策略分享

ai-自动-一只权0号测试

me_robot 15分钟前 #1

克隆策略

```
# 基础参数配置
class conf:
    start_date = '2006-01-01'
    end_date='2017-07-19'

    # split_date 之前的数据用于训练，之后的数据用作效果评估
    split_date = '2012-01-01'

    # D.instruments: https://bigquant.com/docs/data_instruments.html
    instruments = D.instruments(start_date, split_date)

    # 机器学习目标标注函数
    # 如下标注函数等价于 min(max((持有期间的收益 * 100), -20), 20) + 20 （后面的M.fast_auto_labeler会做取整操作）
    # 说明：max/min这里将标注分数限定在区间[-20, 20]，+20将分数变为非负数 （StockRanker要求标注分数非负整数）
    label_expr = ['return * 100', 'where(label > {0}, {0}, where(label < -{0}, -{0}, label)) + {0}'.format(22)]

    # 持有天数，用于计算label_expr中的return值(收益)
    hold_days = 5

    # 特征 https://bigquant.com/docs/data_features.html，你可以通过表达式构造任何特征
    features = [
        'close_5/close_0',  # 5日收益
        'close_10/close_0',  # 10日收益
        'close_20/close_0',  # 20日收益
        'close_30/close_0',  # 30日收益
        'close_60/close_0',  # 60日收益
        'close_120/close_0',  # 120日收益

        'avg_amount_0/avg_amount_5',  # 当日/5日平均交易额
        'avg_amount_5/avg_amount_10',  # 5日/20日平均交易额
        'avg_amount_10/avg_amount_20',  # 10日/20日平均交易额
        'avg_amount_20/avg_amount_30',  # 20日/30日平均交易额
```

7月21日

1 / 1
7月21日

12分钟前

创建新的主题

一句话告诉讨论什么...

策略&研究

https://i.bigquant.com/user/me_robot/lab/share/AI%E9%80%89%E8%82%A1%E7%AD%96%E7%95%A5-%E8%87%AA%E5%8A%A8.ipynb?_t=1500633526414

克隆策略

```
# 基础参数配置
class conf:
    start_date = '2006-01-01'
```

已保存

« 隐藏预览

策略分享

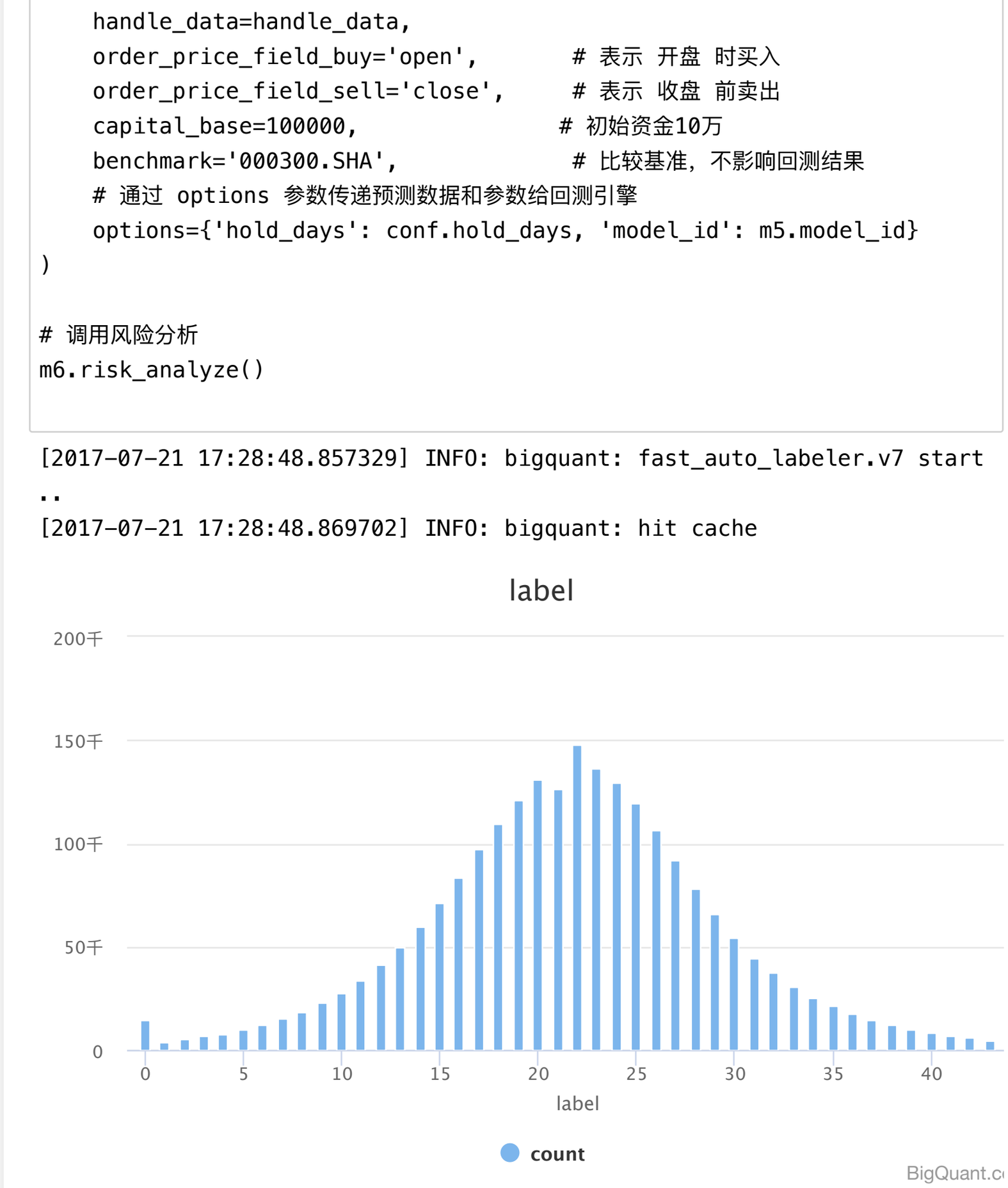
创建主题 取消

```
        'rank_return_30/rank_return_60',  # 30日/60日收益排名
        'rank_return_60/rank_return_120',  # 60日/120日收益排名

        'pe_ttm_0 < 50',
        'st_status_0 >0',
        'list_days_0 >99',

        'pe_ttm_0',
        'st_status_0',
        'list_days_0 > 99',
    ]

# 给数据做标注：给每一行数据（样本）打分，一般分数越高表示越好
m1 = M.fast_auto_labeler.v7(
    instruments=conf.instruments, start_date=conf.start_date, end_date=
conf.split_date,
    label_expr=conf.label_expr, hold_days=conf.hold_days,
    benchmark='000300.SHA', sell_at='open', buy_at='open')

# 计算特征数据
m2 = M.general_feature_extractor.v5(
    instruments=conf.instruments, start_date=conf.start_date, end_date=
conf.split_date,
    features=conf.features)

# 数据预处理：缺失数据处理，数据规范化，T.get_stock_ranker_default_transforms
为StockRanker模型做数据预处理
m3 = M.transform.v2(
    data=m2.data, transforms=T.get_stock_ranker_default_transforms(),
    drop_null=True, astype='int32', except_columns=['date', 'instrument
'],
    clip_lower=0, clip_upper=200000000)

# 合并标注和特征数据
m4 = M.join.v2(data1=m1.data, data2=m3.data, on=['date', 'instrument'],
sort=True)

# StockRanker机器学习训练
m5 = M.stock_ranker_train.v3(training_ds=m4.data, features=conf.feature
s)

# 每一个节点都可以点击展开
m5.plot_model()

## 量化回测 https://bigquant.com/docs/module_trade.html
# 回测引擎：准备数据，只执行一次
def prepare(context):
    # context.start_date / end_date，回测的时候，为trader传入参数；在实盘运行
的时候，由系统替换为实盘日期
    n1 = M.general_feature_extractor.v5(
        instruments=D.instruments(),
        start_date=context.start_date, end_date=context.end_date,
        model_id=context.options['model_id'])

    n2 = M.transform.v2(
        data=n1.data, transforms=T.get_stock_ranker_default_transforms(
),
        drop_null=True, astype='int32', except_columns=['date', 'instru
ment'],
        clip_lower=0, clip_upper=200000000)

    n3 = M.stock_ranker_predict.v2(model_id=context.options['model_id']
, data=n2.data)

    context.instruments = n3.instruments
    context.options['predictions'] = n3.predictions
```

```
# 回测引擎：初始化函数，只执行一次
def initialize(context):

    # 加载预测数据
    context.ranker_prediction = context.options['predictions'].read_df(
)

    # 系统已经设置了默认的交易手续费和滑点，要修改手续费可使用如下函数
    context.set_commission(PerOrder(buy_cost=0.0003, sell_cost=0.0013,
min_cost=5))

    # 预测数据，通过options传入进来，使用 read_df 函数，加载到内存 (DataFrame)
    # 设置买入的股票数量，这里买入预测股票列表排名靠前的5只
    stock_count = 5

    # 每只的股票的权重，如下的权重分配会使得靠前的股票分配多一点的资金，[0.339160,
0.213986, 0.169580, ..]
    context.stock_weights = T.norm([1 / math.log(i + 2) for i in range(
0, stock_count)])

    # 设置每只股票占用的最大资金比例
    context.max_cash_per_instrument = 0.2

# 回测引擎：每日数据处理函数，每天执行一次
def handle_data(context, data):

    # 按日期过滤得到今日的预测数据
    ranker_prediction = context.ranker_prediction[
        context.ranker_prediction.date == data.current_dt.strftime('%Y-
%m-%d')]

    # 1. 资金分配
    # 平均持仓时间是hold_days，每日都将买入股票，每日预期使用 1/hold_days 的资金
    # 实际操作中，会存在一定的买入误差，所以在前hold_days天，等量使用资金；之后，尽
量使用剩余资金（这里设置最多用等量的1.5倍）
    is_staging = context.trading_day_index < context.options['hold_days
'] # 是否在建仓期间（前 hold_days 天）
    cash_avg = context.portfolio.portfolio_value / context.options['hol
d_days']
    cash_for_buy = min(context.portfolio.cash, (1 if is_staging else 1.
5) * cash_avg)
    cash_for_sell = cash_avg - (context.portfolio.cash - cash_for_buy)
    positions = {e.symbol: p.amount * p.last_sale_price
                 for e, p in context.perf_tracker.position_tracker.posi
tions.items()}

    # 2. 生成卖出订单：hold_days天之后才开始卖出；对持仓的股票，按StockRanker预
测的排序末位淘汰
    if not is_staging and cash_for_sell > 0:
        equities = {e.symbol: e for e, p in context.perf_tracker.positi
on_tracker.positions.items()}
        instruments = list(reversed(list(ranker_prediction.instrument[r
anker_prediction.instrument.apply(
                lambda x: x in equities and not context.has_unfinished_
sell_order(equities[x]))])))

        # print('rank order for sell %s' % instruments)
        for instrument in instruments:
            context.order_target(context.symbol(instrument), 0)
            cash_for_sell -= positions[instrument]
            if cash_for_sell <= 0:
                break

    # 3. 生成买入订单：按StockRanker预测的排序，买入前面的stock_count只股票
    buy_cash_weights = context.stock_weights
```

```
    buy_instruments = list(ranker_prediction.instrument[:len(buy_cash_w
eights)])
    max_cash_per_instrument = context.portfolio.portfolio_value * conte
xt.max_cash_per_instrument
    for i, instrument in enumerate(buy_instruments):
        cash = cash_for_buy * buy_cash_weights[i]
        if cash > max_cash_per_instrument - positions.get(instrument, 0
):

            # 确保股票持仓量不会超过每次股票最大的占用资金量
            cash = max_cash_per_instrument - positions.get(instrument,
0)
        if cash > 0:
            context.order_value(context.symbol(instrument), cash)



# 调用交易引擎
m6 = M.trade.v1(
    instruments=None,
    start_date=conf.split_date,
    end_date=conf.end_date,
    prepare=prepare,
    initialize=initialize,
    handle_data=handle_data,
    order_price_field_buy='open',       # 表示 开盘 时买入
    order_price_field_sell='close',     # 表示 收盘 前卖出
    capital_base=100000,                # 初始资金10万
    benchmark='000300.SHA',             # 比较基准，不影响回测结果
    # 通过 options 参数传递预测数据和参数给回测引擎
    options={'hold_days': conf.hold_days, 'model_id': m5.model_id}
)

# 调用风险分析
m6.risk_analyze()
```

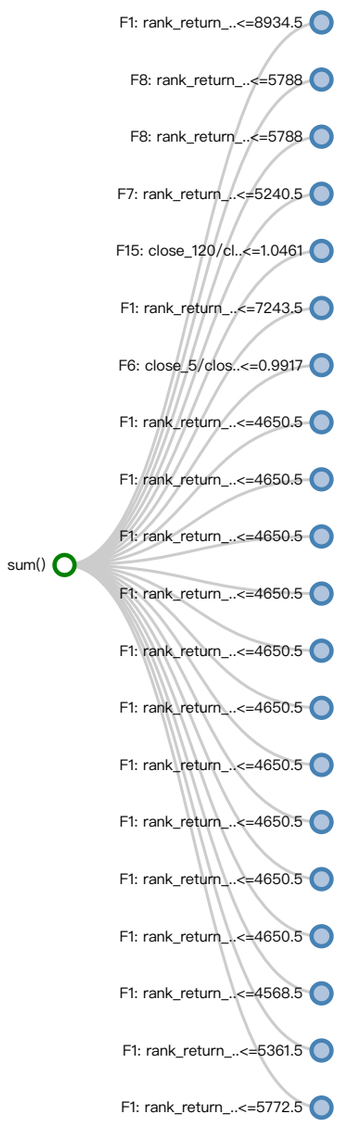
```
[2017-07-21 17:28:48.857329] INFO: bigquant: fast_auto_labeler.v7 start
..
[2017-07-21 17:28:48.869702] INFO: bigquant: hit cache
```




```
[2017-07-21 17:28:48.885747] INFO: bigquant: fast_auto_labeler.v7 end [
0.028408s].
[2017-07-21 17:28:48.902168] INFO: bigquant: general_feature_extractor.
v5 start ..
[2017-07-21 17:28:48.906013] INFO: bigquant: hit cache
[2017-07-21 17:28:48.911699] INFO: bigquant: general_feature_extractor.
v5 end [0.009547s].
[2017-07-21 17:28:48.943617] INFO: bigquant: transform.v2 start ..
```

[2017-07-21 17:28:48.948747] INFO: bigquant: hit cache

[2017-07-21 17:28:48.954018] INFO: bigquant: transform.v2 end [0.010407s].
[2017-07-21 17:28:48.979850] INFO: bigquant: join.v2 start ..
[2017-07-21 17:28:48.988797] INFO: bigquant: hit cache
[2017-07-21 17:28:48.990943] INFO: bigquant: join.v2 end [0.011116s].
[2017-07-21 17:28:49.033084] INFO: bigquant: stock_ranker_train.v3 start ..
[2017-07-21 17:28:49.044840] INFO: bigquant: hit cache
[2017-07-21 17:28:49.052518] INFO: bigquant: stock_ranker_train.v3 end [0.019436s].



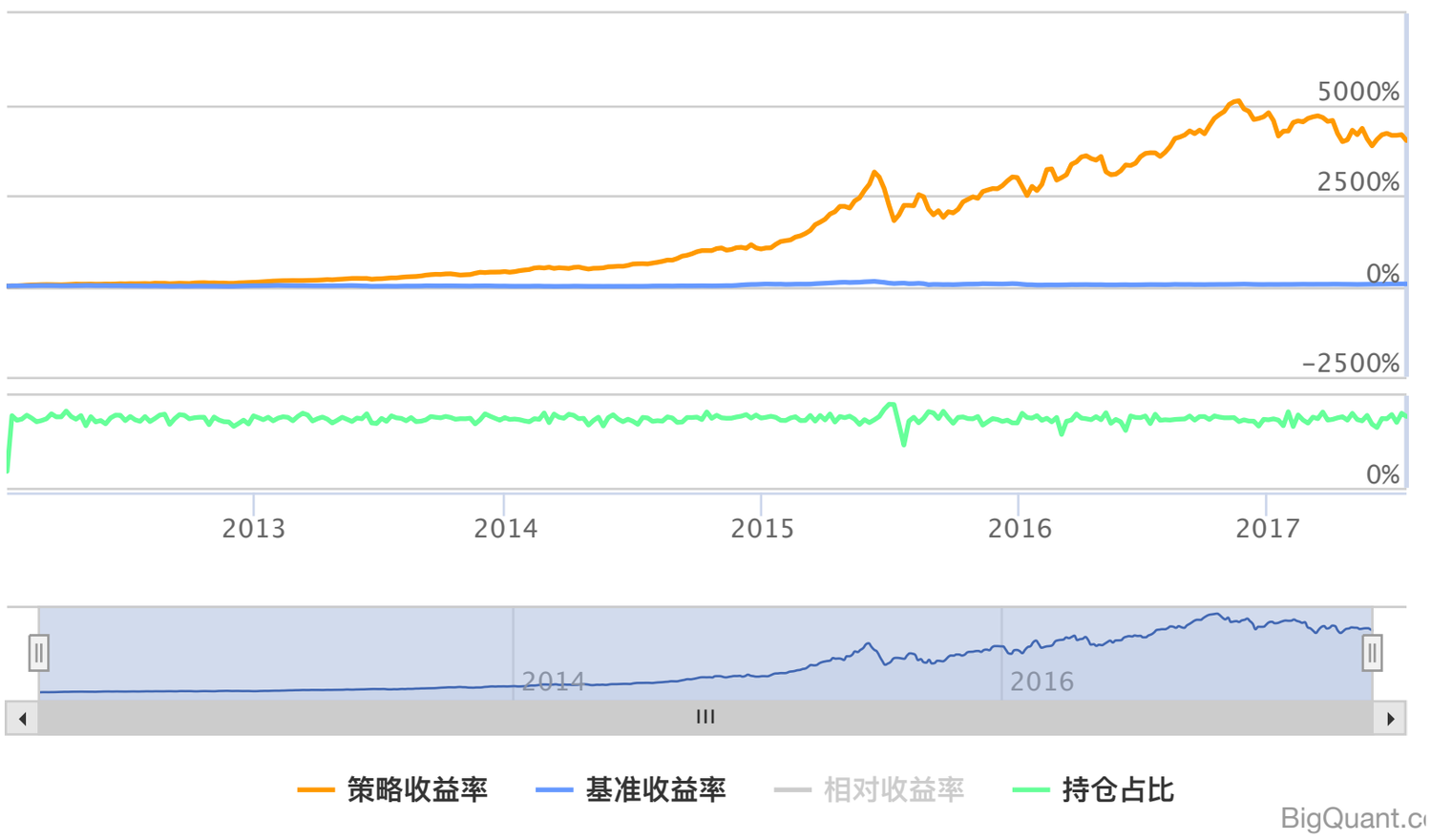
[2017-07-21 17:28:49.182029] INFO: bigquant: backtest.v6 start ..
[2017-07-21 17:28:49.495158] INFO: bigquant: general_feature_extractor.v5 start ..
[2017-07-21 17:29:08.294680] INFO: general_feature_extractor: year 2012, featurerows=565675
[2017-07-21 17:29:39.686384] INFO: general_feature_extractor: year 2013, featurerows=564168
[2017-07-21 17:30:13.096025] INFO: general_feature_extractor: year 2014, featurerows=569948
[2017-07-21 17:30:41.782595] INFO: general_feature_extractor: year 2015, featurerows=569698
[2017-07-21 17:31:09.182378] INFO: general_feature_extractor: year 2016, featurerows=641546
[2017-07-21 17:31:26.686571] INFO: general_feature_extractor: year 2017, featurerows=388437
[2017-07-21 17:31:26.702310] INFO: general_feature_extractor: total feature rows: 3299472
[2017-07-21 17:31:26.710580] INFO: bigquant: general_feature_extractor.v5 end [157.21541s].
[2017-07-21 17:31:26.723589] INFO: bigquant: transform.v2 start ..
[2017-07-21 17:31:33.760937] INFO: transform: transformed /y_2012, 541917/565675
[2017-07-21 17:31:39.645461] INFO: transform: transformed /y_2013, 562660/564168
[2017-07-21 17:31:45.480195] INFO: transform: transformed /y_2014, 559457/569948
[2017-07-21 17:31:51.342031] INFO: transform: transformed /y_2015, 541527/569698
[2017-07-21 17:31:57.706345] INFO: transform: transformed /y_2016, 622280/641546
[2017-07-21 17:32:02.700660] INFO: transform: transformed /y_2017, 357763/388437

[2017-07-21 17:32:02.726165] INFO: transform: transformed rows: 3185604/3299472
[2017-07-21 17:32:02.761094] INFO: bigquant: transform.v2 end [36.037516s].
[2017-07-21 17:32:02.770371] INFO: bigquant: stock_ranker_predict.v2 start ..
[2017-07-21 17:32:06.559653] INFO: df2bin: prepare data: prediction ..
[2017-07-21 17:33:09.546235] INFO: stock_ranker_predict: prediction: 3185604 rows
[2017-07-21 17:34:31.189525] INFO: bigquant: stock_ranker_predict.v2 end [148.419163s].
[2017-07-21 17:35:36.633891] INFO: Performance: Simulated 1346 trading days out of 1346.
[2017-07-21 17:35:36.635703] INFO: Performance: first open: 2012-01-04 14:30:00+00:00
[2017-07-21 17:35:36.636863] INFO: Performance: last close: 2017-07-19 19:00:00+00:00
[注意] 有 547 笔卖出是在多天内完成的。当日卖出股票超过了当日股票交易的2.5%会出现这种情况。

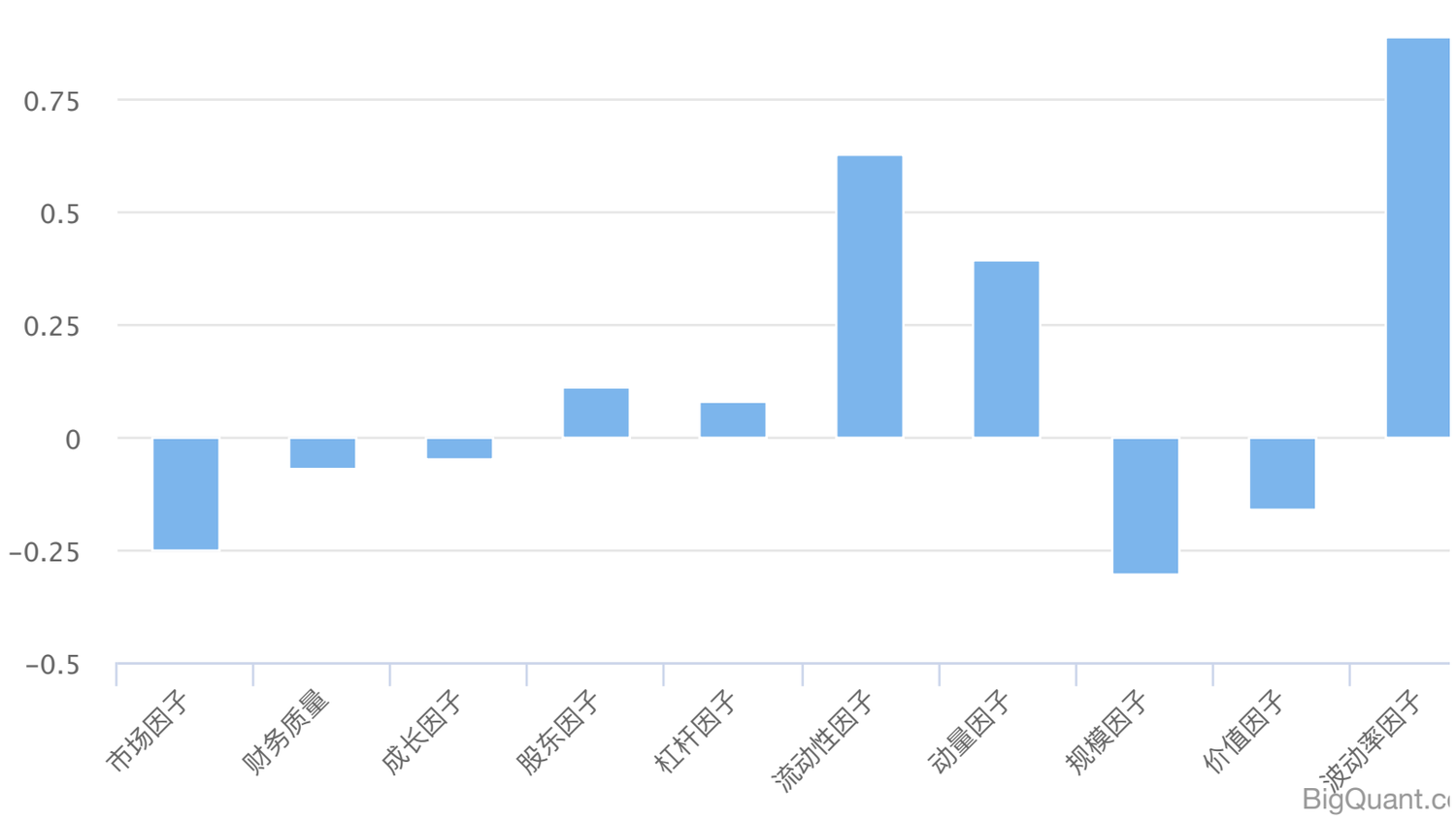
收益概况 | 交易详情 | 每日持仓和收益 | 输出日志

收益率	年化收益率	基准收益率	阿尔法	贝塔	夏普比率	收益波动率	信息比率	最大回撤
4031.94%	100.71%	59.0%	0.92	0.86	2.89	33.3%	3.53	44.71%

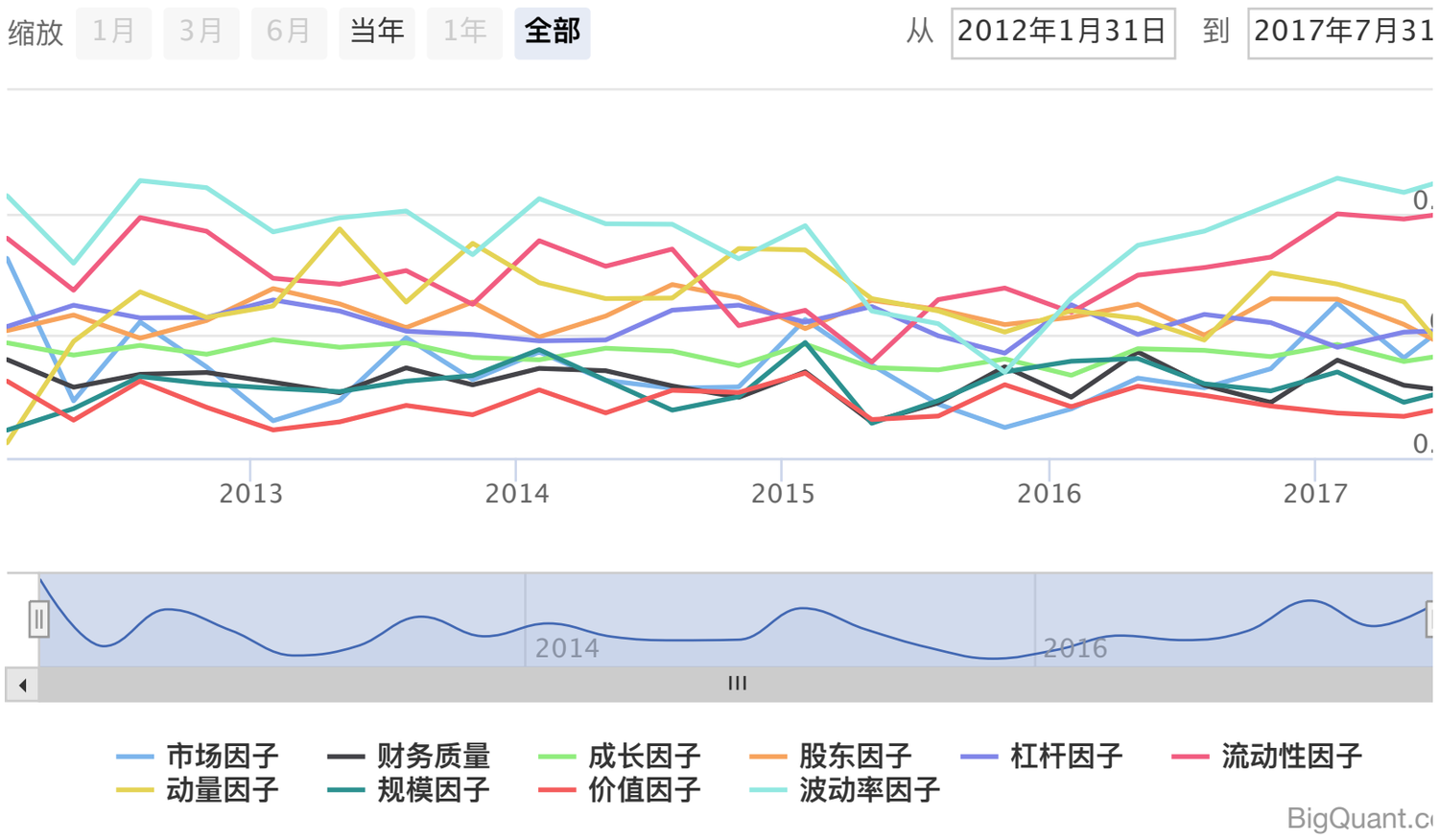
缩放 1月 3月 6月 当年 1年 全部 从 2012年1月 2日 到 2017年7月18



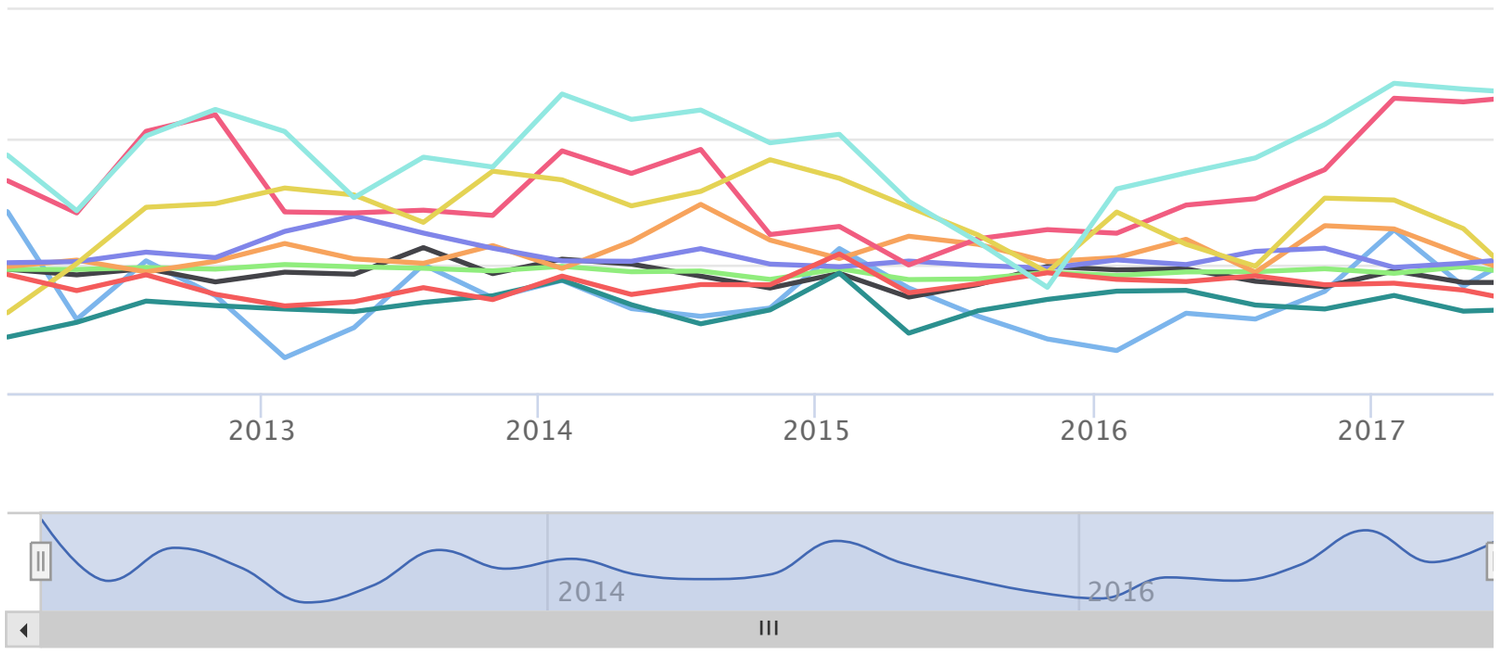
[2017-07-21 17:35:42.609953] INFO: bigquant: backtest.v6 end [413.427921s].



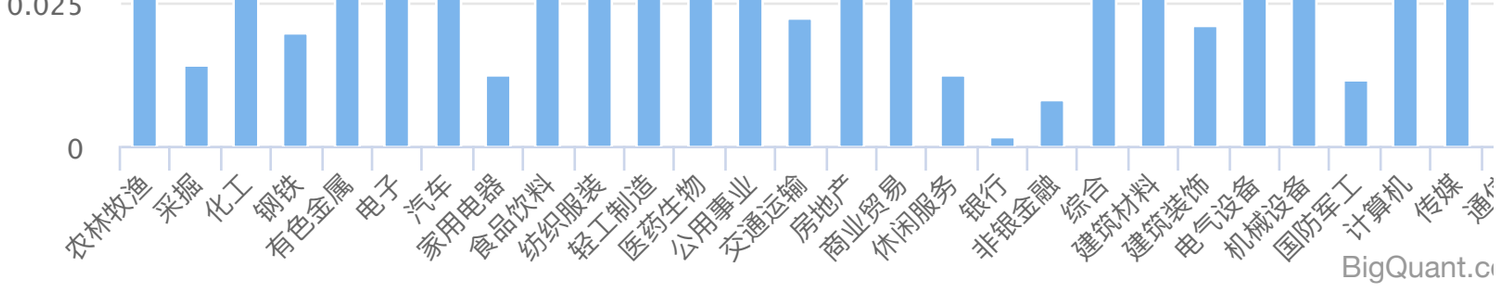
风险暴露趋势（排名）

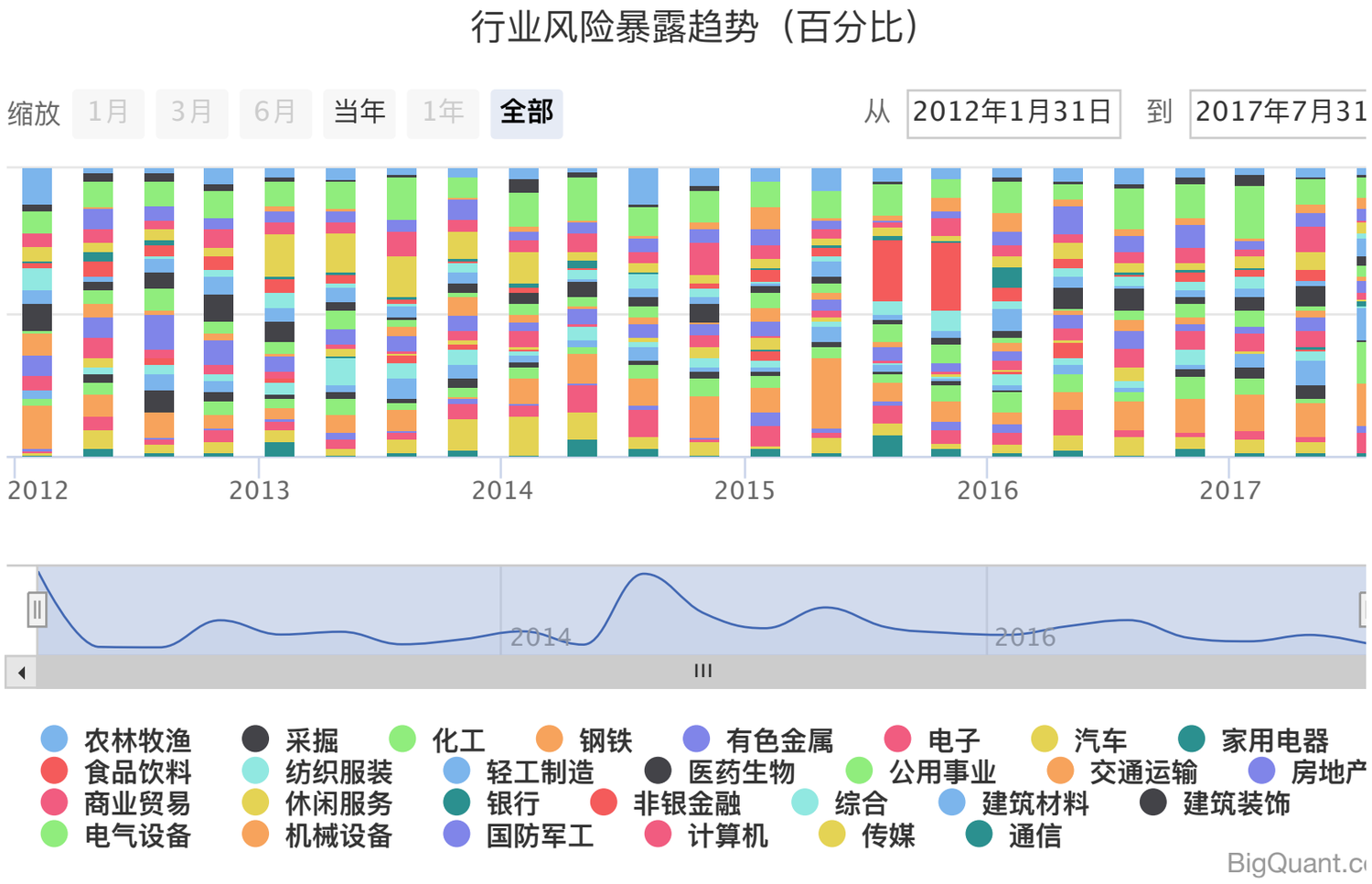


风险暴露趋势（绝对值）



区间行业风险暴露





主题	分类	用户	回复	活动
first blood • 近期 策略分享			1	2小时前
ai-自动 测试——初来乍到，第一个AI策略 策略分享			1	7小时前
上次访问				
[量化学堂-数学知识]峰度和偏度 策略分享, 峰度, 偏度, 正态性检验			0	2天前
[量化学堂-策略开发]使用cvxopt包实现马科维茨投资组合优化:以一个股票策略为例 策略分享, 组合优化, cvxopt, markowitz			0	2天前
在历史股票数据中寻找和目前沪深300表现相似K线 策略分享			1	3天前
Tensorflow第二讲 - MNIST 策略分享			0	3天前
收益率150%的测试策略，大家看看 策略分享			2	3天前
Tensorflow第三讲 - 深入MNIST（CNN） 策略分享			3	3天前
LSTM Networks应用于股票市场之Functional Model 策略分享, lstm			4	8天前
ROE策略 策略分享			2	13天前
策略研究平台常用快捷键使用指南 策略分享, 策略研究平台, 快捷键			0	13天前
电梯策略更新 策略分享			6	15天前

基于卷积神经网络的多因子预测 策略分享, 机器学习	0	17天前
Tensorflow第一讲 - 介绍及基本用法 策略分享, tensorflow	2	18天前
多策略组合收益分析 策略分享, 策略组合	2	18天前
电梯策略 策略分享	2	22天前
求助闭包调用失败，请大家帮助 策略分享	1	22天前
新手策略，大牛请指教 策略分享	3	23天前
大家看看这个策略如何? 策略分享	2	24天前
神经网络不胜语， M-P模型似可寻（深度学习入门系列之三） 策略分享, 机器学习, 用户成长系列	0	25天前
神经网络交易算法 策略分享, 机器学习	0	27天前
“漂亮50”策略尝试 v1 策略分享	0	28天前
“漂亮50”策略尝试 策略分享	0	28天前
小试牛刀 策略分享	1	6月16日
标准化、规范化、二值化等多种机器学习数据预处理方法 策略分享, 数据预处理	0	6月14日
贵州茅台的净利润相当于整个创业板的20%!!! 策略分享	0	6月13日
在计算平均换手率数据时出现报错 策略分享	1	6月5日
策略小测试，增加20日均线向上过滤 策略分享	4	6月4日
LSTM Networks应用于股票市场之Sequential Model 策略分享, lstm	1	5月25日
调试策略代码运行出bug,大家帮忙看看 策略分享	2	5月19日