## **vn.py社区精选14 - 量化策略评估的显微镜：逐笔对冲统计（上）**

原创 KeKe VNPY  2019-12-03

收录于话题

#vn.py社区精选

25个

**逐日盯市统计的痛点**

在实盘交易中，逐日盯市（Marking-to-Market）是基于当日的收盘价、仓位数据、成交数据等来统计每日的盈亏，用于交易所对于客户盈亏情况的每日清算以及保证金管理。

在策略回测中，逐日盯市统计的算法也可以用于对策略盈亏曲线的计算和绘制。一个好的策略，资金曲线总是整体向上，并且相对平滑无太大回撤的，换句话说，就是夏普比率和收益回撤比都比较高。2.0版本的vn.py框架的CTA回测引擎，为了更直观的评估策略的整体效果，内置的盈亏统计采用了逐日盯市的模式。

但因为将每日所有的成交数据都映射到了最终收盘时的结果，逐日盯市统计的方式，无法在每笔开平仓交易的层面来分析盈亏情况，例如：手续费和滑点相对平均盈亏的占比、策略交易胜率和盈亏比等统计指标。

考虑到以上信息对于策略开发和研究的重要性，在本文中我们设计了一种新的逐笔开平对冲算法，来解决相关回测统计指标计算的问题。

**逐笔对冲统计的概念**

在讲解代码前，先通过例子来简单介绍一下逐笔对冲统计这个概念：

在交易中，若判断某个标的物上涨，可以买入开仓，如以100元的价格买入1手；

若价格继续上涨到110，我们“赚”了10元，但这属于浮动盈亏，只有真正平仓的时候盈利才会真正落在口袋里面，也就是从浮动盈亏变成实际盈亏；

价格继续上涨到112时候，我们发出卖出平仓指令，此时开平仓盈亏为（112-100）\*1 = 12。

上面的例子中可以知道该笔开平仓赚了12元，在实际成交中我们还需要考虑手续费和滑点：

手续费：每次交易都需要给交易所和经纪商支付的费用，对于股票等证券产品还需要交付印花税和过户费；

滑点：交易中策略的目标成交价和最终的实际成交价可能会有一定的偏差，这个偏差就是滑点，背后的原因是盘口上的买卖价差以及实际交易中的延时。

计算完每笔交易的手续费和滑点后，我们就可以最终得到该笔开平仓交易的净盈亏情况。除了最简单的一开一平外，现实中许多策略的开平交易情况可能复杂得多，总体上可以分为：

一笔开仓，接着是一笔平仓；

一笔开仓或者平仓包含多次成交；

逐步开仓后，有一定仓位再逐步平仓

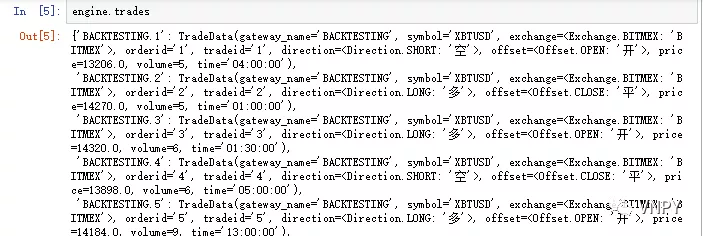
计算完逐笔开平仓盈亏后，我们就可以统计每次交易的胜率和盈亏比了，更进一步还可以对交易方向进行筛选，来看看纯多头交易和纯空头交易的盈亏情况。

**回测的原始成交结果**

以下是vn.py中CTA回测引擎缓存回测成交信息的代码：

*# Push trade update*self.trade\_count += 1  
if long\_cross: trade\_price = min(order.price, long\_best\_price) pos\_change = order.volumeelse: trade\_price = max(order.price, short\_best\_price) pos\_change = -order.volume  
trade = TradeData( symbol=order.symbol, exchange=order.exchange, orderid=order.orderid, tradeid=str(self.trade\_count), direction=order.direction, offset=order.offset, price=trade\_price, volume=order.volume, time=self.datetime.strftime("%H:%M:%S"), gateway\_name=self.gateway\_name,)trade.datetime = self.datetime  
self.strategy.pos += pos\_changeself.strategy.on\_trade(trade)  
self.trades[trade.vt\_tradeid] = trade

每一笔成交信息都以TradeData的数据格式缓存在trades字典中，我们先通过打印输出该字典来直观地看看TradeData的数据结构：



**打印输出成交记录**

对成交缓存数据的结构有个大概的了解后，接下来遍历engine.trades的值，依次打印：

成交时间：value.datetime；

成交方向：value.direction.value;

成交开平：value.offset.value;

成交价格：value.price;

成交数量：value.volume。

在遍历过程中，若检测到是平仓操作，即value.offset.value == "平”，则另外打印分隔线，便于肉眼观察每笔开平仓所对应的时间、价格等：

trade = engine.tradesfor value in trade.values(): print("时间:",value.datetime,value.direction.value,value.offset.value, "价格：",value.price, "数量：",value.volume) if value.offset.value == "平": print("---------------------------------------------------------")



尽管只是在Jupyter Notebook中简单的进行打印输出，已经能够一目了然的看到每次开平仓交易的基本信息。

其中的回测价格信息可以对照实盘成交回报信息进行对比，去计算真实成交和回测成交的价差，统计真实滑点，每隔一段时间（如一个月后）对回测中用到的滑点参数进行调整，力求回测尽量与实盘交易一致。

**一开一平的统计逻辑**

这里我们从简单的情况开始着手，首先做出假设条件：

每次委托都能完全成交，即一次委托对应一次成交；

开仓成交后接下来的就是平仓委托，即每2笔成交是开平仓关系；

由于不需要考虑较为复杂的一次委托多次成交情况，每次开仓成交后的下一笔必定是成交量相等的平仓成交，那么可以设计出如下的计算逻辑：

构建原始成交数据DataFrame，其中包括：日期时间、成交方向、开平仓、价格、数量；

把【价格】列表向后平移一个单位，得到上一笔成交记录；

对DataFrame进行条件判断：

若【成交方向】、【开平仓】为“多平”，意味着本次交易为空开->多平，那么盈亏=（开仓价格-平仓价格）\* 成交数量；持仓时间=平仓时间-开仓时间；

若【成交方向】、【开平仓】为“空平”，意味着本次交易为多开->空平，那么盈亏=（平仓价格-开仓价格）\* 成交数量；持仓时间=平仓时间-开仓时间；

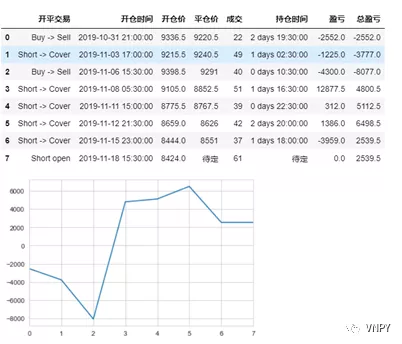
对最后一行进行额外处理：若【开平】为“开”，【平仓价】和【持仓时间】设置为"待定"，【盈亏】设为0；

使用dropna把【盈亏】为空值的行去掉；

对DataFrame的索引重新排序；

计算【累计盈亏】并画出图。

同时在实盘交易中做每日收盘后的统计时，我们可以设置DataFrame只显示当月的成交记录，便于重点观察最近一段时间的盈亏情况、持仓时间等：



上图中，我们可以一目了然地看出每一组完整的开平仓交易的最终盈亏以及持仓时间。

搞定了简单的情况，接下来我们可以将算法变得更加通用化，满足更多场景：一次开仓多次平仓、多次开仓一次平仓以及更加复杂的多次开仓多次平仓。