## **vn.py社区精选16 - 策略移植，从TB到vn.py！**

原创 用Python的交易员 VNPY  2019-12-20

收录于话题

#vn.py社区精选

25个

原文作者：KeKe

TradeBlazer交易开拓者（简称TB），可能是许多投资者开始接触量化时的第一根拐杖，也是国内用户量最大的量化平台之一。

但随着时间过去，国内量化投资者编程水平的逐渐提高，越来越多的人开始转向Python这样的开源生态体系。

在转换平台的过程中，由于编程语法、数据结构、驱动机制等方面的区别，不少人遇到了各种困难，掉在某些坑里可能几周都爬不出来。

本篇文章中我们就来通过一个的经典趋势跟踪策略AtrRsiStrategy，来详细讲解如何一步步将TB策略代码移植到vn.py上的过程。

**ATR-RSI策略**

完整的ATR-RSI策略逻辑如下：

开仓过滤：当前波动率（ATR）大于历史平均波动率（ATR均值）时，我们认为后续走出趋势的机会变大，只有此时才考虑开仓交易；

多头开仓：当RSI指标进入超买区域（比如RSI > 66），说明多头力量已取得上风，此时选择立即做多开仓，为了保证能够立刻成交，使用超价的限价委托来下单。

多头平仓：采用固定百分比的移动止损，在持有多头仓位情况下，跟踪价格曾经到达的最高点，当价格从最高点回落到固定百分比（比如0.8%）的一瞬间立刻平掉多头仓位。

空头开仓：当RSI指标进入超卖区域（比如RSI < 34），说明空头已取得上风，此时选择立即做空开仓，同样使用超价限价单保证立刻成交。

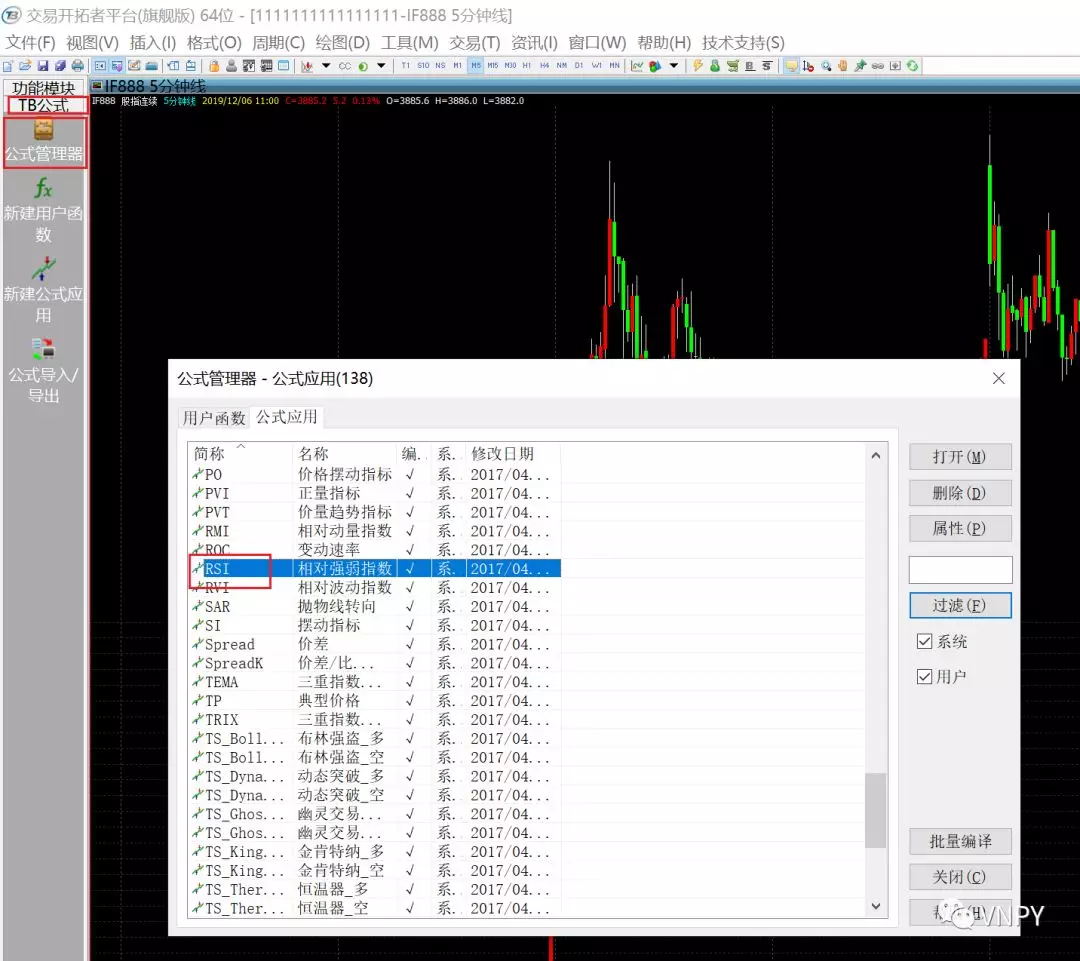
空头平仓：同样采用固定百分比的移动止损，当价格当价格从最低点反弹超过固定百分比（比如0.8%）的一瞬间立刻平掉空头仓位。

注意点：我们总是假设在当前K线走完计算信号并且发出委托，成交永远发生在下一根K线。即T时刻计算信号，发出委托；最快也要T+1时刻该委托才能成交。这也是下面停止单和限价单撮合的充分条件。

**TB中的策略实现**

**创建RSI指标函数**

打开TB，在【TB公式】->【公式管理器】->【公式应用】里面找到RSI指标



打开RSI指标公式应用，复制里面的代码。

同样在【TB公式】->【新建用户函数】里面创建新的RSI指标函数，这类命名为rsirsi，把代码粘贴到新的函数里面。

修改函数代码：变量输出类型修改成NumericSeries，删除最后4行的画图函数。

编译保存后，退出。

**创建ATR-RSI策略**

在【TB公式】->【新建公式应用】打开新的策略模板。

定义变量输出类型统一为NumericSeries。由于很多信号是基于T计算，在T+1时间成交的，故我们需要取前一个时刻的数据，比如前一个时刻的RSI指标，即rsi\_value[1]。

计算当前ris指标，rsi\_value = rsi\_array[1]

Params Numeric rsi\_length(5); Numeric rsi\_entry(16);Vars NumericSeries rsi\_array(0); NumericSeries rsi\_value(0); NumericSeries rsi\_buy(0); NumericSeries rsi\_sell(0);  
Begin *// Calculate Rsi Value* rsi\_buy = 50 + rsi\_entry; rsi\_sell = 50 - rsi\_entry; rsi\_array = rsirsi(rsi\_length); rsi\_value = rsi\_array[1];

计算当前ATR指标，atr\_value = atr\_array[1]；以及当前ATR均值，atr\_ma= atr\_ma\_array[1]

Params Numeric atr\_length(22); Numeric atr\_ma\_length(10);  
Vars NumericSeries atr\_value(0); NumericSeries atr\_ma(0); NumericSeries atr\_arry(0); NumericSeries atr\_ma\_array(0);  
Begin // Calculate Atr Value and Atr Ma atr\_arry = AvgTrueRange(atr\_length); atr\_ma\_array = Average(atr\_arry[atr\_ma\_length], atr\_ma\_length); atr\_value = atr\_arry[1]; // last bar for atr\_value atr\_ma = atr\_ma\_array[1]; // last bar for atr\_ma\_value

空仓情况下，发出限价单委托开仓：

当波动率上涨并且RSI指标>66时，使用当前收盘价+5的限价单，超价买入保证成交；

当波动率上涨并且RSI指标<34时候，使用当前收盘价-5的限价单，超价卖出保证成交。

If(MarketPosition == 0) { intra\_trade\_low = Low[1]; intra\_trade\_high = High[1]; *// 【Long condition】* If(rsi\_value > rsi\_buy AND atr\_value > atr\_ma) { long\_limit = Close[1] + 5; If(long\_limit>=Low) { Buy(fixed\_size, Min(Open, long\_limit)); } } *// 【Short condition】*    Else If(rsi\_value < rsi\_sell AND atr\_value > atr\_ma) { short\_limit = Close[1] - 5; If(short\_limit <=High) { SellShort(fixed\_size, Max(Open, short\_limit));      }       }  }

百分比移动止盈止损离场：

多仓情况下，当价格从最高点回落0.8%的一瞬间触发条件单离场；

空仓情况下，当价格从最低点回调0.8%的一瞬间触发条件单离场；

*// postition >0*  Else If(MarketPosition >0) { intra\_trade\_high = Max(intra\_trade\_high, High[1]); intra\_trade\_low = Low[1]; long\_stop = intra\_trade\_high \* (1 - trailing\_percent / 100);  
 If(Low <= long\_stop) { Sell(MarketPosition, Min(Open, long\_stop)); } } *// postiton < 0*  Else If(MarketPosition <0) { intra\_trade\_low = Min(intra\_trade\_low, Low[1]); intra\_trade\_high = High[1]; short\_stop = intra\_trade\_low \*(1+ trailing\_percent /100); If(High >= short\_stop) { BuyToCover(-MarketPosition, Max(Open, short\_stop)); } }

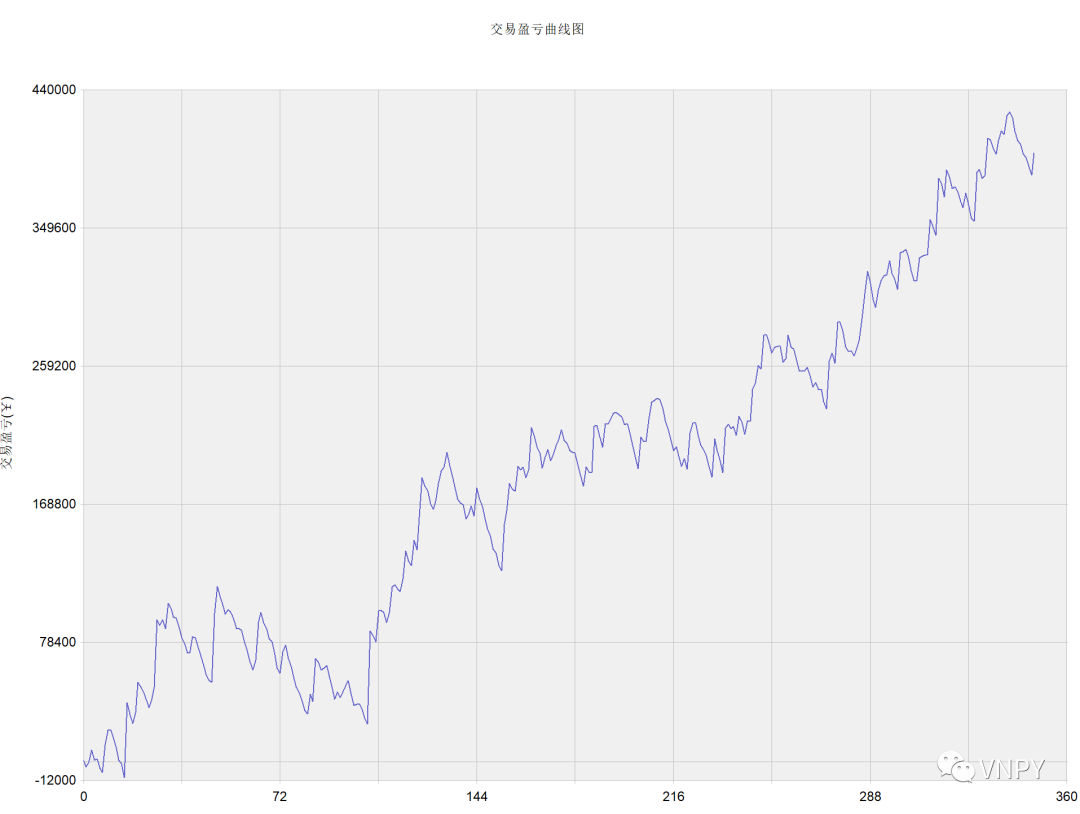
**策略回测结果**

数据：沪深300股指连续（IF888）

时间区间：2019年1月~12月

K线周期：1分钟

策略效果：资金曲线整体向上，平均盈亏比为2.08。



**TB完整代码**

Params  Numeric atr\_length(22); Numeric atr\_ma\_length(10); Numeric rsi\_length(5); Numeric rsi\_entry(16); Numeric trailing\_percent(0.8); Numeric fixed\_size(1); Vars NumericSeries rsi\_array(0); NumericSeries atr\_value(0); NumericSeries atr\_ma(0); NumericSeries rsi\_value(0); NumericSeries rsi\_buy(0); NumericSeries rsi\_sell(0); NumericSeries intra\_trade\_high(0); NumericSeries intra\_trade\_low(0); NumericSeries atr\_arry(0); NumericSeries atr\_ma\_array(0); NumericSeries long\_stop(0); NumericSeries short\_stop(0); NumericSeries long\_limit(0); NumericSeries short\_limit(0);Begin  
 // Calculate Rsi Value rsi\_buy = 50 + rsi\_entry; rsi\_sell = 50 - rsi\_entry; rsi\_array = rsirsi(rsi\_length); rsi\_value = rsi\_array[1]; // Calculate Atr Value and Atr Ma atr\_arry = AvgTrueRange(atr\_length); atr\_ma\_array = Average(atr\_arry[atr\_ma\_length], atr\_ma\_length);   atr\_value = atr\_arry[1]; // last bar for atr\_value   atr\_ma = atr\_ma\_array[1]; // last bar for atr\_ma\_value If(MarketPosition == 0) { intra\_trade\_low = Low[1]; intra\_trade\_high = High[1]; // 【Long condition】 If(rsi\_value > rsi\_buy AND atr\_value > atr\_ma) { long\_limit = Close[1] + 5; If(long\_limit>=Low) { Buy(fixed\_size, Min(Open, long\_limit)); } } // 【Short condition】    Else If(rsi\_value < rsi\_sell AND atr\_value > atr\_ma) { short\_limit = Close[1] - 5; If(short\_limit <=High) { SellShort(fixed\_size, Max(Open, short\_limit));      }          } } // postition >0 Else If(MarketPosition >0) { intra\_trade\_high = Max(intra\_trade\_high, High[1]); intra\_trade\_low = Low[1]; long\_stop = intra\_trade\_high \* (1 - trailing\_percent / 100);  
 If(Low <= long\_stop) { Sell(MarketPosition, Min(Open, long\_stop)); } } // postiton < 0 Else If(MarketPosition <0) { intra\_trade\_low = Min(intra\_trade\_low, Low[1]); intra\_trade\_high = High[1]; short\_stop = intra\_trade\_low \*(1+ trailing\_percent /100); If(High >= short\_stop) { BuyToCover(-MarketPosition, Max(Open, short\_stop)); } }End

**vn.py中的策略实现**

TB策略的逻辑完全由行情驱动，即每次有行情变化（Tick更新、K线走完）时会完整执行代码中的所有逻辑。与之不同的是，vn.py内置的CTA策略模板，提供了诸多的事件驱动回调函数，如：Tick更新驱动（on\_tick函数）、K线驱动（on\_bar函数）、成交驱动（on\_trade）、委托驱动（on\_order）等。

要移植TB上的策略，只需在vn.py策略代码的on\_bar回调函数中实现对应的策略逻辑即可：

调用cancel\_all()函数撤销未成交委托，保证当前委托状态的干净和唯一性；

基于K线时间序列容器ArrayManager，来维护K线历史数据，计算需要的技术指标数据；

委托方式同样分为4种，下单时的可选参数中，stop=True意味着停止单，stop=False意味着限价单：

buy：买入开仓

sell：卖出平仓

short：卖出开仓

cover：买入平仓

def on\_bar(self, bar: BarData): """ Callback of new bar data update. """ self.cancel\_all()  
 am = self.am am.update\_bar(bar) if not am.inited: return  
 atr\_array = am.atr(self.atr\_length, array=True) self.atr\_value = atr\_array[-1] self.atr\_ma = atr\_array[-self.atr\_ma\_length:].mean() self.rsi\_value = am.rsi(self.rsi\_length)  
 if self.pos == 0: self.intra\_trade\_high = bar.high\_price self.intra\_trade\_low = bar.low\_price  
 if self.atr\_value > self.atr\_ma: if self.rsi\_value > self.rsi\_buy: self.buy(bar.close\_price + 5, self.fixed\_size) elif self.rsi\_value < self.rsi\_sell: self.short(bar.close\_price - 5, self.fixed\_size)  
 elif self.pos > 0: self.intra\_trade\_high = max(self.intra\_trade\_high, bar.high\_price) self.intra\_trade\_low = bar.low\_price  
 long\_stop = self.intra\_trade\_high \* \ (1 - self.trailing\_percent / 100) self.sell(long\_stop, abs(self.pos), stop=True)  
 elif self.pos < 0: self.intra\_trade\_low = min(self.intra\_trade\_low, bar.low\_price) self.intra\_trade\_high = bar.high\_price  
 short\_stop = self.intra\_trade\_low \* \ (1 + self.trailing\_percent / 100) self.cover(short\_stop, abs(self.pos), stop=True)  
 self.put\_event()

完整的代码实现请参考Github仓库中的[策略源代码](https://github.com/vnpy/vnpy/blob/master/vnpy/app/cta_strategy/strategies/atr_rsi_strategy.py" \t "/Users/huangning/Documents\\x/_blank)。

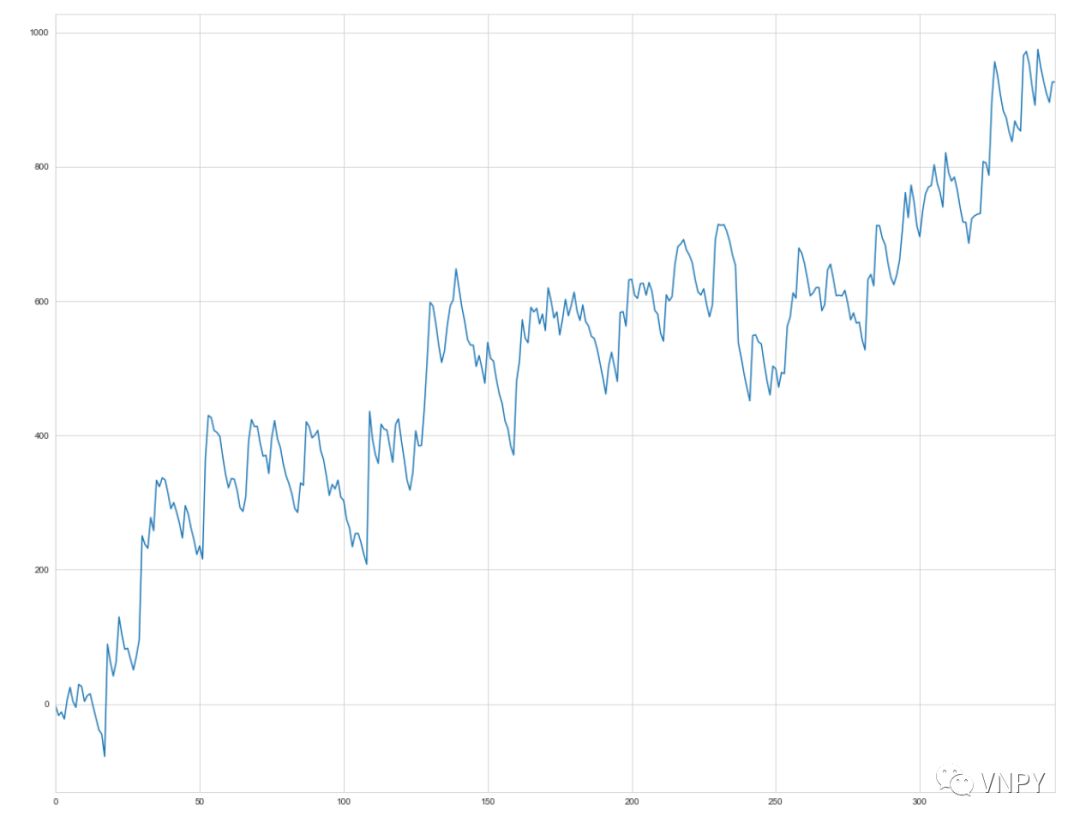
**策略回测结果**

数据：RQData的沪深300主力拼接合约（IF888）

时间区间：2019年1月~12月

K线周期：1分钟

策略效果：资金曲线整体向上，与TB的资金曲线几乎一致。



**两个平台的对比总结**

**K线数据访问区别**

TB

默认访问的是当前最新时间点的数据，如使用Close函数访问的是当前最新一根尚未走完的K线数据；

如果需要访问最近一根已经走完的K线收盘价，则必须使用Close[1]；

同样，对于最近一根已经走完K线的技术指标，以RSI指标举例，则必须使用rsi\_array[1]。

vn.py

默认访问的是最近一根已经走完的K线数据；

当前最新一根尚未走完的K线数据，在策略内禁止访问，杜绝TB上的信号闪烁问题（未来函数）。

**委托撮合逻辑区别**

TB

需要策略开发者在策略内，自行编写相对复杂的委托撮合逻辑，来尽量逼近真实交易情况；

vn.py

内置了详尽的停止单、限价单撮合逻辑，在调用委托函数时，只需调整可选参数stop即可实现委托的转变。

**策略回测结果区别**

即使在策略逻辑层面已经做到一致，TB和vn.py的回测资金曲线图依旧可能存在某些细节方面的区别。主要原因是数据源方面的不同，TB使用的是自身提供的历史数据源，而vn.py默认推荐使用的是RQData数据服务。