（动量分差小→更容易“误换仓”），交易成本侵蚀收益。

2）经济直觉与产品可解释性

-股-金天然是“风险开/关”双闸门：牛市抓股指趋势，避险期切到黄金；

-国债的慢趋势特征更适合配置型（固定配比），但在月频动量中常被“短期噪音”触发；

-对客户与渠道，“赢家通吃=100%/0%”更易理解、执行成本更低，利于产品化与 A/B 测试落地。

结论：在该样本与执行频率下，双资产的“趋势纯度+低噪音换仓”>三资产的“名义分散”>单资产，这是实证结果而非主观偏好。

注：研究中亦测试了加入 10 年期国债 ETF（511260） 的三资产策略，但核心分析聚焦“股-金”双资产配置。

四． 策略设计与实现

1. 策略逻辑

- 动量择优 + 赢者通吃。
- 每月评比沪深 300ETF 与黄金 ETF 的动量表现，择强满仓。

2. 操作频率

- 月频（月末评比，次月初执行），避免过高换手率。

3. 关键参数

- 回看期（LB）：在 3/6/9/12 个月不同窗口作出实证对比的测试中，发现 6 个月动量窗口表现最佳，作为最终设定。

4. 核心切换节点

- 当沪深 300 的过去 6 个月累计收益 > 黄金，则下月满仓切换至沪深 300；反之切换至黄金。

五． 回测与效果分析（2015－2024）

5.1 双资产 vs. 单一资产

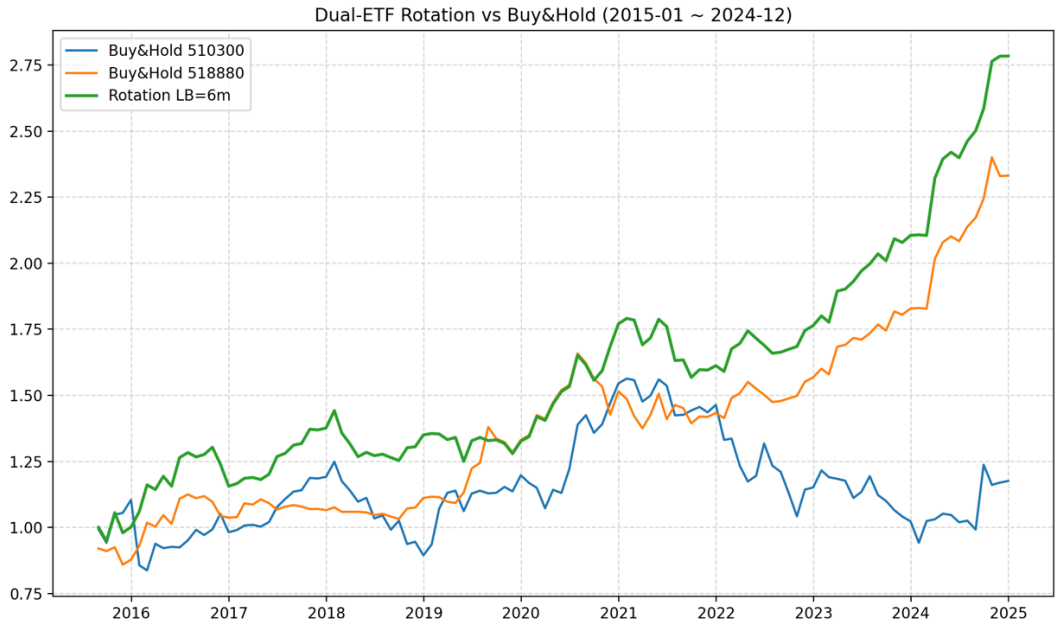
-年化收益：双资产 11.5% > 黄金 8.9% > 沪深 300

-最大回撤：双资产 -13% < 黄金 -17% < 沪深 300

-Sharpe：双资产 0.79 > 黄金 0.60 > 沪深 300

结论：风险收益比明显更优，验证了动量有效性。

*虽然有来回切换成本，但在大趋势阶段能明显减小回撤。



5.2 双资产轮动 vs. 三资产轮动

-双资产 Sharpe 0.793 > 三资产 Sharpe 0.738

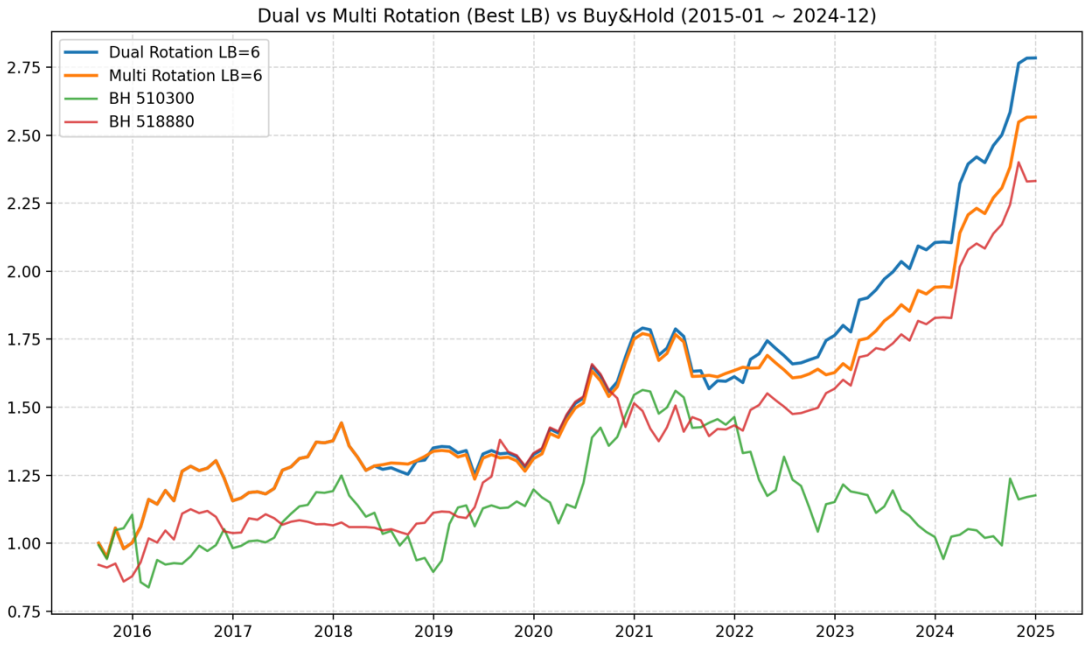
-年化收益 11.49% > 10.53%

结论：

1）动量轮动适合“趋势感强的资产”，不是越多越好。国债虽稳健（Sharpe≈0.80），但收益有限（≈4–5%）以及在动量排名中容易“误触发”， 常常在股和金差不多时被“误选”。由于国债缺乏趋势性，所以换过去收益有限，反而拖累整体表现（Sharpe 降到 0.738）。因此，国债适合长期配置，不适合动量轮动；
2）双资产动量的优势在于其 Sharpe≈0.79，虽然略低于国债，但年化收益≈11.5%，远超国债。并且回撤比股市小一半以上，收益又比国债高一倍以上。

- 单买股 = 高收益+高风险
- 单买金 = 防御性强+中等收益
- 单买债 = 稳定+低收益

- 动量双资产 = 收益和风险的最佳平衡（验证了动量择优在趋势性资产中的有效性与边界条件。）



六． 不同市场阶段的表现

市场阶段	时间区间	动量轮动表现	机制解释

上涨趋势阶段	2015Q1－Q2、2019Q1、2020Q2	动量信号快速切入沪深300，明显捕捉上涨行情，跑赢单一持仓	动量信号捕捉上涨趋势
下跌趋势阶段	2015Q3－2016Q1、2018 全年	策略切换至黄金，显著降低组合回撤	黄金避险属性发挥作用
震荡阶段	2017 全年、2021H2－2022H1	策略频繁切换，整体收益持平或略低	信号在无趋势时失真，成本侵蚀收益

结论：策略不预测市场，而是自然顺应市场阶段，自动通过“过去 n 个月谁涨得更多”来决定持仓。

换句话说，动量轮动就是把“判断市场阶段”的任务交给价格本身，它自动帮我们完成“牛市追股，熊市避险”的切换，以及震荡承认边界。

七． 中泰证券业务落地分析

1.历史验证与未来可复制性：十年跨周期验证有效；高不确定性环境下复制性强。

2.落地基础：月频低成本、ETF 流动性好、逻辑简单透明，适合智能投顾与中小客户。

3.业务机会：

-智能投顾产品：APP 上线信号、一键调仓；

-客户教育：用低门槛、跨周期适应吸引投资者；

-差异化竞争：区别于固定配比，结合中泰研究优势形成品牌壁垒。

八． 核心结论与建议

- **比单资产优越：**收益更高，回撤更低。

- **比固定配比优越**：风险相近但回报更高。
- **跨周期适应**：牛市抓股、熊市避险、震荡承认边界。
- **适用人群**：中等偏保守、希望收益高于债券/存款、又不想高频交易的投资者。
- **可持续性**：十年多轮验证，基础逻辑清晰，执行可行。

*可能涉及到的相关提问：

Q1：为什么要回测十年？

A：覆盖了股灾、熊市、疫情、加息等全周期，证明策略不是短期凑巧，而是长期有效。

Q2：为什么不用三资产？

A：国债本身稳健，但趋势弱，加入后增加噪音，反而削弱动量轮动效果。

Q3：为什么不是直接买国债？

A：国债稳但收益低（≈4－5%）；本策略收益率≈11.5%，风险仍可控，长期更优。

Q4：是不是频繁调仓？

A：不是，每月最多一次调仓；大趋势阶段往往连续持有，平均换仓次数有限。

Q5：真正的优势是什么？

A：极简规则（每月择强），实现比任何单一资产更优的风险收益比，既能讲清楚，也能产品化。

附录 A：策略回测代码及说明

- **A.1：双 ETF 轮动 vs 单资产；**
- **A.2：双 ETF 轮动 vs 多资产轮动。**

A.1 双 ETF 轮动 vs 单资产回测代码（dual_rotation_10y_gold_ak.py）

```
import os

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import akshare as ak

# ---------------- 参数 ----------------
TICKS = {
    "510300.SH": "510300",  # 沪深 300 ETF
    "518880.SH": "518880",  # 黄金 ETF
}
START = "20150101"
END   = "20241231"  # 固定到年末，月频刚好覆盖 2015-01 ~ 2024-12

LOOKBACKS = [3, 6, 9, 12]

LEVER    = 1.0      # 杠杆倍数
FIN_ANN  = 0.045    # 年化融资利率（LEVER>1 时，对超出 1 的部分计提）
COMM_BPS = 1.0      # 手续费（万分比）
SLIP_BPS = 2.0      # 滑点（万分比）
RF_ANN   = 0.02     # 年化无风险利率（Sharpe 用）

OUTDIR = "outputs_dual_etf"
os.makedirs(OUTDIR, exist_ok=True)

# ---------------- 工具函数 ----------------
def get_close(symbol: str, start: str, end: str) -> pd.Series | None:
    """用 akshare 获取日频收盘价，返回 Series(index=Date, name=symbol)"""
    code = TICKS[symbol]
    df = ak.fund_etf_hist_em(
        symbol=code,
        period="daily",
        start_date=start,
        end_date=end,
        adjust=""
    )
    if df is None or df.empty:
```

```
        return None
    df = df.rename(columns={"日期": "Date", "收盘": "Close"})
    df["Date"] = pd.to_datetime(df["Date"])
    df = df.sort_values("Date").set_index("Date")
    s = df["Close"].astype(float)
    s.name = symbol
    return s

def metr(r: pd.Series, rf: float = 0.02) -> dict:
    """年化收益、波动、夏普、最大回撤、胜率、样本月数"""
    r = r.dropna()
    n = len(r)
    if n == 0:
        return {"CAGR": np.nan, "Vol": np.nan, "Sharpe": np.nan,
                "MDD": np.nan, "WinRate": np.nan, "Months": 0}
    cagr  = (1 + r).prod() ** (12 / n) - 1
    vol   = r.std() * np.sqrt(12) if n > 1 else np.nan
    rf_m  = rf / 12.0
    shar  = ((r.mean() - rf_m) / r.std() * np.sqrt(12)) if r.std() > 0 else np.nan
    curve = (1 + r).cumprod()
    mdd   = (curve / curve.cummax() - 1.0).min()
    win   = (r > 0).mean()
    return {"CAGR": cagr, "Vol": vol, "Sharpe": shar,
            "MDD": mdd, "WinRate": win, "Months": n}

# ---------------- 获取/整理数据 ----------------
series = []
for sym in TICKS.keys():
    s = get_close(sym, START, END)
    if s is None or s.empty:
        raise RuntimeError(f"no data: {sym}")
    series.append(s)

prices = pd.concat(series, axis=1).dropna(how="all")
prices = prices[~prices.index.duplicated(keep="last")].sort_index()

# 月末频率（ME = Month End）
m_px  = prices.resample("ME").last().dropna(how="all")  # 月末价格
m_ret = m_px.pct_change().dropna(how="all")             # 月度收益

# ---------------- 买入持有基准 ----------------
bh_curves, bh_m = {}, {}
for sym in TICKS.keys():
    r = m_ret[sym].copy()
```

```
    bh_curves[sym] = (1 + r).cumprod()
    bh_m[sym]      = metr(r, RF_ANN)

# 用于保存每个 Lookback 的月度收益序列（用于后续画“最佳轮动”曲线）
rotation_rets = {}

# ---------------- 轮动策略（赢家通吃） ----------------
all_metrics = []

for lb in LOOKBACKS:
    # 动量 = 过去 lb 个月累计涨幅；用“上月末”的动量决定“本月”持仓（再 shift 一月执行）
    mom = (m_px / m_px.shift(lb)) - 1.0
    mom = mom.shift(1).dropna(how="all")
    ret = m_ret.loc[mom.index]  # 对齐收益

    # 赢家=1，其余=0；下月初执行
    w = pd.DataFrame(0.0, index=mom.index, columns=mom.columns)
    win_asset = mom.idxmax(axis=1)
    for d, sym in win_asset.items():
        w.loc[d, sym] = 1.0
    w = w.shift(1).fillna(0.0).loc[ret.index]

    # 换仓检测
    prev   = w.shift(1).fillna(0.0)
    switch = pd.Series((w.values != prev.values).any(axis=1), index=w.index)

    # 成本设置：首次进场单边，之后换仓双边
    one_side  = (COMM_BPS + SLIP_BPS) / 10000.0
    roundtrip = one_side * 2.0
    tc = pd.Series(0.0, index=w.index)
    active_mask = (w.sum(axis=1) > 0)
    if active_mask.any():
        first_active = active_mask.idxmax()  # 第一个 True 的时间戳
        tc.loc[first_active] = -one_side
    tc.loc[switch] += -roundtrip  # 若与首月重叠，此处会被下面单边覆盖
    # 确保首次进场最终是单边（覆盖可能的叠加）
    if active_mask.any():
        tc.loc[first_active] = -one_side

    # 杠杆 & 融资
    gross = (w * ret).sum(axis=1)
    fin = (FIN_ANN / 12.0) if LEVER > 1.0 else 0.0
```

```
    rot = LEVER * gross - max(0.0, LEVER - 1.0) * fin + tc

    # 存收益用于后续画“最佳轮动”曲线
    rotation_rets[lb] = rot.copy()

    # 记录指标
    rot_m = metr(rot, RF_ANN); rot_m["LookbackM"] = lb
    all_metrics.append(rot_m)

    # 可选：逐LB净值/回撤图（如不需要可注释）
    curve = (1 + rot).cumprod()
    plt.figure(figsize=(10, 6))
    plt.plot(curve, label=f"Rotation LB={lb}m")
    for sym in TICKS.keys():
        plt.plot(bh_curves[sym].loc[curve.index], label=f"BH {sym}")
    plt.title(f"Dual-ETF Rotation (LB={lb}m): 510300.SH vs 518880.SH")
    plt.grid(True, ls="--", alpha=0.5); plt.legend(); plt.tight_layout()
    plt.savefig(os.path.join(OUTDIR, f"nav_curve_gold_lb{lb}.png"), dpi=200);
plt.close()

    peak = curve.cummax(); dd = curve / peak - 1.0
    plt.figure(figsize=(10, 4)); plt.plot(dd, label=f"DD LB={lb}m")
    plt.title(f"Drawdown Rotation (LB={lb}m) Gold Pair")
    plt.grid(True, ls="--", alpha=0.5); plt.tight_layout()
    plt.savefig(os.path.join(OUTDIR, f"drawdown_gold_lb{lb}.png"), dpi=200);
plt.close()

# ---------------- 汇总输出 ----------------
met_df = pd.DataFrame(all_metrics).set_index("LookbackM").sort_index()
bh_df  = pd.DataFrame({f"BH_{sym}": bh_m[sym] for sym in TICKS.keys()})

met_df.to_csv(os.path.join(OUTDIR, "metrics_rotation_10y_gold.csv"), encoding="utf-
8-sig")
bh_df.to_csv(os.path.join(OUTDIR, "metrics_buyhold_10y_gold.csv"), encoding="utf-8-
sig")

# Sharpe vs Lookback（直接用内存）
plt.figure(figsize=(10, 6))
for lb in LOOKBACKS:
    plt.scatter(lb, met_df.loc[lb, "Sharpe"], s=80)
plt.xticks(LOOKBACKS)
plt.title("Sharpe vs Lookback (10Y) Gold Pair")
plt.grid(True, ls="--", alpha=0.5)
plt.tight_layout()
```

```
plt.savefig(os.path.join(OUTDIR, "sharpe_vs_lookback_10y_gold.png"), dpi=200)
plt.close()

# ---------------- 三线净值对比图 + 最佳轮动回撤图 ----------------
# 选“最佳轮动”：优先用 LB=6（预设参数），否则取 Sharpe 最高的
if 6 in LOOKBACKS:
    lb_best = 6
else:
    lb_best = met_df["Sharpe"].idxmax()

bh_300  = bh_curves["510300.SH"]
bh_gold = bh_curves["518880.SH"]
rot_best_curve = (1 + rotation_rets[lb_best]).cumprod()

# 对齐索引并画三线净值图
common_idx =
bh_300.index.intersection(bh_gold.index).intersection(rot_best_curve.index)
plt.figure(figsize=(10, 6))
plt.plot(bh_300.loc[common_idx],  label="Buy&Hold 510300")
plt.plot(bh_gold.loc[common_idx], label="Buy&Hold 518880")
plt.plot(rot_best_curve.loc[common_idx], label=f"Rotation LB={lb_best}m",
linewidth=2)
plt.title("Dual-ETF Rotation vs Buy&Hold (2015-01 ~ 2024-12)")
plt.grid(True, ls="--", alpha=0.5)
plt.legend()
plt.tight_layout()
plt.savefig(os.path.join(OUTDIR, "nav_curve_BH_vs_rotation_best.png"), dpi=200)
plt.close()

# 最佳轮动回撤图
peak = rot_best_curve.cummax()
dd_best = rot_best_curve / peak - 1.0
plt.figure(figsize=(10, 4))
plt.plot(dd_best, label=f"Drawdown Rotation LB={lb_best}m")
plt.title(f"Drawdown (Rotation LB={lb_best}m)")
plt.grid(True, ls="--", alpha=0.5)
plt.tight_layout()
plt.savefig(os.path.join(OUTDIR, f"drawdown_rotation_best_lb{lb_best}.png"),
dpi=200)
plt.close()

# ---------------- 打印检查 ----------------
print("m_px range (月末价格):", m_px.index.min().date(), "~",
m_px.index.max().date())
```

```
print("m_ret range (月度收益):", m_ret.index.min().date(), "~",
m_ret.index.max().date())
print("months (月度收益样本数):", len(m_ret))
print("\nBuy & Hold:\n", bh_df)
print("\nRotation (fixed 10Y):\n", met_df)
print("最佳LB:", lb_best)
print("输出目录:", os.path.abspath(OUTDIR))
```

```
回测结果：m_px range (月末价格): 2015-01-31 ~ 2024-12-31
m_ret range (月度收益): 2015-02-28 ~ 2024-12-31
months (月度收益样本数): 119

Buy & Hold:
          BH_510300.SH  BH_518880.SH
CAGR          0.016455      0.089136
Vol           0.216286      0.121400
Sharpe        0.089422      0.600233
MDD          -0.405562     -0.171015
WinRate       0.554622      0.537815
Months      119.000000    119.000000

Rotation (fixed 10Y):
                CAGR       Vol    Sharpe       MDD   WinRate  Months
LookbackM
3          -0.008017  0.157075 -0.096942 -0.499105  0.543103     116
6           0.114886  0.121619  0.793305 -0.133420  0.646018     113
9           0.069536  0.141505  0.408027 -0.224457  0.600000     110
12          0.093212  0.116105  0.654894 -0.179115  0.598131     107
最佳LB: 6
```

A.2 双ETF轮动 vs 多资产轮动（compare_dual_vs_multi_ak.py）

```
# -*- coding: utf-8 -*-
"""
Dual vs Multi ETF Rotation (AKShare only) 2015-01-01 ~ 2024-12-31
- 双ETF：沪深300(510300.SH) + 黄金(518880.SH)
- 多ETF：沪深300 + 黄金 + 国债ETF(511260.SH)
- 月度动量赢家通吃（回看期 L ∈ {3,6,9,12}；上月末动量、下月执行）
- 交易成本：首月单边、换仓双边（手续费+滑点）
"""
```

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import akshare as ak
from typing import Optional, Dict, List

# ==================== 参数 ====================
TICKS = {
    "510300.SH": "510300",   # 沪深 300 ETF
    "518880.SH": "518880",   # 黄金 ETF
    "511260.SH": "511260",   # 国债 ETF
}

PAIR  = ["510300.SH", "518880.SH"]
MULTI = ["510300.SH", "518880.SH", "511260.SH"]

START = "20150101"
END   = "20241231"   # 覆盖 2015-01 ~ 2024-12

LOOKBACKS = [3, 6, 9, 12]

LEVER    = 1.0       # 杠杆倍数
FIN_ANN  = 0.045     # 年化融资利率（LEVER>1 时，对超出 1 的部分计提）
COMM_BPS = 1.0       # 手续费（万分比，单边）
SLIP_BPS = 2.0       # 滑点（万分比，单边）
RF_ANN   = 0.02      # 年化无风险利率（Sharpe 用）

OUTDIR = "outputs_dual_vs_multi_ak"
os.makedirs(OUTDIR, exist_ok=True)

# ==================== 工具函数 ====================
def get_close(symbol: str, start: str, end: str) -> Optional[pd.Series]:
    """
    仅用 akshare 获取日频收盘价，返回 Series(index=Date, name=symbol)
    """
    code = TICKS[symbol]
    df = ak.fund_etf_hist_em(
        symbol=code, period="daily", start_date=start, end_date=end, adjust=""
    )
    if df is None or df.empty:
        return None
```

```
    # 兼容列名
    rename_map = {}
    if "日期" in df.columns:   rename_map["日期"] = "Date"
    if "收盘" in df.columns:   rename_map["收盘"] = "Close"
    if "收盘价" in df.columns: rename_map["收盘价"] = "Close"
    df = df.rename(columns=rename_map)

    if "Date" not in df.columns or "Close" not in df.columns:
        return None

    df["Date"] = pd.to_datetime(df["Date"])
    df = df.sort_values("Date").set_index("Date")
    s = pd.to_numeric(df["Close"], errors="coerce").replace([np.inf, -np.inf],
np.nan).dropna()
    s.name = symbol
    return s

def metr(r: pd.Series, rf: float = 0.02) -> dict:
    """年化收益、波动、夏普、最大回撤、胜率、样本月数"""
    r = r.dropna()
    n = len(r)
    if n == 0:
        return {"CAGR": np.nan, "Vol": np.nan, "Sharpe": np.nan,
                "MDD": np.nan, "WinRate": np.nan, "Months": 0}
    cagr  = (1 + r).prod() ** (12 / n) - 1
    vol   = r.std() * np.sqrt(12) if n > 1 else np.nan
    rf_m  = rf / 12.0
    shar  = ((r.mean() - rf_m) / r.std() * np.sqrt(12)) if r.std() > 0 else np.nan
    curve = (1 + r).cumprod()
    mdd   = (curve / curve.cummax() - 1.0).min()
    win   = (r > 0).mean()
    return {"CAGR": cagr, "Vol": vol, "Sharpe": shar,
            "MDD": mdd, "WinRate": win, "Months": n}

def ensure_mon_end(prices: pd.DataFrame) -> pd.DataFrame:
    """
    转月末频率（ME = Month End）
    """
    prices = prices[~prices.index.duplicated(keep="last")].sort_index()
    m_px = prices.resample("ME").last().dropna(how="all")
    return m_px

def winner_take_all_rotation(
    m_px: pd.DataFrame,
```

```
    m_ret: pd.DataFrame,
    universe: List[str],
    lb: int,
    lever: float = 1.0,
    fin_ann: float = 0.045,
    comm_bps: float = 1.0,
    slip_bps: float = 2.0,
) -> pd.Series:
    """
    赢家通吃轮动：动量 = 过去 lb 个月累计涨幅（上月末动量，下月执行）
    返回：月度收益序列（含交易成本与融资成本）
    """
    px = m_px[universe].copy()
    ret = m_ret[universe].copy()

    # 上月末动量（过去 lb 个月累计）
    mom = (px / px.shift(lb)) - 1.0
    mom = mom.shift(1).dropna(how="all")
    ret = ret.loc[mom.index]  # 对齐到动量可用期

    # 当月仓位 = 上月末动量赢家；下月初执行（再 shift(1)）
    w = pd.DataFrame(0.0, index=mom.index, columns=universe)
    win_asset = mom.idxmax(axis=1)
    for d, sym in win_asset.items():
        if sym in w.columns:
            w.loc[d, sym] = 1.0
    w = w.shift(1).fillna(0.0).loc[ret.index]

    # 换仓与成本
    prev = w.shift(1).fillna(0.0)
    switch = pd.Series((w.values != prev.values).any(axis=1), index=w.index)

    one_side  = (comm_bps + slip_bps) / 10000.0
    roundtrip = one_side * 2.0
    tc = pd.Series(0.0, index=w.index)

    active_mask = (w.sum(axis=1) > 0)
    if active_mask.any():
        first_active = active_mask.idxmax()
        tc.loc[first_active] = -one_side
    tc.loc[switch] += -roundtrip
    if active_mask.any():
        tc.loc[first_active] = -one_side  # 覆盖可能叠加
```

```
    # 杠杆与融资（仅对超出 1 倍部分计息）
    gross = (w * ret).sum(axis=1)
    fin   = (fin_ann / 12.0) if lever > 1.0 else 0.0
    rot   = lever * gross - max(0.0, lever - 1.0) * fin + tc
    return rot

def pick_best_lb(met_df: pd.DataFrame, prefer_lb: int = 6) -> int:
    if prefer_lb in met_df.index:
        return prefer_lb
    return int(met_df["Sharpe"].idxmax())

def pick_metric_row(met_df: pd.DataFrame, prefer_lb: int = 6) -> pd.Series:
    if prefer_lb in met_df.index:
        return met_df.loc[prefer_lb]
    return met_df.loc[met_df["Sharpe"].idxmax()]

# ==================== 获取/整理数据 ====================
series = []
for sym in TICKS.keys():
    s = get_close(sym, START, END)
    if s is None or s.empty:
        raise RuntimeError(f"no data: {sym}")
    series.append(s)

prices = pd.concat(series, axis=1).dropna(how="all")

# 关键：先做月末采样并裁剪目标区间，再计算月度收益，避免 KeyError
m_px_all = ensure_mon_end(prices)
m_px = m_px_all.loc[(m_px_all.index >= pd.Timestamp("2015-01-31")) &
                    (m_px_all.index <= pd.Timestamp("2024-12-31"))]
m_ret = m_px.pct_change().dropna(how="all")

# ==================== 买入持有（BH）基准 ====================
bh_curves: Dict[str, pd.Series] = {}
bh_m: Dict[str, dict] = {}
for sym in TICKS.keys():
    if sym in m_ret.columns:
        r = m_ret[sym].copy()
        bh_curves[sym] = (1 + r).cumprod()
        bh_m[sym] = metr(r, RF_ANN)

# ==================== 轮动：双 ETF & 多 ETF ====================
rotation_dual: Dict[int, pd.Series] = {}
rotation_multi: Dict[int, pd.Series] = {}
```

```
metrics_dual, metrics_multi = [], []

for lb in LOOKBACKS:
    # 双 ETF
    rot_d = winner_take_all_rotation(
        m_px, m_ret, PAIR, lb,
        lever=LEVER, fin_ann=FIN_ANN,
        comm_bps=COMM_BPS, slip_bps=SLIP_BPS
    )
    rotation_dual[lb] = rot_d.copy()
    m_d = metr(rot_d, RF_ANN); m_d["LookbackM"] = lb; m_d["Group"] =
"Dual(300+Gold)"
    metrics_dual.append(m_d)

    # 多 ETF
    rot_m = winner_take_all_rotation(
        m_px, m_ret, MULTI, lb,
        lever=LEVER, fin_ann=FIN_ANN,
        comm_bps=COMM_BPS, slip_bps=SLIP_BPS
    )
    rotation_multi[lb] = rot_m.copy()
    m_m = metr(rot_m, RF_ANN); m_m["LookbackM"] = lb; m_m["Group"] =
"Multi(300+Gold+Tbond)"
    metrics_multi.append(m_m)

    # —— 每个 LB 的净值/回撤图 ——
    for group_name, curve in [
        (f"Dual LB={lb}", (1 + rot_d).cumprod()),
        (f"Multi LB={lb}", (1 + rot_m).cumprod())
    ]:
        plt.figure(figsize=(10, 6))
        plt.plot(curve, label=group_name, linewidth=2)
        # 参考：两只资产的买入持有
        for sym in PAIR:
            if sym in bh_curves:
                plt.plot(bh_curves[sym].loc[curve.index], label=f"BH {sym}",
alpha=0.7)
        plt.title(f"{group_name} Rotation vs BH (2015-01 ~ 2024-12)")
        plt.grid(True, ls="--", alpha=0.5)
        plt.legend()
        plt.tight_layout()
        fname = group_name.replace(" ", "_").replace(":", "").replace("=", "")
        plt.savefig(os.path.join(OUTDIR, f"nav_{fname}.png"), dpi=200)
        plt.close()
```

```
        peak = curve.cummax()
        dd = curve / peak - 1.0
        plt.figure(figsize=(10, 4))
        plt.plot(dd, label=f"DD {group_name}")
        plt.title(f"Drawdown {group_name}")
        plt.grid(True, ls="--", alpha=0.5)
        plt.tight_layout()
        plt.savefig(os.path.join(OUTDIR, f"dd_{fname}.png"), dpi=200)
        plt.close()

# ==================== 汇总输出 ====================
met_dual_df  = pd.DataFrame(metrics_dual).set_index("LookbackM").sort_index()
met_multi_df = pd.DataFrame(metrics_multi).set_index("LookbackM").sort_index()
bh_df        = pd.DataFrame({f"BH_{sym}": bh_m[sym] for sym in bh_m.keys()})

met_dual_df.to_csv(os.path.join(OUTDIR, "metrics_rotation_dual_10y.csv"),
encoding="utf-8-sig")
met_multi_df.to_csv(os.path.join(OUTDIR, "metrics_rotation_multi_10y.csv"),
encoding="utf-8-sig")
bh_df.to_csv(os.path.join(OUTDIR, "metrics_buyhold_10y.csv"), encoding="utf-8-sig")

# —— Sharpe vs Lookback（双 & 多）——
plt.figure(figsize=(10, 6))
for lb in LOOKBACKS:
    if lb in met_dual_df.index:
        plt.scatter(lb, met_dual_df.loc[lb, "Sharpe"], s=90)             # Dual: 点
    if lb in met_multi_df.index:
        plt.scatter(lb, met_multi_df.loc[lb, "Sharpe"], s=90, marker="x")# Multi:
叉
plt.xticks(LOOKBACKS)
plt.title("Sharpe vs Lookback (Dual vs Multi, 2015-2024)")
plt.grid(True, ls="--", alpha=0.5)
plt.tight_layout()
plt.savefig(os.path.join(OUTDIR, "sharpe_vs_lookback_dual_vs_multi.png"), dpi=200)
plt.close()

# ==================== 选最佳LB并三线图对比 ====================
lb_best_dual  = pick_best_lb(met_dual_df, 6)
lb_best_multi = pick_best_lb(met_multi_df, 6)

curve_dual  = (1 + rotation_dual[lb_best_dual]).cumprod()
curve_multi = (1 + rotation_multi[lb_best_multi]).cumprod()
```

```
common_idx = curve_dual.index.intersection(curve_multi.index)
for sym in PAIR:
    if sym in bh_curves:
        common_idx = common_idx.intersection(bh_curves[sym].index)

plt.figure(figsize=(10, 6))
plt.plot(curve_dual.loc[common_idx],  label=f"Dual Rotation LB={lb_best_dual}",
linewidth=2)
plt.plot(curve_multi.loc[common_idx], label=f"Multi Rotation LB={lb_best_multi}",
linewidth=2)
if "510300.SH" in bh_curves:
    plt.plot(bh_curves["510300.SH"].loc[common_idx], label="BH 510300", alpha=0.8)
if "518880.SH" in bh_curves:
    plt.plot(bh_curves["518880.SH"].loc[common_idx], label="BH 518880", alpha=0.8)
plt.title("Dual vs Multi Rotation (Best LB) vs Buy&Hold (2015-01 ~ 2024-12)")
plt.grid(True, ls="--", alpha=0.5)
plt.legend()
plt.tight_layout()
plt.savefig(os.path.join(OUTDIR, "nav_dual_vs_multi_best.png"), dpi=200)
plt.close()

# —— 回撤：双/多最佳 ——
for label, curve in [
    (f"Dual LB={lb_best_dual}", curve_dual),
    (f"Multi LB={lb_best_multi}", curve_multi),
]:
    peak = curve.cummax()
    dd = curve / peak - 1.0
    plt.figure(figsize=(10, 4))
    plt.plot(dd, label=f"DD {label}")
    plt.title(f"Drawdown {label}")
    plt.grid(True, ls="--", alpha=0.5)
    plt.tight_layout()
    tag = label.replace(" ", "_")
    plt.savefig(os.path.join(OUTDIR, f"dd_{tag}.png"), dpi=200)
    plt.close()

# ==================== 文本对比（报告用） ====================
def _fmt(x):
    return "NA" if pd.isna(x) else f"{x*100:,.2f}%"

def pick_metric_row_print(met_df: pd.DataFrame, prefer_lb: int = 6) -> pd.Series:
    return met_df.loc[prefer_lb] if prefer_lb in met_df.index else
met_df.loc[met_df["Sharpe"].idxmax()]
```

```
best_dual_row  = pick_metric_row_print(met_dual_df, 6)
best_multi_row = pick_metric_row_print(met_multi_df, 6)

print("====== 数据区间 ======")
print("月末价格范围:", m_px.index.min().date(), "~", m_px.index.max().date())
print("月度收益样本数:", len(m_ret))

print("\n====== 买入持有（BH）指标 ======")
print(pd.DataFrame({f"BH_{k}": v for k, v in bh_m.items()}))

print("\n====== 轮动（Dual：沪深300+黄金）各LB ======")
print(met_dual_df)

print("\n====== 轮动（Multi：沪深300+黄金+国债）各LB ======")
print(met_multi_df)

print("\n====== 最佳方案对比（优先LB=6，否则Sharpe最高） ======")
print(f"Dual  : LB={lb_best_dual}  Sharpe={best_dual_row['Sharpe']:.3f}
CAGR={_fmt(best_dual_row['CAGR'])}")
print(f"Multi : LB={lb_best_multi} Sharpe={best_multi_row['Sharpe']:.3f}
CAGR={_fmt(best_multi_row['CAGR'])}")

better_by_sharpe = "Dual" if (best_dual_row["Sharpe"] > best_multi_row["Sharpe"])
else "Multi"
better_by_cagr   = "Dual" if (best_dual_row["CAGR"]  > best_multi_row["CAGR"])
else "Multi"

print("\n—— 汇总判断 ——")
print(f"按夏普(Sharpe)：{better_by_sharpe} 更优")
print(f"按年化收益(CAGR)：{better_by_cagr} 更优")

print("\n输出目录:", os.path.abspath(OUTDIR))
```

回测结果：

```
====== 数据区间 ======
月末价格范围: 2015-01-31 ~ 2024-12-31
月度收益样本数: 119

====== 买入持有（BH）指标 ======
        BH_510300.SH  BH_518880.SH  BH_511260.SH
CAGR        0.016455      0.089136      0.042357
Vol         0.216286      0.121400      0.027370
Sharpe      0.089422      0.600233      0.801099
```

```
MDD          -0.405562      -0.171015      -0.042087
WinRate       0.554622       0.537815       0.727273
Months      119.000000     119.000000      88.000000

====== 轮动（Dual：沪深300+黄金）各LB ======
                CAGR       Vol     Sharpe  ...    WinRate  Months           Group
LookbackM                                  ...
3          -0.008017  0.157075 -0.096942  ...   0.543103     116  Dual(300+Gold)
6           0.114886  0.121619  0.793305  ...   0.646018     113  Dual(300+Gold)
9           0.069536  0.141505  0.408027  ...   0.600000     110  Dual(300+Gold)
12          0.093212  0.116105  0.654894  ...   0.598131     107  Dual(300+Gold)

[4 rows x 7 columns]

====== 轮动（Multi：沪深300+黄金+国债）各LB ======
                CAGR       Vol  ...  Months                  Group
LookbackM                       ...
3          -0.008185  0.147525  ...     116  Multi(300+Gold+Tbond)
6           0.105305  0.118394  ...     113  Multi(300+Gold+Tbond)
9           0.056100  0.139287  ...     110  Multi(300+Gold+Tbond)
12          0.082500  0.114478  ...     107  Multi(300+Gold+Tbond)

[4 rows x 7 columns]

====== 最佳方案对比（优先LB=6，否则Sharpe最高） ======
Dual  : LB=6  Sharpe=0.793  CAGR=11.49%
Multi : LB=6 Sharpe=0.738  CAGR=10.53%

—— 汇总判断 ——
按夏普(Sharpe)：Dual 更优
按年化收益(CAGR)：Dual 更优
```