量化投资策略评估与归因分析

厦门大学王亚南经济研究院
陈海强 教授

目录

一、交易策略收益与风险评估

交易策略验证与风控

- 量化投资策略设计最后一步就是回测，在交易策略，投资策略或风险建模中，回测旨在估计策略或模型在过去一段时间内的表现。
- 回测是一种反向测试，应用于前一时间段的特殊类型的交叉验证；回溯测试提供了在合成数据上测试模型和策略时无法获得的信息，同时可以进一步对策略参数进行优化。

量化投资策略验证与风控

策略的标准和绩效陷阱

a. 基准指数

b. 回报的连续性

c. 夏普比率

d. 最大回撤（区间）

e. 交易费用（频率）

f. 生存期偏差

g. 市场结构转换

h. 数据过拟合偏差

交易策略验证与风控

回测陷阱

a. 数据偷窥偏差(AB 测试法, 模拟交易)

b. 数据过拟合偏差 (参数个数, 样本规模, 样本外测试, 模拟交易, 自适应参数)

c. 敏感度分析 (参数敏感度 , 简化模型)

交易策略验证与风控

交易成本分析

a. 佣金

b. 市场流动性（bid-ask spread）

c. 机会成本（限价成交）

d. 市场冲击（大单成本）

e. 滑点（延时成本）

量化投资策略验证与风控

降低交易成本

a. 低价股票（流动性、佣金）

b. 限制订单大小（市场冲击，1%的市场交易量）

c. 按照股票市值调整订单大小

d. 算法交易（导致延时成本）

量化投资策略验证与风控

模拟交易服务

a. 策略Debug

b. 数据过拟合偏差

c. 数据偷窥偏差

d. 更加直观的认识策略

交易策略验证与风控

实盘表现差于预期

a. bugs

b. 真实交易与回验匹配检查?

c. 执行成本偏高？

d. 市场冲击偏大?

数据过拟合偏差和市场结构转换？

交易策略验证与风控

仓位风险

a. 止损（金融危机蔓延)

b. half-Kelly 公式

c. 处理黑天鹅

软件系统风险

物理和自然灾害

模型风险

a. 数据过拟合

b. 市场结构转换

c. 数据生存期偏差

d. 竞争加剧

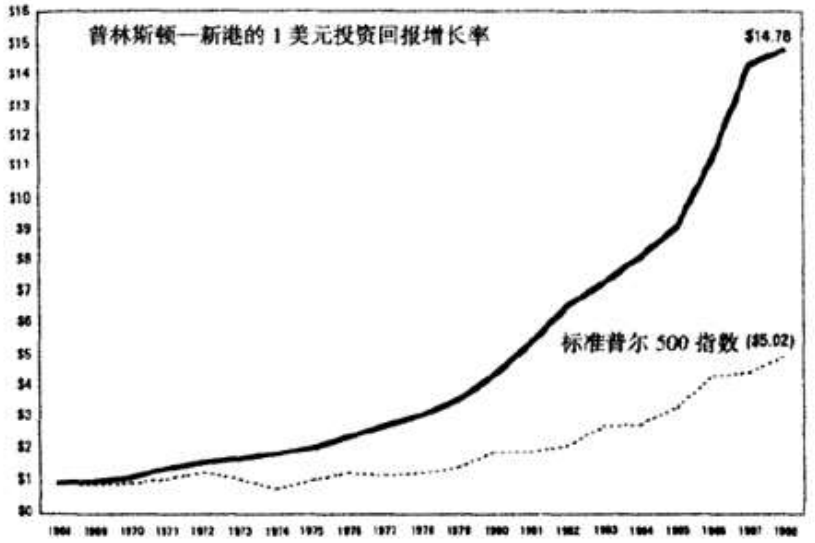
交易策略验证与风控

资金分配和杠杆管理：Kerry公式

在概率论中，凯利公式（也称 “凯利方程式”）是一个在期望净收益为正的独立重复赌局中，使本金的长期增长率最大化的投注策略。如果我们每次有p的概率盈利，那么盈利后我们的资金额就是本金×（1+盈利率*投资比例），而相反当出现1-p的亏损概率时，我们亏损后的资金额度为本金×（1-亏损率*投资比例）。作为股市的长期投资者，我们的目标是总收益的几何平均值最大，于是就可以得到下面这个结果：

$$\text{投资比例} = \frac{\text{盈利概率}}{\text{盈利额}} - \frac{\text{亏损概率}}{\text{亏损额}}$$

世界上第一只真正意义上的对冲基金普林斯顿-新港（Prinseton-Newport）就是靠凯利公式的策略赚的盆满钵盈



交易策略常用评判标准

- Alpha（阿尔法）

投资中面临着系统性风险（即Beta）和非系统性风险（即Alpha），Alpha是投资者获得与市场波动无关的回报。比如投资者获得了15%的回报，其基准获得了10%的回报，那么Alpha或者价值增值部分就是5%

公式：

$$Alpha = \alpha = R_p - [R_f + \beta_p(R_m - R_f)]$$

image

- R_p = 策略年化收益率
- R_m = 基准年化收益率
- R_f = 无风险利率（默认是0.04）
- β_p = 策略beta值

交易策略常用评判标准

- Beta（贝塔）

Beta表示投资的系统性风险，反映了策略对大盘变化的敏感性。例如一个策略的Beta为1.5，则大盘涨1%时，策略可能涨1.5%，反之亦然；如果一个策略的Beta为-1.5，说明大盘涨1%的时候，策略可能跌1.5%，反之亦然

公式:

$$Beta = \beta_p = \frac{Cov(D_p, D_m)}{Var(D_m)}$$

image

- D_p = 策略每日收益
- D_m = 基准每日收益
- $Cov(D_p, D_m)$ = 策略每日收益与基准每日收益的协方差
- $Var(D_m)$ = 基准每日收益的方差

交易策略常用评判标准

- Sharpe（夏普比率）

Sharpe表示每承受一单位总风险，会产生多少超额报酬，可以同时对策略的收益与风险进行综合考虑

公式:

$$SharpeRatio = \frac{R_p - R_f}{\sigma_p}$$

image

- R_p = 策略年化收益率
- R_f = 无风险利率（默认值为0.04）
- σ_p = 策略收益波动率

交易策略常用评判标准

- Sortino（索提诺比率）

Sortino 表示每承担一单位的下行风险，将会获得多少超额回报

公式：

$$SortinoRatio = \frac{R_p - R_f}{\sigma_p d}$$

- R_p = 策略年化收益率
- R_f = 无风险利率（默认值为0.04）
- σ_{pd} = 策略下行波动率

交易策略常用评判标准

- **Information Ratio（信息比率）**

Information Ratio 是用来衡量单位超额风险带来的σ_p = 策略收益波动率。信息比率越大，说明该策略单位跟踪误差所获得的超额收益越高，因此，信息比率较大的策略的表现要优于信息比率较低的基准。合理的投资目标应该是在承担适度风险下，尽可能追求高信息比率

$$InformationRatio = \frac{R_p - R_m}{\sigma_t}$$

- R_p = 策略年化收益率
- R_m = 基准年华收益率
- σ_t = 策略与基准每日收益差值的年化标准差

交易策略常用评判标准

- **Volatility（策略波动率）**

Information Ratio 是用来衡量单位超额风险带来的σ_p = 策略收益波动率。信息比率越大，说明该策略单位跟踪误差所获得的超额收益越高，因此，信息比率较大的策略的表现要优于信息比率较低的基准。合理的投资目标应该是在承担适度风险下，尽可能追求高信息比率

$$Volatility = \sigma_p = \sqrt{\frac{250}{n-1}\sum_{i=1}^{n}(r_p - \overline{r_p})^2}$$

- r_p = 策略每日收益第
- $\bar{r}_p$ = 策略每日收益的平均值 = lnΣi=lnrp
- n = 策略执行天数

交易策略常用评判标准

- **Benchmark Volatility（基准波动率）**

Benchmark Volatility 用来测量基准的风险性，波动越大代表基准风险越高

$$Volatility = \sigma_m = \sqrt{\frac{250}{n-1}\sum_{i=1}^{n}(r_m - \overline{r_m})^2}$$

- r_m = 其次每日收益第
- $\bar{r}_m$ = 基准每日收益的平均值 = lnΣi=lnrm
- n = 策略执行天数

交易策略常用评判标准

- **Max Drawdown（最大回撤）**

Max Drawdown 描述策略可能出现最糟糕的情况，最极端可能的亏损

$$MaxDrawdown = Max(P_x - P_v)/P_x$$

P_x, P_v = 策略某日股票和现金的总价值

最大回撤大了会导致直接击穿投资者的心里底线导致割肉离场，从而失去后续盈利的机会。如果投资者都是机器人，无视中间波动，这个指标就没有意义，但可惜不是，所以明明一个收益很好的策略，如果最大回撤大，也不是好策略，因为投资者落实不了

二、交易策略或基金业绩归因

业绩归因

- 业绩归因通常用于分析组合的收益来源，是投资经理进行策略评估与修正、投资者度量基金经理各方面能力的重要工具
- 业绩归因主要有两大类方法，基于净值的业绩归因和基于持仓的业绩归因。前者主要是对基金的收益率序列进行分析，而后者则是根据组合的实际持仓进行分析。

基于净值的业绩归因

- 基于净值的业绩归因是将基金的收益序列对风格因子进行时间序列回归，然后根据回归结果考察每种风格对组合收益的贡献，以及基金经理的主动管理能力（Alpha）对组合收益的贡献。

  - Treynor 和 Mazuy（1966）提出的 T-M 模型，将基金经理的能力分为选股和择时两类。他们在 CAPM 的基础上，增加了市场风险溢价的二次项，并以二次项的回归系数代表择时能力，以整个模型的截距项代表选股能力。
  - Henriksson 和 Merton（1981）对 T-M 模型进行修正，将二次项改为市场风险溢价与虚拟变量（市场风险溢价大于 0 时，为 1；反之，为 0）的乘积，提出了 H-M 模型。类似地，虚拟变量的系数代表择时能力，而回归截距项代表选股能力。
  - Chang 和 Lewellen（1984）在 H-M 模型的基础上作了进一步修改，加入两个虚拟变量，得到基金在市场风险溢价大于 0 和小于 0 时的 beta，并通过比较这两个 beta 的差值来分析基金的特点。
  - Sharpe（1992）通过一个二次规划问题估计基金的风格头寸，并首次提出基于风格分析来为每只基金单独制定一个业绩基准，以便更加精确地衡量基金经理的贡献。

净值因子回归模型

- 基于净值的业绩归因通过时间序列回归，将基金超额收益分解为风格因子收益和特质收益两部分，后者一般被认为来自于基金经理选股和择时的能力。

净值归因模型的一般形式为：

$$R_i = a_i + b_{i,1}F_1 + b_{i,2}F_2 + \ldots + b_{i,n}F_n + e_i$$

其中，R_i 为组合收益，F_1、F_2、…、F_n 为因子（各风格资产收益），$b_{i,1}$、…、$b_{i,j}$、…、$b_{i,n}$ 为组合对因子的敏感度。$a_i + e_i$ 是不能为因子（风格资产）所解释的部分（non-factor）。

上述模型中，常用的因子包括，市场收益、市值因子收益、估值因子收益、国家收益、行业收益等。在实际应用中，因子收益通常采用分散化的模拟组合进行估算

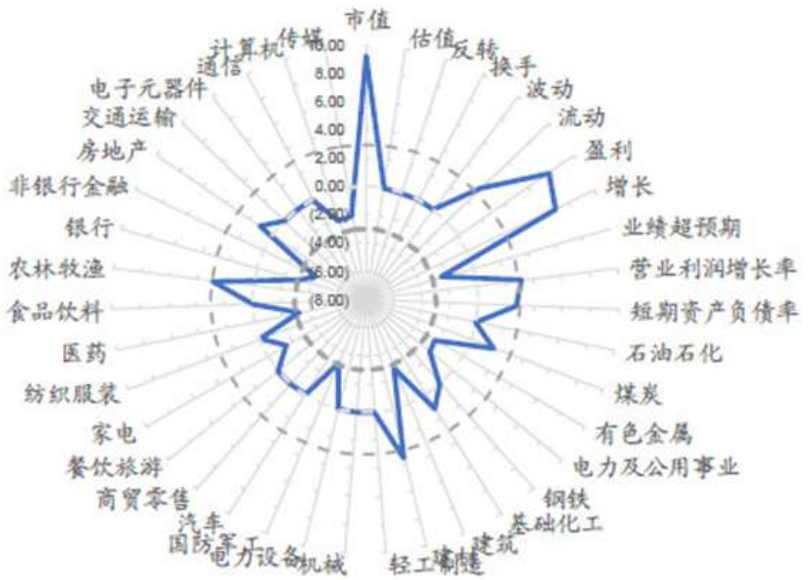
案例分析

- 基金 A 为例，对常用业绩归因模型的应用方法和结果进行演示和说明
- 基金 A 的投资目标是在控制跟踪误差的基础上，采用量化方法对基准指数进行增强。按照基金契约，股票资产的投资比例不低于基金资产的 90%，基准指数指数成分股和备选成分股的资产不低于股票资产的 80%，投资范围不包括股指期货品种
- 采用基金收益率序列对因子进行回归，可以得到基金相对于基准指数的行业和风格暴露

案例分析

- 该基金在行业上的控制非常严格，相对于基准指数没有明显的偏离。若从显著性的角度考虑，仅在非银和银行两个板块上存在轻微的低配，在建材上存在轻微的高配。由此推测，该基金在构建投资组合时，可能采取了行业中性的处理方法。从风格角度来看，该基金相对基准指数存在明显的小市值偏离，但在估值/成长维度上的偏离并不显著。此外，该基金还在盈利与增长两个基本面因子上有非常明显的偏离。

图2 基金 A 的风格与行业暴露（2013.01-2018.10）

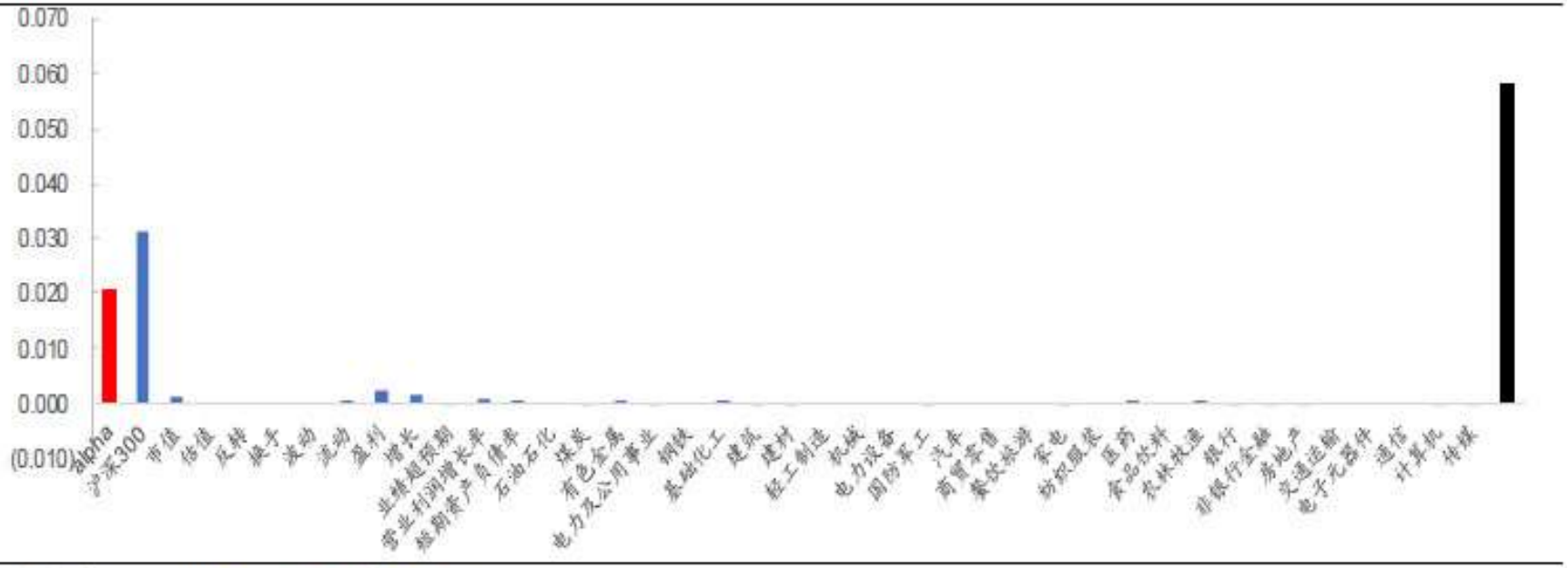


资料来源：Wind，海通证券研究所

案例分析

基金 A 近 5 年的业绩归因结果，其中，黑色柱状图为基金收益，由各因子的贡献（蓝色柱状图）和 Alpha（红色柱状图）组成。市场因子外，该基金最主要的收益来源是风格因子中的小市值和基本面因子中的盈利与增长，而行业因子几乎不存在明显的收益贡献。按照净值归因模型的定义，因子贡献以外的剩余收益即为基金 Alpha，是基金经理通过选股与择时获取的收益

图3 基金 A 基于净值的业绩归因结果（2013.01-2018.10）



资料来源：Wind，海通证券研究所

基于持仓的业绩归因

- 归因基于持仓的业绩归因方法是对投资组合在不同时点上的实际持仓进行分析，并将其映射到不同风格中。经典的模型有 Brinson 绩效归因模型及其多期改进，以及多因子归因模型。

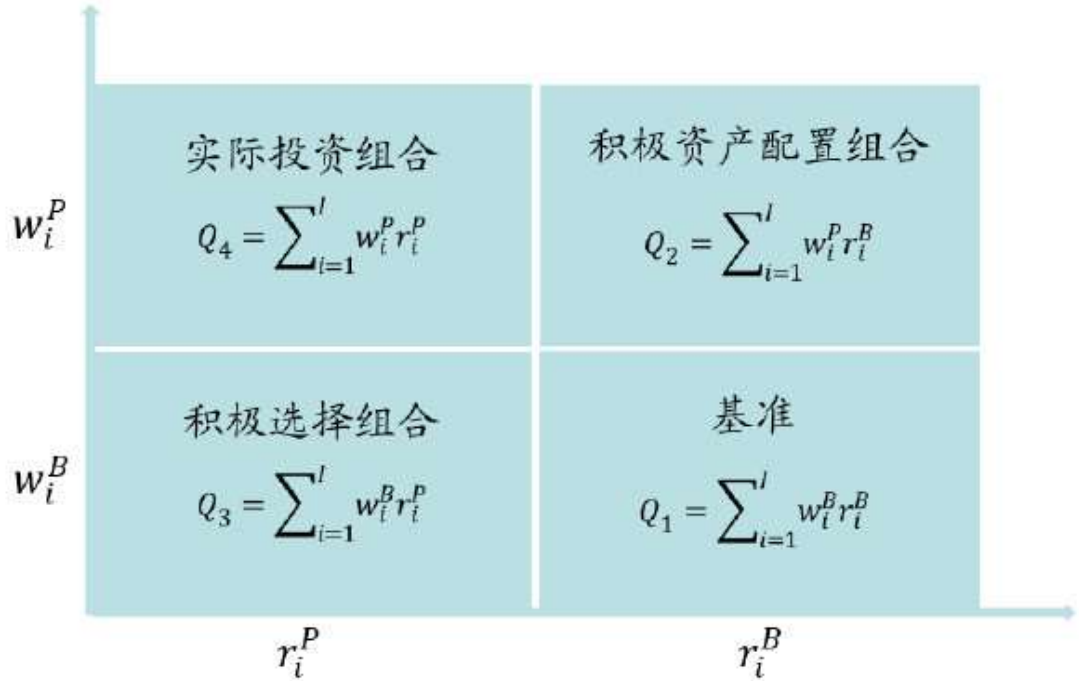
Brinson 模型

- Brinson 模型最早由 Brinson、Hood 和 Beebower 提出，BHB 模型模型从自上而下的角度将组合相对于基准的超额收益分解为资产配置效应、标的选择效应以及交互效应三部分。
- 随后 Brinson和 Fachler 将交互效应与选择效应合并，提出 BF 版本的 Brinson 模型。基于业绩归因的连续性要求，通常需要将单期归因结果合并为多期。
- 但从单期收益到多期收益需要考虑再投资的影响，而不是简单累加。为此，许多学者都对这个问题进行了深入研究，并提出了多种可由单期拓展至多期的归因算法。

单期 Brinson 归因模型

假设组合中共有 I 类资产，如，股票、债券、行业（或风格）、基金、衍生品等。以 w_i^P 表示实际投资组合中大类资产 i 的权重，w_i^B 表示基准组合中大类资产 i 的权重；r_i^P 表示实际组合中大类资产 i 的收益率，r_i^B 表示基准组合中大类资产 i 的收益率。Brinson 模型的收益分解过程如下图所示。

图5 Brinson 归因模型框架



资料来源：海通证券研究所整理

单期 Brinson 归因模型

实际投资组合相对于基准的超额收益记为 R^A，按照定义有 $R^A = Q_4 - Q_1$。基于上述 4 个概念组合可以将超额收益分解为大类资产配置收益（Allocation Return，AR）、选择收益（Selection Return，SR）和交互收益（Interaction Return，IR），即

$$R^A = AR + SR + IR,$$

$$\begin{cases} \text{资产配置收益：} AR = Q_2 - Q_1 = \sum_{i=1}^{I}(w_i^P - w_i^B)r_i^B \\ \text{选择收益：} SR = Q_3 - Q_1 = \sum_{i=1}^{I} w_i^B(r_i^P - r_i^B) \\ \text{交互收益：} IR = R^A - AR - SR = \sum_{i=1}^{I}(w_i^P - w_i^B)(r_i^P - r_i^B) \end{cases}$$

由以上定义可知，

- 配置效应（AR）= 超额权重 * 基准资产收益率，是在不进行证券选择的情况下，通过积极的资产配置，即超配具有正收益、低配具有负收益的资产所能获取的收益。
- 选择效应（SR）= 基准权重 * 资产超额收益，是各大类资产配置比例与基准相同，通过在每类资产内部进行积极的证券选择所能获取的超额收益。
- 交互效应（IR）= 超额权重 * 超额收益率，是投资组合在资产配置和个股选择逐层归因后的剩余部分。

多期 Brinson 归因模型

- 实际的投资组合评价过程中，往往需要对一只基金在多个时间段内的整体业绩表现进行归因，这就需要将模型由单期拓展到多期

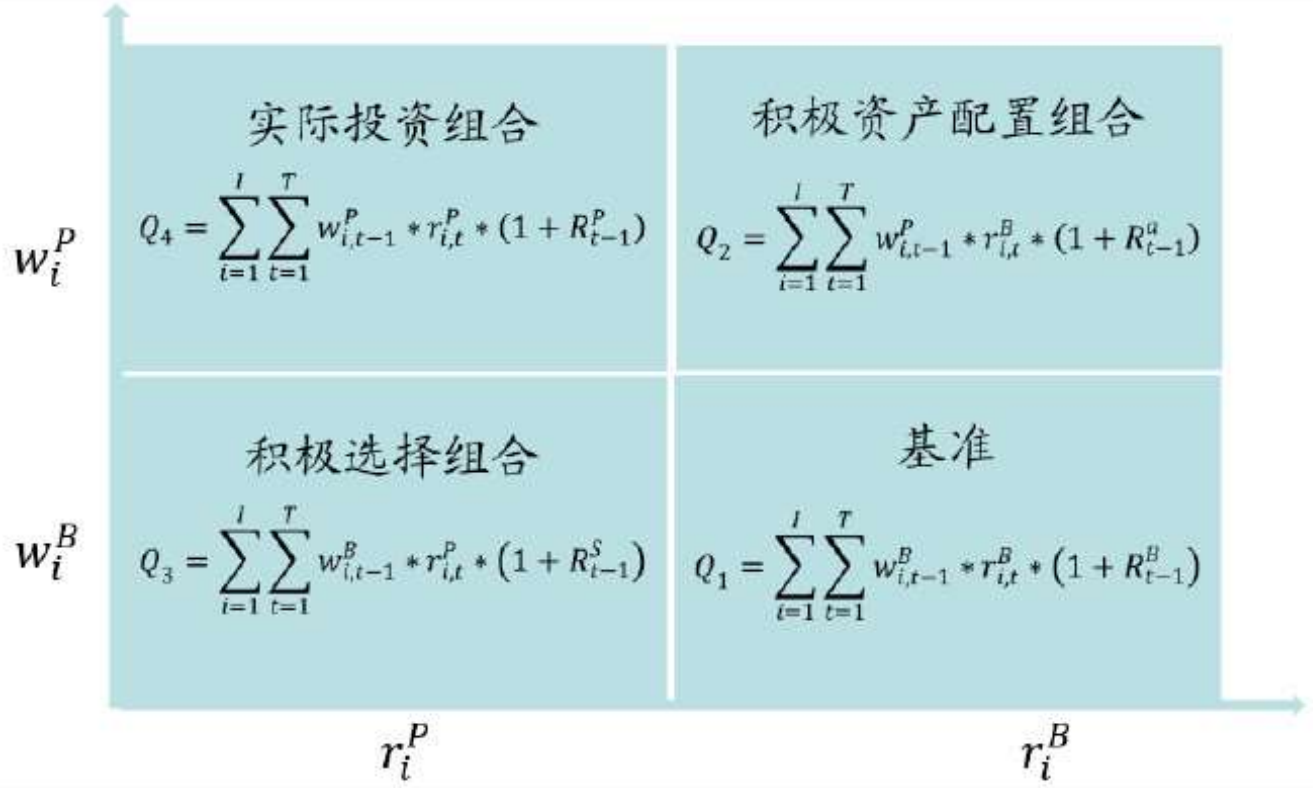
从单期归因到多期归因需要考虑再投资的影响。组合 T 期的累计收益并不是简单的单期收益加总，而是经过前期累计收益放缩后的单期收益之和，即

$$R_T^P = \sum_{t=1}^{T} (1 + R_{t-1}^P) \times r_{i,t}^P$$

其中，$r_{i,t}^P$ 为组合在第 t 个子期间的收益。因此，业绩归因也并不是将每个子时段的归因效应简单加总，而是需要通过适当放缩来达到目的。

AKH算法将各个概念组合的累计收益率作为各自绩效的放缩因子

图6 Brinson 多期归因模型的 AKH 算法框架



资料来源：海通证券研究所整理

案例分析

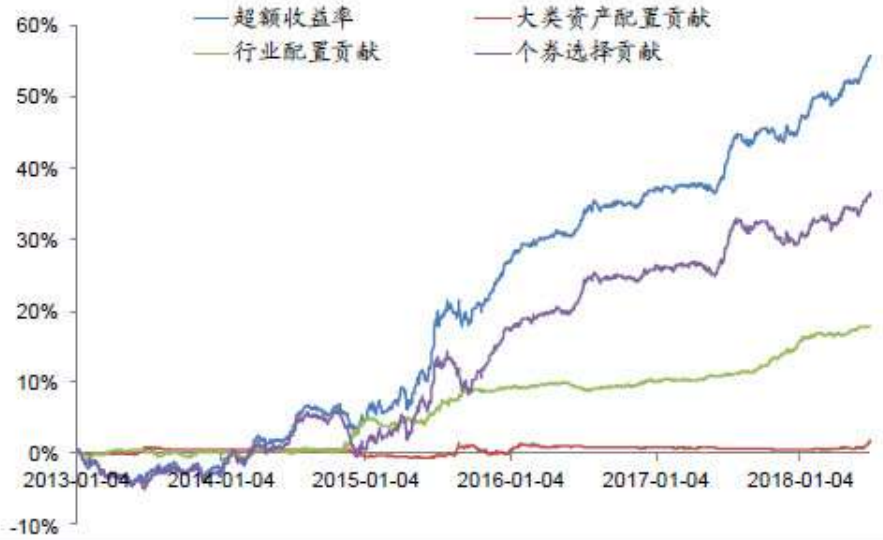
- 对于基金 A，我们分别根据季报（仓位）、半年报（行业和个股，其中行业采用中信一级分类）披露的持仓数据，计算其资产配置和行业配置的比例。并假定相邻报告期之间，持仓比例是均匀变化的。

表 1 基金 A 超额收益的 Brinson 模型分解（2013.01-2018.06）

名称	收益率	基准收益率	超额收益	资产配置贡献	股票超额收益贡献	
					行业配置贡献	个股选择贡献
基金 A	94.33%	38.47%	55.86%	1.51%	17.82%	36.53%

资料来源：Wind，海通证券研究所

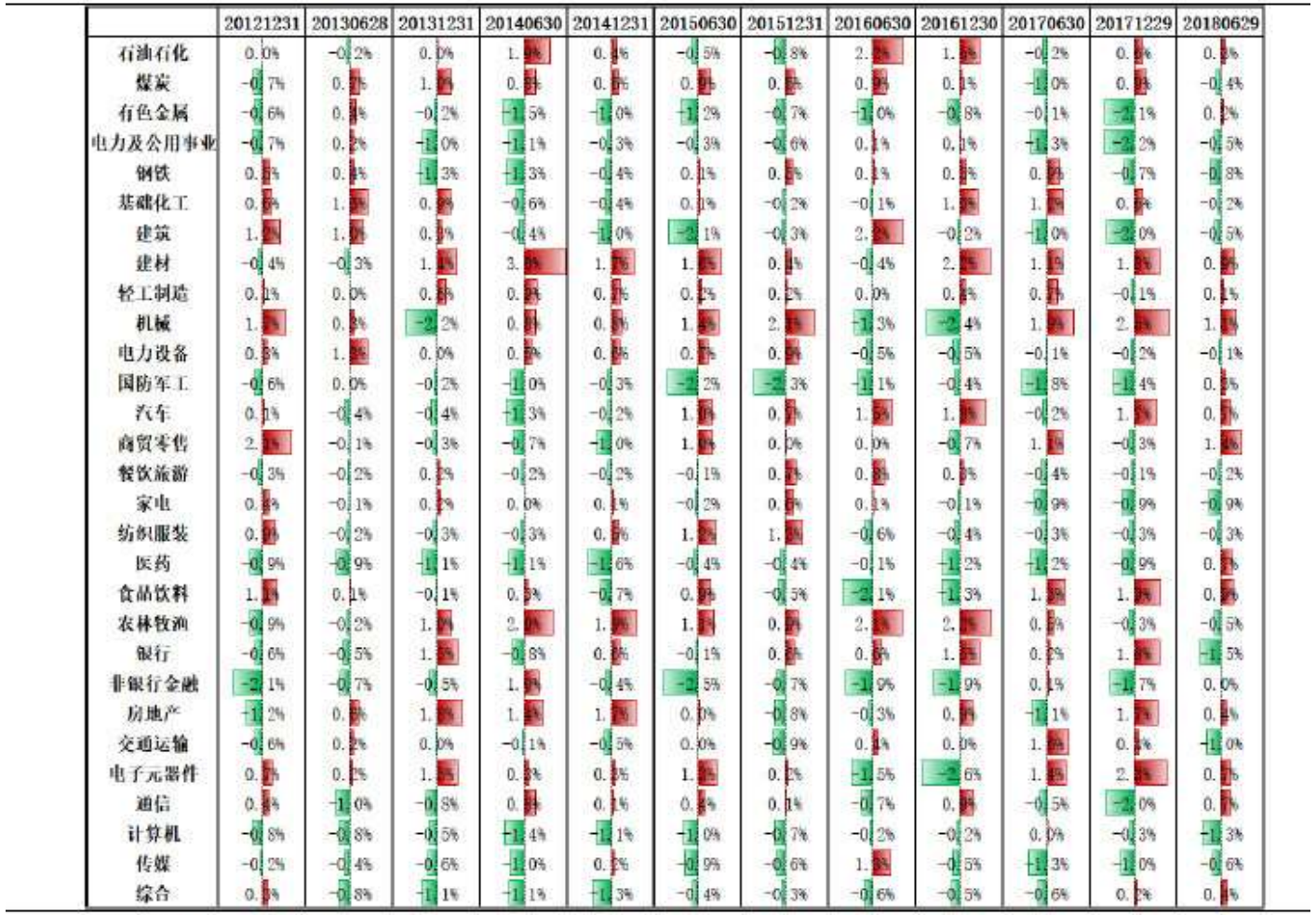
图7 基金 A 超额收益及 Brinson 模型分解后各部分的历史表现



资料来源：Wind，海通证券研究所

该基金的行业配置变化

图8 基金A的行业配置变化（2013.01-2018.06）



资料来源：Wind，海通证券研究所

三、万矿回测平台介绍

万矿回测框架

量化策略步骤

- 回测：将策略的思路进行策略的开发，在历史数据中运行、验证结果

万矿量化平台：https://www.windquant.com/

回测框架

万矿的回测框架使用前需要先导入包：

```
1  from WindAlgo import * #引入回测框架
```

要将一个策略进行回测主要分为两个基本步骤：

- **（1）：通过调用BackTest实例化回测对象**
- **（2）：运行上述的回测对象**

```
1  wa = BackTest(init_func = initialize, handle_data_func=handle_data) #实例化回测对象
2  res = wa.run(show_progress=True)   #运行回测对象
```

回测对象

- **BackTest有两个参数：初始化函数initialize和策略函数handle_data，回测时要把initialize和handle_data两个函数传入该对象，当用户传入参数，实例化BackTest对象之后就可以在handle_data等函数中使用BackTest实例。**

参数	类型	说明
init_func	function	initialize函数
handle_data_func	function	handle_data函数

```
wa = BackTest(init_func = initialize, handle_data_func=handle_data) #实例化回测对象
res = wa.run(show_progress=True)    #运行回测对象
```

完整的策略对象

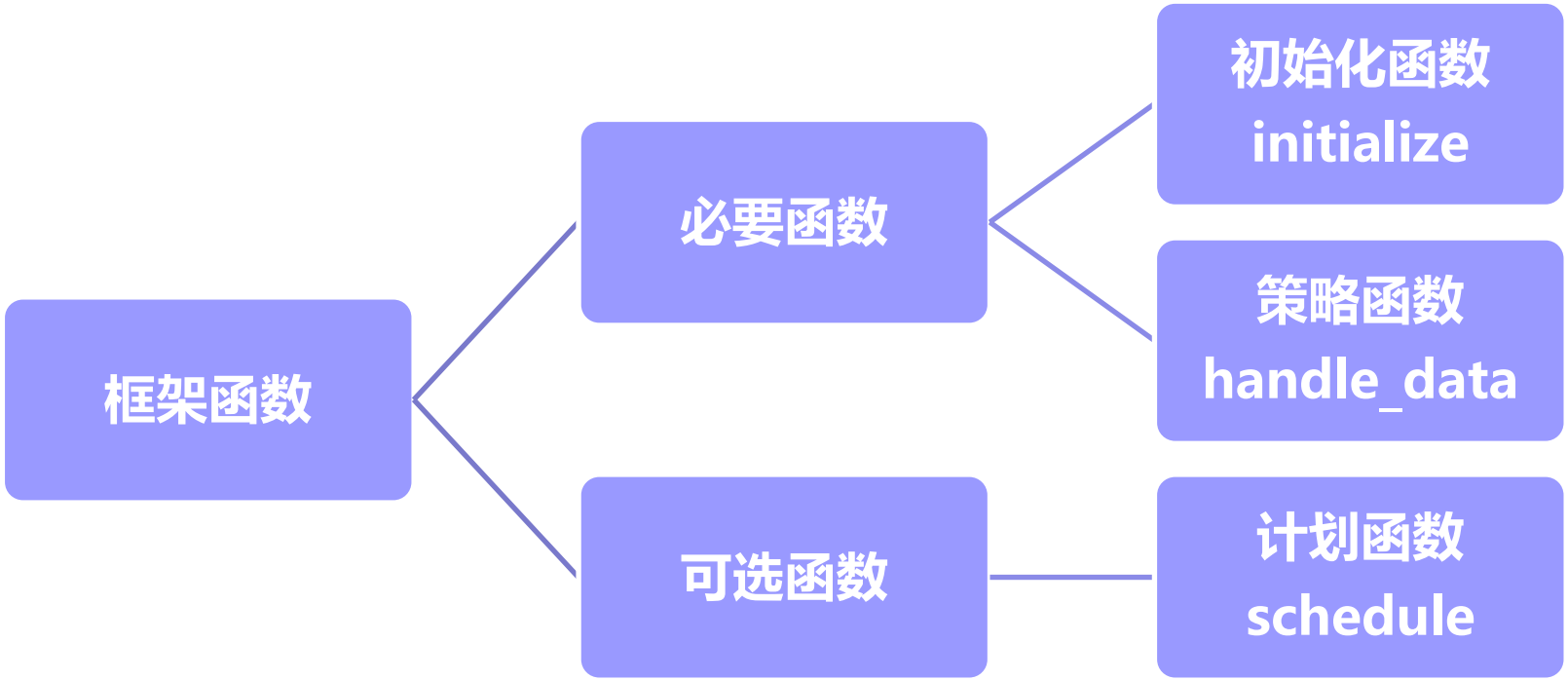
所以要实例化一个完整的策略对象需要如下结构：

```
from WindAlgo import *
def initialize(context):
    # 此处进行策略初始化设置
    pass

def handle_data(bar_datetime, context, bar_data):
    # 此处实现策略逻辑
    pass

wa = BackTest(init_func = initialize, handle_data_func=handle_data)
```

```
wa = BackTest(init_func = initialize, handle_data_func=handle_data) #实例化回测对象
res = wa.run(show_progress=True)   #运行回测对象
```

框架函数

- 框架函数：指的是构成整个策略所需要的结构函数，分为必要函数（策略必备）和可选函数。



框架函数

初始化函数

- **初始化函数initialize(context):**
- **(1)在初始化函数中，用于在handle_data运行之前对策略进行初始化设置，也可在其中自定义一些变量，该函数只在策略初始化时运行一次**
- **(2)该函数用于策略运行前的初始化，用于在各函数之间传递参数。其中context用于设置策略，同时可用于在各函数之间传递数据**

- 参数说明

参数	类型	说明
context	对象	initialize唯一且必须的参数，在策略中可作为全局变量使用

- 返回说明

默认无返回，用户可以根据自己的需要在函数中实现。

初始化函数

初始化函数的一般设置科目如下：

```
def initialize(context):
    context.capital = 1000000              #回测的初始资金
    context.securities = ["600519.SH"] #（1）用于设置回测证券池，否则将无法接收到回测行情（2）change_securities(code_list)

    context.start_date = "20160101"     #回测开始时间，字符串类型
    context.end_date = "20170501"       #回测结束时间，字符串类型
    context.period = 'd'                #策略运行周期，'d' 代表日，'m'代表分钟，默认为 'd'
    #下面设置比较少用，均有默认设置值
    context.benchmark = '000300.SH'   #设置回测基准，默认为沪深300，用于设置策略的业绩基准，可选常见的指数的Wind代码
    context.commission = 0.0003   #浮点数，默认0.0003，用于设置回测的股票交易手续费率。
    context.fee_multi = 3          # 浮点或者整形数字，默认3，用于设置期货手续费倍率。
    context.slippage_setting=SlippageSetting(stock_slippage_type='byRate',stock_slippage=0.0001) #用于设置滑点
    #以上为initialize函数中参数context自带的的属性，另外自己还可以根据需要自定义新的属性，如下：
    context.feature = 'TURN'           #比如策略中需要用到感兴趣的因子，可以在这里添加，本例中为"TURN" 换手率
```

策略函数

- 策略函数handle_data():
- handle_data(bar_datetime, context, bar_data)，该函数是策略函数，每个回测周期都会调用该函数。

- 函数中有三个参数:bar_datetime,context,bar_data
- (1)bar_datetime:
- 该参数用于传递当前回测周期的时间，日期类型为datetime
- (2)context:
- 用于在各函数之间传递变量,也就是用于接收初始化函数所定义的参数context的属性
- (3)bar_data:
- 用于传递对应回测周期的行情数据，也就是初始化函数中context.securities中的证券行情，根据context.period决定是日线还是分钟线数据

策略函数

- **策略函数，每天/每分钟(根据context.period是日线还是分钟线定)回测框架都会调用该函数，并将对应的时间和行情数据传递给该函数。**
- **可在该函数中实现你的投资策略。例如：获取数据、计算因子或者均线、确定要买卖的股票、调用下单函数买卖股票**

```
import talib as ta
def handle_data(bar_datetime, context, bar_data):#定义策略函数
    his= wa.history('600519.SH',30) #使用history函数获取近期历史行情
    ma5=ta.MA(np.array(his.get_field('close')), timeperiod=5, matype=0) #使用talib技术指标库计算5日均线
    ma20=ta.MA(np.array(his.get_field('close')), timeperiod=20, matype=0)
    position=wa.query_position() #查询当前持仓
    if('600519.SH' in position.get_field('code')): #如果当前持仓中有600519.SH
        if(ma5[-1]<ma20[-1]): #如果5日均线下穿20日均线，则卖出所有股票
            wa.batch_order.sell_all()
    else:
        if('600519.SH' not in position.get_field('code')): #如果当前持仓中没有600519.SH
            if(ma5[-1]>ma20[-1]): #如果5日均线上穿20日均线，则买入股票
                res = wa.order('600519.SH', 4000, 'buy')
```

一个简单的策略示例

运行 | 中断 | 保存 | 折叠 | 块复制 | 块剪切 | 删除 | API函数 ▾ | 代码 | 文本

```
from WindAlgo import * #引入回测框架
import talib as ta

def initialize(context):#定义初始化函数
    context.capital = 1000000 #回测的初始资金
    context.securities = ["000001.SZ","600519.SH"] #回测标的
    context.start_date = "20150101" #回测开始时间
    context.end_date = "20170501" #回测结束时间
    context.period = 'd' #策略运行周期, 'd' 代表日, 'm'代表分钟

def handle_data(bar_datetime, context, bar_data):#定义策略函数
    his= bkt.history('600519.SH',30) #使用history函数获取近期历史行情
    ma5=ta.MA(np.array(his.get_field('close')), timeperiod=5, matype=0) #使用talib技术指标库计算5日均线
    ma20=ta.MA(np.array(his.get_field('close')), timeperiod=20, matype=0)
    position=bkt.query_position() #查询当前持仓
    if('600519.SH' in position.get_field('code')): #如果当前持仓中有600519.SH
        if(ma5[-1]<ma20[-1]): #如果5日均线下穿20日均线，则卖出所有股票
            bkt.batch_order.sell_all()
    else:
        if('600519.SH' not in position.get_field('code')): #如果当前持仓中没有600519.SH
            if(ma5[-1]>ma20[-1]): #如果5日均线上穿20日均线，则买入股票
                res = bkt.order('600519.SH', 4000, 'buy')

bkt = BackTest(init_func = initialize, handle_data_func=handle_data) #实例化回测对象
res = bkt.run(show_progress=True) #调用run()函数开始回测,show_progress可用于指定是否显示回测净值曲线图
```

策略示例回测结果


绝对收益：71.41%
相对收益：74.07%
年化收益：26.93%
最大回撤：-18.53%
Alpha：24.82%
Beta：0.213
Sharpe：1.179
1.8
1.6
1.4
1.2
1
0.8
2015-04-20
2015-07-29
2015-11-13
2016-02-29
2016-06-08
2016-09-20
2017-01-04
2017-04-21
策略净值
业绩基准

功能函数

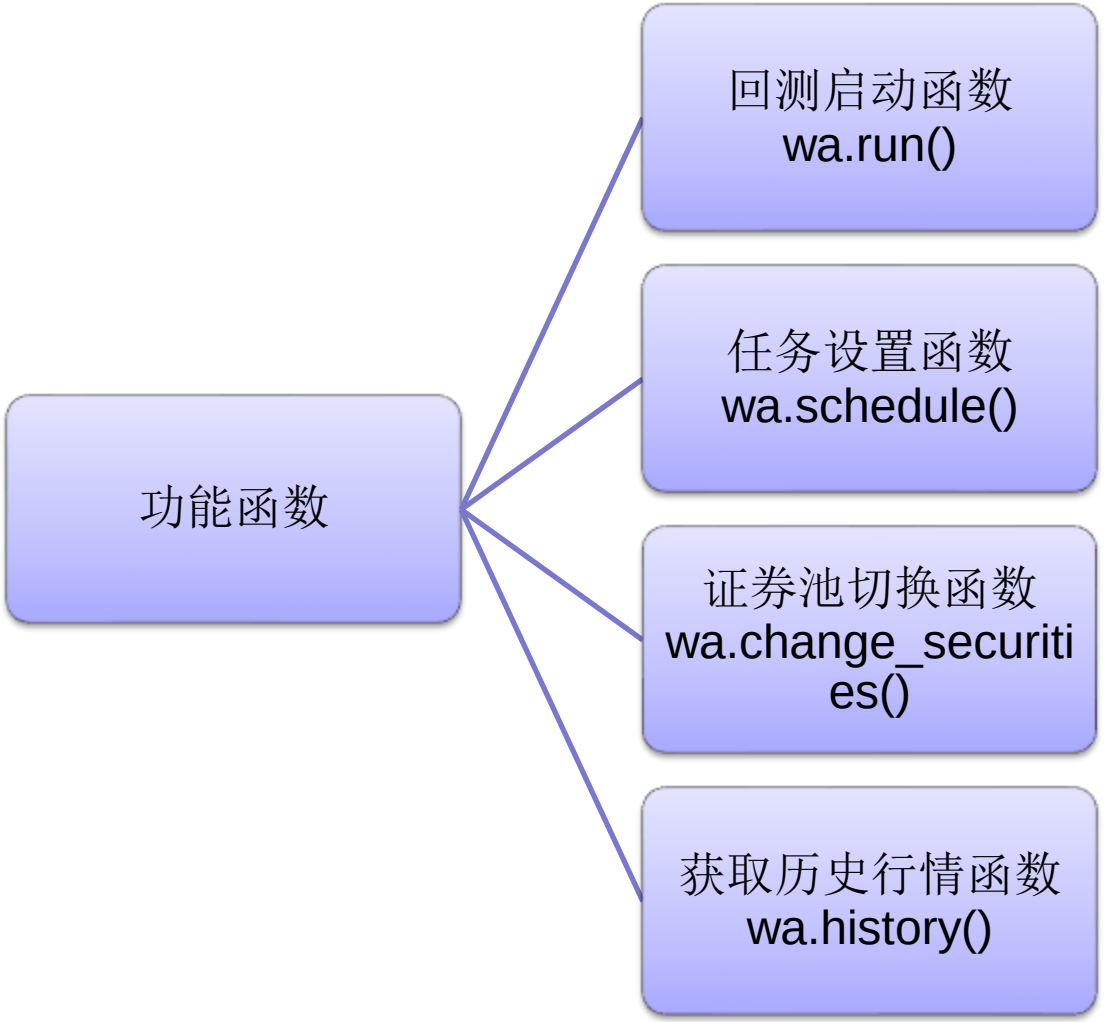
BackTest子函数

- 严格来讲，除了框架函数外，万矿回测框架的其他内置函数均为BactTest的子函数，具体可分为如下四类：（1）功能函数；（2）交易函数；（3）查询函数；（4）回测结果统计函数
- 所以上述函数在调用前均需要先实例化BackTest对象，然后再调用。

- 例：我们实例化一个BackTest对象，命名为wa，以后可以使用wa在任何函数中调用该实例，例如wa.run()启动回测，wa.history()获取历史行情，wa.batch_order.change_to()进行批量下单等。

功能函数

功能函数有四个：



功能函数

- **run(show_process=True)**，该函数用于启动回测；
- run函数有唯一参数show_process，为bool类型，设置是否显示动态生成回测曲线；
- 代码用法：res = wa.run(show_progress=True)

- 证券池切换函数：**change_securities(code_list)**
- 该函数用于在回测过程中改变证券池(即context.securities)，在多因子选股等应用场景中，

每一期选择的股票都不一样，可使用该函数来将每一期的选股结果加入到证券池中，以便能正常接收行情和下单交易。

- 该函数有唯一参数code_list，为list类型，如['000001.SZ' ,' IF.CFE']，用于设置证券池中包含哪些证券。
- 代码用法：is_success, bar_data=wa.change_securities(code_list)

功能函数

- 计划任务设置函数**schedule(schedule_func, period, offset = 0, market='SH')**
- (1)该函数用于设置回测过程中需要运行的计划任务。借助该函数可实现定期调仓、定期改变股票池等操作。
- (2)执行时间比较灵活，该函数会在handle_data函数之前调用。

- 参数介绍：
- （1）schedule_func：用于指定被调用的用户自定义函数,自定义函数必读包括如下三个参数（bar_datetime，context,bar_data），
- 作用相当于策略函数handle_data()。可以理解为执行时间比较灵活的策略函数。
- （2）period：用于指定计划任务执行周期
- 用于指定函数被调用的周期，可为’q’，‘m’，‘w’，‘d’，‘h’。分别表示季度，月，周，日和小时，该参数与offset参数配合使用
- （3）offset时间偏移参数 （可选参数，默认为0）
- 该参数用于指定时间偏移量，与period参数配合使用，用于指定在period基础之上，进一步指定计划任务具体运行的时间规则
- （4）market 日期规则对应的交易所（可选参数，默认为’SH’）
- 因每个交易所的交易日有可能不一样，因此需用matket指定period和offset对应的交易所

策略可写在计划任务中

```python
from WindPy import *
from datetime import *
import pandas as pd
w.start(show_welcome=False)
from WindAlgo import * #引入回测框架

list1 = pd.read_csv("data/pool/2009pool.csv")
pool1 = list(list1[:round(len(list1)/10)]["stock"])

def initialize(context):#定义初始化函数
    context.capital = 1000000 #回测的初始资金
    context.securities = pool1 #回测标的
    context.start_date = "20100501" #回测开始时间
    context.end_date = "20180401" #回测结束时间
    context.period = 'd' #策略运行周期, 'd' 代表日, 'm'代表分钟
    context.benchmark = '000300.SH'  #设置回测基准为沪深300

def handle_data(bar_datetime, context, bar_data):#定义策略函数
    pass

def my_schedule1(bar_datetime, context, bar_data):                # 注意：schedule函数里不能加入新的参数

    #global code_list
    bar_datetime_month = bar_datetime.month             #设置时间
    bar_datetime_year = str(bar_datetime.year-1)
    if bar_datetime_month==5:
        list2 = pd.read_csv("data/pool/"+bar_datetime_year+"pool.csv")
        code_list = list(list2[:round(len(list2)/10)]["stock"])
        wa.change_securities(code_list)
        context.securities = code_list     #改变证券池
        list_sell = list(wa.query_position().get_field('code'))      #此处可改进 有些下个月打算买的股票可以不用卖出只需要调仓
        for code in list_sell:
            volumn = wa.query_position()[code]['volume']     #找到每个code 的 持仓量
            res = wa.order(code,volumn,'sell',price='close', volume_check=False)  # 卖出上一个月初 买入的所有的股票
    ## '卖出上个月所有仓位'  为本月的建仓做准备

def my_schedule2(bar_datetime, context,bar_data):
    bar_datetime_month = bar_datetime.month             #设置时间
    if bar_datetime_month==5:
        buy_code_list=list(set(context.securities)-(set(context.securities)-set(list(bar_data.get_field('code')))))  # 
        for code in buy_code_list:
            res = wa.order_percent(code,1/len(buy_code_list),'buy',price='close', volume_check=False)
            #对最终选择出来的股票建仓 每个股票仓位相同     '本月建仓完毕'



wa = BackTest(init_func = initialize, handle_data_func=handle_data)    #实例化回测对象
wa.schedule(my_schedule1, "m", 0)          #    m表示在每个月执行一次策略 0表示偏移   表示月初第一个交易日往后0天
wa.schedule(my_schedule2, "m", 1)          #在月初第2个交易日进行交易
res = wa.run(show_progress=True)           #调用run()函数开始回测,show_progress可用于指定是否显示回测净值曲线图
nav_df = wa.summary('position')                    #获取回测结果   回测周期内每一天的组合净值
```

功能函数

- 获取历史行情函数：**history(code, bar_count, period = None, bar_datetime = None)**
- 该函数可用于在回测过程中获取历史行情数据
- 参数说明：

（1）code: 字符串类型，指定需要获取行情的标的代码，例如'000001.SZ'
（2）bar_count: 非负整数型，用于指定获取多少个行情周期的数据，例如要获取最近30个回测周期的行情数据，则设置bar_count为30
（3）period：用于指定周期，默认为回测设置中的context.period
（4）bar_datetime:设置为None时，系统默认以回测设置中的 context.period 为周期，或者设置为'd'(日线)，'m'（分钟线）

用法示例：his= bkt.history('600519.SH',30)

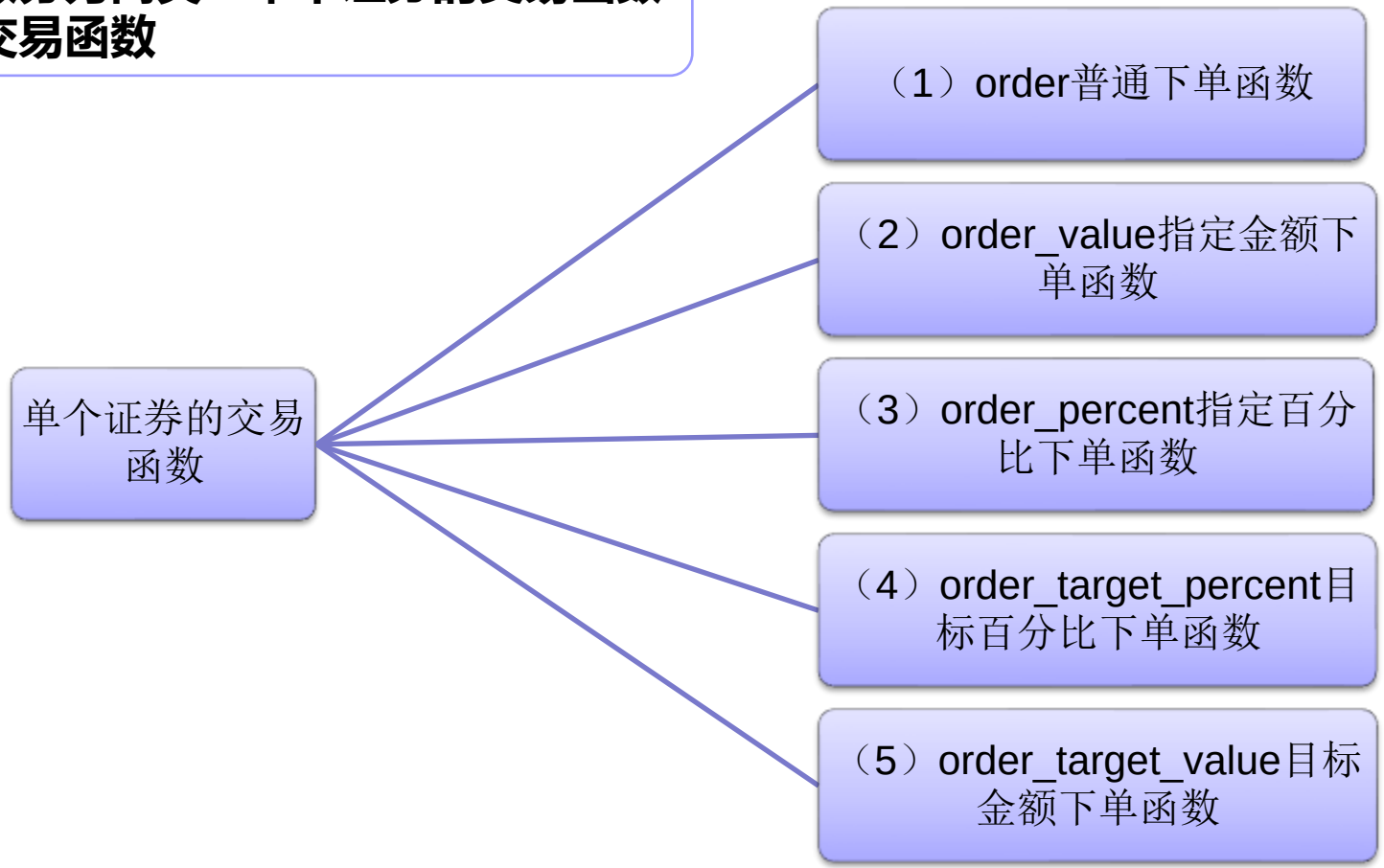
一般在策略函数handle_data中使用：

```
11  def handle_data(bar_datetime, context, bar_data):#定义策略函数
12      his= bkt.history('600519.SH',30) #使用history函数获取近期历史行情
13      ma5=ta.MA(np.array(his.get_field('close')), timeperiod=5, matype=0) #使用talib技术指标库计算5日均线
14      ma20=ta.MA(np.array(his.get_field('close')), timeperiod=20, matype=0)
15      position=bkt.query_position() #查询当前持仓
16      if('600519.SH' in position.get_field('code')): #如果当前持仓中有600519.SH
17          if(ma5[-1]<ma20[-1]): #如果5日均线下穿20日均线，则卖出所有股票
18              bkt.batch_order.sell_all()
19      else:
20          if('600519.SH' not in position.get_field('code')): #如果当前持仓中没有600519.SH
21              if(ma5[-1]>ma20[-1]): #如果5日均线上穿20日均线，则买入股票
22                  res = bkt.order('600519.SH', 4000, 'buy')
```

交易函数

交易函数

- **交易函数分为两类：单个证券的交易函数和批量交易函数**



单个证券交易

- （1）order(code,volume,trade_side,price='close', volume_check=False)
- 买卖时主要根据交易股数进行交易
- code：字符串类型，如'000001.SZ'，单个股票或者期货代码
- volume：非负整数型，交易股票/期货数量
- trade_side：字符串，买卖方向，可取参数如下：
- ‘buy’ :买/开多单
- ‘short’ :开空单（期货）
- ‘cover’ :平空单（期货）
- ‘sell’ :卖/平多单
- price：字符串类型或者浮点数，默认为'close',用于设置回测交易时，使用的委托价格
- volume_check：bool类型，默认为Flase，用于设置下单时，是否做成交量检查。
- True:检查委托的量是否超过当天对应股票或者期货的成交总量，如果超过则无法成交。False:不检查当日成交总量，直接成交。

单个证券交易

- （2）order_value(code,value,trade_side,price='close', volume_check=False)
- 该函数用于买卖一定金额的股票或者期货，通过value指定金额。
- value：浮点数或者整数，用于指定买卖的金额
- 其他参数同order

- （3）order_percent(code,percent,trade_side,price='close', volume_check=False)
- 该函数用于买卖一定百分比资产的股票或者期货，买卖的金额等于前一日总资产乘上percent。
- percent：0~1之间的浮点数，用于指定买卖的金额，买卖的金额等于前一日总资产乘上percent
- 其他参数同order

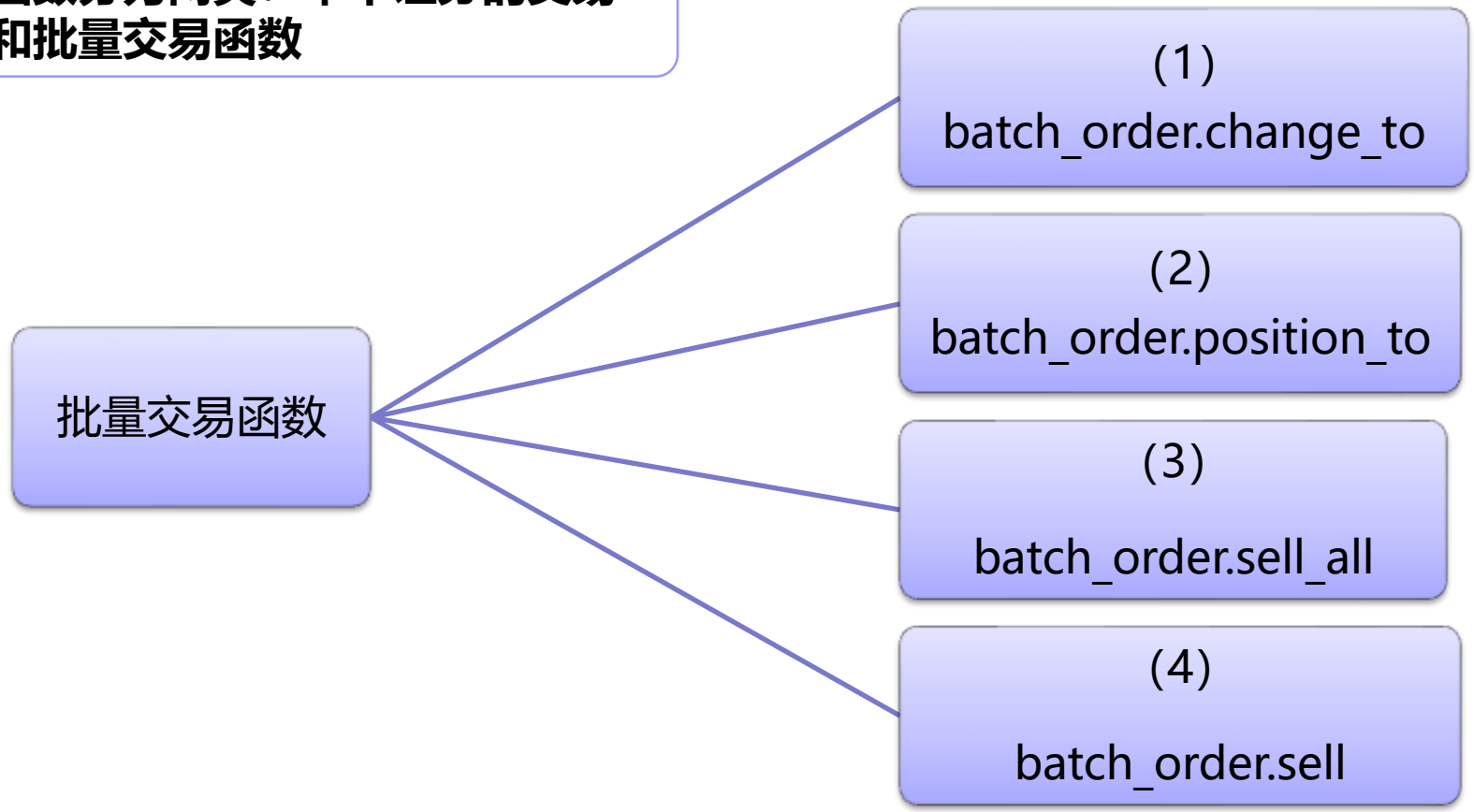
单个证券交易

- （4）order_target_value(code,target_value,price='close', volume_check=False)
- 该函数用于按持仓目标买卖股票或者期货，使买卖操作完成后，对应股票或者期货的持仓市值等于target_value。
- target_value：大于0的整数或者浮点数，用于指定目标持仓市值，买卖操作完成后，对应股票或者期货的持仓市值等于target_value。
- 其他参数同order

- （5）order_target_percent(code,target_percent,price='close', volume_check=False)
- 该函数用于按持仓目标买卖股票或者期货，使买卖操作完成后，对应股票或者期货的持仓市值等于前一日总市值乘上target_percent。
- target_percent：0~1之间的浮点数，用于指定目标持仓市值，使买卖操作完成后，对应股票或者期货的持仓市值等于前一日总市值乘上target_percent。
- 其他参数同order

交易函数

- **交易函数分为两类：单个证券的交易函数和批量交易函数**



批量交易

- （1）batch_order.change_to(code_list,position,price='close', volume_check=False, no_quotation='error')
- 该函数用于股票批量调仓（只支持股票），使得调仓后，账户中的持仓变为code_list里面的股票，资金平均分配，
- 所有持仓的总市值等于前一日总资产乘上position，该函数常用于选股回测。
- code_list：字符串list，如[’000001.SZ’，’600000.SH’]，指定调仓后的持仓股票
- position：0~1之间的浮点数，用于指定持仓仓位，调仓后，所有持仓的总市值等于前一日总资产乘上position。
- price：同order函数
- volume_check：同order函数
- no_quotation: 对于停牌等原因导致当日股票无行情的处理方式，可选'error'或者'skip'，
- 如果为'error' 即返回报错信息，如果为'skip'，则会跳过停牌股票，通过买卖剩余股票来尽量逼近预定的仓位目标，但无法保证一定达到该目标，这个参数在持股数量较大的回测中较实用。

批量交易

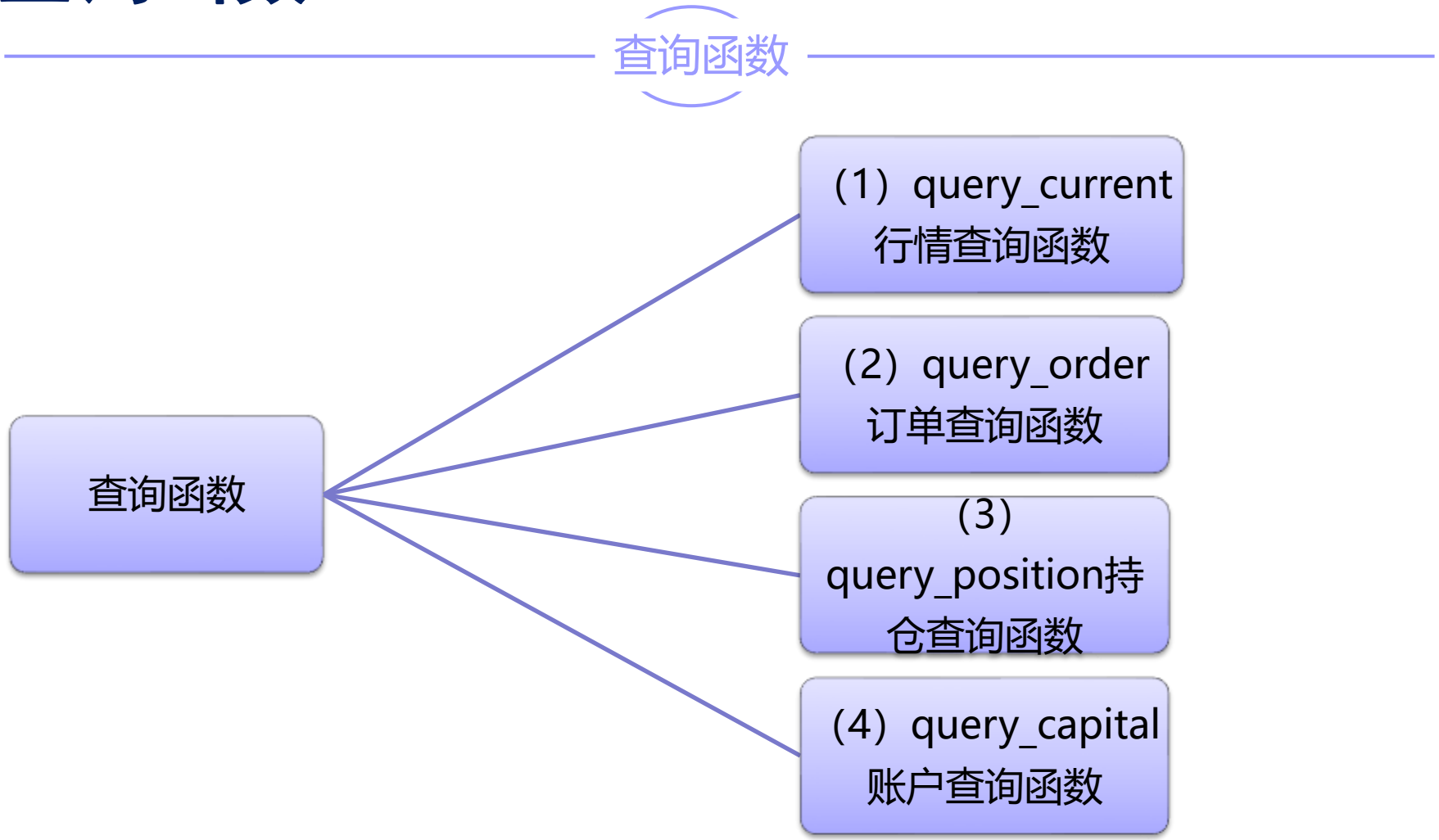
- （2）batch_order.position_to(position,price='close', volume_check=False, no_quotation='error')
- 该函数用于调整仓位，在不改变持仓股票的情况下，等比例调整股票的持仓仓位，使得调整后，持仓总市值等于前一日总资产乘上position。
- position：0~1之间的浮点数，用于指定持仓仓位，调仓后，所有持仓的总市值等于前一日总资产乘上position。
- 其他参数同order

- （3）batch_order.sell_all(price='close', volume_check=False, no_quotation='error')
- 该函数用于卖掉账户中的所有持仓。
- 其他参数同order

批量交易

- （4）batch_order.sell (code_list,price='close', volume_check=False, no_quotation='error')
- 该函数用于卖掉持仓中的一批股票。
- code_list：字符串list，如['000001.SZ','600000.SH']，指定需要卖掉的股票
- 其他参数同order

查询函数


查询函数
查询函数
（1）query_current
行情查询函数
（2）query_order
订单查询函数
（3）
query_position持
仓查询函数
（4）query_capital
账户查询函数

行情查询

- 行情查询函数：**query_current(code_list, field_list)**，用于查询当前回测周期的行情数据。

- 参数说明：
- code_list：字符串list，如['000001.SZ','600000.SH']，需要查询的行情的证券代码。
- field_list：字符串list，可包含字符串如下：'pre_close', 'open', 'high', 'low', 'close', 'volume', 'pct_chg', 'vwap'，其中‘vwap’只对日级别回测有效

- 返回值：

包含相应行情数据的WindFrame

- 用法：

```
sq = wa.query_current("600036.SH", "close")
```

订单查询

- 订单查询函数：**query_order(order_id)**，用于订单信息。

- 参数说明：
- order_id：需要查询的订单的ID。

- 返回值：

None或者包含订单信息的WindFrame

返回的 **WindFrame数据格式如下：**

	code	order_time	trade_side	volume	err_code	price	amount	err_msg
0	600036.SH	2015-10-08 00:00:00	'buy'	500	0	17.9147	8957.35	'OK'

持仓查询

- 持仓查询函数：**query_position()**，用于查询当前持仓。

- 参数说明：
  该函数无参数

- 返回值：
  如果当前无持仓返回None，如果有持仓返回包含持仓信息的WindFrame数据

返回的 **WindFrame数据格式如下：**

	code	cost_price	revenue	amount	volume	side
0	600036.SH	17.9993	-8.09967	27116.5	1500	'long'

账户查询

- 账户查询函数：**query_capital（）**，用户查询当前资金账户信息

- 参数说明：
该函数无参数

- 返回值：
返回包含当前资金账户信息的WindFrame数据

返回的 WindFrame数据格式如下：

available_fund	holding_value	margin	total_asset
99643.0912	919124.73	0.0	1.018768e+06

available_fund:可用资金
holding_value:持仓市值
margin:保证金
total_asset:总资产

回测结果汇总函数

交易汇总查询

- 回测结果汇总函数：
  summary(summary_type,start_date=None,end_date=None)，用于获取回测结果，该函数在回测结束后才能使用。

- 参数说明：
  (1)summary_type 字符串，指定获取的回测结果类型
可选参数有：
  'result': 回测概览(暂不支持日期参数)
  'nav':获取对应区间回测净值
  'trade'获取回测交易明细
  'position':获取历史持仓记录
  'monthly_profit':月度盈亏
  'position_rate': 每日仓位
  'stat_month':策略表现月度统计
  'stat_quarter':策略表现季度统计
  'stat_year':策略表现年度统计

毛利率下降

- 参数说明：

(2)start_date： 指定查询的开始日期，默认为回测开始日期
(3)end_date：指定查询的结束日期，默认为回测结束日期

- 返回值：

包含结果数据的WindFrame

代码用法示例：

```
nav_df=wa.summary('nav') #获取回测结果
```

本章总结

- 量化风控是量化投资和高频交易的核心，如何设定相关指标是关键
- 评估量化投资交易策略是重要环节，其中回测是常用手段
- 业绩归因能够回答你的利润来源是选股、择时或是运气
- 量化交易代码需要的是耐心，所以希望大家未来都能写出属于自己得财富代码