交易策略回测框架

2019-4-18

决策人员入门：

Q：请简述框架的作用。

A：通过开发人员书写的一个交易策略函数，框架可以自动进行交易模拟，并记录交易明细和追踪动态价值，最后作出简单的评估。

Q：目前已经有许多在线的回测平台，为何要自建框架?

A：首先，自建的框架可以实现高度自动化，因此可以极大地简化策略测试流程。其次，自建的框架允许我们使用自己的数据和计算资源，这可以保证我们对整个测试过程有更多的控制力，同时不受平台上对计算资源的限制。最后，自建的框架可以在保证轻量高效的同时，按需增添功能。

开发人员入门：

Q：回测过程是如何进行的?

A：首先传入一个价格矩阵，然后指定一个交易策略函数，最后启动回测，回测结束后即可获取模拟过程中的交易日志等信息。

Documentation

Requirements:

Python >= 3.6

Pandas >= 0.24.0

Constructor:

```
your_instance = BackTestFramework(**kwargs)
```

Available parameters: see Public Read-Write Attributes.

Class Attributes:

1. `print_debugging`：若为 `True`，则打印每笔交易明细，包括交易日期、资产类型、数量、当前价格、总费用、剩余现金、当前组合价值。默认为 `False`。

   > 用法建议：开启该功能有助于追踪策略运行过程中的错误。

2. `print_debugging_detail`：若为 `True`，则打印更细致的信息，包括每天开始时打印日期、交易开始前打印交易资产类型。默认为 `False`。

   > 用法建议：开启该功能有助于检查框架自身是否存在 bug，通常不需要开启。

3. `enable_security_check`：资产检查指检查待交易的资产是否在 `price` 中有定义，以及交易发生时价格是否为 `NaN`。在发生错误时抛出错误明细。若为 `False`，则禁用资产检查。默认为 `True`。

   > 用法建议：资产检查通常仅占用极少的计算资源，通常耗时不会超过数百毫秒，因此若无特殊要求，无需禁用资产检查。

4. `disable_short_sell`：若为 `False`，则允许卖空操作。默认为 `True`。

   > 用法建议：在默认情况下，所有卖空操作都是被禁止的，包括卖空股票、期指和股指。在不需要卖空的情况下禁用卖空，有助于迅速追踪策略可能存在的错误。

Public Read-Write Attributes:

1. `price`：价格矩阵，定义了每日的资产价格。交易会参照矩阵中的价格数据进行结算。行指标为日期，列指标为资产名称。
2. `start_date`：回测的开始日期。如果不作明确定义，自动寻找价格矩阵的第一个日期。
3. `end_date`：回测的结束日期。如果不作明确定义，自动寻找价格矩阵的最后一个日期。

   > 用法建议：在测试策略函数是否存在 bug 时，如果时间跨度较大的回测用时较长，可以先在短时间内进行测试，如果策略函数没有问题，再拓展到更长的时间区间上。

4. `securities`：回测考虑的资产列表。如果不作明确定义，自动寻找价格矩阵的所有列名。
5. `stock_list`：回测考虑的股票列表。如果不作明确定义，自动寻找价格矩阵的所有列名。
6. `futures_list`：回测考虑的期货列表。默认为空表。
7. `benchmark`：回测所用的基准资产价格序列，如果定义了，则在作图时会对应作出该基准的价格变化。如果基准资产已经在价格矩阵中了，也可以只传入该资产的名称。默认为空。
8. `factors`：因子，计划框架可以自动产生一些因子，或者收集用户指定的因子。目前没有作用。
9. `commission`：手续费比例，对应一个 0～1 的小数。默认为 0。
10. `slippage`：滑点，目前没有作用。
11. `cash`：现金，表示回测开始时的现金量。默认为 1 亿元。

Public Read-only Attributes:

1. `log`：交易日志，记录了每笔交易的发生日期、资产类型、数量、当前价格、交易后仓位、剩余现金、当前组合价值。仅在回测后可以访问。
2. `position`：动态历史仓位，记录了每天结束时，每种资产以及现金的仓位。仅在回测后可以访问。
3. `current_position`：当前仓位，记录了上笔交易发生后的仓位。

   > 用法建议：由于可以实时访问当前仓位，灵活使用有助于构建比较复杂的交易策略。

4. `value`：动态历史价值，记录了每天结束时，每种资产以及现金的分量价值。仅在回测后可以访问。
5. `current_value`：当前价值，记录了上笔交易发生后的组合价值。
6. `cash_spent_on_commission`：手续费支出。仅在回测后可以访问。
7. `today`：今天的日期。
8. `yesterday`：上个交易日的日期。

   > 用法建议：灵活使用这两个属性，有助于构建比较复杂的交易策略。常用于根据前一日的收盘情况进行交易。

Public Methods:

1. `buy(security: str or list, volume)`：买入指定手数的资产。
2. `sell(security: str or list, volume)`：卖出指定手数的资产。
3. `adjust_to(security: str or list, new_position)`：将资产的仓位调整到指定的手数。
4. `adjust_to_value(security: str or list, new_value)`：将资产的仓位调整到指定的价值。
5. `adjust_to_portion_of_value(security: str or list, portion_of_value)`：将资产的仓位调整到对当前价值的指定比例上。
6. `adjust_to_by_portion(security: str or list, portion, otherwise_position=0)`：按比例扩大或缩小当前仓位。
7. `strategy()`：策略函数，用户需要重载该函数，让框架在每个交易日作判断，进行交易。
8. `user_initialize()`：用户指定的初始化函数，默认为 `pass`。
9. `user_finalize()`：用户指定的最后处理函数，默认为 `pass`。
10. `run()`：启动回测。
11. `summary(save_pic=False, pic_name='image.png')`：作图，汇报最大回撤、年化收益率和夏普比率。
12. `log_output(file_name='result.xlsx')`：将交易日志、动态历史仓位、动态历史价值、所用的价格矩阵输出到 Excel 文档中。