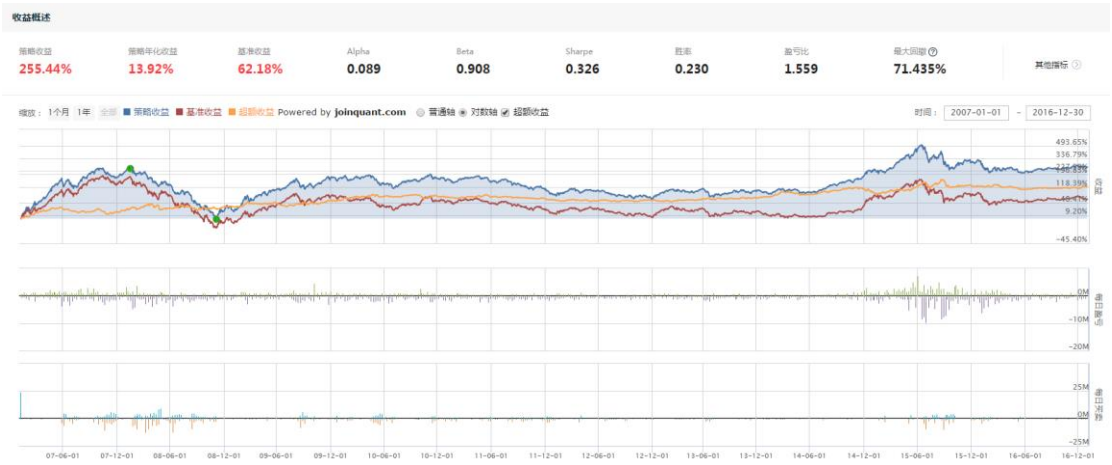
策略介绍：

该布林策略经回测，2007-01-01 到 2017-01-01，选用沪深 300 前 100 只股作为样本，在聚宽平台上回测结果如下：



该策略十年收益为 255.44%，策略年化收益为 13.92%，Alpha 是投资者获得与市场波动无关的回报，即超额收益为 8.9%，Beta 为 0.908，表示投资组合和基准的走向相同，但是比基准的移动幅度更小。最大回撤是用来描述买入产品后可能出现的最糟糕的情况，本策略为 71%，说明策略还是有一定风险的。

指标介绍：

布林线（Boll）指标是股市技术分析的常用工具之一，通过计算股价的“标准差”，再求股价的“信赖区间”。该指标在图形上画出三条线，其中上下两条线可以分别看成是股价的压力线和支撑线，而在两条线之间还有一条股价平均线。一般来说，股价会运行在压力线和支撑线所形成的通道中。

在众多技术分析指标中，BOLL 指标属于比较特殊的一类指标。绝大多数技术分析指标都是通过数量的方法构造出来的，它们本身不依赖趋势分析和形态分析，而 BOLL 指标却与股价的形态和趋势有着密不可分的联系。BOLL 指标中的“股价

通道”概念正是股价趋势理论的直观表现形式。BOLL 是利用“股价通道”来显示股价的各种价位，当股价波动很小，处于盘整时，股价通道就会变窄，这可能预示着股价的波动处于暂时的平静期；当股价波动超出狭窄的股价通道的上轨时，预示着股价的异常激烈的向上波动即将开始；当股价波动超出狭窄的股价通道的下轨时，同样也预示着股价的异常激烈的向下波动将开始。

策略流程：

流程图 1

正文：

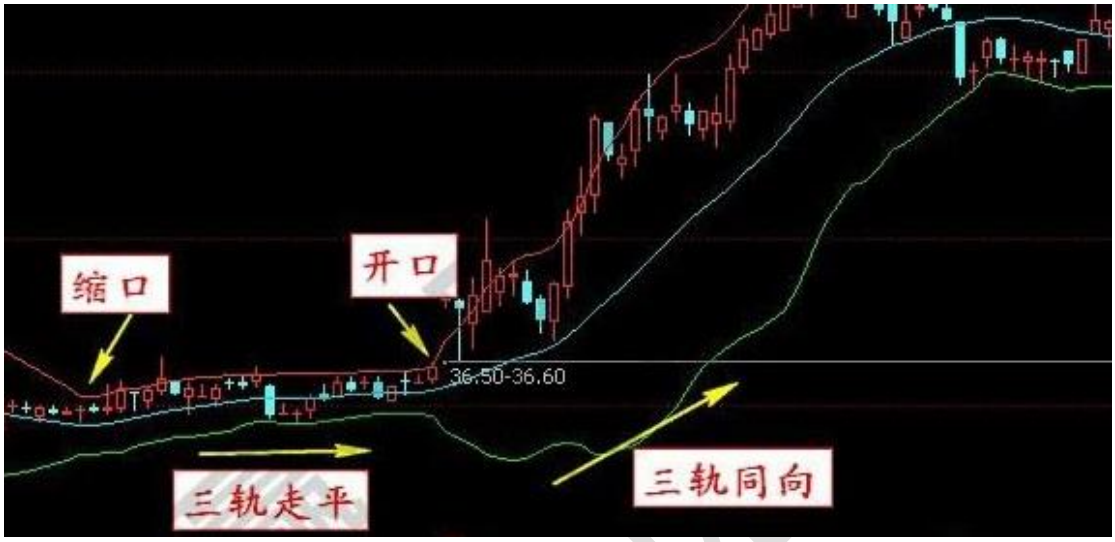
根据小市值原则，利用 get_fundamentals 函数获取指定股票池（沪深 300）的财务指数，根据自身情况选择股票数，此处不增加条件，仅选用前 100 只股。

利用 history 函数获取多只股票的收盘价数据，然后计算 N 日收盘价平均值、标准差，计算前一日上下轨线。

- 中轨线=N 日的移动平均线
- 上轨线=中轨线+两倍的标准差
- 下轨线=中轨线－两倍的标准差

```
for i in range(0,num):
    for j in range(0,days):
        close_data[i][j]=price[security][i+j]
    mid[i]=np.mean(close_data[i])
    std[i]=np.std(close_data[i])
up=mid[num-1]+2*std[num-1]
down=mid[num-1]-2*std[num-1]
```

利用 M 日标准差判断布林线是开口或者闭口：



开口形态常出现在市场短期内暴涨行情的初期。当市场经过长时间的底部整理后，市场价格突然出现向上急速飙升的行情，此时布林线上轨线也同时急速向上扬升，而下轨线却加速向下运动就是开口。

收口形态常出现在市场暴跌行情的初期。收口型喇叭口形态的确立是以市场的上轨线开始掉头向下、价格向下跌破短期均线为准。对于收口型喇叭口形态的出现，投资者如能及时卖出则能保住收益、减少较大的下跌损失

```
boll=0
for i in range(0,num-1):
    if std[i]>std[i+1]:
        boll=boll+1
    else:
        boll=boll-1
```

市场开口或者收口需要用不同的择时条件。当开口形态出现时，如果当前价格超过昨日的上轨且高于均价的，则使用当前所选股相应的现金买入对应数量，如果当前价格跌破了昨日的下轨且价格低于均价则卖出。如果连续多日收口，即则股价超过上轨时卖，跌破下轨时买。

```
if boll==-4:
    if current_price>up and current_price>mid[num-1]:
        num_of_shares=int(cash/current_price)
        if num_of_shares>0 and paused==False:
            order(security,+num_of_shares)
    elif current_price<down and current_price<mid[num-1]:
        if paused==False:
            order_target(security,0)
```

```
if boll==4:
    if current_price>up and current_price<mid[num-1]:
        if paused==False:
            order_target(security,0)
    elif current_price<down and current_price>mid[num-1]:
        num_of_shares=int(cash/current_price)
        if num_of_shares>0 and paused==False:
            order(security,+num_of_shares)
```

由于开口闭口都属于 较为特殊的情况，大部分情况下，市场都是平稳波动的，因此我们在择时的时候增加了平口情况。在平口情况下，股价波动不剧烈，少有穿过上下轨的情况，因此，我们设定择时条件为股价在中轨以上且价格高于三天前的买入，股价在中轨以下且价格低于三天前的卖出。

```
if boll in range(-1,1):
    if current_price>mid[num-1] and mid[num-1]>mid[num-3]:
        num_of_shares=int(cash/current_price)
        if num_of_shares>0 and paused==False:
            order(security,+num_of_shares)
    if current_price<mid[num-1] and mid[num-1]<mid[num-3]:
        if paused==False:
            order_target(security,0)
```

策略改良建议：

该策略历史回测成绩良好，然而在进入 2017 年以后，和 10-15 年前相比，如今更严的监管以及激烈的竞争，令很多流行的量化投资策略不再有效，导致量化投资超额收益锐减，该策略的情况也不够理想。

从回测情况看，策略总体走势与基准想符合，但是策略的收益仍然是低于基准的。由于每个策略对不同的股票的效果不同，我们在执行回测时使用的是沪深 300 前 100 只股票，在实际操作中从收益方面考虑，我们可以适当选取一些运行该策略历史收益较好的股票。此外，从回测结果可以看出，该策略在回测时间内只有少数买卖操作，一方面是因为 2017 年市场较为平缓，没有什么特别大的波动，另一方面这说明我们在策略中所选用的择时条件还有不足，需要加以改进。

附录：

```
import numpy as np
import pandas as pd
from pandas import Series,DataFrame

def initialize(context):
    # 设定沪深 300 作为基准
    set_benchmark('000300.XSHG')
    # 开启动态复权模式(真实价格)
    set_option('use_real_price', True)
    # 股票类交易手续费是：买入时佣金万分之三，卖出时佣金万分之三加千分之一印花税, 每笔交易佣金最低扣 5 块钱
    set_order_cost(OrderCost(open_tax=0, close_tax=0.001, \
                             open_commission=0.0003, close_commission=0.0003,\
                             close_today_commission=0,              min_commission=5), type='stock')
def handle_data(context, data):
    #获取沪深 300 股票池
    stock_set=get_index_stocks('000300.XSHG')
    #此处可增加选股条件
    q = query(
        valuation.code, # 股票代码
    ).filter(
        valuation.code.in_(stock_set),#只对设定股票池执行
    )
```

```
#获取财务数据，指定日期为回测当天
current_date=context.current_dt.strftime('%Y-%m-%d')
fdf = get_fundamentals(q,current_date)
#取前 50 只股
fdf=fdf.head(100)
#获取股票列表
get_stocks=list(fdf['code'])
#  去除 ST，*ST
st=get_extras('is_st',get_stocks,current_date,current_date, df=True)
st=st.loc[current_date]
get_stocks=list(st[st==False].index)
#考虑 5 天的历史数据
num=5
#轨线用前面 20 天数据计算
days=20
#每只股票可用资金为当前资金除以 50
cash=context.portfolio.available_cash/100
#获取所有股票前 num+days 天收盘价数据
price=history(num+days,'1d','close',get_stocks,skip_paused=True)
where_are_nan = np.isnan(price)
where_are_inf = np.isinf(price)
price[where_are_nan] = 0
price[where_are_inf] = 0
#循环每只股
for security in get_stocks:
    #用数组保存均值，大小为 num
    mid=np.arange(num)
    #标准差
    std=np.arange(num)
    #定义数组大小
    close_data=np.arange(num*days).reshape(num,days)
    for i in range(0,num):
        for j in range(0,days):
            close_data[i][j]=price[security][i+j]
        #中轨线即均值为 days 天收盘价数据平均
        mid[i]=np.mean(close_data[i])
        #计算标准差
        std[i]=np.std(close_data[i])
    #上轨线=中轨线+两倍的标准差
    up=mid[num-1]+2*std[num-1]
    #下轨线
    down=mid[num-1]-2*std[num-1]
    #保存 num 天数据，判断开口收口或平口
    boll=0
```

```
for i in range(0,num-1):
    if std[i]>std[i+1]:
        boll=boll+1
    else:
        boll=boll-1

#判断目前股票是否停牌
paused=data[security].paused

#取得当前股票价格
current_price=data[security].price
#如果连续 num 天开口
if boll==-4:
        #如果当前价格超过昨日的上轨且价格高于均线
    if current_price>up and current_price>mid[num-1]:
        #计算可以买多少股票
        num_of_shares=int(cash/current_price)
            #如果可以买的数量超过 0 并且股票未停牌
        if num_of_shares>0 and paused==False:
            #购买股票
            order(security,+num_of_shares)
        #如果当前价格跌破了昨日的下轨且价格低于均线
    elif current_price<down and current_price<mid[num-1]:
        #如果股票未停牌
        if paused==False:
            #将股票卖空
            order_target(security,0)
#如果连续 num 天收口
if boll==4:
    #股价超过上轨且价格低于均线时卖
    if current_price>up and current_price<mid[num-1]:
        if paused==False:
            order_target(security,0)
    #跌破下轨且价格高于均线时买
    elif current_price<down and current_price>mid[num-1]:
        num_of_shares=int(cash/current_price)
        if num_of_shares>0 and paused==False:
            order(security,+num_of_shares)
#连续平口
if boll in range(-1,1):
    #价格在中轨线上且昨日均价高于三天前
    if current_price>mid[num-1] and mid[num-1]>mid[num-3]:
        num_of_shares=int(cash/current_price)
        if num_of_shares>0 and paused==False:
```

```
            order(security,+num_of_shares)
    if current_price<mid[num-1] and mid[num-1]<mid[num-3]:
        if paused==False:
            order_target(security,0)
```