法律声明

本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

加班主任入群

《初阶！量化交易：策略编写及系统搭建》第6期

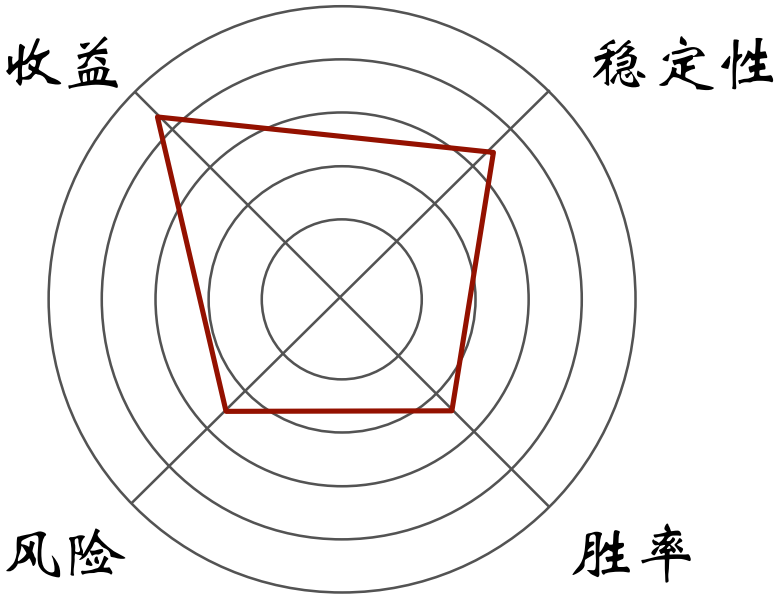
第5课：怎么评价和诊断交易策略

主　讲：刘英斐

内容介绍

几种常用评价指标的工程实现

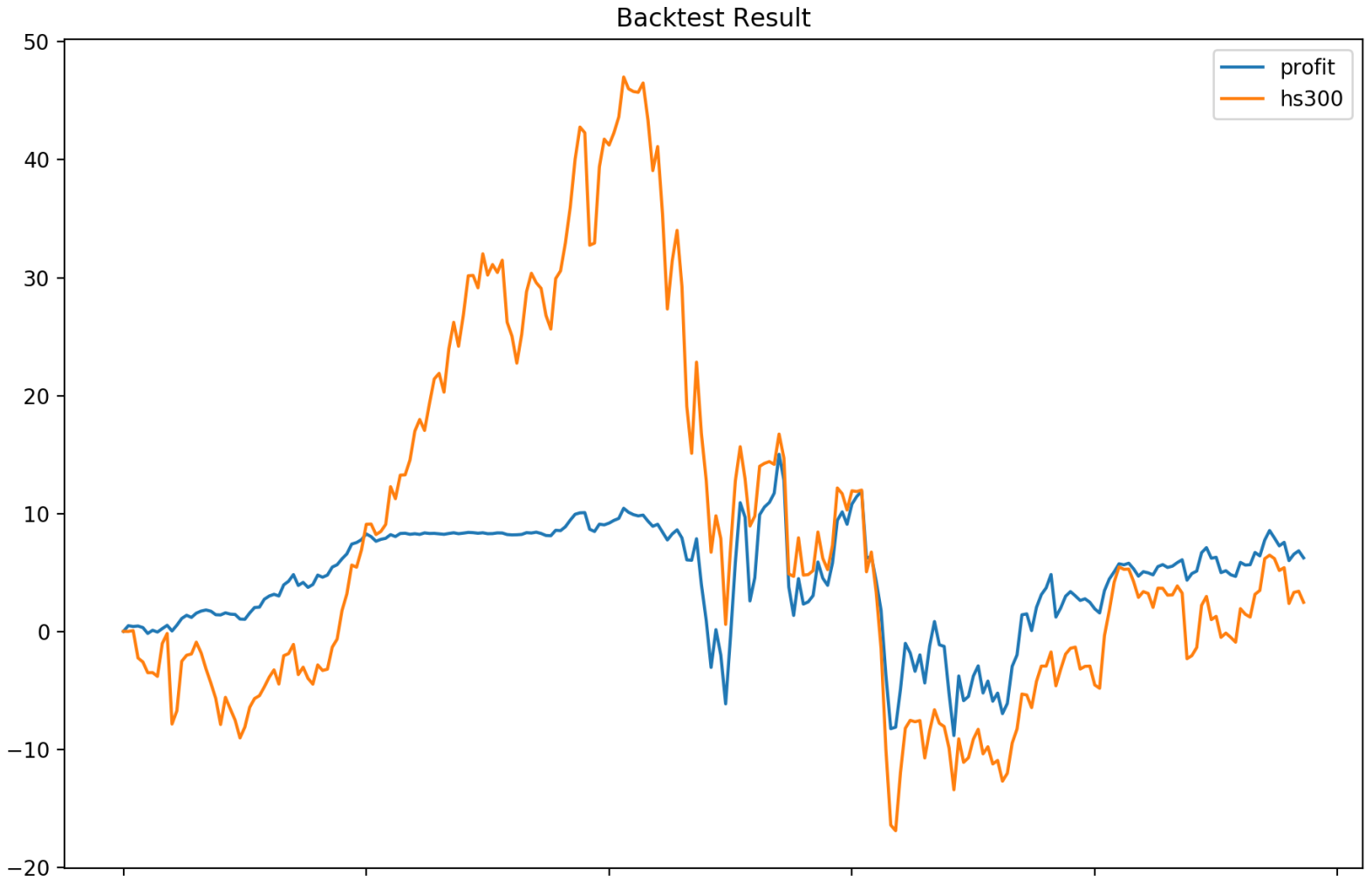
策略评价方法



代码

backtest.py

净值曲线



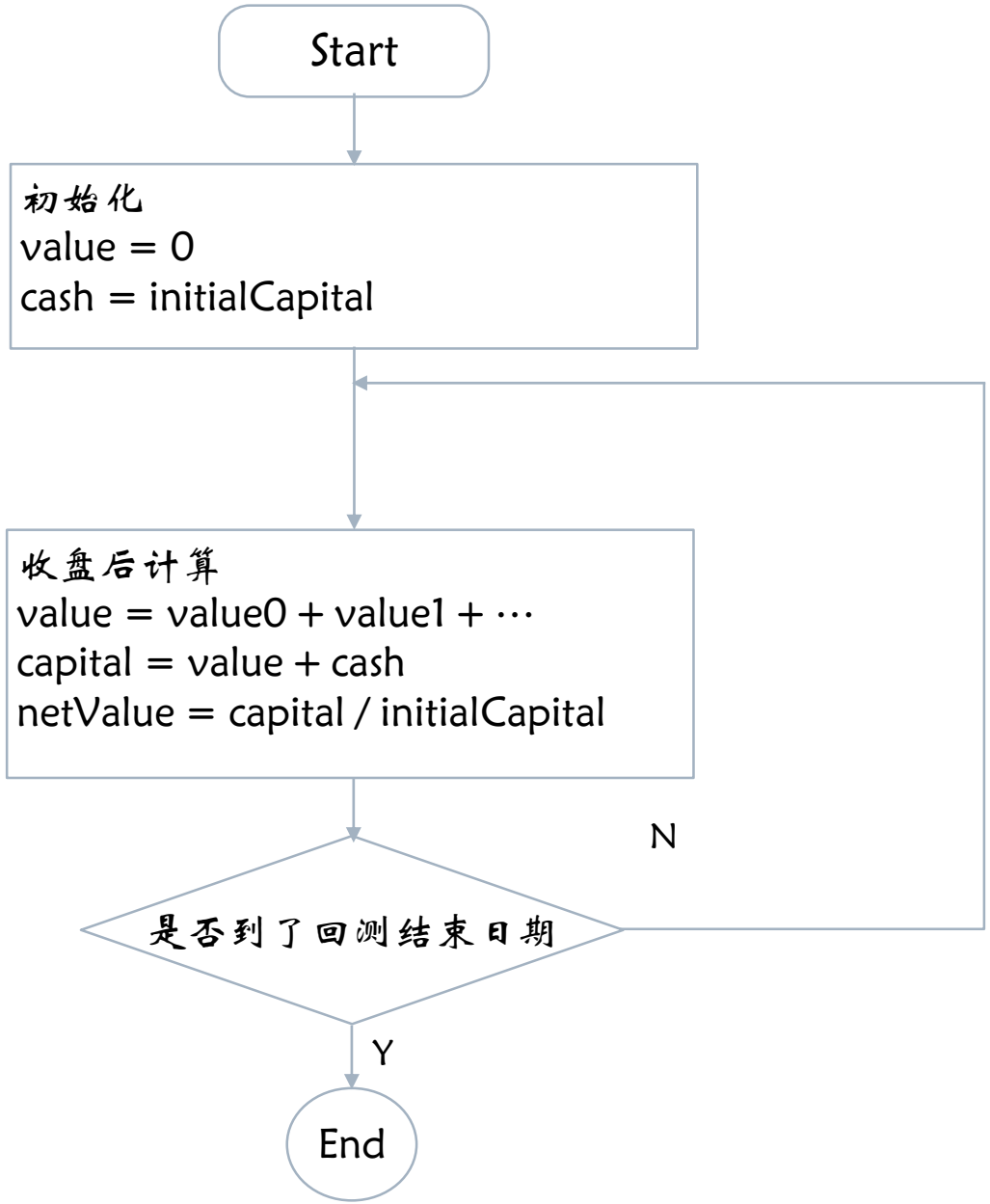
净值计算

InitialCapital - 初始总资产

Capital - 账户总资产
Value - 持仓股总市值
Cash - 账户可用现金
Capital = Value + Cash

NetValue - 账户净值
NetValue = Capital / InitialCapital

ProfitPct - 账户总收益率
ProfitPct = (NetValue - 1) * 100%


Start
初始化
value = 0
cash = initialCapital
收盘后计算
value = value0 + value1 + …
capital = value + cash
netValue = capital / initialCapital
N
是否到了回测结束日期
Y
End

年化收益

- 年化收益率是指把一段时间内的收益率换算成年收益率
- 单利/复利年化收益

计算公式 – 单利

- 单利年化收益率 = 当前投资收益率 /（投资天数/365）×100%
- 举例
  - 某策略在2017年6月1日到6月30日期间的收益为3%
  - 年化收益率 = 3%/(30/365) X 100%，即36.5%

计算公式 – 复利

$$Years = \frac{TradingDays}{AnnualTradingDays} \quad (1)$$

$$NetValue = (1 + AnnualProfit)^{Years} \quad (2)$$

$$AnnualProfit = \sqrt[Years]{NetValue} - 1 \quad (3)$$

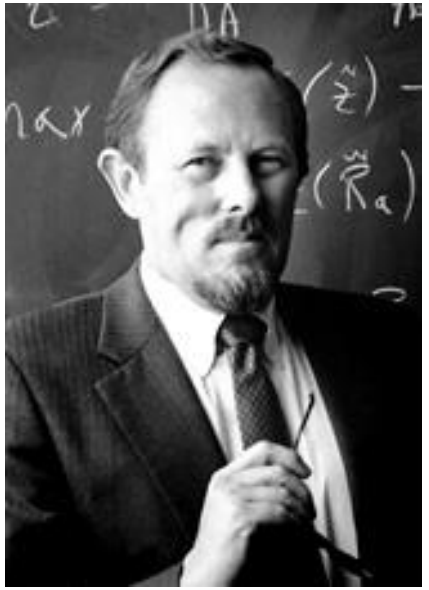
$$AnnualProfit = NetValue^{\frac{1}{Years}} - 1 \quad (4)$$

```
def compute_annual_profit(trading_days, net_value):
    """
    计算年化收益
    """

    annual_profit = 0
    if trading_days > 0:
        # 计算年数
        years = trading_days/245
        # 计算年化收益
        annual_profit = pow(net_value, 1/years) - 1

    annual_profit = round(annual_profit * 100, 2)

    return annual_profit
```

年化收益

Sharpe Ratio（夏普比率）

- 夏普比率（SHARPE）是一个可以同时对收益与风险加以综合考虑的指标
  - 在给定的风险水平下使期望回报最大化
  - 在给定期望回报率的水平上使风险最小化



计算公式

$$\text{Sharpe Ratio} = \frac{E(R_P) - R_f}{\sigma_P}$$

- $E(R_p)$：投资组合预期收益率
- R_f：无风险利率，一般指国债或定期存款利率
- σ_p：投资组合的收益标准差，即风险

策略的评价指标—夏普比率

$$ProfitMean = \frac{1}{N}\sum_{i=0}^{N} Profit_i$$

$$ProfitStd = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(Profit_i - ProfitMean)^2}$$

$$SharpeRatio = \frac{ProfitMean - R_f}{ProfitStd}$$

R_f - 无风险收益

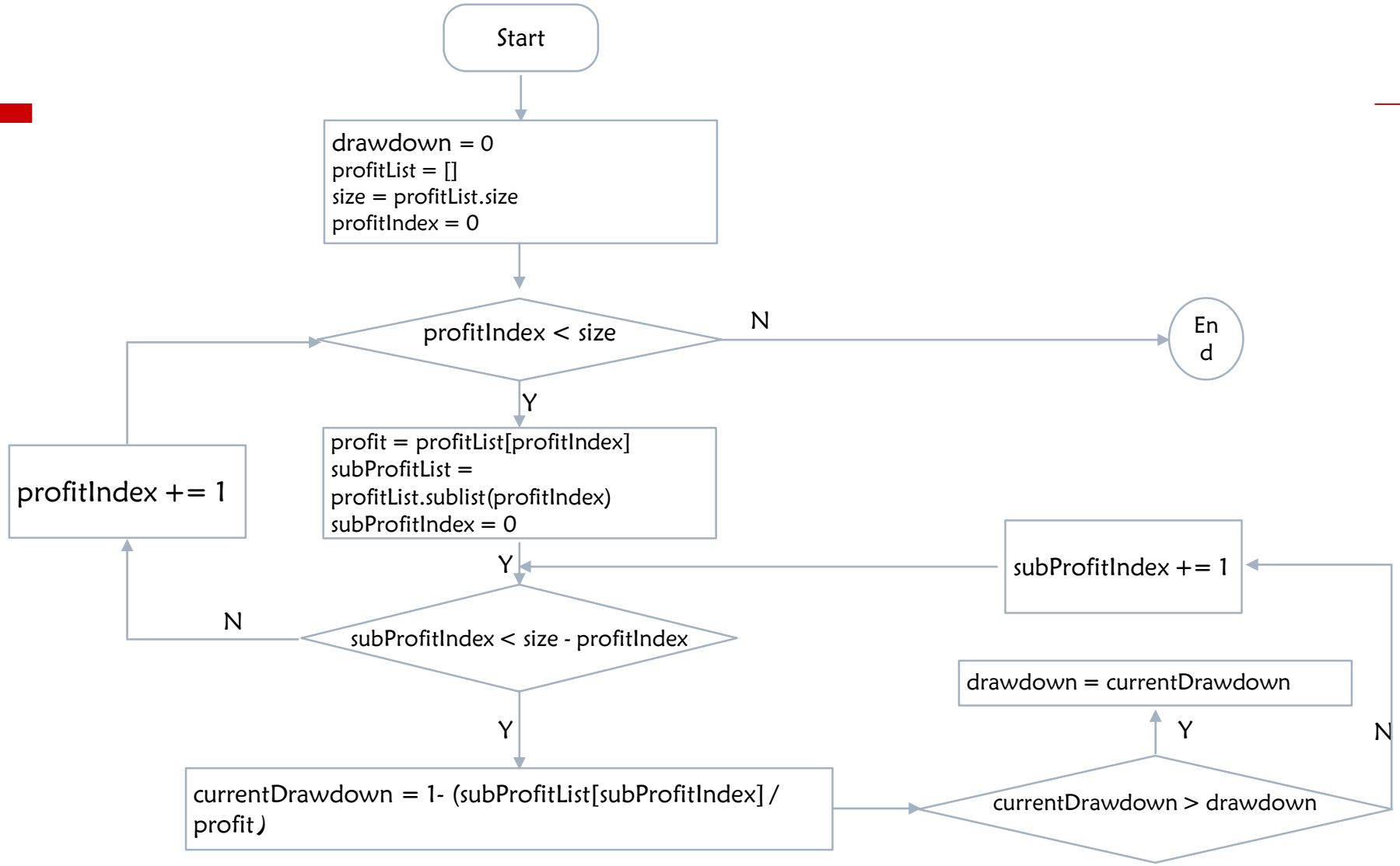
夏普比率

```
def compute_sharpe_ratio(net_value, df_day_profit):
    """
    计算夏普比率
    :param net_value: 最后的净值
    :param df_day_profit: 单日的收益，profit：策略单日收益，hs300：沪深300的单日涨跌
幅
    """

    # 总交易日数
    trading_days = df_day_profit.index.size

    # 计算单日收益标准差
    profit_std = round(df_day_profit['profit'].std(), 4)
    print(profit_std)

    # 年化收益
    annual_profit = compute_annual_profit(trading_days, net_value)

    # 夏普比率
    sharpe_ratio = (annual_profit - 4.75) / (profit_std * pow(245, 1 / 2))

    return annual_profit, sharpe_ratio
```

最大回撤

在选定周期内任一历史时点往后推，策略的净值走到最低点时的收益率回撤幅度的最大值

用来描述策略可能出现的最糟糕情况，衡量了最极端可能的亏损。


Start
drawdown = 0
profitList = []
size = profitList.size
profitIndex = 0
profitIndex < size
N
End
Y
profit = profitList[profitIndex]
subProfitList =
profitList.sublist(profitIndex)
subProfitIndex = 0
profitIndex += 1
Y
subProfitIndex += 1
N
subProfitIndex < size - profitIndex
drawdown = currentDrawdown
Y
Y
N
currentDrawdown = 1- (subProfitList[subProfitIndex] / profit)
currentDrawdown > drawdown

```
def compute_drawdown(net_values):
    """
    计算最大回撤
    :param net_values: 净值列表
    """
    # 最大回撤初始值设为0
    max_drawdown = 0
    size = len(net_values)
    index = 0
    # 双层循环找出最大回撤
    for net_value in net_values:
        for sub_net_value in net_values[index:]:
            drawdown = 1 - sub_net_value/net_value
            if drawdown > max_drawdown:
                max_drawdown = drawdown

        index += 1

    return max_drawdown
```

最大回撤

信息率

- 信息率用来衡量承担主动风险所带来的超额收益，表示单位主动风险所带来的超额收益。
- 在承担适度风险的情况下，尽量追求高信息率

计算公式

$IR = \alpha_p / \omega_p$

α_p 组合的超额收益

ω_p 主动风险

```
def compute_ir(df_day_profit):
    """
    计算信息率
    :param df_day_profit: 单日收益，profit - 策略收益 hs300 - 沪深300的
    :return: 信息率
    """
    # 计算单日的无风险收益率
    base_profit = 4.5 / 245

    df_extra_profit = pd.DataFrame(columns=['profit', 'hs300'])
    df_extra_profit['profit'] = df_day_profit['profit'] - base_profit
    df_extra_profit['hs300'] = df_day_profit['hs300'] - base_profit

    # 计算策略的单日收益和基准单日涨跌幅的协方差
    cov = df_extra_profit['profit'].cov(df_extra_profit['hs300'])
    # 计算策略收益和基准收益沪深300的方差
    var_profit = df_extra_profit['profit'].var()
    var_hs300 = df_extra_profit['hs300'].var()
    # 计算Beta
    beta = cov / var_hs300
    # 残差风险
    omega = pow((var_profit - pow(beta, 2) * var_hs300) * 245, 1/2)
    # Alpha
    alpha = (df_extra_profit['profit'].mean() - (beta * df_extra_profit['hs300'].mean())) * 245
    # 信息率
    ir = round(alpha / omega)

    print('cov：%10.4f，var_profit：%10.4f，var_hs300：%10.4f，beta：%10.4f，omega：%10.4f，alpha：
%10.4f，ir：%10.4f' %
          (cov, var_profit, var_hs300, beta, omega, alpha, ir), flush=True)

    return ir
```

休息一下

5分钟后回来

差得离谱和好得离谱都是问题

什么样的策略是好策略

不以目标需求为依据的策略性能评估
都是耍流氓

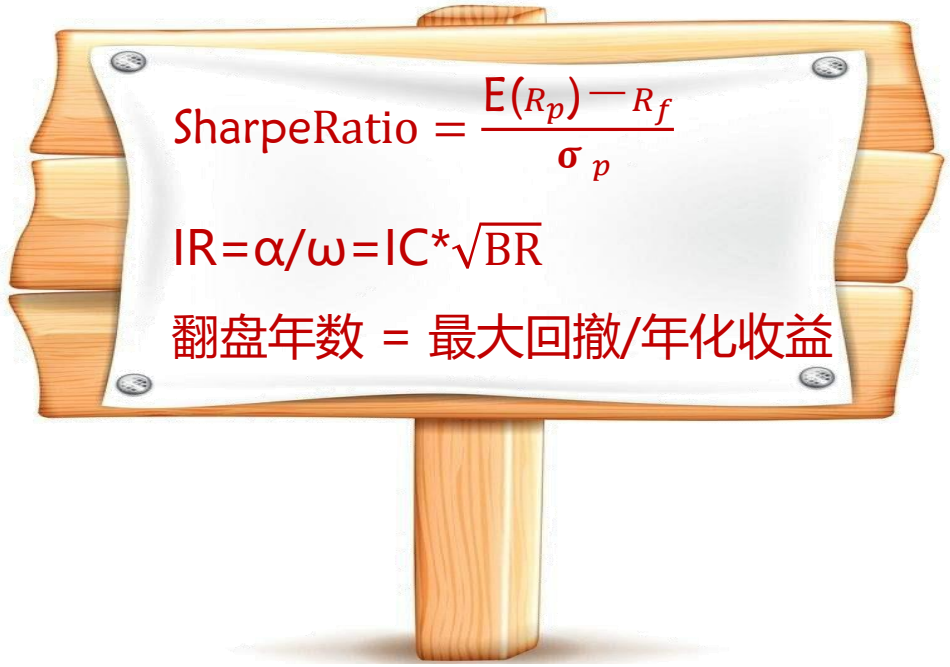
没有好人和坏人
也没有好策略和坏策略
只有适合与不适合

风险 vs. 收益

- 首先要确定你的目标
- 看看你的策略性能是否符合你的目标，从风险和收益两个方面来衡量
- 再来看看你的策略是否能够稳定地保持该性能
- 做好心理准备

如何评价

- 夏普比率
  - 风险与收益的权衡
  - 索提诺比率
- 最大回撤
  - 最坏的情况
- 年化收益
  - 按复利计算
- 心理因子
  - 舒适度
- 信息率
  - 投资经理的能力



量化系统工作过程


策略制定

行情判断


股票池筛选


信号产生


头寸管理


交易执行


人工干预


数据


评价和优化

运气or实力

回测阶段不足200次交易

经验值最低也需要70次

停牌也能成交

还有临时停牌

涨停也能买？！

跌停也能卖？！

ST的票5%不算涨跌停

还有新股哦

单笔成交量超过当日该股票成交量的10%

进得去出不来

日内出的信号以开盘价成交

肿么做到的呢？

收盘价能成交吗？

这个需要具体分析

相同条件每次回测结果不一致

可以解释吗？

究竟要满足什么要求呢？

稳健型

- 正收益
- 回撤小

进取型

- 风险收益平衡
- 风险至上

激进型

- 在可承受的风险范围内追求收益最大化
- 极端行情配套的风控和严格的压力测试

从交易品种和策略类型出发

股票多头策略	• Sharpe > 1，最好大于1.5
Alpha策略	• 最大回撤 < 5%
商品期货	• Sharpe > 1.7
趋势型	• 赔率：股票型 > 2，商品期货 > 5 • 心理因子 > 3
回复型	• 高胜率 > 60%

重要的是逻辑

从哪些方面去改进策略

一个完整的交易系统

- 市场——买卖什么
- 逻辑——买卖思路
- 头寸规模——买卖多少
- 入市——何时买进
- 止损——何时退出亏损的头寸
- 离市——何时退出赢利的头寸

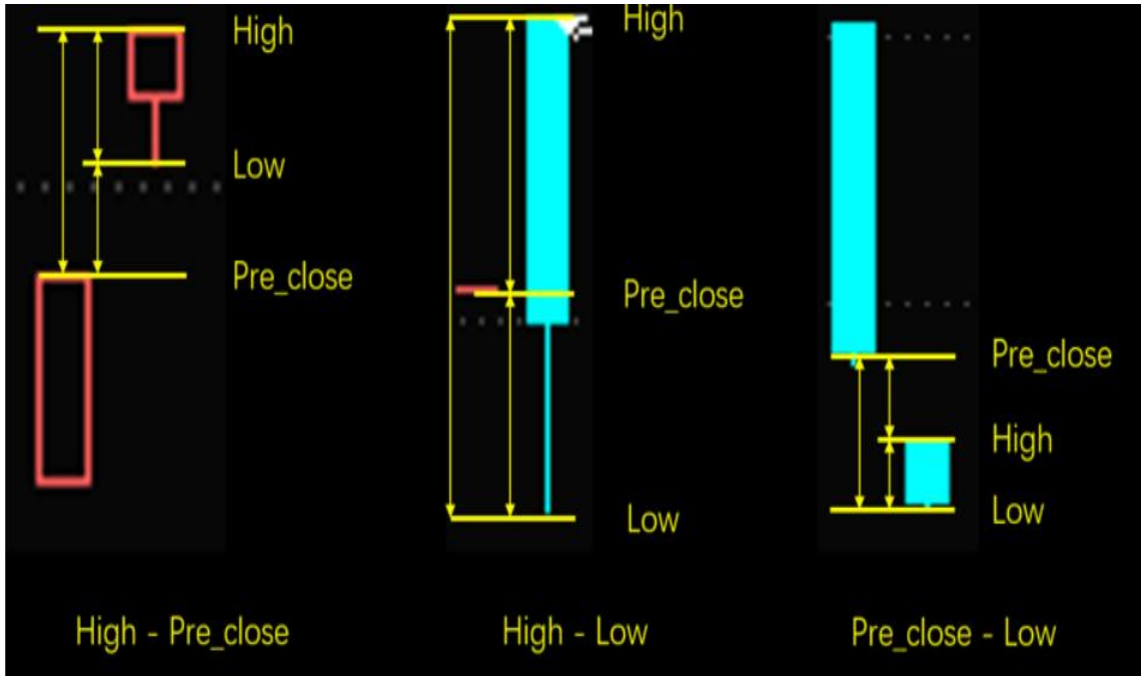
还是先来看一个策略

随机入市策略

- 每次凭掷硬币随机进入市场，或者做多头，或者做空头
- 一旦得到一个退出市场的信号，就基于随机信号再次入市
- 用EMA(ATR,10)指标来确定市场的波动性
- 用这一波动性得数的3倍作为初始止损
- 跟踪止损，只按照对盈利有利的方式移动（多头向上，空头向下）
- 在多个期货市场上运行检验
- 只进行1笔合约的交易 vs. 使用1%风险法则来确定头寸规模

真实波幅

$$TrueRange = Max(High - Low), | preclose - High |, | preclose - Low |)$$



ATR与EMA

$$\mathrm{N}(ATR) = \frac{1}{\mathrm{n}} \times \sum_{i=1}^{n} TrueRange_i$$

$$\mathrm{EMA}_{\mathrm{today}} = a * ATR_{today} + (1 - a) * EMA_{yesterday}$$

1%风险法则

$$U = \frac{C \times 1\%}{NV}$$

其中：

- C = 投入的资本 x 杠杆，可以理解为可承受的亏损
- 价值量波动性NV = 波动性代表的货币价值
- 波动性N = EMA

初始止损和跟踪止损

初始止损【多头】：

$$\text{Price}Mark = Buy\,\text{Pr}\,ice - k \times \text{EMA}$$

跟踪止损【多头】：

$$\text{Price}Mark = \text{MAX}(\text{R}ealP\text{rice - k} * \text{EMA}, \text{Pr}\,iceMark)$$

只按照对盈利有利的方向移动

设置全局参数

```
def set_params(context):
    # 设置合约简称
    g.future_symbol = 'J'
    # 当日的主力合约
    g.future = None
    # 最近一次交易的合约
    g.last_future = None
    # 多头仓位
    g.long_position = False
    # 空头仓位
    g.short_position = False
    # 设置时间窗口
    g.atr_window = 10
    # 波动性倍数
    g.ema_times = 3
    # 计算波动性得数的时间窗口
    g.ema_window =10
    # 止损价格
    g.price_mark = [0, 0]
    # 头寸风险因子
    g.pos_factor = 0.01
```

初始化函数

```
def initialize(context):
    # 设置参数
    set_params(context)
    # 设定基准[主力合约]
    set_benchmark(g.future)
    # 开启动态复权模式
    #set_option('use_real_price', True)

    # 设定账户为金融账户
    set_subportfolios([SubPortfolioConfig(cash=context.portfolio.starting_cash,
type='index_futures')])
    # 期货类每笔交易时的手续费是：买入时万分之0.23,卖出时万分之0.23,平今仓为万分之23
    set_order_cost(OrderCost(open_commission=0.000023,
close_commission=0.000023,close_today_commission=0.0023), type='index_futures')
    # 设定保证金比例
    set_option('futures_margin_rate', 0.15)
    # 开盘前运行
    run_daily( before_market_open, time='before_open', reference_security=g.future)
    # 开盘时运行
    run_daily( market_open, time='every_bar', reference_security=g.future)
    # 收盘后运行
    run_daily( after_market_close, time='after_close', reference_security=g.future)
```

移动止损

```
# 最近1个K线的收盘价
close_price = attribute_history(g.future , 1, '1m', 'close').values[0]

if g.long_position:
    # 多头仓位上浮止损线
    g.price_mark[0] = max(close_price - g.price_mark[1], g.price_mark[0])
    #最新价触发止损
    if get_current_data()[g.future].last_price < g.price_mark[0]:
        order_target(g.future, 0, side = 'long')
        g.long_position = False

if g.short_position:
    # 空头仓位下调止损线
    g.price_mark[0] = min(close_price + g.price_mark[1], g.price_mark[0])
    #最新价触发止损
    if get_current_data()[g.future].last_price > g.price_mark[0] :
        order_target(g.future, 0, side = 'short')
        g.short_position = False
```

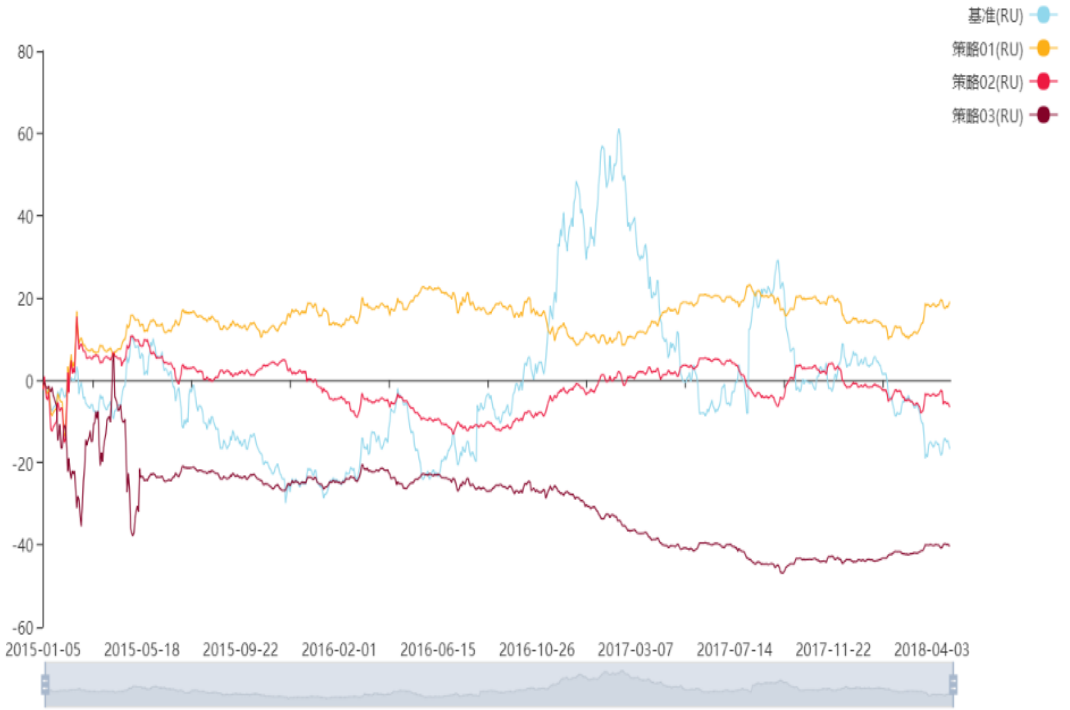
开仓

```
## 开仓信号：模拟随机掷硬币，0是开空仓，1是开多仓
open_signal = random.randint(0, 1)

# 多头开仓
if open_signal ==1:
    # 确定开仓头寸
    unit = get_unit(context.portfolio.total_value,ATR,g.future_symbol)
    order(future,unit,side='long')
    if context.portfolio.positions[future].total_amount>0:
        g.price_mark = [context.portfolio.long_positions[future].price -
          g.ema_times*EMA, g.ema_times*EMA]
        g.long_position = True
        g.last_future= future

# 空头开仓
elif open_signal == 0:
    unit = get_unit(context.portfolio.total_value,ATR,g.future_symbol)
    order(future,unit,side='short')
    if context.portfolio.short_positions[future].total_amount > 0:
        g.price_mark = [context.portfolio.short_positions[future].price  +
          g.ema_times*EMA, g.ema_times*EMA]
        g.short_position = True
        g.last_future= future
```

计算ATR和EMA

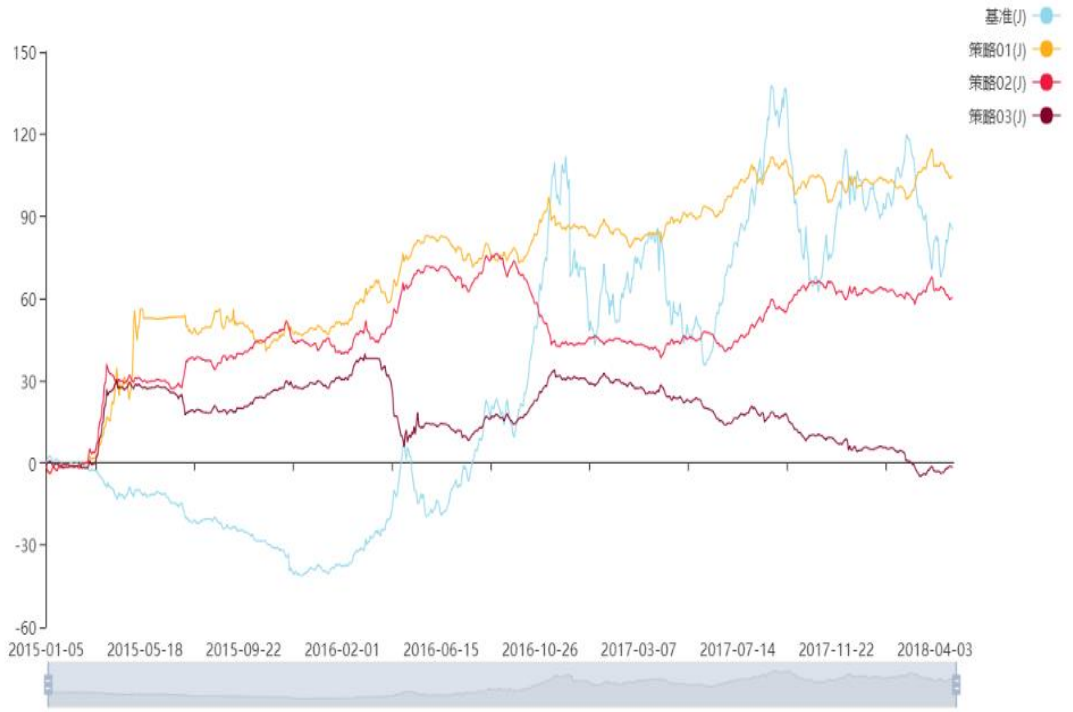
```
# 查询历史数据
price_list = attribute_history(g.future, g.atr_window+1+g.ema_window-1, '1d',
['close','high','low'])


#计算EMA
def get_EMA(price_list):
    atr_list = []
    # 计算连续n日的ATR
    for i in range(0, g.ema_window):
        atr_list .append(get_ATR(price_list[i: g.atr_window+i+1],g.atr_window))

    # 计算EMA(ATR, n)
    EMA = pd.ewma(DataFrame({"atr":atr_list}), g.ema_window)['atr'].iloc[-1]

# 计算ATR
def get_ATR(price_list):
    TR_list = [max(price_list[‘high’].iloc[i]-
              price_list['low'].iloc[i],abs(price_list['high'].iloc[i]-price_list['close'].iloc[i-
              1]),abs(price_list['close'].iloc[i-1]-price_list['low'].iloc[i])) for i in range(1,
              g.atr_window+1)]
    ATR = np.array(TR_list).mean()
    return ATR
```

橡胶



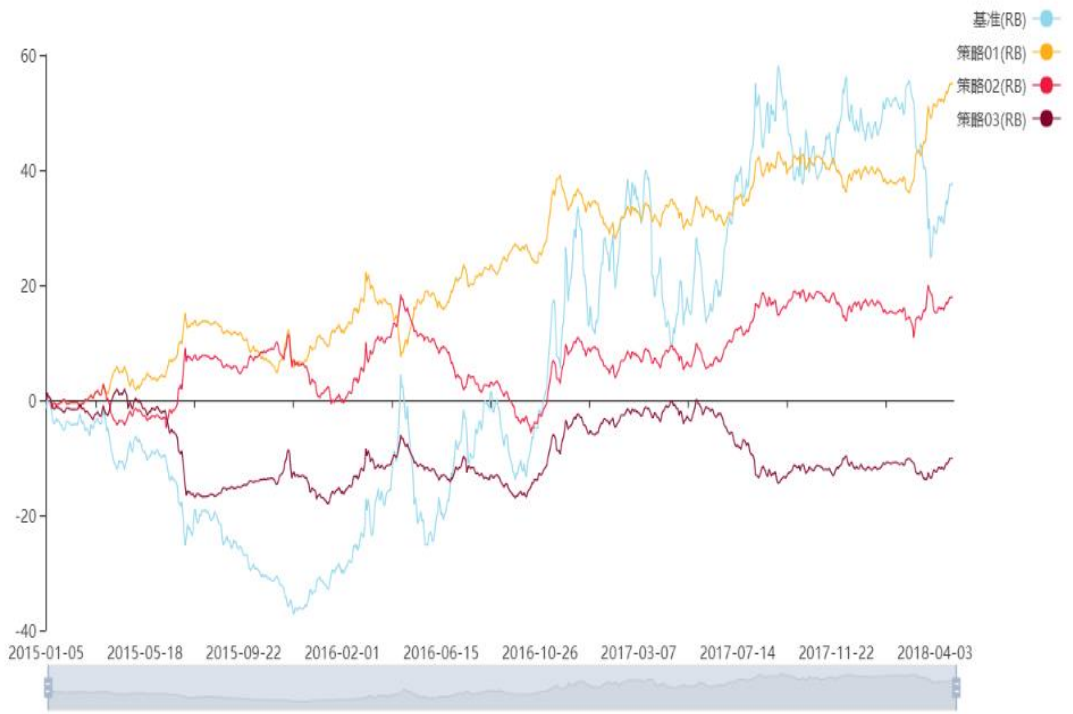
	年化	夏普	最大回撤	总收益
橡胶(RU)	-5.50%	-0.139	48.78%	-16.74%
策略01(RU)	5.54%	0.107	13.8%	19.07%
策略02(RU)	-2.07%	-0.397	24.61%	-6.55%
策略03(RU)	-14.77%	-0.740	49.98%	-40.39%

焦炭



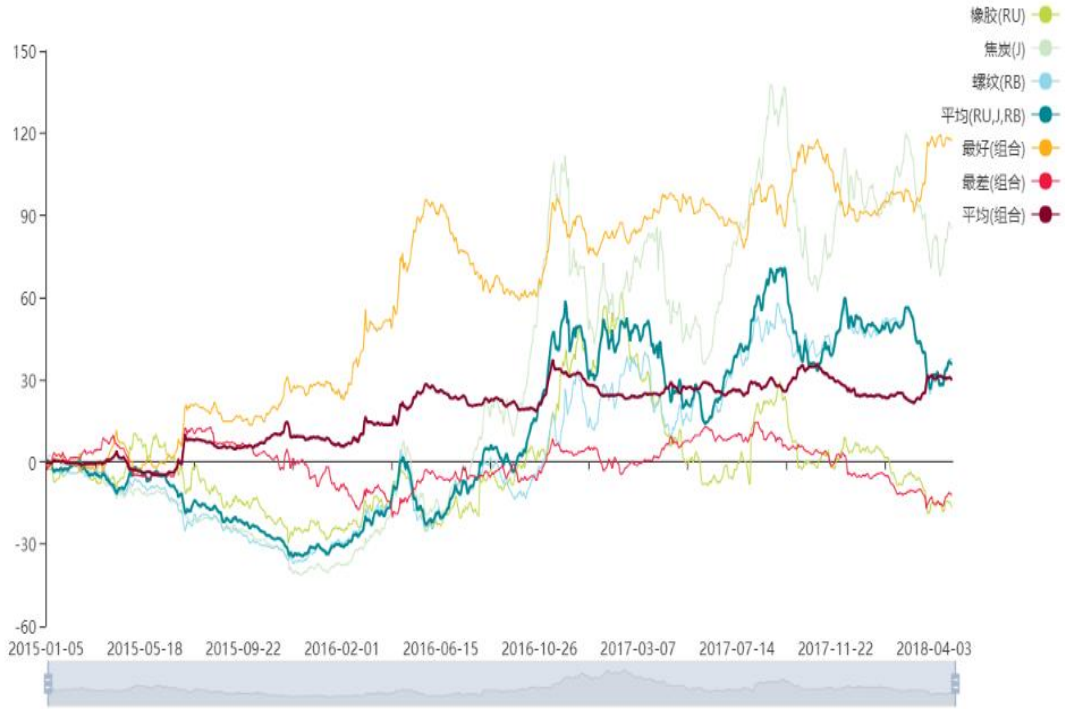
	年化	夏普	最大回撤	总收益
焦炭(J)	20.94%	0.413	34.96%	85.03%
策略01(J)	24.77%	1.392	9.92%	104.63%
策略02(J)	15.69%	1.058	21.6%	60.26%
策略03(J)	-0.48%	-0.363	31.88%	-1.55%

螺纹钢



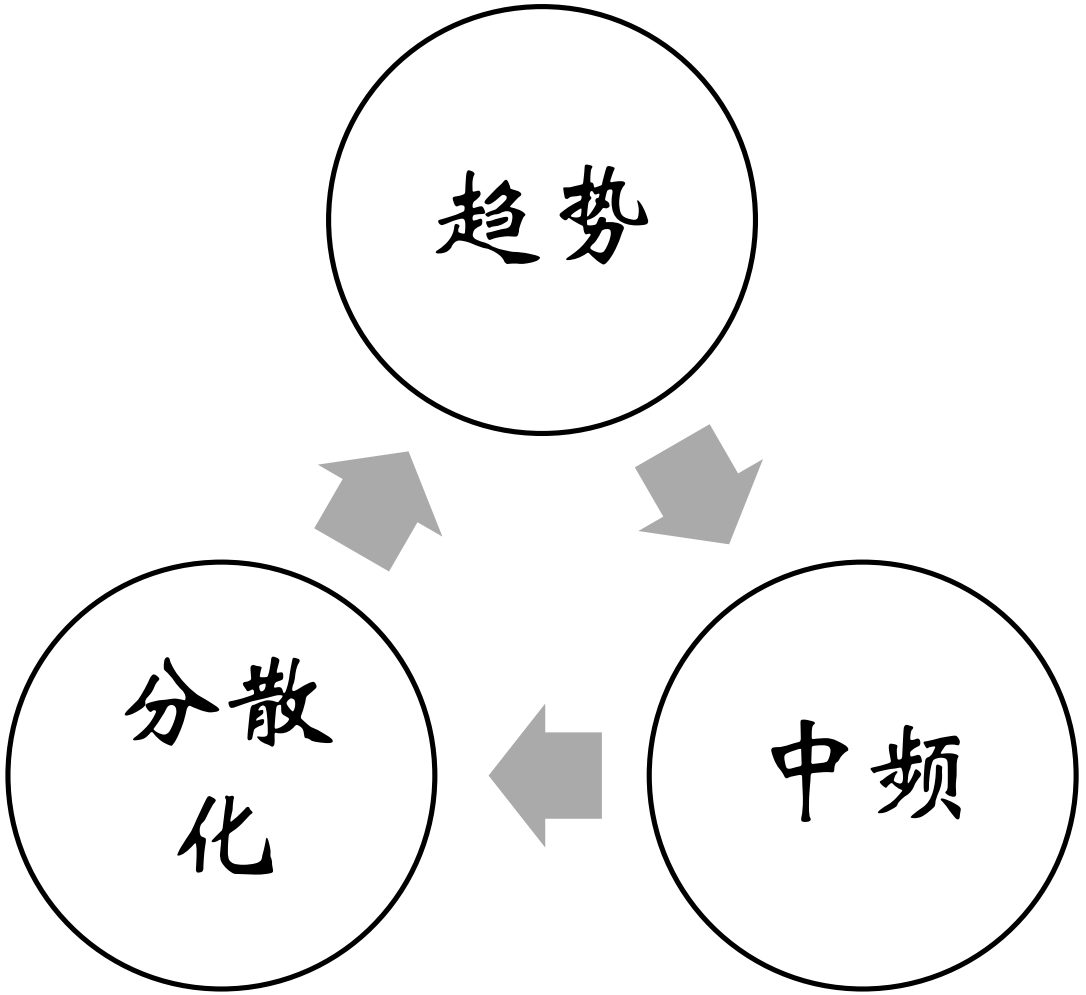
	年化	夏普	最大回撤	总收益
螺纹(RB)	10.41%	0.333	38.57%	37.78%
策略01(RB)	14.52%	1.147	12.03%	55.08%
策略02(RB)	5.23%	0.126	20.27%	17.15%
策略03(RB)	-3.24%	-0.783	19.55%	-10.11%

组合

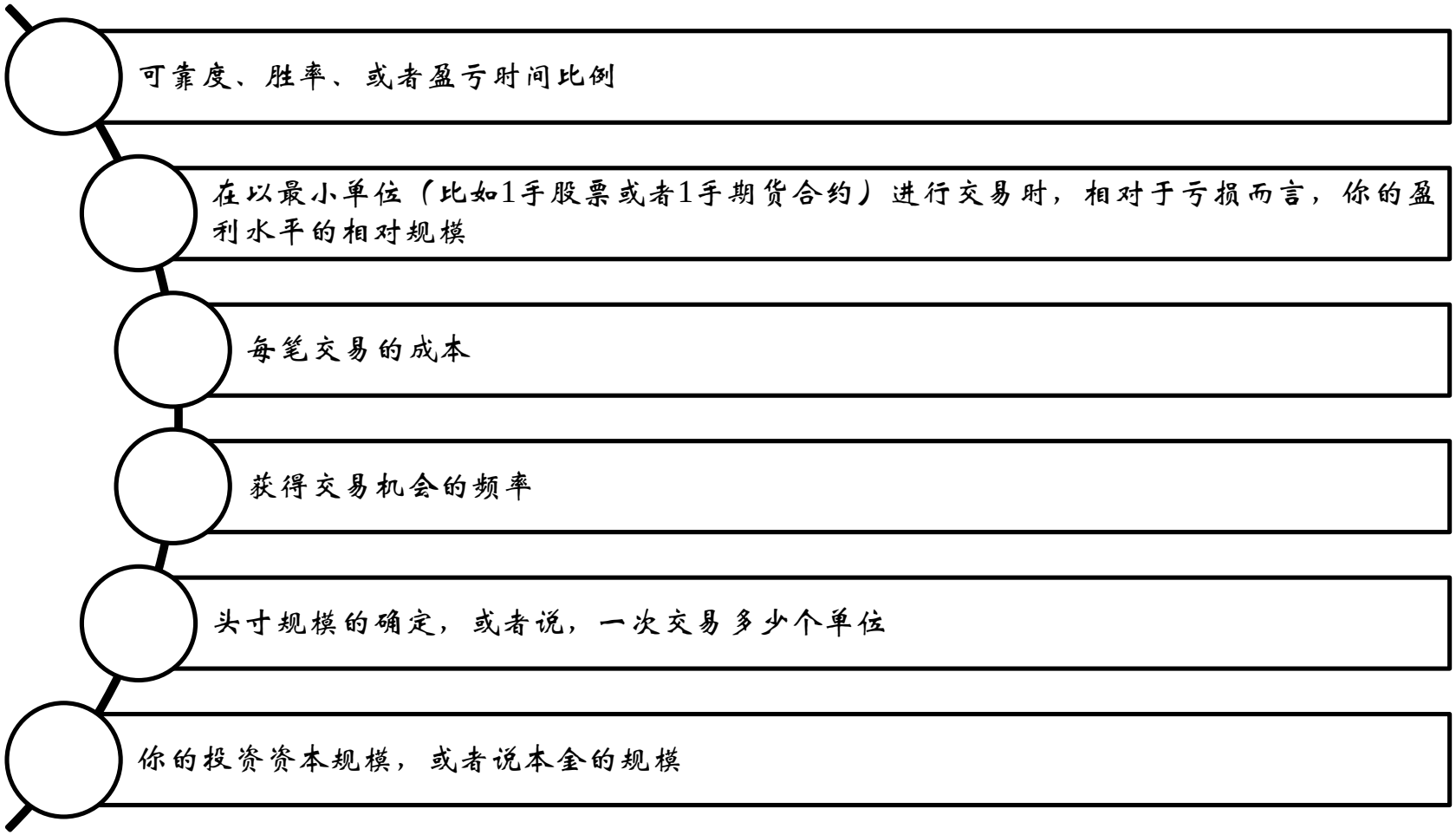


	年化收益	夏普	最大回撤	总收益
橡胶(RU)	-5.50%	-0.139	48.78%	-16.74%
焦炭(J)	20.94%	0.413	34.96%	85.03%
螺纹(RB)	10.41%	0.333	38.57%	37.78%
基准平均	9.80%	0.271	34.81%	35.35%
组合	8.38%	0.729	11.59%	29.75%

随机策略的本质思考



要素


可靠度、胜率、或者盈亏时间比例
在以最小单位（比如1手股票或者1手期货合约）进行交易时，相对于亏损而言，你的盈利水平的相对规模
每笔交易的成本
获得交易机会的频率
头寸规模的确定，或者说，一次交易多少个单位
你的投资资本规模，或者说本金的规模

从逻辑出发 – 以趋势型策略为例

- 胜率不高
- 赔率要大
- 不要浪费过多精力在信号系统的胜率上
- 头寸管理：顺势加仓
- 退出方式：止盈止损
- 截断亏损 让利润奔跑
- 控制亏损 把利润交给市场
- 没有行情的时候不要大亏 行情来了要赚足

总结

- 年化收益、夏普比率、最大回撤、信息率的实现
- 评价策略性能要从投资目标出发
- 策略的改进要从逻辑下手

问答互动

在所报课的课程页面，
1、点击“全部问题”显示本课程所有学员提问的问题。
2、点击“提问”即可向该课程的老师和助教提问问题。

联系我们

小象学院：互联网新技术在线教育领航者

– 微信公众号：**小象学院**

THANKS