

浙江省大学生证券投资竞赛 团队赛报告（决赛）

报告名称：基于动态多因子选股的 Alpha 策略设计

团队赛类别：量化组

学校名称：杭州电子科技大学

团队名称：索罗斯队

指导教师：胡文彬

团队队长：杨小敏

团队成员：蒙景辉 沈一帆 蒋慧凯 左利霞

2017 年 10 月

浙江省大学生证券投资竞赛委员会

浙江省大学生证券投资 竞赛团队报告

报告题目： 基于动态多因子选股的 Alpha 策略设计

参赛成员：杨小敏 蒙景辉 沈一帆 蒋慧凯 左利霞

指导老师：胡文彬

学校名称：杭州电子科技大学

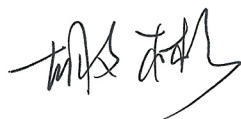
浙江省大学生证券投资竞赛

团队报告独创性声明及决赛操作独立性声明

本团队声明所呈交的投资报告是本团队成员在导师指导下，共同进行研究工作所取得的成果。除了文中特别加以标注的内容外，投资报告中不包含其他人已经发表或撰写过的作品成果。对本报告的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本团队成员完全意识到本声明的法律结果由团队成员承担。

本团队声明所呈交的投资操作报告是本团队成员在导师指导下，共同进行实际操作所取得的成果。对本报告涉及的实际操作都是真实的，没有虚构和造假。本团队成员完全意识到本声明的法律结果由团队成员承担。

指导老师签名：



团队成员签名：

杨小敏 左利霞 蒙景辉 沈一帆 蒋慧凯

签字日期：2017 年 10 月 23 日

基于动态多因子选股的 Alpha 策略设计

报告简介

策略介绍：本组模型是以 Alpha 策略为主体，运用动态多因子模型构建动态股票池，以反向操作股指期货的方式对冲掉系统性风险，并应用布林线进行风险控制的量化交易策略模型。

模型构建：本组首先采用动态多因子选股模型，以一个月为周期进行因子有效性检验并挑选出下一期的有效因子，再根据有效因子对待选股票进行排序打分，等权重动态构建股票池。然后在期货市场做空等额的沪深 300 股指期货，从而对冲掉系统性风险，得到 Alpha 收益。此外，在风险控制上，本组运用布林线策略进行止盈止损，结合一旦触及止盈止损线全仓卖出的仓位控制方法，以控制收益的波动性。以上几种策略的综合运用构成了本组的投资模型。

实证结果：本组策略有效对冲了系统性风险，取得了良好的超额收益，同时有效的控制了模型的波动性，收益稳定。

主要创新点：本组的研究逻辑是通过动态多因子选股挑选出具有超额收益的股票，并利用沪深 300 股指期货对冲掉市场风险，从而得到相对超额收益，收益相对稳定。通过市场回测检验，本组的策略风险可控，回撤较小，超额收益相对稳定。

目录

1. 量化投资策略的构建	4
1.1 策略构建整体流程	4
1.2 策略构建的总体思路	4
2. 策略构建的具体过程	6
2.1 动态多因子选股策略	6
2.1.1 候选因子池的建立	6
2.1.2 候选因子的有效性检验	9
2.1.3 股票池的建立	11
2.2 股指期货对冲	12
2.2.1 股指期货对冲流程	12
2.2.2 股指期货对冲方案	13
2.3 风险控制	14
2.3.1 止盈止损	14
2.3.2 仓位管理	15
3. 回测结果分析	17
3.1 部分模型回测检验	17
3.1.1 动态多因子选股模型回测检验	17
3.1.2 动态多因子选股及布林线组合模型回测检验	18
3.2 整体模型回测检验	19
3.2.1 综合市场行情长线回测	19
3.2.2 牛市市场行情回测	19
3.2.3 熊市市场行情回测	20
3.2.4 震荡市场行情回测	21
4. 总结	23
参考文献	24
附录	25

图表目录

图 1.1	策略流程图	4
图 2.1	动态多因子选股策略模型.....	6
表 2.2	因子类别.....	7
表 2.3	候选因子池.....	7
表 2.4	超低配组合数三种情况回测结果.....	11
表 2.5	测试期四种情况回测结果.....	11
表 2.6	整体策略回测结果.....	12
图 2.7	股指期货对冲方案流程.....	13
表 2.8	布林线情况分析.....	14
表 2.9	布林线上下轨五种情况回测结果.....	14
图 3.1	动态多因子选股回测图.....	17
图 3.2	动态多因子及布林线组合回测图.....	18
图 3.3	整体模型长线回测图.....	19
图 3.4	整体模型牛市市场行情回测图.....	20
图 3.5	整体模型熊市市场行情回测图.....	21
图 3.6	整体模型震荡市场行情回测图.....	22

1. 量化投资策略的构建

1.1 策略构建整体流程

策略的整体流程图见图 1.1:

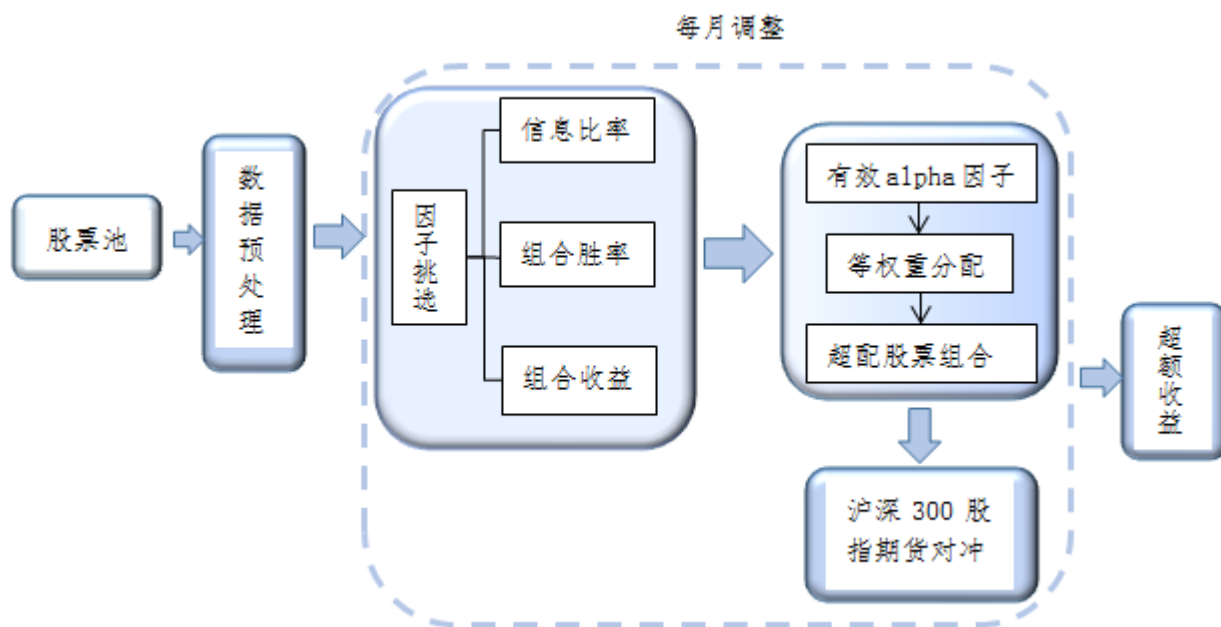


图 1.1 策略流程图

1.2 策略构建的总体思路

考虑到中性策略不同于一般的单方向策略，能够有效控制收益的波动性，本组选择的策略主体是 Alpha 策略，策略主要分为以下两个方面。

(1) 动态多因子选股

在 Alpha 策略的大背景下，本组选择动态多因子模型作为选股策略。选股在 Alpha 策略中是至关重要的一环，考虑到动态多因子选股不仅简单易行，有较好的稳健性，还兼具灵活性强、操作性强的特点，样本外的表现也较好。同时，动态多因子模型能更好的把握市场的变化。因此我们尝试构建动态多因子选股模型进行动态选股，帮助我们建立优质的股票组合，为策略的后续进行打下良好基础。

(2) 风险规避

考虑到在动态多因子选股模型中,候选股票池的建立是以沪深 300 成分股为基础进行的,本组在对系统风险的分离中,选择卖空等额的沪深 300 股指期货,过程中通过合约的动态调整以及同步的等额资金配置力求达到 1:1 完全对冲的对冲效果。

在利用股指期货对冲系统风险的同时,还应该对非系统风险进行有效控制。市场上的风险控制主要包括止盈止损和仓位管理两方面。止盈止损上,我们选择布林线通道法作为主要的止盈止损工具,针对不同的布林线三轨状态,判断是否买入卖出,或继续持有股票。仓位控制方面,我们是在止盈止损的前提下进行的,即当其触及止盈止损线时就全仓卖出。同时固定每月第 8 个交易日重新进行因子打分,构建股票池,调整股指期货,进行动态仓位控制。

2. 策略构建的具体过程

2.1 动态多因子选股策略

动态多因子选股策略模型如下图 2.1

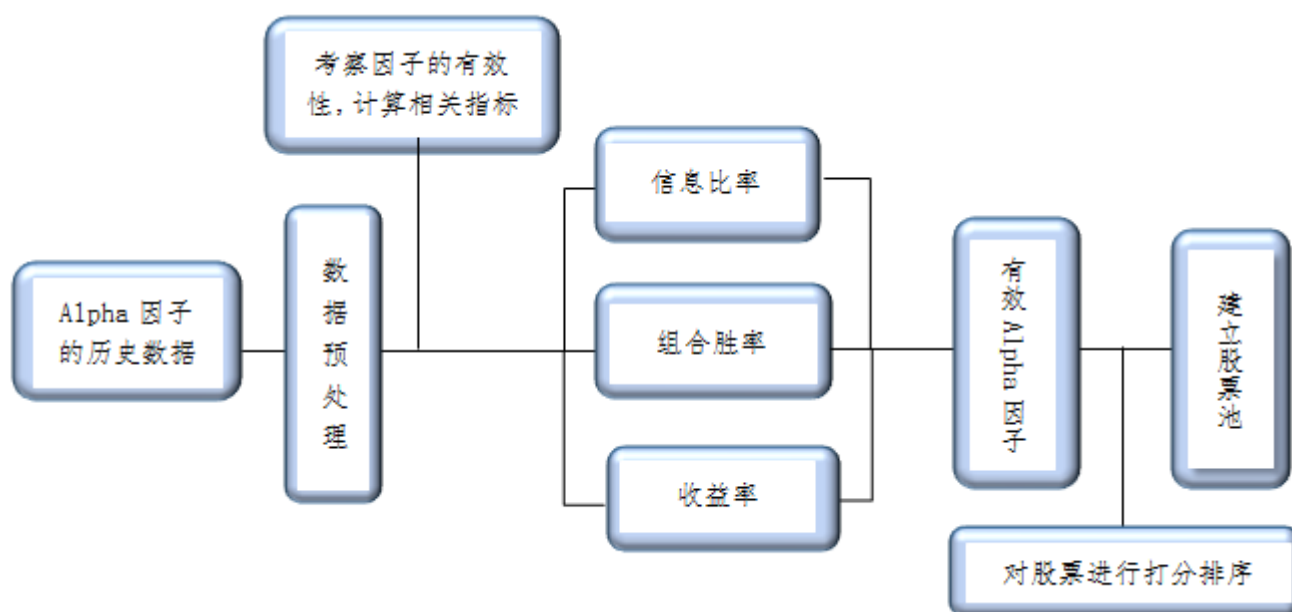


图 2.1 动态多因子选股策略模型

2.1.1 候选因子池的建立

考虑到不同的 alpha 因子会对个股的超额收益产生不同的影响，所以为了确保因子选择的有效性，本组基于从多角度多方向进行全面选择的基本策略思路来构建候选因子池。

我们最终选择两大类因子，包括财务因子和技术因子，其中又涵盖 8 类具体的因子，包括：估值因子、偿还能力因子、成长能力因子、营运能力因子、盈利能力因子、能量指标因子、量价指标因子、趋势型因子。

各类因子的意义描述如下表：

表 2.2 因子类别

因子类别		因子概述
财务因子	估值因子	企业内在价值水平
	偿还能力因子	企业以资抵债能力
	成长能力因子	企业未来发展能力
	营运能力因子	企业经营运营能力
	盈利能力因子	企业盈利能力
技术因子	能量指标因子	股市多空双方力量对比
	量价指标因子	股市量价关系对比
	趋势型因子	股市趋势走向

在上述 8 类因子中，我们选择了我们认为具有代表性的 74 个因子构建候选因子池。具体因子如下表：

表 2.3 候选因子池

编号	因子类别	候选因子	因子方向
1	估值因子	市盈率 PE	-1
2		市净率 PB	-1
3		市现率 PCF	-1
4		市销率 PS	-1
5		股息率	1
6		总市值	-1
7		流通市值	-1
8		总股本	-1
9		流通股本	-1
10	偿还能力因子	流动比率	1
11		产权比率	-1
12		速动比率	1
13		有形资产/负债合计	1
14		有形资产/带息债务	1
15		净负债/股权价值	-1

16		长期债务与营运资金比率	-1
17		有形资产/净债务	1
18	成长能力因子	营业总收入增长率（同比）	1
19		每股经营活动产生的现金流量净额	1
20		营业利润(同比增长率)	1
21		利润总额(同比增长率)	1
22		净资产收益率(摊薄)(同比增长率)	1
23		经营活动产生的现金流量净额(同比)	1
24		净利润增长率（同比）	1
25		基本每股收益增长率（同比）	1
26	营运能力因子	总资产周转率	1
27		应付账款周转率	1
28		流动资产周转率	1
29		固定资产周转率	1
30		现金循环周期	-1
31		存货周转率	1
32		应收账款周转率	1
33	盈利能力因子	净资产收益率 ROE（同比）	1
34		总资产净利率 ROA	1
35		销售净利率	1
36		息税前利润/营业总收入	1
37		销售费用/营业总收入	-1
38		总资产报酬率 roa	1
39		销售成本率	-1
40		净利润/营业总收入	1
41		营业利润/营业总收入	1
42		营业总成本/营业总收入	-1
43		管理费用/营业总收入	-1
44		财务费用/营业总收入	-1
45	能量指标因子	VR 成交量比率	1
46		VSTD	-1
47		ARBR 人气意愿指标	1
48		SRDM30	1
49		VROC12	1

50		VRSI6	1
51		CR 能量指标	1
52		MF1 资金流向指标	1
53		5 日量比	1
54		MASS 梅丝线	1
55	量价指标因子	OBV 能量潮	1
56		PVT 量价趋势指标	1
57		WAD30	1
58	趋势型因子	BB1 多空指数	1
59		MTM5	1
60		DMA 平均线差	1
61		MA5	1
62		MACD 指数平滑异同平均	1
63		EXPMA5	1
64		PRICEOSC 价格振荡指标	1
65		TRIX12	1
66		DBCD 异同离差乖离率	1
67		DPO 区间震荡线	1
68		PSY12	1
69		VMA5	1
70		VMACD 量指数平滑异同平均	1
71		VOsc 成交量震荡	1
72		TAPI 加权指数成交值	1
73		MICD 异同离差动力指数	1
74		RCCD 异同离差变化率指数	1

2.1.2 候选因子的有效性检验

有效的选股因子能够帮助策略获得长期、稳定的 Alpha 收益，与此同时，该因子在各个时间段应该具备较低的波动性。动态多因子选股模型的核心就是通过因子有效性的检验挑选出能获得超额收益的因子组合，并通过动态变化因子把握市场形势。我们对备选因子进行有效性评价，旨在通过比较股票的超额收益与因子之间的关系，找出能够有效预测股票未来收益的因子。

1) 数据的预处理

获取当天的因子数据,再根据因子的大小及方向对沪深 300 的成分股进行排序,取其一定数量的股票作为超配组合,最后相同数量的股票作为低配组合,以 22 个交易日为周期,获取在此之前一定期限内的股价数据,并计算月收益率,取其均值,分别得到超配组合收益率和低配组合收益率。

2) 因子度量指标的选择

本组使用多个指标相结合的方式筛选候选因子是否有效。以下是我们选取的三个评价因子有效性的指标:

a) 信息比率: 信息比率 = (超配组合收益率 - 低配组合收益率) / 超额收益标准差, 该指标的值越高, 表明因子在挑选具有 Alpha 收益的个股的表现越好。

b) 组合胜率: 组合胜率 = (超配组合收益率 - 低配组合收益率) > 0 的次数 / 测试期 (按月), 该指标表示超配组合跑赢低配组合的时间比例, 也就是表明有多少时间该因子是有效的。该指标的值越高, 因子的有效性越好。

c) 收益率: 即超配组合的收益率, 该指标的大小表示因子的收益是否稳定和可持续。该指标的值越高, 则因子的有效性越好。

3) 候选因子的初步筛选

初步筛选过程如下:

a) 依据每个指标的大小对因子进行排序, 因子的顺序即因子得分

b) 依据加权法算出最后得分

由于信息比率和胜率相对于其他两个指标对超额收益的预测影响更大, 所以我们在计算每个因子的综合得分时, 为其分配的比重较大。因此我们按照如下比重计算因子最后得分: 信息比率 2/5, 组合胜率 2/5, 收益率 1/5。

c) 确定有效因子

算出综合得分后对因子按照得分从大到小进行排序, 最终选择得分排名前五的因子。

4) 确定超低配组合数及测试期

为确定测试因子有效性时构建的超低配组合数为多少及测试期为多久时, 动态多因子策略表现最为突出, 我们用了动态多因子的程序进行回测。

以下是超低配组合数三种情况回测结果:

表 2.4 超低配组合数三种情况回测结果

持股数	超低配组合	策略收益(%)	Alpha (%)	Bata	最大回撤(%)
15	5	79.49	0.08	0.86	50.06
15	10	41.08	-0.00	0.75	45.82
15	15	31.25	-0.05	0.85	41.38

分析上表可得，当持股是为 15 时，测试因子指标时构建的超低配组合为 5 只个股时，表现总体相对更好。其策略收益为 79.49，并且取得了 0.08 的超额收益，相较于其他两种情况，表现均是最好的。但是 Beta 和最大回撤都稍大于其他两种情况，说明其对冲效果没有其他情况好，但也并非相差甚大。

综合考虑各因素，我们确定超低配组合采用 5 只个股。

以下是测试期四种情况回测结果：

表 2.5 测试期四种情况回测结果

时间段	策略收益 (%)	Alpha (%)	Bata	最大回撤 (%)
3 个月	31.25	-0.05	0.85	41.38
6 个月	41.08	0.00	0.75	45.82
9 个月	79.49	0.08	0.86	50.06
12 个月	34.14	-0.05	0.92	62.70

分析上表可知，在对用以计算组合收益率的股价数据的时间段进行选择时，选择前九个月的数据进行计算，相较于其他时间段，无论是策略收益还是阿尔法收益都有明显的优势，但最大回撤数值略高。综合考虑各因素的重要性，我们选择前九个月的股价数据进行计算。

2.1.3 股票池的建立

1) 候选股票池的建立

备选股票池从沪深 300 指数所包含的成分股中选取。

2) 单个因子对股票运用五分法进行排序打分

经过冗余因子的剔除后，我们已经挑选出了最终的有效因子，同时备选股票池也已经建立。在此基础上，我们在考虑因子方向的同时，对所有股票进行排序

打分，打分方法采用五分法打分方法，即将所有的股票从上到下划分为五等份，并从上到下给每份的股票依次给予 5 分、4 分、3 分、2 分、1 分的打分。

3) 根据得分对股票排序

前面对所有股票打分完成后，我们再根据等权重的方法算出每只股票的最终得分，然后按照股票得分从高到低的顺序对所有股票进行排序。

4) 确定持股数，构建最终股票池

为确定持股数为多少时，动态多因子策略表现最为突出，我们用了动态多因子的程序进行回测。

以下是持股数三种情况回测结果：

表 2.6 持股数三种情况回测结果

持股数	超配组合	策略收益(%)	Alpha (%)	Bata	最大回撤(%)
10	5	58.40	0.01	0.98	51.08
15	5	79.49	0.08	0.86	50.06
20	5	55.00	0.00	1.01	56.62

在动态多因子策略下，持股数为 15 时的策略收益及 alpha 收益明显要优于其他两种情况。其最大回撤虽和其他两种情况差异不大，但是仍是其中最低。

综上，我们确定最终的持股数为 15。

确定好持股数后，便可以构建我们最后的股票池。在此后的每个月第 8 个交易日，我们根据前一天筛选的因子对股票进行打分，构建新的股票池，然后进行一次调仓，即剔除掉不符合条件的股票，同时买进新的符合筛选条件的股票。这样我们交易的股票池也随着市场的变化在不断调整。

2.2 股指期货对冲

2.2.1 股指期货对冲流程

股指期货对冲方案的原理及流程如下图 2.7

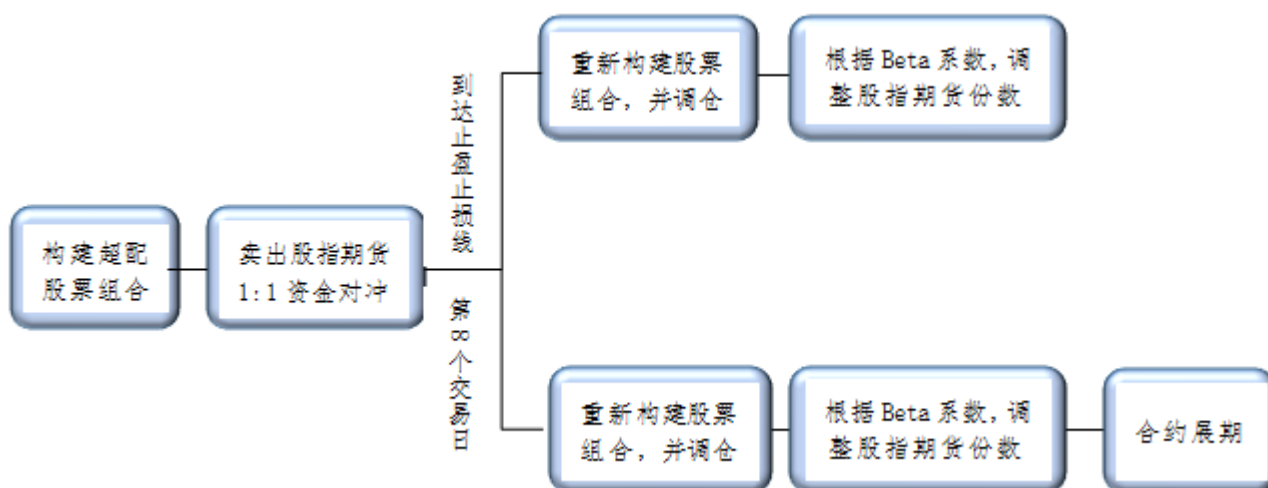


图 2.7 股指期货对冲方案流程

2.2.2 股指期货对冲方案

1) 保证金的管理

由于固定初始资金为 1000 万元，考虑到股票收益及保证金政策，我们小组决定固定预留总资金的 40% 设立保证金账户，剩余 60% 的资金用来构建股票超配组合。

2) 具体策略实现

策略执行初期，首先构建一个股票组合，等权重分配资金。同时在期货市场，做空等额的沪深 300 股指期货进行对冲。

在计算对冲所需股指期货合约数量的过程中，我们首先根据 $\beta = X_1\beta_1 + X_2\beta_2 + \dots + X_n\beta_n$ （股票组合中个股的 β 系数记作 β_i ）计算超配股票组合的 β 系数。

继而根据 套期保值买卖期货合约数 = $\frac{\text{现货总价值}}{\text{期货指数点} \times \text{每点乘数}} \times \beta$ 系数 公式

计算所需购买的合约份数，力求实现 1:1 的完全对冲。

策略执行过程中，每月第 8 个交易日重新构建股票组合，调整期货合约。另外，当触及止盈止损线时也对持仓和股指期货进行调整。在合约数量的选择上，

我们根据期初超配组合股票市值的规模来计算持有股指期货空头合约的数量，在合约存续期内均保持此合约数量不变。

3) 合约的调整

由于股指期货合约的最后结算交易日为合约到期月份的第三个周五，所以我们选定合约到期月份的第 8 个交易日根据新筛选的因子组合构建新的超配股票组合，并进行调仓，先买入当月股指期货进行平仓，再根据超配股票组合的市场价值计算股指期货和约的数量，后卖空股指期货，从而实现完全对冲。

若股票组合达到其止盈止损要求，则重新构造超配股票组合。与此同时，根据超配股票组合的市场价值调整股指期货和约的数量，若在每月第 8 个交易日之前达到止盈止损要求，则买入或卖空当月期货合约，若在第 8 个交易日之后达到止盈止损要求，则买入或卖空下月期货合约。

2.3 风险控制

2.3.1 止盈止损

本组的止盈止损是采用的布林线通道法。

布林线由上轨、下轨和中轨组成，中轨就是简单的移动平均线。本组将布林线通道使用于股票组合的止盈止损，中轨上的每一点均使用前 20 天的股票组合的股价平均值。我们的上下轨通过中轨加减一定倍数的标准差计算得到。我们根据判断当前前五天的股价平均值来确认布林线三轨的状态。

布林线通道法在使用中共有四种情况，具体表现及分析如下：

表 2.8 布林线情况分析

	上下轨表现	形成原因
开口	上轨向上，下轨向下	价格快速大幅运行，突破上下轨
收口	上下轨向中轨聚拢	价格陷入行情后整理阶段
三轨同向	上中下三轨指向同一方向	价格经过一波上涨或下跌后，小幅回调在中轨得到支撑或阻力，

再次向上或向下运行		
走平	三轨同时横向运行	行情陷入窄幅区间整理

为了及时准确的判断布林线的四种情况，本组决定五天判断一次是否触发布林线，经过实践，其效果最好。

当布林线通道处于开口阶段，止损点设在中轨的略上方，一旦股价突破中轨，便将股票组合全部卖出，越早出场越好。而收口阶段则是高抛低吸区间操作的良机，本组将止损点设置在判断日当天的中轨的略上方和略下方。当布林线通道处于三轨同向阶段，本组选择将止损点设在中轨的略下方，一旦股价跌破中轨，就尽快出场。而走平阶段，本组则先继续持有股票，根据具体情况，再进一步做出判断。

为确定上下轨为中轨加减多少倍标准差时，总体表现最为突出，我们用了动态多因子加布林线的程序进行回测。

以下是布林线上下轨五种情况回测结果：

表 2.9 布林线上下轨五种情况回测结果

上轨	下轨	策略收益(%)	Alpha (%)	Bata	最大回撤(%)
1	1	43.97	-0.02	0.92	52.93
2	1	109.98	0.15	0.90	50.85
3	1	85.06	0.08	0.97	53.54
2	2	127.34	0.19	0.82	43.92
3	2	52.42	0.02	0.77	39.74

分析上表可得，当上轨采用中轨加 2 倍标准差，下轨采用中轨减 2 倍标准差时，表现总体相对较好。其最大回撤和 Bata 都较令人满意，对冲效果相对将会更好。同时也取得了 0.19 的超额收益，是这 5 种情况中最高的，且策略收益相较于其他情况更为亮眼。

综上，我们确定上下轨采用中轨加减两倍标准差得出。

2.3.2 仓位管理

考虑到手续费等因素，我们小组采用的仓位管理策略是在策略执行初期，全仓买入超配股票组合。策略执行期间，一旦股票组合触及止盈止损线，即重新构建超配股票组合。比较前后两个超配股票组合的股票范围差异，卖出前股票组合中不在新超配股票组合范围内的股票，买入新超配股票组合中不在前股票组合中的股票，进行动态调仓。

3. 回测结果分析

为了验证策略的有效性，本组分别针对动态多因子选股，动态多因子选股与布林线风险控制的组合进行部分策略回测，用以证明其在整体策略中的必要性以及其对最终收益的贡献程度。同时，本组针对不同的市场行情，对总体策略分别进行综合市场行情回测、牛市市场行情回测、熊市市场行情回测和震荡市场行情回测，用以证明本组策略在不同市场行情中的具体表现，验证策略的有效性。

3.1 部分模型回测检验

3.1.1 动态多因子选股模型回测检验

回测股票区间：沪深 300 成分股

回测时间段：2014 年 3 月 1 日—2017 年 2 月 1 日

回测结果：



图 3.1 动态多因子选股回测图

回测结果指标如下：

策略收益	基准收益	α 值	夏普比率	最大回撤
68.49%	55.11%	0.05	0.54	56.17%

回测分析：动态多因子选股模型的策略表现如上图所示。从总体来看，本组的动态多因子选股模型能有效跟踪大盘趋势，并且不论市场行情如何变化，都能

能够超越大盘，取得稳定收益。最终策略收益较基准收益超出 13.38%，并取得 0.05 的超额收益。表明本组策略在候选因子有效性检验的打分机制上是有效的，最终选择的择股因子能够挑选出我们需要的具有超额收益的股票。但是由于未加入风险控制，所以最大回撤偏高。

3.1.2 动态多因子选股及布林线组合模型回测检验

回测股票区间：沪深 300 成分股

回测时间段：2014 年 3 月 1 日—2017 年 2 月 1 日

回测结果：



图 3.2 动态多因子选股及布林线组合回测图

回测结果指标如下：

策略收益	基准收益	α 值	夏普比率	最大回撤
103.72%	55.11%	0.13	0.78	45.98%

回测分析：单从两条线的走势看，本组的风险控制在各个时间点与市场有较大的分离，可见我们的布林线风险控制方法起到了一定的止损并控制风险的作用。加入了布林线进行风险控制后，策略收益较基准收益高了 48.61%，并且取得了 0.13 的超额收益，且其最大回撤相较于单多因子选股有了一定的下降，但还是较高。

3.2 整体模型回测检验

3.2.1 综合市场行情长线回测

回测股票区间：沪深 300 成分股

回测时间段：2014 年 3 月 1 日—2017 年 2 月 1 日

回测结果：



图 3.3 整体模型长线回测图

回测结果指标如下：

策略收益	基准收益	α 值	夏普比率	最大回撤
68.36%	55.11%	0.14	0.99	17.07%

回测分析：本组整体模型的回测包括动态多因子选股、布林线风险控制以及股指期货期货对冲。在复杂市场行情中，取得 68.36% 的策略收益和 0.14 的超额收益，夏普比率 0.99。同时有效控制收益的波动性，相较于部分模型回测的结果，最大回撤显著降低，控制在 17.07%。 β 值从 0.91 下降到 0.18，虽未实现完全对冲，但基本实现了对系统性风险的分离。

3.2.2 牛市市场行情回测

回测股票区间：沪深 300 成分股

回测时间段：2014 年 3 月 1 日—2015 年 6 月 7 日

回测结果：



图 3.4 整体模型牛市市场行情回测图

回测结果指标如下：

策略收益	基准收益	α 值	夏普比率	最大回撤
74.37%	139.46%	0.49	1.80	9.00%

回测分析：本身阿尔法策略并不适合牛市市场行情，无法有效的捕捉迅速上涨的市场整体收益，在牛市市场上存在局限性。本组的整体策略在市场行情大幅走高时期，能够有效获得 0.49 的超额收益，收益波动性较低，单位风险收益良好，但也丧失了获得大幅受益的良机，模型本身存在不足之处。

3.2.3 熊市市场行情回测

回测股票区间：沪深 300 成分股

回测时间段：2015 年 6 月 7 日—2016 年 2 月 1 日

回测结果：



图 3.5 整体模型熊市市场行情回测图

回测结果指标如下：

策略收益	基准收益	α 值	夏普比率	最大回撤
-13.44%	-45.44%	-0.17	-1.16	18.63%

回测分析：在市场行情大幅走低时，本组模型通过股指期货对冲，成功分离系统性风险，有效控制收益的波动性，最大回撤控制在 **18.63%**，并且成功止损。但选股策略表现欠佳，**alpha** 值为负，未能获得预期的超额收益，模型本身有待改进。

3.2.4 震荡市场行情回测

回测股票区间：沪深 300 成分股

回测时间段：2016 年 2 月 1 日—2017 年 2 月 1 日

回测结果：



图 3.6 整体模型震荡市场行情回测图

回测结果指标如下：

策略收益	基准收益	α 值	夏普比率	最大回撤
12.33%	14.41%	0.05	0.89	6.42%

回测分析：在市场行情趋势不明，区间震荡时期，本组模型能够有效控制收益波动性，最大回撤控制在 6.24%，取得 0.05 的超额收益。但是策略 β 值偏高，对冲效果的优势并不明显。

4. 总结

本组策略的主体是 **Alpha** 策略，即通过动态多因子模型选出具有 **Alpha** 收益的股票，在股指期货的有效对冲下，在止盈止损保障收益的情况下，我们基本能够获取到所要的 **Alpha** 收益，并且是在最大回撤在适当范围内的情况下。但是在一些细节上，还有一些问题有待优化。

在选股上，本组策略采用的是动态多因子选股方法。动态调整因子的方法，使得最终筛选出来的股票能够把握市场的形势。本组的因子选择在全面性这方面上还做得不够，因为每个因子的影响程度是不同的。

在仓位控制上，我们采取的是全仓买入、全仓卖出。虽然这种做法不够灵活，对市场的适应性也不够强，但 **Alpha** 策略对择时的要求并不高，太复杂的仓控策略反而会成为累赘。因此，综合考虑下，这种方法在 **Alpha** 策略下还是比较适用的。

在止盈止损方面，我们的采取布林线控制风险，但效果还是不佳，只是零星的触发了几次，回撤仍然较大，不能有效地阻止较大亏损的出现。

综合以上几方面，我们的策略能获得稳定的风险极低的 **Alpha** 收益，虽然有些小问题，但对整体策略影响不大，还是较成功地实现了 **Alpha** 策略。

参考文献

- [1] 分析师徐莉莉. 量化投资在中国的发展现状. 渤海证券. 2012.9
- [2] 丁鹏. 量化投资. 策略与技术[M]. 电子工业出版社. 2012
- [3] 分析师罗军、胡海涛. 2011 年金融工程研讨会专题报告系列之二——大浪淘金, Alpha 因子何处寻. 广发证券. 2011.8
- [4] 浅析布林线四大状态及止损设置技巧. 2013.11
- [5] 阿尔法策略的三大优势和瓶颈. 2013
- [6] 首席分析师罗军. 基于多因子选股的量化对冲方案分析——转融通: 双刃剑之“惑”. 广发证券. 2012.6
- [7] 量化课堂——股指期货对冲策略 - 知乎专栏. 2017
- [8] 罗军、李明、胡海涛、蓝昭钦. 2010 年中期量化投资专题系列报告三——沪深 300 成分股的多因子模型有效因子选择框架构建. 山西证券. 2013.11
- [9] 分析师罗军、胡海涛. 沪深 300 成分股的应用分析. 广发证券. 2011.5
- [10] 分析师史庆盛. 基于风格特征归因的动态因子策略. 2013.6
- [11] 分析师史庆盛. 基于风格回复的多因子动态调仓策略. 2014.8

附录

```
import pandas as pd
import numpy as np
import datetime
import statsmodels.api as sm
import talib
import math

def initialize(account):
    set_log_level('warn')
    #智能选股
    get_iwencai('沪深 300')
    #设置交易周期
    account.pp=5
    #设置最大持股数
    account.max_stocks=10
    #设置筛选因子个数
    account.max_need=5
    #设置因子测试期数
    account.fpn=9
    #设置检验因子所用的股票数
    account.samt=5
    #设置子账户

    set_subportfolios([{'cash':6000000,'type':'stock'},{'cash':4000000,'type':'future'}]
)

    #设置计数起点
    account.cday=0
    #设置因子池
    account.need=['factor.pe',
                  'factor.pb',
                  'factor.pcf_cash_flow_ttm',
                  'factor.ps',
                  'factor.dividend_rate',
                  'factor.market_cap',
                  'factor.current_market_cap',
                  'factor.capitalization',
                  'factor.circulating_cap',
                  'factor.current_ratio',
                  'factor.equity_ratio',
                  'factor.quick_ratio',
                  'factor.tangible_assets_liabilities',
```

'factor.tangible_assets_int_liabilities',
'factor.net_debt_equity',
'factor.long_term_debt_to_opt_capital_ratio',
'factor.tangible_assets_net_liabilities',
'factor.overall_income_growth_ratio',
'factor.net_cashflow_psg_rowth_ratio',
'factor.opt_profit_grow_ratio',
'factor.total_profit_growth_ratio',
'factor.diluted_net_asset_growth_ratio',
'factor.net_cashflow_from_opt_act_growth_ratio',
'factor.net_profit_growth_ratio',
'factor.basic_pey_ear_growth_ratio',
'factor.turnover_of_overall_assets',
'factor.turnover_ratio_of_account_payable',
'factor.turnover_of_current_assets',
'factor.turnover_of_fixed_assets',
'factor.cash_cycle',
'factor.inventory_turnover_ratio',
'factor.turnover_ratio_of_receivable',
'factor.weighted_roe',
'factor.overall_assets_net_income_ratio',
'factor.net_profit_margin_on_sales',
'factor.before_tax_profit_div_income',
'factor.sale_cost_div_income',
'factor.roa',
'factor.ratio_of_sales_to_cost',
'factor.net_profit_div_income',
'factor.opt_profit_div_income',
'factor.opt_cost_div_income',
'factor.administration_cost_div_income',
'factor.financing_cost_div_income',
'factor.vr_rate',
'factor.vstd',
'factor.arbr',
'factor.srdm',
'factor.vroc',
'factor.vrsi',
'factor.cr',
'factor.mfi',
'factor.vr',
'factor.mass',
'factor.obv',
'factor.pvt',
'factor.wad',

```
'factor.bbi',  
'factor.mtm',  
'factor.dma',  
'factor.ma',  
'factor.macd',  
'factor.expma',  
'factor.priceosc',  
'factor.trix',  
'factor.dbcd',  
'factor.dpo',  
'factor.psy',  
'factor.vma',  
'factor.vmacd',  
'factor.vosc',  
'factor.tapi',  
'factor.micd',  
'factor.rccd']
```

#设置股票因子及其方向

```
account.mfactors={'factor.pe':-1,  
                  'factor.pb':-1,  
                  'factor.pcf_cash_flow_ttm':-1,  
                  'factor.ps':-1,  
                  'factor.dividend_rate':1,  
                  'factor.market_cap':-1,  
                  'factor.current_market_cap':-1,  
                  'factor.capitalization':-1,  
                  'factor.circulating_cap':-1,  
                  'factor.current_ratio':1,  
                  'factor.equity_ratio':-1,  
                  'factor.quick_ratio':1,  
                  'factor.tangible_assets_liabilities':1,  
                  'factor.tangible_assets_int_liabilities':1,  
                  'factor.net_debt_equity':-1,  
                  'factor.long_term_debt_to_opt_capital_ratio':-1,  
                  'factor.tangible_assets_net_liabilities':1,  
                  'factor.overall_income_growth_ratio':1,  
                  'factor.net_cashflow_psg_rowth_ratio':1,  
                  'factor.opt_profit_grow_ratio':1,  
                  'factor.total_profit_growth_ratio':1,  
                  'factor.diluted_net_asset_growth_ratio':1,  
                  'factor.net_cashflow_from_opt_act_growth_ratio':1,  
                  'factor.net_profit_growth_ratio':1,  
                  'factor.basic_peg_ratio':1,  
                  'factor.turnover_of_overall_assets':1,
```

'factor.turnover_ratio_of_account_payable':1,
'factor.turnover_of_current_assets':1,
'factor.turnover_of_fixed_assets':1,
'factor.cash_cycle':-1,
'factor.inventory_turnover_ratio':1,
'factor.turnover_ratio_of_receivable':1,
'factor.weighted_roe':1,
'factor.overall_assets_net_income_ratio':1,
'factor.net_profit_margin_on_sales':1,
'factor.before_tax_profit_div_income':1,
'factor.sale_cost_div_income':-1,
'factor.roa':1,
'factor.ratio_of_sales_to_cost':-1,
'factor.net_profit_div_income':1,
'factor.opt_profit_div_income':1,
'factor.opt_cost_div_income':-1,
'factor.administration_cost_div_income':-1,
'factor.financing_cost_div_income':-1,
'factor.vr_rate':1,
'factor.vstd':-1,
'factor.arbr':1,
'factor.srdm':1,
'factor.vroc':1,
'factor.vrsi':1,
'factor.cr':1,
'factor.mfi':1,
'factor.vr':1,
'factor.mass':1,
'factor.obv':1,
'factor.pvt':1,
'factor.wad':1,
'factor.bbi':1,
'factor.mtm':1,
'factor.dma':1,
'factor.ma':1,
'factor.macd':1,
'factor.expma':1,
'factor.priceosc':1,
'factor.trix':1,
'factor.dbcd':1,
'factor.dpo':1,
'factor.psy':1,
'factor.vma':1,
'factor.vmacd':1,

```

        'factor.vosc':1,
        'factor.tapi':1,
        'factor.micd':1,
        'factor.rccd':1}
account.lastlong={}      #超配股票池
account.lastshort={}    #低配股票池
account.lastlong=account.lastlong.fromkeys(account.need)
account.lastshort=account.lastshort.fromkeys(account.need)

account.winr={}
account.ir={}
account.incomer={}
#log.info(account.winr)
#设置默认因子
account.realneed=['factor.pe','factor.weighted_roe',

'factor.net_cashflow_from_opt_act_growth_ratio','factor.overall_income_growt
h_ratio',

'factor.turnover_ratio_of_account_payable']
#设置打分机制
account.score=5
#定期运行函数
schedule_function(reallocate,date_rule=date_rules.month_start(8))

def reallocate(account,data):
    #log.info('各单位注意，这是每月换血，请主动配合')

    tempstocks=give_me_stocks(account,data,is_re=1)      #      前
maxstocks 只股票带综合分数
    nstocks=[]
    for i in range(len(tempstocks)):
        nstocks.append(tempstocks[i][0])

    if(account.cday==0):
        cash=account.cash/account.max_stocks
    else:
        cash=account.positions_value/account.max_stocks

    for i in list(account.positions):
        if(i not in nstocks[:account.max_stocks]):

```

```

        order_target(i,0)
    for i in nstocks:
        order_target_value(i,cash)
        if(len(account.positions)==account.max_stocks):
            break
    Astocks=[]
    tstocks=dict(tempstocks)
    log.info(list(account.positions))
    log.info(tstocks)
    ctr=0
    for i in list(account.positions):
        if i not in tstocks:
            continue
        Astocks.append((i,tstocks[i]))
        ctr+=1
    log.info(ctr)
    Amount=give_me_amount(account,data,*Astocks)
    code=get_future_code('IF','next_month')      #下月期货
    if(len(account.subportfolios.positions)==0):    #若子账户期货
为    空，说明当月期货已经交割，入手下月
        order(code,Amount,pindex=1,type='short')
    else:
        current_code=list(account.subportfolios.positions)[0]
#手头期货

        number=account.subportfolios.positions[current_code].total_amount    #手 头
期货份数

        if(code==current_code):        #如果手头的是下月期
货，直接调仓

            if Amount>number:

                order(current_code,-number+Amount,pindex=1,type='short')    # 做 空 期
货，卖出

            else:

                order(current_code,number-Amount,pindex=1,type='long')    #做多期货，买
回

            else:
                order(current_code,-number,pindex=1,type='long')

#平仓买回

                order(code,Amount,pindex=1,type='short')    #
做空下月期货

```

```

def give_me_amount(account,data,*tempstocks):
    #计算 Beta 值
    beta=[]
    percent=[]
    nstocks=[]
    rstocks=[]
    sum1=0.0

    for i in range(len(tempstocks)):
        sum1+=tempstocks[i][1]
        rstocks.append(tempstocks[i][1])
        nstocks.append(tempstocks[i][0])

    #计算权重
    for i in range(len(rstocks)):
        percent.append(rstocks[i]/sum1)

    for i in nstocks:      #迭代股票池

        iclose=get_price(['000300.SH',i],None,get_datetime().strftime('%Y%m%d'),'1d',['
close'],None,None,220) #证券
        xclose=iclose['000300.SH']['close']
        yclose=iclose[i]['close']

        if(len(yclose)==0):
            #log.info('糟糕,yclose 长度是 0')
            beta.append(0)
            continue

        if(len(xclose)!=len(yclose)):
            #log.info(['iclose 是',iclose])
            l=min(len(xclose),len(yclose))
            xclose=xclose[-l:]
            yclose=yclose[-l:]

        x=np.array(xclose)
        y=np.array(yclose)

        rm=(x[1:]-x[0:-1])/x[0:-1]
        r1=(y[1:]-y[0:-1])/y[0:-1]
        est=sm.OLS(r1,sm.add_constant(rm))
        est=est.fit() #单只股票的 beta
        beta.append(est.params[1])

```

```

Beta=0.0
for i in range(len(beta)):
    Beta+=beta[i]*percent[i] #Beat 等于因子打分权重之和
    #log.info(['Beta 是',Beta])
    close=data.attribute_history('000300.SH',['close'],1,'1d')    # 期货
开盘价

    #log.info(['沪深 300 股指期货的开盘价是',close])
    if(account.cday==0):
        cash=account.cash*1.0/account.max_stocks
    else:
        cash=account.positions_value*1.0/account.max_stocks

Amount=math.ceil((Beta*(cash*account.max_stocks))/(300*close['close'][0]))
    #log.info(['Amount 是',Amount])
    return int(Amount)

def give_me_need(account,data):
    n=len(account.mfactors)
    querc=', '.join(account.need)
    #log.info(querc)

    q=query(querc).filter(factor.symbol.in_(account.iwencai_securities),factor.date==
get_datetime().strftime('%Y-%m-%d'))

    df=get_factors(q).fillna(0)
    #log.info(df)
    df['factor_symbol']=account.iwencai_securities[:len(df)]

    n=len(df)
    m=len(df.columns)-1

    for k in range(m):

        tempdf=df.sort_values(df.columns[k])
        name=list(tempdf['factor_symbol'])
        #log.info('排序后的 df')
        #log.info(tempdf)
        #log.info(name)

        f='factor.'+df.columns[k]

```

```
log.info(f)

if account.mfactors[f]<0:
    account.lastlong[f]=name[:account.samt*2]
    account.lastshort[f]=name[-2*account.samt:]

else:
    account.lastlong[f]=name[-2*account.samt:]
    account.lastshort[f]=name[:2*account.samt]
log.info(['long',len(account.lastlong[f])])
log.info(['short',len(account.lastshort[f])])

value=get_price(account.lastlong[f],None,get_datetime().strftime("%Y%m%d"),'
22d',['close'],True,None,account.fpn)

rsum1=np.zeros((account.fpn-1,1))
ctr=0
for stk in account.lastlong[f]:
    #log.info(['这是 value[stk]',value[stk]])
    x=np.array(value[stk])
    #log.info(x)
    if(len(x)!=account.fpn):
        log.info('超配之力不从心')
        continue
    if(ctr==account.samt):
        break
    rsum1+=(x[1:]-x[:-1])/x[:-1]
    ctr=ctr+1
log.info([rsum1,ctr])

value=get_price(account.lastshort[f],None,get_datetime().strftime("%Y%m%d"),'
22d',['close'],True,None,account.fpn)
rsum2=np.zeros((account.fpn-1,1))
ctr=0
for stk in account.lastshort[f]:
    x=np.array(value[stk])
    #log.info(x)
    if(len(x)!=account.fpn):
        log.info('低配之力不从心')
        continue
    if(ctr==account.samt):
        break
    rsum2+=(x[1:]-x[:-1])/x[:-1]
```

```

        ctr=ctr+1
        log.info([rsum2,ctr])

        account.winr[f]=((np.sum(rsum1-rsum2>0)/(account.fpn-1)))
        account.ir[f]=(np.mean(rsum1-rsum2)/np.std(rsum1-rsum2))
        account.incomer[f]=(np.mean(rsum1))
        #log.info(account.winr)
        #log.info(account.ir)

    log.info(account.winr)
    log.info(account.ir)
    log.info(account.incomer)

    score={}
    score=score.fromkeys(list(account.need),0.0)
    #log.info(score)
    temp=sorted(account.winr.items(),key=lambda item:item[1])
    for i in range(len(temp)):
        score[temp[i][0]]+=0.4*i

    temp=sorted(account.ir.items(),key=lambda item:item[1])
    for i in range(len(temp)):
        score[temp[i][0]]+=0.4*i
    #log.info(score)

    temp=sorted(account.incomer.items(),key=lambda item:item[1])
    for i in range(len(temp)):
        score[temp[i][0]]+=0.2*i
    log.info(score)
    tempcore=sorted(score.items(),key=lambda item:item[1],reverse=True)
    log.info(tempcore)
    realneed=[]
    for i in range(account.max_need):
        realneed.append(tempcore[i][0])
    #log.info(realneed)
    return realneed
'''
def testneed(account,data):
    log.info('欢迎来到 testneed')
    samp=account.iwencai_securities
    m=len(account.realneed)
    tempneed={}
    tempneed=tempneed.fromkeys(list(account.need),0)

```

```
log.info(['m 是',m])
for i in range(m-1):
    for j in range(i+1,m):
        log.info('for 循环待命中……')
        log.info([i,j])
        log.info(len(account.need))
        x={}
        x=x.fromkeys(samp,0.0)
        y=x
        name1=account.need[i]
        name2=account.need[j]

    q1=query(name1).filter(factor.symbol.in_(samp),factor.date==get_datetime().strftime('%Y-%m-%d'))

    q2=query(name2).filter(factor.symbol.in_(samp),factor.date==get_datetime().strftime('%Y-%m-%d'))
    df1=get_factors(q1)
    df2=get_factors(q2)
    df1['factor_symbol']=samp[:len(df1)]
    df2['factor_symbol']=samp[:len(df2)]
    #log.info(df1)
    #log.info(df2)
    if(account.mfactors[name1]<0):
        tempdf1=df1.sort_values(df1.columns[0])
    else:
        tempdf1=df1.sort_values(df1.columns[0],ascending=False)
    security1=list(tempdf1['factor_symbol'])
    if(account.mfactors[name2]<0):
        tempdf2=df2.sort_values(df2.columns[0])
    else:
        tempdf2=df2.sort_values(df2.columns[0],ascending=False)
    security2=list(tempdf2['factor_symbol'])

    n=len(tempdf1)
    for k in range(n):
        rank=int(k/(n/account.score))
        x[security1[k]]=account.score-rank

    n=len(tempdf2)
    for k in range(n):
        rank=int(k/(n/account.score))
        y[security2[k]]=account.score-rank
    log.info([x])
```

```

mx=my=3
sx=sy=np.std(list([5,4,3,2,1]*60))

r=0.0
for k in samp:
    r+=(x[k]-mx)*(y[k]-my)
r=r/(len(samp)*sx*sy)
log.info(['r=',r])

if(r>0.5):
    tempneed[name2]=1
#log.info('将会看到 tempneed')
log.info(tempneed)
return list(tempneed.keys())
'''
def give_me_stocks(account,data,is_re):
    samp=account.iwencai_securities
    if(is_re==1):
        account.realneed=give_me_need(account,data)
        #从数据库查询因子,获取 q
        log.info(account.realneed)
        need=','.join(account.realneed)
        #log.info(['这是 need',need])

    q=query(need).filter(factor.symbol.in_(samp),factor.date==get_datetime().strftime('%Y-%m-%d'))
    df=get_factors(q).fillna(0) #结合当天时间,形成 dataframe,缺失值
    填充为 0

    samp=samp[:len(df)]
    df['factor_symbol']=samp

    n=len(df)    #获取行数,即总共获取的股票数目
    m=len(df.columns)-1    #获取列数(因子数目)=总列数-1

    stocks={}
    stocks=stocks.fromkeys(samp,0.0) #预设打分股票池

    for i in range(m):    #先对因子一列从小到大进行排序,方便打
        分,逐列展开
        f='factor.'+df.columns[i]
        if(account.mfactors[f]<0):

```

```

        tempdf=df.sort_values(df.columns[i]) #从小到大
    else:
        tempdf=df.sort_values(df.columns[i],ascending=False)

    security=list(tempdf['factor_symbol']) #排序后的股票代码，重新构建序列

    for j in range(n): #确定 n，即确定的股票代码
        name=security[j]#字符串格式，股票代码

        rank=int((j/(n/account.score))) #确定等级
        stocks[name]+=(5-rank)

    tempstocks=sorted(stocks.items(),key=lambda
item:item[1],reverse=True) #给已经拥有分数的股票池从大到小排序
    return tempstocks[:2*account.max_stocks]

def boll(account,data):
    flag=0 #设置触发变量
    up=[],[]
    mid=[],[]
    low=[],[]
    cprice=[],[]
    last=-6
    now=-1
    for i in list(account.positions):
        values=data.attribute_history(i,['close'],21,'1d',False,None).fillna(0)

        upper,middle,lower=talib.BBANDS(values['close'].values,timeperiod=15,nbdevup
=2,nbdevdn=2,matype=0)
        up[0].append(upper[last])
        up[1].append(upper[now])

        mid[0].append(middle[last])
        mid[1].append(middle[now])

        low[0].append(lower[last])
        low[1].append(lower[now])

        cprice[0].append(values['close'][last])
        cprice[1].append(values['close'][now])

    up=np.mean(up,1)

```

```

mid=np.mean(mid,1)
low=np.mean(low,1)
cprice=np.mean(cprice,1)
#log.info(up)

if(up[1]>up[0] and low[1]<low[0]):    #开轨
    if(cprice[1]<cprice[0] and cprice[1]>mid[1]):
        flag=1
elif(up[1]<up[0] and low[1]>low[0]):    #收轨
    if((cprice[0]<up[0] and cprice[1]>up[1]) or (cprice[0]>low[0] and
cprice[1]<low[1])):
        flag=1

elif((up[1]-up[0])*(low[1]-low[0])>0):    #三轨同向
    if(cprice[0]>mid[0] and cprice[1]<mid[1]):
        flag=1

return flag

def handle_data(account,data):
    ctime=get_datetime()    #今日时间
    dd=ctime-account.start_date

    if(int(dd.days)>=account.cday):    #如果超过计时点，就进行判断
        if(account.cday==0):
            reallocate(account,data)    #月初处理
            account.cday+=account.pp    #调整计时起点

        else:    #非初始日

            #布林线通道
            if(boll(account,data)==1):    #布林线判断出要构建新的股票池
                tempstocks=give_me_stocks(account,data,is_re=0)
                nstocks=[]
                for i in range(len(tempstocks)):
                    nstocks.append(tempstocks[i][0])

                cash=account.positions_value/account.max_stocks

            for i in list(account.positions):
                if(i not in nstocks[:account.max_stocks]):#股票不存在
                    现有账户中，此回合需卖出

```



```

        order_target(i,0)    #全部卖出
    for i in nstocks:
        order_target_value(i,cash) #调整股票使满足目标金
额

        if(len(account.positions)==account.max_stocks):
            break

    Astocks=[]
    tstocks=dict(tempstocks)
    for i in list(account.positions):
        if i not in tstocks:
            continue
        Astocks.append((i,tstocks[i]))
    Amount=give_me_amount(account,data,*Astocks)
    if(len(account.subportfolios.positions)==0):    # 若 子
账户期货为空，说明当月期货已经提前交割，入手下月
        code=get_future_code('IF','next_month')
        order(code,Amount,pindex=1,type='short')
    else:
        current_code=list(account.subportfolios.positions)[0]

#手头期货

    number=account.subportfolios.positions[current_code].total_amount
    if Amount>number:
        #log.info('布林线期货做空')

    order(current_code,-number+Amount,pindex=1,type='short')
    else:
        #log.info('布林线期货做多')

    order(current_code,number-Amount,pindex=1,type='long')
    account.cday+=account.pp    #调整计时起点

```

基于动态多因子选股的 Alpha 策略设计实测报告

目录

1 投资策略回顾	2
1.1 投资策略概述.....	3
1.2 选股方法.....	3
1.3 交易计划.....	3
2 实测操作总结及收益分析	5
2.1 实测情况说明.....	5
2.2 盈亏原因分析.....	5
2.2.1 盈利分析	6
2.2.2 亏损分析	7
2.3 实测操作业绩评估.....	8
2.4 操作亮点.....	8
2.5 心得体会.....	8
3 回测操作总结及交易分析	10
3.1 回测结果分析.....	10
3.2 回测盈亏分析.....	11
3.2.1 盈利分析	13
3.2.2 亏损分析	14
4 投资策略效益分析	16
4.1 策略优势.....	16
4.2 策略不足.....	16
4.3 改进方法.....	16
附录	19

图表目录

表 2.1 证券交易基本情况汇总表（实测）	5
图 2.2 中国平安 6 月 9 日—9 月 6 日 K 线图	6
图 2.3 中国国航 6 月 9 日—9 月 6 日 K 线图	6
图 2.4 沪深 300 指数 6 月 9 日—9 月 6 日 K 线图	7
图 2.5 招商银行 6 月 9 日—9 月 6 日 K 线图	7
图 3.1 整体策略回测图	10
表 3.2 股票交易基本情况汇总表（回测）	11
表 3.3 股指期货交易基本情况汇总表（回测）	13
图 3.4 上海石化 6 月 9 日—8 月 10 日 K 线图	13
图 3.5 南方航空 6 月 12 日—7 月 12 日 K 线图	14
图 3.6 万科 A 股 7 月 19 日—8 月 10 日 K 线图	15
图 4.1 整体策略回测图（股票池构建周期：一个月）	17
图 4.2 整体策略回测图（股票池构建周期：十五天）	18

1 投资策略回顾

1.1 投资策略概述

考虑到中性策略不同于一般的单方向策略，能够有效控制收益的波动性，本组选择的策略主体是 Alpha 策略，整体策略主要分为两个部分：选股与风险控制。在选股策略上，我们依据动态多因子选股的方法，通过对沪深 300 成分股进行筛选构建策略所持股票池。在对风险的控制上，我们一方面做空等额的股指期货进行对冲，以控制系统风险，获取超额收益；另一方面选择布林线通道法作为止盈止损工具对非系统风险进行控制，当触及止盈止损线即全仓卖出重建股票池，同时固定每月第八个交易日重新构建股票池与股指期货，进行动态仓位管理。

1.2 选股方法

本组选择动态多因子模型作为选股模型，以沪深 300 成分股为候选股票池进行筛选。

动态多因子模型选股具体如下：

我们获取当天的因子数据，再根据因子的大小及方向对沪深 300 的成分股进行排序，取其排名最前的一定数量的股票作为超配组合，排名最末相同数量的股票作为低配组合，以 22 个交易日为周期，获取在此之前一定期限内的股价数据，并计算月收益率，取其均值，分别得到超配组合收益率和低配组合收益率。其后运用超低配组合收益率计算信息比率、组合胜率、收益率三个度量指标，依据指标综合得分的大小对因子进行排序打分，最终选择得分排名前五的因子。再根据因子对股票进行排序打分，最终选择得分最高的 10 支股票构建最终股票池。

1.3 交易计划

策略执行初期，首先构建一个股票组合，同时在期货市场，做空等额的股指期货进行对冲。在此后的每月第 8 个交易日，重新构建股票池，进行调仓，同时

对股指期货进行调整，买入上月股指期货进行平仓后卖空当月股指期货，从而实现完全对冲。

若在此期间股票组合达到止盈止损要求，则重新构造超配股票组合，并且调整股指期货合约的数量，若在每月第 8 个交易日之前达到止盈止损要求，则买入或卖空当月期货合约，若在第 8 个交易日之后达到止盈止损要求，则买入或卖空下月期货合约。

2 实测操作总结及收益分析

2.1 实测情况说明

针对本组实测情况，现做如下说明。本组于6月9日成功开户，在6月12日即6月的第八个交易日按照程序设定，通过动态多因子筛选的方法成功构建股票池，持股包括中国平安、建设银行在内的10支股票。6月28日，由于触及止盈止损线全仓卖出所持股票，但在随后的运行中，虽然成功卖空股指期货，却并未重新构建股票池。之后，小组成员按照系统帮助的提示尝试重新启动程序，并联系相关负责人，遗憾情况未得到改善，直至实测结束前模型均未重新构建股票池。

2.2 盈亏原因分析

针对实测结果，小组对交易明细数据进行整理，进而对策略运行及盈亏结果进行分析。数据整理结果如下图所示：

表 2.1 证券交易基本情况汇总表（实测）

交易股票个数		10	盈利股票 个数	4	亏损股票个数	6
序号	股票代码	股票名称	盈亏金额	持股天数	买卖理由	收益率
1	601318	中国平安	13997.8	16	符合选股条件	2.1897%
2	601939	建设银行	10231.67	16	符合选股条件	1.6046%
3	601166	兴业银行	-11518.84	16	符合选股条件	-1.8059%
4	601111	中国国航	-24803.94	16	符合选股条件	-3.9681%
5	600036	招商银行	31656.71	16	符合选股条件	4.9997%
6	600029	南方航空	-20132.73	16	符合选股条件	-3.1787%
7	601288	农业银行	-5622.81	16	符合选股条件	-0.8834%
8	600028	中国石化	-25354.76	16	符合选股条件	-3.9797%
9	601398	工商银行	-3218.77	16	符合选股条件	-1.8426%

表 2.1 证券交易基本情况汇总表（实测）续表

序号	股票代码	股票名称	盈亏金额	持股天数	买卖理由	收益率
10	600030	中信证券	6062.59	16	符合选股条件	3.5062%
11	IF1707	IF1707	-78978.45	9	对冲	-2.6326%
账户总收益率：						-0.2870%

2.2.1 盈利分析

本组策略在实测初期成功通过动态多因子选股模型实现了对候选股票池的筛选，挑选出了有效的股票进行交易，其中包括中国平安（601318）、招商银行（600036）等在内的部分股票在实测期间有不错的涨幅，把握了市场行情的发展变化，实现了部分盈利。中国平安股票在实测期间内走势如下图所示：



图 2.2 中国平安 6 月 9 日—9 月 6 日 K 线图

但在对候选股票池的构建中也存在不足之处，导致误判，例如中国国航（601111）在实测期间表现较差，一度连跌，但得益于止盈止损的程序控制，及时止损，维护了盈利成果。



图 2.3 中国国航 6 月 9 日—9 月 6 日 K 线图

2.2.2 亏损分析

本组策略在最终的实测过程中，整体呈现亏损的状态，造成亏损的主要原因是在实际运行中，股票池的构建与股指期货的交易在时间与价值上未能实现完全匹配。6月12日买入股票，却未能卖出等额的股指期货进行对冲。此外，6月28日清空了股票池，卖空股指期货却未能重新构建新的股票池，以至于未能对冲掉系统性风险，没有实现阿尔法策略的优势。而单独卖空沪深300股指期货的操作适得其反，同期沪深300板块表现良好，如图2.4所示，直接导致策略的大幅亏损。



图 2.4 沪深 300 指数 6 月 9 日—9 月 6 日 K 线图

在程序实测初期结果中，也体现了策略本身的不足之处，由于对股票止盈止损的控制以及股票长线考虑存在不足之处，导致其对盈利效果造成了冲击，未能维护住胜利果实。如下图所示，在招商银行（600036）、中国平安（601318）等多只股票数日连续上涨的情况下，触及止盈止损，在 28 号全仓卖出，损失了长线利益。



图 2.5 招商银行 6 月 9 日—9 月 6 日 K 线图

2.3 实测操作业绩评估

本组策略在实测期间的成功率为 23.08%，最大回撤控制在 1.19%。策略总体呈现亏损状态，共计亏损 107681.53 元，累计收益率为-1.08%。其中股票亏损占比约 20%，股指期货亏损占比约 80%。这与最初的策略构想背离，一方面由于沪深 300 市场行情一路走高，策略本身对未来行情预判存在偏差；另一方面，实测中对股指期货的卖出也未按照策略的设想进行，错失正确时机，造成了期货的大额亏损。

2.4 操作亮点

本组模型是将有效因子筛选出来后，对股票进行动态多因子打分，这样选择股票的方法更具科学性，旨在使最终筛选出来的股票能够更好的把握市场形势。同时辅以布林线通道法进行止盈止损控制和动态调仓，本组模型在实操上还是有一定的可取之处。

2.5 心得体会

动态多因子选股这一策略通过高效快速地量化大量的数据做出买入卖出决策，克服了手动交易的弱点，较好地规避了投资者因为情绪波动而作出非理性决策的可能，最终能筛选出有效的股票进行交易，获取部分盈利。但同时过分的程序化，也丧失了部分自主选择判断的灵活性优势，略显无力。就仓位控制而言，我们认为其灵活性还有待加强，力求更大程度发挥动态多因子选股的优势，扩大盈利。

对于风险控制，我们认为有效的止盈止损手段是必要的，它在保证收益的同时防止了巨额亏损的出现，确保整体策略收益在一个相对平稳的水平上。但就实测期间的市场行情而言，沪深 300 指数持续上涨，保守的交易策略一定程度上对盈利造成亏损，所以交易策略的选择只有在结合对未来市场行情进行充分预判的前提下才能发挥其最大的优势。

在策略的整体实测中，我们充分认识到阿尔法策略的核心是系统风险的对冲，因此股票交易和股指期货交易不仅要顺利进行，还要保证时间和价值的高度匹配，

力求达到完全对冲，这样才能真正实现阿尔法策略的优势，获取超额收益，否则只会适得其反。

在理想情况下，本组模型应该是能够发挥出其自身的作用，达到既定目标。然而这次实测，结果却不尽人意，与我们的预期差距较大，交易记录也颇为惨淡。在发现程序停止运行之后，我们立刻寻找问题所在，最终发现是由于程序中途运行出错，导致后面很长时间没有进行交易，重启程序之后，又能正常运行。由于我们自身的疏忽，造成了这样的结果，确实十分遗憾，也借此希望平台日后能更加稳定，越做越好。然而数据的缺失无可避免，于是在实测结束后，我们重新对程序进行了回测，并进行了相关的分析以及程序的优化，力求能达到我们最初预期的目标。

3 回测操作总结及交易分析

3.1 回测结果分析

考虑是否存在回测中程序语句在实测环境中不兼容的问题，9月初小组尝试用实测时间段进行程序回测，未出现与实测同样的问题，回测情况一切正常。回测结果基本上实现了本组最初的策略构想，初步展现了基于动态多因子选股的Alpha策略的雏形，策略运行中大体上成功实现构建股票池以及股指期货的同步买卖，对冲系统风险，实现了可观的收益。仅在程序运行之初，股指期货未及时卖出，同时重新构建股票池的频率过小，有待优化。

以下是对整体策略在实测期间的回测结果分析。



图 3.1 整体策略回测图

回测结果指标如下：

策略收益	基准收益	α 值	β 值	夏普比率	最大回撤	胜率
5.09%	8.31%	0.03	0.46	1.76	3.02%	57.14%

回测分析：从图中线的走势来看，本组策略基本能随着市场而变动。策略整体上能够有效获得 0.03% 的超额收益，胜率也较高，为 57.14%，并且取得了 5.09% 的策略收益，说明本组的候选因子有效性检验的打分机制是有效的，最终选择的择股因子能够挑选出我们需要的股票。且本组策略的收益波动性较低，最大回撤仅为 3.02%，单位风险收益良好，这证明本组在风险控制这一方面做的较为优秀。

美中不足的是，策略 β 值偏高，对冲效果并不明显。通过以上分析可以得出，本组策略在股指期货的对冲下，最大回撤在适当范围的情况下，基本能够获取到所要的 Alpha 收益。

3.2 回测盈亏分析

针对回测结果，小组对交易明细数据进行整理，进而对策略运行及盈亏结果进行分析。数据整理结果如下图所示：

表 3.2 股票交易基本情况汇总表（回测）

交易股票个数		33	盈利股票 个数	15	亏损股票个数	18
序号	股票代码	股票名称	盈亏金额	持股天数	买卖理由	收益率
1	601600.SH	中国铝业	-6275.6	3	符合选股条件	-0.90%
2	000060.SZ	中金岭南	-6953.2	3	符合选股条件	-0.99%
3	600585.SH	海螺水泥	513.4	3	符合选股条件	0.07%
4	000725.SZ	京东方 A	331.3	3	符合选股条件	0.05%
5	000800.SZ	一汽轿车	-9234.5	3	符合选股条件	-1.32%
6	000858.SZ	五粮液	29149.2	33	符合选股条件	4.17%
7	600688.SH	上海石化	45313.31	62	符合选股条件	6.30%
8	600010.SH	包钢股份	-7816	3	符合选股条件	-1.12%
9	000778.SZ	新兴铸管	-403.9	3	符合选股条件	-0.06%
10	601899.SH	紫金矿业	-1400	3	符合选股条件	-0.20%
11	601111.SH	中国国航	-73618.7	30	符合选股条件	-10.51%
12	600690.SH	青岛海尔	-25244.1	30	符合选股条件	-3.60%
13	600887.SH	伊利股份	-5393.4	37	符合选股条件	-0.77%
14	600104.SH	上汽集团	12639	30	符合选股条件	1.81%

表 3.2 股票交易基本情况汇总表（回测）续表

序号	股票代码	股票名称	盈亏金额	持股天数	买卖理由	收益率
15	600029.SH	南方航空	-54046.8	30	符合选股条件	-7.72%
16	000568.SZ	泸州老窖	-7095.9	30	符合选股条件	-1.06%
17	600115.SH	东方航空	-23928.1	30	符合选股条件	-3.41%
18	000651.SZ	格力电器	105777.6	30	符合选股条件	15.13%
19	600309.SH	万华化学	4816.5	7	符合选股条件	0.69%
20	600674.SH	川投能源	-28586.9	7	符合选股条件	-4.10%
21	601225.SH	陕西煤业	16069.4	7	符合选股条件	2.30%
22	600177.SH	雅戈尔	-10223.3	7	符合选股条件	-1.47%
23	601899.SH	紫金矿业	26676.6	7	符合选股条件	3.83%
24	600415.SH	小商品城	-12008.5	7	符合选股条件	-1.72%
25	601006.SH	大秦铁路	15222.8	7	符合选股条件	2.18%
26	600585.SH	海螺水泥	-789.4	22	符合选股条件	-0.12%
27	600837.SH	海通证券	2811.5	22	符合选股条件	0.41%
28	600157.SH	永泰能源	47487.3	22	符合选股条件	6.93%
29	600221.SH	海航控股	17745.4	22	符合选股条件	2.59%
30	000001.SZ	平安银行	-19418.7	22	符合选股条件	-2.83%
31	000002.SZ	万科A	-52465.4	22	符合选股条件	-7.67%
32	600705.SH	中航资本	4505.6	22	符合选股条件	0.66%
33	600352.SH	浙江龙盛	79616.7	22	符合选股条件	11.62%
账户总收益率：						0.28%

表 3.3 股指期货交易基本情况汇总表（回测）

交易期货个数		3	盈利期货个数	0	亏损期货个数	2
序号	期货代码	期货名称	盈亏金额	持股天数	买卖理由	收益率
1	IF1707	IF1707	-214705	31	对冲	-3.25%
2	IF1708	IF1708	-77682.2	8	对冲	-1.75%
3	IF1709	IF1709	--	--	对冲	--
账户总收益率：						-2.50%

3.2.1 盈利分析

策略运行过程中，部分股票取得了一定的盈利。一方面得益于动态多因子选股，挑选出的股票在回测期间表现良好，紧跟市场行情，取得了不错的收益，另一方面得益于及时的止盈止损有效控制了损失。在诸多盈利的股票中，上海石化（600688）的表现最具有代表性，下图是上海石化 6 月 9 日至 8 月 10 日的走势图。



图 3.4 上海石化 6 月 9 日—8 月 10 日 K 线图

从交易明细以及股票 k 线图可以看出，程序依据多因子选股，预期上海石化近期有上涨趋势，于 6 月 9 日开仓买入 107000 股，持股期间，抓住了小幅度上涨趋势，尤其在 7 月下旬存在几个大幅上涨，取得良好收益。8 月重建股票池，清仓所持上海石化全部股票。之后，上海石化股价一路大跌至 6.39。从持有该股

票的表现可以看出，动态多因子选股在股票选择上存在一定优势，及时的止盈止损是取得盈利的有效保障。

纵观整体策略的盈利表现可以发现，策略在 7 月 19 日触及布林线重新构建股票池至 8 月 10 日按照程序设定对股票池再次进行重建期间，持股的诸如永泰能源（600157）、浙江龙盛（600352）等股票均具有很强劲的表现，策略在该段行情下无论是股票的选择还是时机的把握均表现亮眼。

3.2.2 亏损分析

本组策略在回测过程中，总体上是盈利的，策略收益为 5.09%。但也有部分股票呈现的是亏损状态，例如中国国航、南方航空、万科 A 等股票。通过对其中两支股票的分析，我们不难发现，亏损的主要原因集中在选股和止盈止损上。虽然本组采用的动态多因子选股方法大体上是具有一定科学性的，使得最终筛选出来的股票能够把握市场的形势，但在因子选择的全面性上有所欠缺，因为每个因子的影响程度是不同的，因此对部分股票的未来收益预测还是存在偏差。在止盈止损上，本组采用的方法还是不够灵活，效果并没有达到我们所预期的，并且只是零星的触发了几次，不能有效地阻止较大亏损的出现，这方面还是有待改进。

例如南方航空（600029）这支股票，我们在 6 月 12 买入，于 7 月 12 日卖出。高进低出，股票价格一路下跌，这表明，本组策略在选股上还有一定的改进空间，南航这支股票并不能实现预期的超额收益，反而跌势明显，且未能成功止损，不能将损失降到最低。



图 3.5 南方航空 6 月 12 日—7 月 12 日 K 线图

本组于7月19日买入万科A（000002），8月10日卖出。从下图可以看出，在开始两天内，万科A确实有小额涨幅，但之后便开始下跌。7月27日到8月3日的股价较为稳定，没有大幅变动，在这之后，又开始一路下跌。由于没有及时止损，导致了最终的亏损。



图 3.6 万科A股7月19日—8月10日K线图

4 投资策略效益分析

4.1 策略优势

本组策略通过动态多因子模型能有效把握市场的变化，通过有效的选股因子帮助策略实现长期、稳定的 Alpha 收益。在对有效因子的筛选中，我们构建了数量较为庞大的候选因子池，并采用多个指标相结合的方式对候选因子进行筛选。通过实测数据可以看出我们的程序对候选因子的筛选是有效的，最终选择的股票可以通过整体策略的运行获得可观的超额收益。

此外，在风险控制上，我们运用布林线通道法进行止盈止损，在收益波动率和最大回撤方面取得良好效果。

4.2 策略不足

首先，在总体趋势的判断上，我们错误地估计了实测期间的市场行情，选择了在震荡行情与熊市行情具有较大优势的阿尔法策略，未能把握实测期间的市场机遇。

其次，本组程序尚未完全解决股票停牌问题，在构建超低配组合时出现不完全对称的情况，因此在计算因子度量指标时可能存在误差，从而降低了动态多因子筛选的有效性。此外，由于筛选出的有效股票可能处于停牌的情况，程序也就无法实现运用股指期货完全对冲系统性风险，进而，通过 Alpha 策略获得超额收益的效果打了折扣。

再次，本组策略在最初设定的时候，考虑到股指期货的交割日期，规定每月的第八个交易日重新构建股票池，同时对期货数额进行相应调整。在实测中发现每月一次的调整，时间间隔过长，不能有效把握短期市场行情的变化，灵活性不足，过于被动。同样的问题也体现在策略的仓位控制上，最初设计模型时考虑到程序的实现，我们采取的是全仓买入，全仓卖出的策略，灵活性不足，导致错失良机。

4.3 改进方法

我们主要针对原始策略在三方面进行改进，第一方面是解决了实测交易中出现的初始开户构建股票池时，股指期货未同步开仓卖出的问题；第二方面是股票停牌的问题；第三方面是对策略中股票池重建周期长度不适宜的问题进行优化。

（一）股指期货交易问题的解决

股指期货应该是在构建股票池，买卖股票的同时进行买入卖出。但从交易记录中，我们发现在策略进行第一次股票交易时，股指期货并未如我们所预期的那样开仓卖出。所以我们对程序进行了改进，查找出了其中的漏洞并解决。

（二）股票停牌问题

本组策略经常受到股票停牌的限制，导致股票池与预想不符以及动态多因子的选股有效性降低。我们采取的方法是构建股票池之初，实时核对是否存在停牌股，当发现股票停牌，便及时将排名紧随其后的股票纳入股票池，从而解决了股票停牌问题。

（三）股票交易周期的优化

本组原始策略是每月构建一次股票池，除触及止盈止损线会重新构建股票池外，第2个月的第8个交易日才会重新构建股票池。但在实际操作中发现，这种做法虽然能减少交易次数，削减手续支出，但是不能随着市场行情的变动及时对所持股票做出调整，使得我们处于一个较为被动的位置。因此我们决定对股票交易周期进行调整，最终决定将构建股票池的周期从一个月调整为十五天，其对比结果如下图所示。从图中可知，周期缩短为十五天后，整体策略表现明显变好，能更加灵活地适应市场行情的变化，追求超额收益。

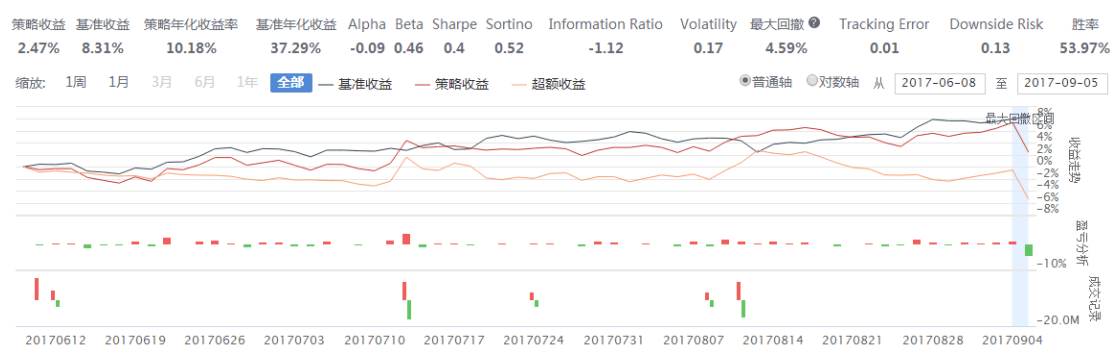


图 4.1 整体策略回测图（股票池构建周期：一个月）

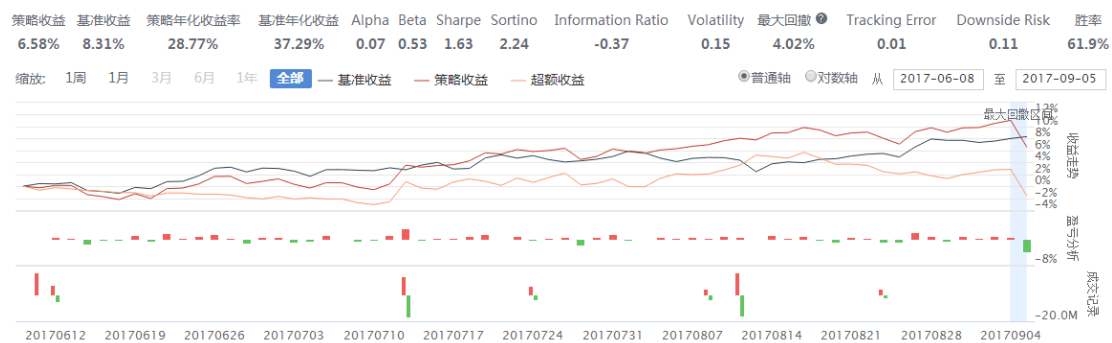


图 4.2 整体策略回测图（股票池构建周期：十五天）

附录

```
import pandas as pd
import numpy as np
import datetime
import statsmodels.api as sm
import talib
import math

def initialize(account):
    set_log_level('warn')
    #智能选股
    get_iwencai('沪深 300')
    #设置交易周期
    account.pp=5
    #设置最大持股数
    account.max_stocks=10
    #设置筛选因子个数
    account.max_need=5
    #设置因子测试期数
    account.fpn=9
    account.st=0
    #设置检验因子所用的股票数
    account.samt=5
    #设置子账户
    set_subportfolios([{'cash':7000000,'type':'stock'},{'cash':3000000,'type':'future'}])
    log.warn(['account.cash:',account.cash])
    #设置计数起点
    account.cday=0
    #设置因子池
    account.need=['factor.pe',
                  'factor.pb',
                  'factor.pcf_cash_flow_ttm',
                  'factor.ps',
                  'factor.dividend_rate',
                  'factor.market_cap',
                  'factor.current_market_cap',
                  'factor.capitalization',
                  'factor.circulating_cap',
                  'factor.current_ratio',
                  'factor.equity_ratio',
                  'factor.quick_ratio',
                  'factor.tangible_assets_liabilities',
```

'factor.tangible_assets_int_liabilities',
'factor.net_debt_equity',
'factor.long_term_debt_to_opt_capital_ratio',
'factor.tangible_assets_net_liabilities',
'factor.overall_income_growth_ratio',
'factor.net_cashflow_psg_rowth_ratio',
'factor.opt_profit_grow_ratio',
'factor.total_profit_growth_ratio',
'factor.diluted_net_asset_growth_ratio',
'factor.net_cashflow_from_opt_act_growth_ratio',
'factor.net_profit_growth_ratio',
'factor.basic_pey_ear_growth_ratio',
'factor.turnover_of_overall_assets',
'factor.turnover_ratio_of_account_payable',
'factor.turnover_of_current_assets',
'factor.turnover_of_fixed_assets',
'factor.cash_cycle',
'factor.inventory_turnover_ratio',
'factor.turnover_ratio_of_receivable',
'factor.weighted_roe',
'factor.overall_assets_net_income_ratio',
'factor.net_profit_margin_on_sales',
'factor.before_tax_profit_div_income',
'factor.sale_cost_div_income',
'factor.roa',
'factor.ratio_of_sales_to_cost',
'factor.net_profit_div_income',
'factor.opt_profit_div_income',
'factor.opt_cost_div_income',
'factor.administration_cost_div_income',
'factor.financing_cost_div_income',
'factor.vr_rate',
'factor.vstd',
'factor.arbr',
'factor.srdm',
'factor.vroc',
'factor.vrsi',
'factor.cr',
'factor.mfi',
'factor.vr',
'factor.mass',
'factor.obv',
'factor.pvt',
'factor.wad',

```
'factor.bbi',
'factor.mtm',
'factor.dma',
'factor.ma',
'factor.macd',
'factor.expma',
'factor.priceosc',
'factor.trix',
'factor.dbcd',
'factor.dpo',
'factor.psy',
'factor.vma',
'factor.vmacd',
'factor.vosc',
'factor.tapi',
'factor.micd',
'factor.rccd']
#设置股票因子及其方向
account.mfactors={'factor.pe':-1,
                  'factor.pb':-1,
                  'factor.pcf_cash_flow_ttm':-1,
                  'factor.ps':-1,
                  'factor.dividend_rate':1,
                  'factor.market_cap':-1,
                  'factor.current_market_cap':-1,
                  'factor.capitalization':-1,
                  'factor.circulating_cap':-1,
                  'factor.current_ratio':1,
                  'factor.equity_ratio':-1,
                  'factor.quick_ratio':1,
                  'factor.tangible_assets_liabilities':1,
                  'factor.tangible_assets_int_liabilities':1,
                  'factor.net_debt_equity':-1,
                  'factor.long_term_debt_to_opt_capital_ratio':-1,
                  'factor.tangible_assets_net_liabilities':1,
                  'factor.overall_income_growth_ratio':1,
                  'factor.net_cashflow_psg_rowth_ratio':1,
                  'factor.opt_profit_grow_ratio':1,
                  'factor.total_profit_growth_ratio':1,
                  'factor.diluted_net_asset_growth_ratio':1,
                  'factor.net_cashflow_from_opt_act_growth_ratio':1,
                  'factor.net_profit_growth_ratio':1,
                  'factor.basic_pey_ear_growth_ratio':1,
                  'factor.turnover_of_overall_assets':1,
```

'factor.turnover_ratio_of_account_payable':1,
'factor.turnover_of_current_assets':1,
'factor.turnover_of_fixed_assets':1,
'factor.cash_cycle':-1,
'factor.inventory_turnover_ratio':1,
'factor.turnover_ratio_of_receivable':1,
'factor.weighted_roe':1,
'factor.overall_assets_net_income_ratio':1,
'factor.net_profit_margin_on_sales':1,
'factor.before_tax_profit_div_income':1,
'factor.sale_cost_div_income':-1,
'factor.roa':1,
'factor.ratio_of_sales_to_cost':-1,
'factor.net_profit_div_income':1,
'factor.opt_profit_div_income':1,
'factor.opt_cost_div_income':-1,
'factor.administration_cost_div_income':-1,
'factor.financing_cost_div_income':-1,
'factor.vr_rate':1,
'factor.vstd':-1,
'factor.arbr':1,
'factor.srdm':1,
'factor.vroc':1,
'factor.vrsi':1,
'factor.cr':1,
'factor.mfi':1,
'factor.vr':1,
'factor.mass':1,
'factor.obv':1,
'factor.pvt':1,
'factor.wad':1,
'factor.bbi':1,
'factor.mtm':1,
'factor.dma':1,
'factor.ma':1,
'factor.macd':1,
'factor.expma':1,
'factor.priceosc':1,
'factor.trix':1,
'factor.dbcd':1,
'factor.dpo':1,
'factor.psy':1,
'factor.vma':1,
'factor.vmacd':1,

```

        'factor.vosc':1,
        'factor.tapi':1,
        'factor.micd':1,
        'factor.rccd':1}
account.lastlong={}.fromkeys(account.need) #超配股票池
account.lastshort={}.fromkeys(account.need) #低配股票池


account.winr={}
account.ir={}
account.incomer={}
#设置默认因子
account.realneed=['factor.pe','factor.weighted_roe',

'factor.net_cashflow_from_opt_act_growth_ratio','factor.overall_income_growth_ratio',
        'factor.turnover_ratio_of_account_payable']
#设置打分机制
account.score=5
#定期运行函数
schedule_function(reallocate,date_rule=date_rules.month_start(8))


def reallocate(account,data):
    log.warn('……每月第 8 交易日/起始日……')

    tempstocks=give_me_stocks(account,data,is_re=1) #前 maxstocks 只股票带综合分数
    #log.info('This is the stocks for choosing')
    #log.info(tempstocks)
    nstocks=[]
    for i in range(len(tempstocks)):
        nstocks.append(tempstocks[i][0])

    #起手均分现金，之后均分持仓规模
    log.warn('刷新持仓股票')
    if(account.cday==0):
        cash=account.cash/account.max_stocks
        log.warn(['account.cash:',account.cash])
    else:
        cash=account.positions_value/account.max_stocks

    for i in list(account.positions):

```



```

    if(i not in nstocks[:account.max_stocks]):
        order_target(i,0)
    for i in nstocks:
        order_target_value(i,cash)
        if(len(account.positions)==account.max_stocks):
            break

log.warn('刷新完毕')
log.warn('期货操作')
Astocks=[] #用于计算期货份数的列表，存放持股分数
tstocks=dict(tempstocks)

ctr=0
for i in list(account.positions):
    if i not in tstocks:
        continue
    Astocks.append((i,tstocks[i]))
    ctr+=1
#log.info(ctr)
Amount=give_me_amount(account,data,*Astocks)
#log.info(['The number of Amount is',Amount])
code=get_future_code('IF','next_month') #下月期货
if(len(account.subportfolios.positions)==0): #若子账户期货为空，说明当月期
货已经交割，入手下月
    order(code,Amount,pindex=1,type='short')
    log.warn('下月期货入手')
else:
    current_code=list(account.subportfolios.positions)[0] #手头期货
    number=account.subportfolios.positions[current_code].total_amount #手头
期货份数
    if(code==current_code): #如果手头的是下月期货，直接调仓
        if Amount>number:
            order(current_code,-number+Amount,pindex=1,type='short') #做空期
货，卖出
            log.warn('做空期货')
        else:
            order(current_code,number-Amount,pindex=1,type='long') #做多期货，
买回
            log.warn('做多期货')
    else:
        order(current_code,-number,pindex=1,type='long') #平仓买回
        order(code,Amount,pindex=1,type='short') #做空下月期货
        log.warn('平仓买回 AND 做空下月期货')
log.warn('期货操作完毕')

```

```
def give_me_amount(account,data,*tempstocks):
    log.warn('获取期货买卖份数')
    #计算 Beta 值
    beta=[]
    percent=[]
    nstocks=[]
    rstocks=[]
    sum1=0.0

    for i in range(len(tempstocks)):
        sum1+=tempstocks[i][1]
        rstocks.append(tempstocks[i][1])
        nstocks.append(tempstocks[i][0])

    #计算权重
    for i in range(len(rstocks)):
        percent.append(rstocks[i]/sum1)

    for i in nstocks: #迭代股票池

        iclose=get_price(['000300.SH',i],None,get_datetime().strftime('%Y%m%d'),'1d',['close'],None,None,220) #证券
        xclose=iclose['000300.SH']['close']
        yclose=iclose[i]['close']

        if(len(yclose)==0):
            #log.info('糟糕,yclose 长度是 0')
            beta.append(0)
            continue

        if(len(xclose)!=len(yclose)):
            #log.info(['iclose 是',iclose])
            l=min(len(xclose),len(yclose))
            xclose=xclose[-l:]
            yclose=yclose[-l:]

        x=np.array(xclose)
        y=np.array(yclose)

        rm=(x[1:]-x[0:-1])/x[0:-1]
        r1=(y[1:]-y[0:-1])/y[0:-1]
        est=sm.OLS(r1,sm.add_constant(rm))
        est=est.fit() #单只股票的 beta
```

```
beta.append(est.params[1])

Beta=0.0
for i in range(len(beta)):
    Beta+=beta[i]*percent[i] #Beat 等于因子打分权重之和
    #log.info(['Beta 是',Beta])
    close=data.attribute_history('000300.SH',['close'],1,'1d') #期货开盘价
    #log.info(['沪深 300 股指期货的开盘价是',close])
    cash=account.positions_value*1.0/account.max_stocks
    log.warn(['Beta and cash and 期货价: ',Beta,cash,close['close'][0]])
    Amount=math.ceil((Beta*(cash*account.max_stocks))/(300*close['close'][0]))
    log.warn(['Amount 是',Amount])
    log.warn('期货买卖份数获取完毕')
    return int(Amount*1)

def give_me_need(account,data):
    log.warn('获取新建因子')
    n=len(account.mfactors)
    querc=', '.join(account.need)
    #log.info(querc)

q=query(querc).filter(factor.symbol.in_(account.iwencai_securities),factor.date==get_
_datetime()).strftime('%Y-%m-%d'))

df=get_factors(q).fillna(0)
#log.info(df)
df['factor_symbol']=account.iwencai_securities[:len(df)]

n=len(df)
m=len(df.columns)-1

for k in range(m):

    tempdf=df.sort_values(df.columns[k])
    name=list(tempdf['factor_symbol'])
    #log.info('排序后的 df')
    #log.info(tempdf)
    #log.info(name)

    f='factor.'+df.columns[k]
    #log.info(f)
```

```
if account.mfactors[f]<0:
    account.lastlong[f]=name[:account.samt*2]
    account.lastshort[f]=name[-2*account.samt:]

else:
    account.lastlong[f]=name[-2*account.samt:]
    account.lastshort[f]=name[2*account.samt:]
    #log.info(['long',len(account.lastlong[f])])
    #log.info(['short',len(account.lastshort[f])])

value=get_price(account.lastlong[f],None,get_datetime().strftime("%Y%m%d"),'22d',
['close'],True,None,account.fpn)

rsum1=np.zeros((account.fpn-1,1))
ctr=0
for stk in account.lastlong[f]:
    #log.info(['这是 value[stk]',value[stk]])
    x=np.array(value[stk])
    #log.info(x)
    if(len(x)!=account.fpn):
        #log.info('超配之力不从心')
        continue
    if(ctr==account.samt):
        break
    rsum1+=(x[1:]-x[:-1])/x[:-1]
    ctr=ctr+1
    #log.info([rsum1,ctr])

value=get_price(account.lastshort[f],None,get_datetime().strftime("%Y%m%d"),'22d',
['close'],True,None,account.fpn)
rsum2=np.zeros((account.fpn-1,1))
ctr=0
for stk in account.lastshort[f]:
    x=np.array(value[stk])
    #log.info(x)
    if(len(x)!=account.fpn):
        #log.info('低配之力不从心')
        continue
    if(ctr==account.samt):
        break
    rsum2+=(x[1:]-x[:-1])/x[:-1]
    ctr=ctr+1
```

```
#log.info([rsum2,ctr])

account.winr[f]=((np.sum(rsum1-rsum2>0)/(account.fpn-1)))
account.ir[f]=(np.mean(rsum1-rsum2)/np.std(rsum1-rsum2))
account.incomer[f]=(np.mean(rsum1))
#log.info(account.winr)
#log.info(account.ir)

#log.info(account.winr)
#log.info(account.ir)
#log.info(account.incomer)

score={}
score=score.fromkeys(list(account.need),0.0)
#log.info(score)
temp=sorted(account.winr.items(),key=lambda item:item[1])
for i in range(len(temp)):
    score[temp[i][0]]+=0.4*i

temp=sorted(account.ir.items(),key=lambda item:item[1])
for i in range(len(temp)):
    score[temp[i][0]]+=0.4*i
#log.info(score)

temp=sorted(account.incomer.items(),key=lambda item:item[1])
for i in range(len(temp)):
    score[temp[i][0]]+=0.2*i
#log.info(score)
tempscore=sorted(score.items(),key=lambda item:item[1],reverse=True)
#log.info(tempscore)
realneed=[]
for i in range(account.max_need):
    realneed.append(tempscore[i][0])
log.info(realneed)
log.warn('因子获取完毕')
return realneed

def give_me_stocks(account,data,is_re):
    log.warn('获取新建股票池')
    samp=account.iwencai_securities
    if(is_re==1):
        account.realneed=give_me_need(account,data)
        #从数据库查询因子,获取 q
        #log.info(account.realneed)
```

```

need=', '.join(account.realneed)
#log.info(['这是 need',need])

q=query(need).filter(factor.symbol.in_(samp),factor.date==get_datetime().strftime('%Y-%m-%d'))
df=get_factors(q).fillna(0) #结合当天时间，形成 dataframe,缺失值填充为 0

samp=samp[:len(df)]
df['factor_symbol']=samp

n=len(df) #获取行数，即总共获取的股票数目
m=len(df.columns)-1 #获取列数(因子数目)=总列数-1

stocks={}
stocks=stocks.fromkeys(samp,0.0) #预设打分股票池

for i in range(m): #先对因子一列从小到大进行排序，方便打分 ,逐列展开
    f='factor.'+df.columns[i]
    if(account.mfactors[f]<0):
        tempdf=df.sort_values(df.columns[i]) #从小到大
    else:
        tempdf=df.sort_values(df.columns[i],ascending=False)

    security=list(tempdf['factor_symbol']) #排序后的股票代码，重新构建序列

    for j in range(n): #确定 n，即确定的股票代码
        name=security[j]#字符串格式，股票的代码

        rank=int((j/(n/account.score))) #确定等级
        stocks[name]+=(5-rank)

tempstocks=sorted(stocks.items(),key=lambda item:item[1],reverse=True) #给
已经拥有分数的股票池从大到小排序

log.info(tempstocks)
log.warn('新建股票池获取完毕')
return tempstocks[:2*account.max_stocks]

def boll(account,data):
    log.warn('布林线检测')
    flag=0 #设置触发变量
    up=[],[]
    mid=[],[]

```

```
low=[[],[]]
cprice=[[],[]]
last=-6
now=-1
for i in list(account.positions):
    values=data.attribute_history(i,['close'],21,'1d',False,None).fillna(0)

upper,middle,lower=talib.BBANDS(values['close'].values,timeperiod=15,nbdevup=2,n
bdevdn=2,matype=0)
    up[0].append(upper[last])
    up[1].append(upper[now])

    mid[0].append(middle[last])
    mid[1].append(middle[now])

    low[0].append(lower[last])
    low[1].append(lower[now])

    cprice[0].append(values['close'][last])
    cprice[1].append(values['close'][now])

up=np.mean(up,1)
mid=np.mean(mid,1)
low=np.mean(low,1)
cprice=np.mean(cprice,1)
#log.info(up)

if(up[1]>up[0] and low[1]<low[0]): #开轨
    if(cprice[1]<cprice[0] and cprice[1]>mid[1]):
        flag=1
        log.warn('开轨')
elif(up[1]<up[0] and low[1]>low[0]): #收轨
    if((cprice[0]<up[0] and cprice[1]>up[1]) or (cprice[0]>low[0] and
cprice[1]<low[1])):
        flag=1
        log.warn('收轨')

elif((up[1]-up[0])*(low[1]-low[0])>0): #三轨同向
    if(cprice[0]>mid[0] and cprice[1]<mid[1]):
        flag=1
        log.warn('三轨同向')
log.warn('布林线检测完毕')
if(flag==1):
    log.warn('布林线动作触发')
```

```

else:
    log.warn('无动作')
    return flag

def handle_data(account,data):
    ctime=get_datetime() #今日时间
    if(account.cday==0):
        account.st=ctime
        dd=ctime-account.st

    if(int(dd.days)>=account.cday): #如果超过计时点，就进行判断
        if(account.cday==0):
            reallocate(account,data) #月初处理
            account.cday+=account.pp #调整计时起点
        else: #非初始日

            #布林线通道
            log.warn([account.pp,'/天一检测'])
            if(boll(account,data)==1 or int(dd.days%(15))==0): #布林线判断出要构建新的
股票池或者距离上一次构建达到达到 15 天
                if(int(dd.days%(15))==0):
                    log.warn('15 天/构建')

            tempstocks=give_me_stocks(account,data,is_re=0)
            nstocks=[]
            for i in range(len(tempstocks)):
                nstocks.append(tempstocks[i][0])
            log.info(nstocks)
            cash=account.positions_value/account.max_stocks

            log.warn('刷新持仓股票')
            ctr=0
            for i in list(account.positions):
                if(i not in nstocks[:account.max_stocks]):#股票不存在现有账户中，此回合
需卖出
                    order_target(i,0) #全部卖出
                    ctr+=1
            log.warn(['卖出股票数:',ctr])
            for i in nstocks:
                order_target_value(i,cash) #调整股票使满足目标金额
                if(len(account.positions)==account.max_stocks):
                    break
            log.warn('进行期货操作')

```



```
Astocks=[]
tstocks=dict(tempstocks)
for i in list(account.positions):
    if i not in tstocks:
        continue
    Astocks.append((i,tstocks[i]))
Amount=give_me_amount(account,data,*Astocks)
if(len(account.subportfolios.positions)==0): #若子账户期货为空,说明当月
期货已经提前交割,入手下月
    code=get_future_code('IF','next_month')
    order(code,Amount,pindex=1,type='short')
    log.warn('下月期货入手')
else:
    current_code=list(account.subportfolios.positions)[0] #手头期货
    number=account.subportfolios.positions[current_code].total_amount
    if Amount>number:
        log.warn('做空期货')
        order(current_code,-number+Amount,pindex=1,type='short')
    else:
        log.warn('做多期货')
        order(current_code,number-Amount,pindex=1,type='long')
account.cday+=account.pp #调整计时起点
log.warn([account.pp,'/天一检测(15 天检测)完毕'])
```