当代量化交易系统中期报告

2000015476-杨明宸

2023 年 11 月 17 日

1 系统概述

本系统预期能够为用户提供单因子回测和多因子回测功能，计划将主体回测代码封装到一个 BackTest 类中，使用本系统的用户将主要使用 single_signal_fit 和 multiple_signals_fit 进行回测。这两个方法都需要用户准备好一个或一系列信号作为系统的输入，合法的输入对象要求是一个 dataframe 对象，该 dataframe 对象要求有三个 columns:datetime,stockid,signal. 这意味着无论用户要回测的是时序信号还是截面信号，也无论用户使用了量价数据还是财报数据，本系统都要求用户首先计算好信号并标准化为本系统接受的输入格式。在进行 fit 后，用户可以使用 BackTest 类封装的若干属性和方法得到自己需要的输出。

同时，我看到作为本次课题的基本要求，似乎要求用户输入一个函数就能直接回测。因此我会额外提供一个 API，Reveral(n: int, stock_list:list) 用于帮助用户计算 n 日反转策略并输出主体回测 API 能够接受的合法输入。但我仍需指出，我个人认为让用户传入一个函数做输入不是最佳选择，我个人倾向于系统的输入是标准化的，正如上一段所提及的，使用一个标准的 dataframe 输入，那么不管用户到底要回测截面策略还是时序策略，我的系统都能实现。看上去，让用户自己先整理出标准输入是一件麻烦的事，其实这反而大大丰富了用户自定义信号的自由度，也大大提高了系统的普适性。

2 基本功能

目前预计系统能提供的属性和 API 如下所示：(本部分介绍单因子回测模式下的功能，多因子回测模式将在特色功能部分介绍)

- 属性
  - is_fitted: 实例是否已经拟合了数据，方便用户利用这个属性做 if 分支避免报
  - real_ret：所有输入的信号产生的总的实际收益率
  - exc_ret：所有输入的信号产生的总的超额收益率（基准为全市场收益率）
  - ann_real_ret：将 real_ret 年化
  - ann_exc_ret：将 exc_ret 年化
  - real_vol：计算 real_ret 的年化波动率
  - exc_vol：计算 exc_ret 的年化波动率
  - drawdown：计算整个回测期间的最大回撤
  - Sharpe：计算整个回测期间的实际回报夏普比率

  - exc_Sharpe：计算整个回测期间的超额回报夏普比率
  - R_squared_0：计算整个回测期间信号和真实回报率的 R 方
  - R_squared_1：计算整个回测期间信号和去市场均值超额回报的 R 方

- 方法

  - single_signal_fit(signal_df: DataFrame): 拟和数据并计算上述提到的所有属性，成功调用该函数后，is_fitted 将被赋值为 True。
  - show(excess= False, benchmark= True)：画出 PnL 图并标注必要参数（如年化收益率和夏普比率），当参数 excess 设置为 True 时，将画出超额（以市场回报为基准）PnL 图；当参数 excess 设置为 False 时，将画出原始 PnL 图。当参数 benchmark 为 True 时，将同时画出市场回报曲线作为参考。但是如果 excess 设置为 True，那么 benchmark 参数会被忽略。

3 特色功能及设计动机

在本课题要求实现的基础功能之上，我预期实现如下进阶功能：

- 多空收益 PnL：设计这个功能的动机在于 A 股市场策略往往呈现出空头赚钱容易，多头困难的特点，很多看上去有效的策略其实并不能实盘赚钱，很大一个原因就是多头表现不佳。因此，识别收益来源是多头还是空头颇有重要性。

  BackTest 类下将设计一个方法：LnS_PnL_show(groups: int, exc = True)

  - groups：int 型，要求分多少组
  - exc: 布尔型，如果为 True，返回值和图表都是全市场基准下的超额 R 方和 PnL，否则返回真实值。
  - 返回值：分组 R 方
  - 方法调用过程中会画出分 groups 组后，每组的 PnL 曲线并展示。

  备注：分组方法是根据 signal 大小等分为 groups 组，每组进行计算

- 指增 PnL：设计这个功能的动机在于很多量化策略其实是指数增强策略，那么指增策略在做空上就有限制：如果出现负信号的股票在指数内，那做空没有限制，如果出现负信号的股票在指数外，那就没法做空。指增 PnL 显然是比普通的 PnL 更符合研究者需求的。（这种视角下，普通的 PnL 其实就是空气指增 PnL）。

  BackTest 类下将设计一个方法：AIx_PnL_show(index = 'all'，stocks_list= None, long_limit= False, exc = True)

  - index：字符串类型，在哪支指数上做增强，一些参考是'00905','000300','000852'。而默认值'all' 将画出三个指数上分别的增强结果，放于一张 figure 的三个 ax 上，便于研究者对比不同指数下策略的效果差异。
  - stocks_list：如果这个参数非 None，将忽略 index 参数，改用提供的股票 id 列表作为 benchmark。（这主要是因为有的研究者在研究小市值策略等自定义风格的策略，允许自定义股票池是有帮助的）

  - long_limit: 布尔型参数，如果为 True，只能在股票池里选多头，如果为 False，可以全市场选多头，（但是空头总是有限制的）
  - exc: 布尔型，默认为 True，返回 index 基准下的超额 R 方和 PnL，否则返回真实值
  - 返回值：R 方或 R 方列表（index 参数为'all' 时）
  - 方法调用过程中会画出 PnL 并展示。

此外，单因子回测往往只是策略研发中的第一步，一个更主要的问题是在研究员已经得到大量有效因子后，如何确定最终的策略（即在各个因子之间的权重分配）。一般的量化公司会使用 LightGBM 或者 Lasso 等模型确定权重，但是即便如此，在上模型之前，大部分人还是会看一看各因子之间相关系数，先手动筛选一下。本系统为方便研究员观察多因子之间的结果特别设计了多因子回测模式，这种模式下的标准输入同样是一个 dataframe（columns:[datetime,stockid,signal_1,signal_2,.....]）：

- 属性：
  - R_squared: 整个策略最后的 R 方
  - R_squared_list: 一个列表，降序排列各因子的 R 方（示例：[(signal_1: 0.001, signal_2: 0.0005...)]）
  - reference_weight: 一个列表，通过验证集优化后给出的推荐权重（详见“代码模块组成-多因子优化模块”查看具体是如何进行优化的）
  - corr_matrix: 一个 dataframe，存储了各因子在验证集中的相关矩阵
- 方法
  - multiple_signals_fit(signal_df: DataFrame, valid_ratio: float, optimizer: Optimizer, exc= True): 传入信号数据，规定多少比例的数据用于验证集，选择组合优化使用的优化器后，该方法会拟合数据生成上述属性。exc 选型决定了生成的属性是超额还是原始数据。

    备注：Optimizer 是预期在 BackTest.Optimizer 里写的一些优化器类。
  - show_PnL(exc= True): 展示组合优化后最终策略的 PnL 曲线，可选展示超额 PnL 还是原始 PnL
  - show_corr(): 展示所有因子相关系数的热图，相比调用 corr_matrix 属性查看，可读性更好。

4 代码模块组成

目前本系统计划由如下几个模块组成：

- 主体模块（BackTest）：这个模块主要是封装了 single_signal_fit 和 multiple_signals_fit 这两个 API，是用户最常使用的接口。
- 数据预处理模块（Preprocessing）：利用原始的 HOLC 数据计算每支股票每日的回报，做成标准的 dataframe 格式（columns:[datetime,stockid,signal,volume]，这里的 volume 是流通市值，用于计算 benchmark），以 h5 文件保存到硬盘。
- PnL 计算模块（PnLcalculating）：使用 h5 文件的回报数据和用户传入的信号数据计算每日 PnL，并传输给其他模块。

- BenchMark 计算模块（Benchmark）：由于本系统的进阶功能有自定义股票池做策略研究功能，因此计算 Benchmark 最好自己有一个模块，通过内存中的 h5 文件和用户传入的 stocks_list（如没有，则直接在指数或市场上做），计算回测区间内的基准回报。

- 比较零散的其它属性的计算模块（Utils）：这个模块主要是定义计算 R 方的函数，计算 Sharpee 的函数等等，这些小任务每个都很简单，但数量很多，因此专门开一个模块放置这些函数。

- 多因子优化模块（Optimizers）：这个模块主要在定义多因子回测模式下所要使用到的 Optimizer 对象，本系统计划实现两个优化器：

  一个是简单的线性优化器（Linear_optimizer）其实现方式是，在验证集上用 label 对各个因子做一个线性回归，确定各因子的权重。

  另一个是一个动量优化器（Moment_optimizer），这个优化器只是一个 demo，未必有道理，想法是通过线性模型得到初始权重，然后在回测时间段上每天按照上一天该因子的效果（是赚还是亏）将该因子的权重乘以一个系数（$1 \pm \alpha$），在归一化。

  实际上，真正组合所有因子的时候，肯定是要用到像 LightGBM 这样的模型的，但考虑到调参过程跟具体问题息息相关，很难把调参过程也封装进系统，且这种模型调参似乎也不是一个回测系统应该做的工作。因此我们这里的优化器只是一个便于研究员初步观察策略效果的 demo，是为了研究员做特征筛选工作而准备的，而非专业的组合优化工具。

- 结果呈现模块（Visualization）：这个模块主要是各种 show 方法背后的代码实现部分，包括各个图片的美化等环节，属于一个非核心计算模块，主要用来提高系统的用户友好度。

- 示例函数模块（Demos）：这个模块人如其名，是一些 demo，包括本次课题要求的 Reversal 接口，我也会提供一些其它的基础接口，但必须指出，我并不推荐用户用函数作为输入，因此即使你使用 BackTest.Demos 里的接口，它们也都是返回一个标准 dataframe，然后你必须把这个 dataframe 再扔到 single_signal_fit 或 multiple_signals_fit 去回测。本系统不支持对函数进行回测，可预见的未来内也不会进行这方面的功能拓展。

5 实现途径和系统工作流程

在系统正式工作之前，我事先把提供的原始数据转化为一个 data.h5 文件方便系统使用，这个 data.h5 文件中主要是一个 dataframe，它的 columns 是 [DateTime, StockId, Ret]，同时，如果原始的 HOLC 文件有追加，用户可以通过 BackTest.Preprocessing 提供的接口重新生成 data.h5 文件（详见图一）。

正式使用回测系统时，用户将直接与 BackTest.BackTest 交互，用户应输入一个标准的 DataFrame（信号数据），也可以通过虚线途径跟 BackTest.Demos 交互而不必提供标准的 DataFrame 输入（但不推荐）。BackTest.BackTest 得到输入后会首先判断最早的信号和最晚的信号分别是什么时间，然后将这个信息传给 BackTest.Preprocessing, 要求 BackTest.Preprocessing 从 data.h5 中截取需要的时间段数据到内存中，接着 BackTest.Preprocessing 会将数据传给 BackTest.Utils，BackTest.Benchmark 和 BackTest.PnLCalculate 三个模块，BackTest.Utils 和 BackTest.Benchmark 将按照用户在 BackTest.BackTest 里调用的方法计算各个 Attributes，以及 benchmark 的逐日回报，BackTest.Benchmark 将基准数据传给 BackTest.PnLCalculate，由 BackTest.PnLCalculate 计算逐日 PnL 形成一个 pd.Series 传给 BackTest.Visualization 用于生成图表，当用户调用各种 show 方法时，BackTest.Visualization 会按要求作图展示。（详见图二）

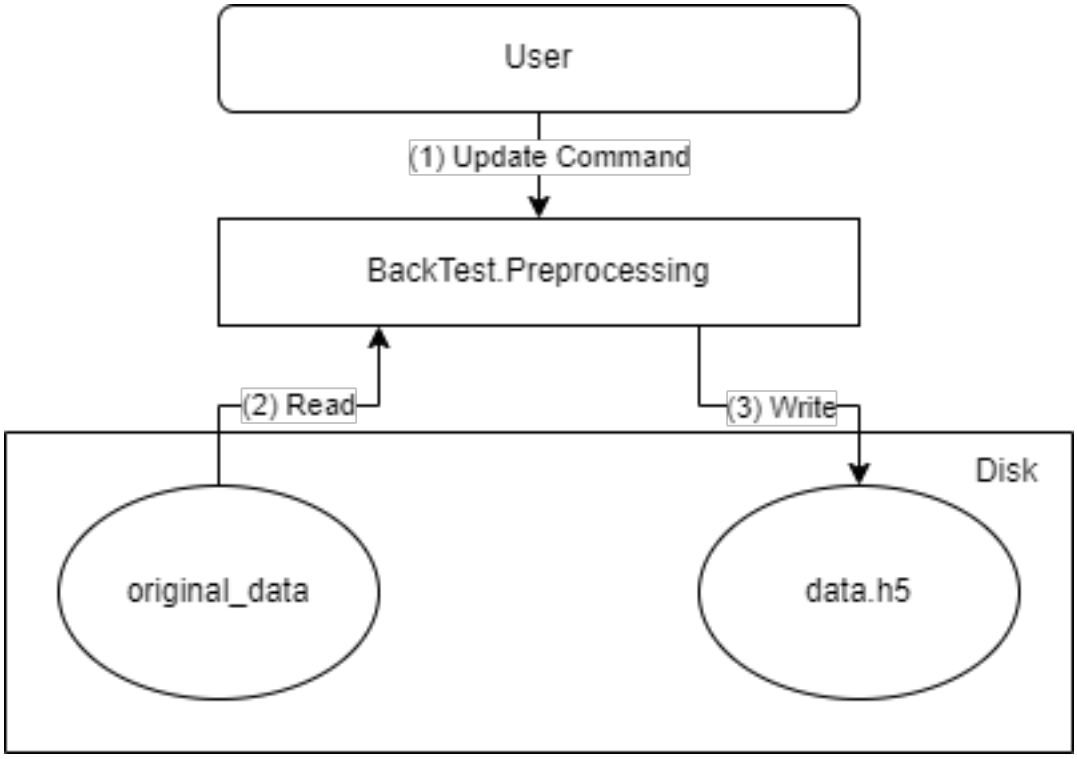

图 1: 数据生成（更新）工作流

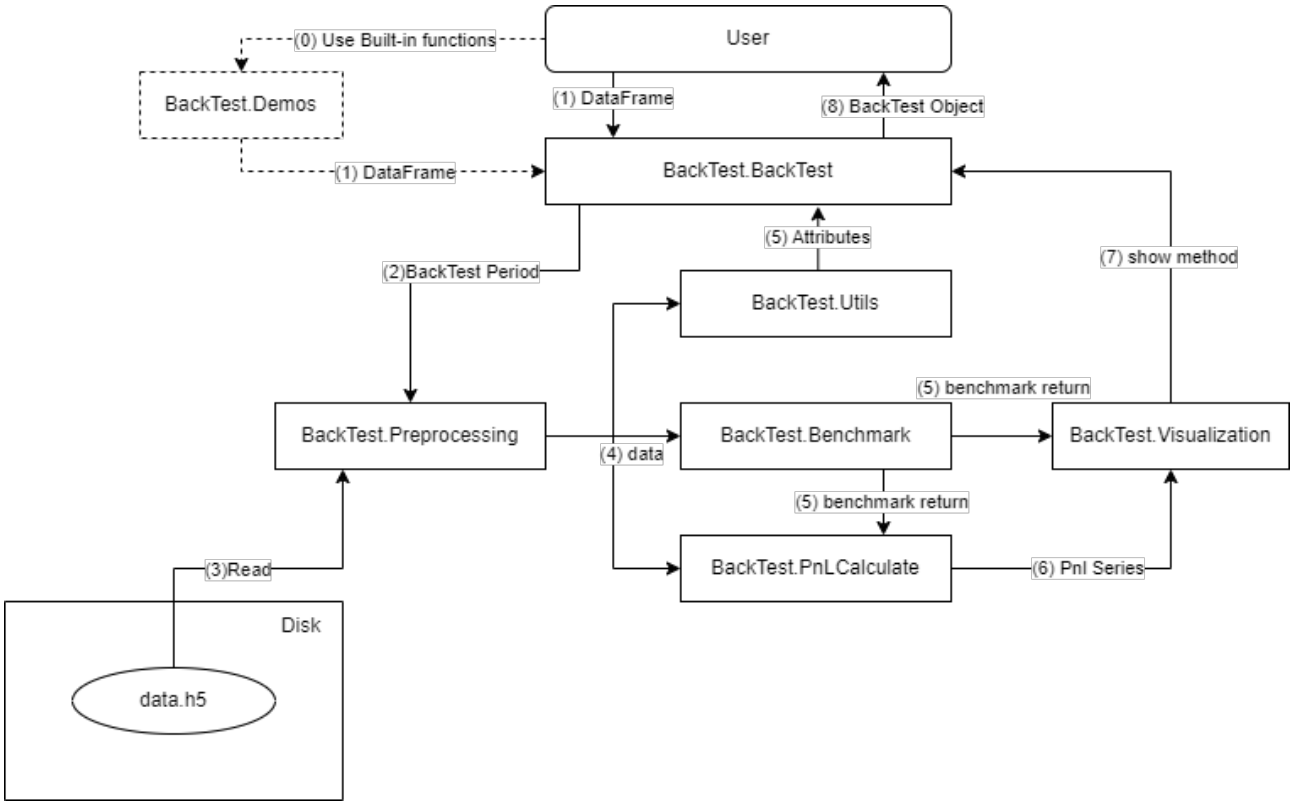

图 2: 回测工作流