RSI-AR 组合策略

杨昀昶[1]

（西安邮电大学经济与管理学院,陕西省西安市 710121）

通信作者联系方式：（Email:yangyunchang001@gmail.com 电话：029-68801085 手机：18292577417）

投稿日期：2018-7-31,

作者简介：

杨昀昶（1997-），男，本科，从事量化投资与市场微观结构理论的研究。

Email:yangyunchang@gmail.com

摘 要 随着量化投资的发展，量化投资策略也越来越多，国内外学者和机构多研究和使用的理论驱动模型，该类策略发展较为成熟，理论驱动型的量化投资策略发展时间久，因此，模型的稳定性更强。本文就构造一种，利用多种技术指标进行选股、择时和止损于一体的量化投资策略。

本文首先介绍了策略概况；其次，以文献综述形式分析了国内外相关的研究进展；再次，介绍了模型基本背景和算法；最后，通过回测分析来评价策略的各方面表现。

策略回测结果较好，但回撤幅度较大，本文认为与当时市场环境存在很大关系。策略 2015 年 1 月 5 日至 2018 年 7 月 30 日的回测结果为：总收益率 159.55%（对比沪深 300 指数 31.45%），年化收益率 53.51%，夏普率 0.899，最大回撤 29.615%。

关键词 技术指标 理论驱动模型 股价预测 量化投资

Abstract With the development of quantitative investment, there are more and more quantitative investment strategies. The theoretically driven models developed and used by scholars and institutions at home and abroad have developed more mature strategies. The theoretically-driven quantitative investment strategy has developed for a long time. Therefore, The stability of the model is stronger. This paper constructs a quantitative investment strategy that uses a variety of technical indicators for stock selection, timing and stop loss.

This paper first introduces the strategy overview. Secondly, it analyzes the relevant research progress at home and abroad in the form of literature review. Thirdly, it introduces the basic background and algorithm of the model. Finally, the back analysis provides the evaluation of various aspects of the strategy.

The strategy backtesting results are better, but the retracement is larger. This paper thinks that it has a great relationship with the market environment at that time. The back-test results of the strategy from January 5, 2015 to July 30, 2018 are: the total return rate is 159.55% (compared with Tongrentang stock 31.45%), the annualized rate of return is 53.51%, the Sharpe rate is 0.899, and the maximum retracement is

29.615%.
Key Word Technical indicators, Theoretical drive model, Stock price forecast, Quantitative investment

1 策略概述

1.1 研究目的

量化投资的策略有很多，量化择时策略是其中之一，量化择时策略不考虑如何选取股票，不考虑构建投资组合，该策略更侧重于在确定股票或者资产组合后买卖时点的选择，甚至可以认为量化择时策略是用来预测任何资产买卖的时点；量化择时策略也不同于套利策略，套利策略往往要求建立投资组合或者资产中加入其它资产来降低组合系统风险来赚取稳健的收益，量化择时策略一般不进行“对冲”，这样以便获取较高的收益，它对于风险的控制主要是通过提高策略模型预测的准确性和设置止盈止损来实现，也有的策略会加入风控模型来降低风险。具体而言，量化择时通过对各项微宏观指标进行数量化分析，挖掘能够预测价格走势的关键信息，并通过历史数据预测未来价格走势，如果预测未来资产价格会上涨，则进行买入；如果预测未来的价格下跌，则进行卖出，如果预测未来是盘整，则保持持有或进行高抛低吸。

虽然，当前市面流行的量化择时策略很多，结构也越来越复杂，但是研究人员却因为技术指标的一些指标而没有将技术指标很少应用在择时模型中。由于中国股市依然没有达到弱有效市场，因此，技术指标依然有足够的发挥空间。

本文就意图利用技术指标构建交易策略，为投资者获得超额收益。

1.2 策略主要思路和创新点

本策略的主要思路是先对大盘进行技术分析，当市场不是特别弱势时，再对每个个股利用技术指标分析，符合条件的就加入买入池。

首先，对沪深 300 指数进行 RSI 和 AR 技术指标的分析，如果市场不是特别弱势，就发出进行下一步的信号。

其次，根据条件筛选出股票池。

最后，再对股票池进行一系列技术指标筛选，将符合全部条件的股票留下，并买入。

本文的创新之处在于利用技术指标进行筛选股票，与通常技术指标的使用方法有所不同。大量的研究者仅仅利用技术指标进行择时，但没有用其很好的选股。同时，本文利用的技术指标要远远多于同类研究。同类的研究大多使用两到三个指标组合，本文则是使用大量的指标组合进行选股。

1.3 研究内容

本文主要内容安排如下：

第一章为策略概述。简单介绍策略进行择时的基本原理，并简单阐述了策略整体思想和整体流程，以及策略的创新点。

第二章为文献综述。通过回顾国内外研究人员利用技术指标进行股价预测的相关文献，整理和分析技术指标的内容及发展现状、相关概念及特点进行了简要的介绍、梳理现有的相关量化策略。

第三章为模型设定。本章通过对策略流程的详细介绍和模型构建，来呈现本策略的具体建模思路和交易流程。

第四章为回测与实证研究。本章通过对策略在广发量化平台上的回测来分析策略的具体表现状况。

第五章为结论。本章阐述了全文的结论以及策略的不足之处和未来完善的方向。

2 文献综述

技术指标在证券投资分析中是否有效要追溯到 Fama(1970)[1]提出的有效市场假说，其中的弱式有效市场阐述了技术交易系统与购买持有策略将获得同等的期望收益并承担同等的风险。在国外，Wong et al. (2003)[2]认为技术指标策略在新加坡证券市场中能产生显著地积极收益；Murphy(1986) [3]、Marshall et al.（2009) [4]、Metghalchiet al.（2012)[5]分别认为在期货市场、美国证券市场以及台湾股票市场并非有利可图。在国内，张华和任若恩(2013)[6]、

周铭山等(2013)[7]验证了技术指标策略在外汇市场以及 A 股市场的有效性; 王志刚、曾勇和李平(2009)[8]发现技术分析在中国股票市场具有非线性预测能力; 孙碧波(2005)[9]分析了移动平均线在上证指数中无法获得超额收益，而陈卓思和宋逢明(2005)[10]则通过 Boostrap 方法发现图形技术分析可以提供十分显著的信息含量。

但真正将技术指标做成投资策略的文献极为稀少，除了少数几份券商研究报告外，几乎找不到相关文献。

由于中国股市弱有效性一直存疑，因此，可以认为技术指标的组合很有可能是有效的。

3 模型设定

3.1 模型假定

假定 3.1 市场组合为全部 A 股和一个空集的并集，$\mathrm{M}=\{s_i \cup \emptyset,\ i=1,2,\ldots\}$

假设 3.2 市场组合的子集的集合的交集、并集、联集、差集运算均是封闭的。即市场集合是一个可测空间。

假设 3.3 市场是有摩擦的。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。但不考虑冲击成本。

假设 3.4 所有证券的价格均是动态复权的。

假设 3.5 策略于每周三的 9 时整开始运行，不考虑时间复杂度。因此策略在该日的 9 时 30 分前便确定了当日的交易策略，并在开盘时执行该策略。

假设 3.6 不考虑分红和相关的税收。

3.2 止损设置

3.2.1 利用大盘止损

策略首先会判断大盘所处的状态（即弱势还是强势），如果大盘较弱势则清仓止损，不要买入任何股票，如果强势，则执行后续的策略。

当然本策略采用的判断条件是技术指标，因此，首先需要定义几个要用到的技术指标。

定义 3.1 RSI 指标

相对强弱指标（RSI 指标）是有韦尔斯·怀尔德(Welles Wilder)提出的。它是根据一定时期内上涨和下跌幅度之和的比率制作出的一种技术曲线。

公式为：

$$\mathrm{RSI}=\frac{上升平均数}{上升平均数+下降平均数}\times 100\%$$

性质 3.1 RSI 指标取值范围是[0,100]。

性质 3.2 RSI 指标高于 50 就表示为强势市场，反之为弱势市场。

性质 3.3 强弱指标多在 70 与 30 之间波动。当 RSI 上升到达 80 时，表示股市已有超买现象，如果一旦继续上升，超过 90 以上时，则表示已到严重超买的警戒区，股价已形成头部，极可能在短期内反转回转。当 RSI 指标下降至 20 时，表示股市有超卖现象，如果一旦继续下降至 10 以下时则表示已到严重超卖区域，股价极可能有止跌回升的机会。

性质 3.4 短期 RSI 上穿长期 RSI，可以认为市场较为强势。

基于以上性质，本文为了将 RSI 判断进行量化描述，因此：

定义 3.2 RSI 判断函数

假定 fr 是 RSI 指数的判断函数。因此：

$$\mathrm{fr}=\begin{cases}50, RSI>55\\ 40, RSI>68\\ 30, 68>\mathrm{RSI}>55\\ 20, 前五日RSI大于\ 50，\ RSI<50\end{cases}$$

当 fr=50 时，可以认为 RSI 判断市场为将上行，40 为市场已经到达高位，30 为持仓，20 为震荡盘整，10 为下行。

本策略需要用到的第二个指标是 AR 指标

定义 3.3 AR 指标

AR 指标是用来反映当前市场的人气状况的指标，公式为：

$$\mathrm{AR}=\frac{\sum_{i=1}^{N}(最高价\text{-}开盘价)}{\sum_{i=1}^{N}(开盘价\text{-}最低价)}$$

N 一般设定为 26 日。

AR 指标在 180 以上时，股市极高活跃；在 120 - 180 时，股市高活跃；在 70 - 120 时，股市盘整；在 60 - 70 以上时，股市走低；在 60 以下时，股市极弱。

由上述的性质，因此对 AR 指标设定类似的判断函数：

定义 3.4 AR 指标的判断函数

假设该函数为 far，因此：

$$\text{far}=\begin{cases}5, AR>180\\ 4, 180\geq AR>120\\ 3, 120\geq \text{AR}>70\\ 2, 70\geq AR>60\\ 1, 60\geq \text{AR}\end{cases}$$

因此，利用大盘指数的止损条件为：

定义 3.5 第一止损条件

利用对沪深 300 指数进行技术分析，设定相关止损条件 stop，当 stop=0 时，卖出全部持仓股票，并保持空仓。

$$\text{stop}=\begin{cases}0, \text{fr}=10\ 或AR连续三日低于60\ 等\\ 1, tx\end{cases}$$

stop=0 的具体条件为，fr=10 或 AR 连续三日低于 60 或三日前 AR 高于 200 当前低于 150.

3.2.2 利用均价止损

策略的第二止损条件是通过判断当日经过后文的选股程序，选出的股票集合（买入池）的平均建仓成本是否高于当前股价，来确定。

当买入池的平均建仓成本高于当前股价时，清仓，并且当日不买入任何股票。反之，不触发止损。

3.3 选股程序

当未触发止损条件时，选股程序会选择出来今日希望买入的股票。本文将选择出的股票集合称为买入池。首先需要获取整个市场的股票引入组合，然后根据相应的条件，筛选出符合买卖条件的买入池。再整体定义买入池前，需要定义一些与单支股票相关的变量。

定义 3.6 最大波浪

定义最大波浪 max_wave 是该股票 s 的一定时期[t,T]内的最高价的最高价减去最低价的最低价，再除以 t 时刻的开盘价。

$$\text{maxwave}=\frac{\max(最高价)-\min(最低价)}{开盘价_{\text{t}}}$$

定义 3.7 常规波浪

定义常规波浪 wave 为该股票 s 的一定时期[t,T]内的收盘价的最高值减去最低值，再除以 t 时刻的开盘价，再除以最大波浪。

$$\text{wave}=\frac{\max(收盘价)-\min(收盘价)}{开盘价_{\text{t}}/\text{maxwave}}$$

定义 3.8 价格极差

定义极差 se 为 T 时刻收盘价减去 t 时刻开盘价，再除以 t 时刻的开盘价，再除以最大波浪。

$$\text{se}=\frac{\text{收盘价}_{T}-开盘价_{\text{t}}}{开盘价_{\text{t}}/\text{maxwave}}$$

定义 3.9 买入池

买入池$\text{B}=\{s_j, j=1,2,\dots\}$是一个证券的集合，它是市场组合的子集，即$\text{B}\subset\text{M}$。∃S 中的元素 s 满足下列条件时，该一系列证券集合 S 中计数测度最大的一个集合即为买入池 B。

1° 流通市值小于 200 亿元。

2° 非 ST 和*ST 类股票。

3° Se>0.

4° t 到 t+15 时刻的最高价等于最高价在时期[t，T]的最高值。

5° t 到 t+15 时刻的最低价等于最低价在时期[t，T]的最低值。

6° 常规波浪在整个股票池中拍名前五的

7° 极差排名最后一位的

8° 极差除以最大波浪的值大于-0.85

9° 最大波浪大于 0.7 的

当买入池确定后，策略将在开盘前下单买入这个买入池。在每日收盘前，全部平仓，并保持空仓，在次日开盘前再重新执行上述步骤。

4 回测和实证研究

4.1 数据说明

本文的策略以中国内地所有 A 股为研究对象，选择 2015 年 1 月 5 日至 2018 年 7 月 30 日合共 700 多个交易日的历史数据进行实证研究。数据来源是广发量化平台的 API 接口。

策略主要的变量选择：

变量名称	说明
输入变量	收盘价，最高价之最高值，15 日的最高价，最低价之最低值，15 日的最低价，常规波浪，最高波浪和价差
输出变量	买入池

表 4.1 策略主要使用的数据

本文的回测是采用滚动回测的方式。回测是模拟从 2015 年 1 月 5 日开始，每一个交易日都模拟运行一次。每个交易日开盘前，先筛选出买入池，并开盘时买入，再收盘前平仓，第二个交易日再执行同样的步骤。

4.2 工具说明

本策略使用 Python 2.X/3.X 语言在广发量化平台上编译运行的（详细代码参见附录 3）。并且需要提前安装如下模块：

模块名称	说明
Numpy	NumPy 系统是 Python 的一种开源的数值计算扩展。
Pandas	pandas 是基于 NumPy 的一种工具，该工具是为了解决数据分析任务而创建的。
Sklearn	基于 python 的机器学习模块
Jqdata	广发量化平台自带模块

表 4.2 Python 模块说明

4.3 回测结果

本文采用广发量化平台进行回测。

假定交易成本只有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。但不考虑冲击成本。

主要结果：

项目	结果
总收益	251.04%
策略年化收益	43.34%
基准收益	-0.53%
Alpha	0.412
Beta	0.460
Sharpe	0.934
Sortino	1.108
Information Ratio	1.476
Algorithm Volatility	1.017
Benchmark Volatility	0.421
胜率	0.256
日胜率	0.508
盈亏比	1.368
盈利次数	67
亏损次数	65
最大回撤	55.208%
最大回撤出现时间	2016-08-19~2018-07-26

表 4.3 策略回测结果概况

从回测结果看，策略大幅跑赢市场，且夏普率为 0.934，也算是较高的水平。另外，策略的风险度较市场略高（策略波动率 1.017，对比市场 0.421），考虑到投资者是风险厌恶的，虽然策略承担了较高的风险，但是考虑到均衡市场下，定价是线性的，策略的收益要远远高于其承担的风险。策略的盈亏比为 1.368，但亏损次数多余盈利次数，说明策略的止损条件设定较好，在盈利时能较好的获利，而损失时，能较好的止损。最大回撤较大，但本文认为出现在特定时期，不代表策略止损条件或风险控制的设定存

在严重漏洞。

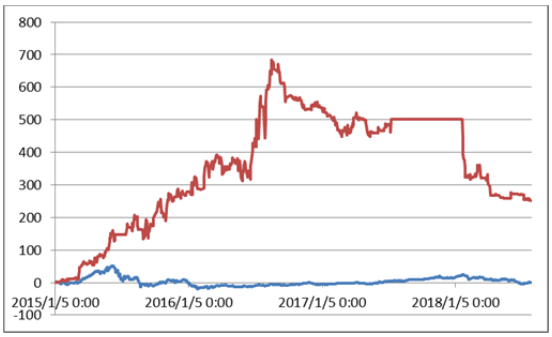

图 4.1 回测表现图（红色为策略，蓝色为市场）

由图中走势看出，策略在牛市表现较好，在熊市中，也能长期跑赢大盘。在其他熊市周期中，策略回撤幅度也小于市场。因此，可以认为策略是可以穿越牛熊周期跑赢市场的。

5 结论

5.1 策略结论

策略明显跑赢大市，可以认为本策略是有效的，且可以获得长期复利增长。策略时间复杂度也在合理范围内，因此可以认为本策略可以在正常情况下完成交易。本文亦证明了支持向量机在股价预测领域有较高的预测准确率和强大的容错能力。

5.2 策略不足与展望

策略存在以下不足：

1.策略止损条件过于简单，需要更好的方法进行止损，减少回撤。

2.策略选股程序过于简单，仅仅通过数个指标的计算配合，进行选择，容易导致买入池的有效性不足。

针对以上情况，策略将来需要做出如下修改：

1.设定更好的止损。例如利用隐马尔可夫链算法，预判当前市场处于牛市、熊市亦或是震荡市，当市场环境为震荡市或熊市时止损。

2. 可采用其他量化择时模型配合判断，例如牛熊线模型、Hurst 指数、分形原理。

参考文献

［1］Fama E.，1970. Efficient Capital Markets: A Review of Theory and Empirical Work. Journal of Finance，Vol. 25: 383 －417.

［2］Wing －Keung Wong，Meher Manzur，Boon －Kiat Chew，2003. How Rewarding is Technical Analysis? Evidence from Singapore Stock Market. Applied Financial Economics，Vol. 13: 543 －551.

［3］Austin，Murphy J.，1986. Futures Fund Performance: A Test of the Effectiveness of Technical Analysis. The Journal of Futures Markets，Vol. 6: 175－185.

［4］Ben R. Marshall，Sun Qian，and Martin Young，2009. Is Technical Analysis Profitable on US Stocks with Certain Size, Liquidity or Industry Characteristic?. Applied Financial Economics，Vol. 19: 1213－1221.

［5］Massoud Metghalchi，Yung － Ho Chang，and Xavier Garza － Gomez，2012. Technical Analysis of the Taiwanese Stock Market. International Journal of Economics and Finance，Vol. 4: 90－102.

［6］张华，任若恩. 基于 Dempster －Shafer 证据理论的外汇交易策略研究［J］. 数理统计与管理，2013(3) .

［7］周铭山，冯新力，林靓，方旭赟，周开国. A 股市场均线策略有效性与收益率随机特征研究［J］. 证券市场导报，2013(1).

［8］王志刚，曾勇，李平. 中国股票市场技术分析非线性预测能力的实证检验［J］. 管理工程学报，2009(1).

［9］孙碧波. 移动平均线有用吗？———基于上证指数的实证研究［J］. 数量经济技术经济研究，2005(2) .

［10］陈卓思，宋逢明. 图形技术分析的信息含量［J］. 数量经济技术经济研究，2005(9) .

附录 1 策略代码实现

```
# 导入模块
import math
import talib as tl
import pandas as pd
import numpy as np
from datetime import timedelta
import talib

def initialize(context):
    # 初始化此策略
    # 设置要操作的股票池为空，每天需要不停变化股票池
    set_universe([])

    # 设置手续费，买入时万分之三，卖出时万分之三加千分之一印花税, 每笔交易最低扣
5 块钱
    set_commission(PerTrade(buy_cost=0.0003, sell_cost=0.0013, min_cost=5))

    # 设置风险基准为沪深 300 指数
    g.riskbench = '000300.XSHG'

    # 设置基准对比为沪深 300 指数
    set_benchmark(g.riskbench)

    # 关闭部分 log
    #log.set_level('order', 'error')

    # 设置真实交易时间
    set_option('use_real_price',True)

    # 初始化日期
    g.d_yesterday = ''
    g.d_today = ''

    # 定义常量
    g.con_START_DATE = 100 # 股票上市时间限制
    g.con_MARKET_CAP = 200 # 流通市值标准

    # 初始化综合风险结果
    g.ris = 10
    g.ar = 1
```

```
    # 记录真实 RSI_F 和 AR
    g.RIS_T = 0
    g.AR_T = 0

    # 上次风险判断标准
    g.risk_flag = 1
    g.buy_flag = 0

    # 初始化风险变化趋势数据
    g.risk_day1 = 0
    g.risk_day2 = 0
    g.risk_day3 = 0
    g.risk_list = [50]

    # 初始化风险参数
    g.con_FAST_RSI = 20
    g.con_SLOW_RSI = 60
    g.con_AR_COUNT = 26 # 以 4 天为期间计算 AR 值（普通为 26 天），反应长期趋势

    # 初始化当日买卖列表
    g.stock_buy = []
    g.stock_sell = []

    g.df_result = pd.DataFrame()

    run_weekly(weekly23, 1, time='open')
    run_weekly(weekly23, 2, time='open')
    run_weekly(weekly23, 3, time='open')
    run_weekly(clean, 4, time='open')
    run_weekly(weekly23, 5, time='open')

    #run_daily(daily, time='open')

# 取得对应股票代码的 RSI 风险指标
def get_RSI(stock):
     # 取得历史收盘价格
    df_hData = attribute_history(stock, 80, unit = '1d', fields = ('close', 'high', 'low'), skip_paused
= True)
    df_hData_today = get_price(stock,start_date = g.d_today, end_date = g.d_today, frequency =
'daily', fields = ['close', 'high', 'low'])

    df_hData = df_hData.append(df_hData_today)

    closep = df_hData['close'].values
```

```
    RSI_F = tl.RSI(closep, timeperiod = g.con_FAST_RSI)
    RSI_S = tl.RSI(closep, timeperiod = g.con_SLOW_RSI)
    isFgtS = RSI_F > RSI_S

    rsiS = RSI_S[-1]
    rsiF = RSI_F[-1]

    '''
    慢速 RSI  在 55 以上时，单边上涨市场，快速 RSI 上穿慢速 RSI 即可建仓
    慢速 RSI  在 55 以下时，调整震荡市场，谨慎入市，取连续 N 天快速 RSI 大于慢速 RSI 建仓
    慢速 RSI  在 60 以上时，牛市，无需减仓操作持仓即可
    '''
    # 基准仓位值
    bsFlag = 10

    if rsiS > 55 and isFgtS[-1]:
        bsFlag = 50 # "上行"
    elif rsiS > 68:
        bsFlag = 40 # "高位"
    elif rsiS > 60:
        bsFlag = 30 # "持仓"
    elif rsiS <= 55 \
        and isFgtS[-1] and isFgtS[-2] \
        and isFgtS[-3] and isFgtS[-4] \
        and isFgtS[-5] :
            bsFlag = 20 # "盘整建仓"
    else:
        bsFlag = 10 # "下行"

    g.RSI_T = rsiS

    return bsFlag

# 取得对应股票代码的 AR 活跃度指标
def get_AR(stock, count):
    df_ar = attribute_history(stock, count - 1, '1d', fields=('open', 'high', 'low'), skip_paused = True)
    df_ar_today = get_price(stock,start_date = g.d_today, end_date = g.d_today, frequency = 'daily', fields = ['open', 'high', 'low'])

    df_ar = df_ar.append(df_ar_today)
```

```
    ar = sum(df_ar['high'] - df_ar['open']) / sum(df_ar['open'] - df_ar['low']) * 100

    '''
    AR 指标  在 180 以上时，股市极高活跃
    AR 指标  在 120 - 180 时，股市高活跃
    AR 指标  在 70 - 120 时，股市盘整
    AR 指标  在 60 - 70 以上时，股市走低
    AR 指标  在 60 以下时，股市极弱
    '''
    brFlag = 1

    if ar > 180 :
        brFlag = 5
    elif ar > 120 and ar <= 180 :
        brFlag = 4
    elif ar > 70 and ar <= 120 :
        brFlag = 3
    elif ar > 60 and ar <= 70 :
        brFlag = 2
    else :
        brFlag = 1

    g.AR_T = ar

    return brFlag

# 取得对应大盘的风险
def get_stock_risk(stock, count):
    # 今日风险估算
    rsi = get_RSI(stock)
    ar = get_AR(stock, count)

    g.ris = rsi
    g.ar = ar

    #record(AR_T=g.AR_T)

    g.risk_day1 = g.risk_day2
    g.risk_day2 = g.risk_day3
    g.risk_day3 = g.AR_T

    buy_flag = 2

    if rsi == 10:
```

```
            buy_flag = 0
        elif rsi == 20:
            buy_flag = 2
        else:
            buy_flag = 1

        # 趋势控制
        if g.risk_day1 < 60 and g.risk_day2 < 60 and g.risk_day3 < 60:
            # 持续低迷
            buy_flag = 0
        elif g.risk_day2 * 0.3 > g.risk_day3:
            # 急跌
            buy_flag = 0
        elif g.risk_day1 < g.risk_day2 and g.risk_day2 < g.risk_day3 and \
            g.risk_day3 < 80 and g.risk_day3 > 65:
            # 弱市回升
            buy_flag = 1
        elif g.risk_day1 > g.risk_day2 and g.risk_day2 > g.risk_day3 and \
            g.risk_day3 < 150 and g.risk_day1 > 200:
            # 强市下跌
            buy_flag = 0
        elif g.risk_day1 > g.risk_day2 and g.risk_day2 > g.risk_day3 and \
            g.risk_day3 < 150 and g.risk_day1 > 150:
            # 中市下跌
            buy_flag = 2
        elif g.risk_day1 > g.risk_day2 * 0.95 and g.risk_day2 > g.risk_day3 and \
            g.risk_day3 < g.risk_day2 * 0.6 and g.risk_day1 > 180:
            # 强市下跌
            buy_flag = 0
        elif g.risk_day1 > 70 and g.risk_day2 > 70 and g.risk_day3 > 70:
            # 维持正常
            buy_flag = 1

        else:
            buy_flag = 2

        return buy_flag

def clean(context):

        for stock in context.portfolio.positions.keys():
            order_target(stock, 0)
        #weekly(context)
        log.info("定期清空")
```

```
# 开盘后止损止盈
def check_price(context, stock):
    price = context.portfolio.positions[stock].price
    avg_cost = context.portfolio.positions[stock].avg_cost

    if avg_cost   > price:
        return False
    else:
        return True

def weekly23(context):
    #if context.current_dt.isoweekday() <> 14:
    if len(context.portfolio.positions.keys()) <= 0:
        weekly(context)

def weekly(context):
    for stock in context.portfolio.positions.keys():
        order_target(stock, 0)

    cash = context.portfolio.cash
    log.info(g.stock_buy)
    for stock in g.stock_buy:
        order_value(stock, cash / len(g.stock_buy))

    if len(g.df_result.index) > 0:
        log.info(g.df_result)

def daily(context):
    for stock in g.stock_sell:
        order_target(stock, 0)

    cash = context.portfolio.cash

    for stock in g.stock_buy:
        order_value(stock, cash / len(g.stock_buy))

# 获得每天的数据
def get_stock_list(context):
    # 取得流通市值 200 亿以下的股票数据
    d_startdate = (context.current_dt - timedelta(days = g.con_START_DATE)).strftime("%Y-%m-%d")

    q = query(
```

```
        valuation.code, valuation.circulating_cap
    ).filter(
        valuation.code.notin_(['002473.XSHE', '000407.XSHE']),
        valuation.circulating_market_cap < g.con_MARKET_CAP
    )

    df = get_fundamentals(q, date = d_startdate)

    df.index = list(df['code'])

    # 去除 ST，*ST
    st = get_extras('is_st', list(df['code']), start_date = g.d_today, end_date = g.d_today, df = True)
    st = st.iloc[0]
    stock_list = list(st[st == False].index)

    days = 29

    # 获得股票池数据
    df_list = get_price(stock_list, start_date=g.d_today, end_date=g.d_today, frequency='daily',
fields=['open', 'close', 'high', 'low', 'paused', 'volume'])

    # 获取收盘价
    df_close = history(days, unit='1d', field='close', security_list = stock_list, skip_paused=True)
    df_close = df_close.append(df_list['close'])

    # 获取收盘价
    df_high = history(days, unit='1d', field='high', security_list = stock_list, skip_paused=True)
    df_high = df_high.append(df_list['high'])

    # 获取收盘价
    df_low = history(days, unit='1d', field='low', security_list = stock_list, skip_paused=True)
    df_low = df_low.append(df_list['low'])

    # 获取历史停牌
    df_paused_sum = history(days, unit='1d', field='paused', security_list = stock_list)
    df_paused_sum = df_paused_sum.append(df_list['paused'])
    df_paused_sum = pd.DataFrame(np.sum(df_paused_sum))
    df_paused_sum.columns = ['paused_sum']

    # 获取成交量
    df_volume = df_list['volume']
    df_volume = df_volume.T

    for col in df_volume.columns:
```

```
        df_volume[col] = df_volume[col] / (df['circulating_cap'] * 100)

df_volume.columns = ['volume']

# 最高价的最高价
df_high_h = pd.DataFrame(df_high.max())
df_high_h.columns = ['high_h']

# 最低价的最低价
df_low_l = pd.DataFrame(df_low.min())
df_low_l.columns = ['low_l']

# 收盘价的最高价
df_close_h = pd.DataFrame(df_high.max())
df_close_h.columns = ['close_h']

# 收盘价的最低价
df_close_l = pd.DataFrame(df_close.min())
df_close_l.columns = ['close_l']

# 前 15 日最高价
df_15_h = pd.DataFrame(df_high.head(15).max())
df_15_h.columns = ['15_h']

# 后 15 日最低价
df_15_l = pd.DataFrame(df_low.tail(15).min())
df_15_l.columns = ['15_l']

# 开始日收盘价
df_start = df_close.head(1)
df_start.index = ['start']
df_start = df_start.T

# 结束日收盘价
df_end = df_close.tail(1)
df_end.index = ['end']
df_end = df_end.T

# 获取停牌
df_paused = df_list['paused'].T
df_paused.columns = ['paused']

df_result = pd.concat([df_start, df_end], axis = 1, join_axes = [df_start.index])
df_result = pd.concat([df_result, df_paused], axis = 1, join_axes = [df_result.index])
```

```
    df_result = pd.concat([df_result, df_volume], axis = 1, join_axes = [df_result.index])
    df_result = pd.concat([df_result, df_high_h], axis = 1, join_axes = [df_result.index])
    df_result = pd.concat([df_result, df_low_l], axis = 1, join_axes = [df_result.index])
    df_result = pd.concat([df_result, df_close_h], axis = 1, join_axes = [df_result.index])
    df_result = pd.concat([df_result, df_close_l], axis = 1, join_axes = [df_result.index])
    df_result = pd.concat([df_result, df_15_h], axis = 1, join_axes = [df_result.index])
    df_result = pd.concat([df_result, df_15_l], axis = 1, join_axes = [df_result.index])

    df_result = df_result[df_result['paused'] == 0]

    df_result['usual_wave'] = (df_result['close_h'] - df_result['close_l']) / df_result['start']
    df_result['max_wave'] = (df_result['high_h'] - df_result['low_l']) / df_result['start']
    df_result['start_end'] = (df_result['end'] - df_result['start']) / df_result['start']



    df_result['usual_wave'] = df_result['usual_wave'] / df_result['max_wave']
    df_result['start_end'] = df_result['start_end'] / df_result['usual_wave']
    #df_result = df_result[df_result['usual_wave'] < 1]

    df_result = df_result[df_result['start_end'] < 0]
    df_result = df_result[df_result['15_h'] == df_result['high_h']]
    df_result = df_result[df_result['15_l'] == df_result['low_l']]

    df_result = df_result.sort(columns = 'usual_wave', ascending = False).head(5)
    df_result = df_result.sort(columns = 'start_end', ascending = False).tail(1)

    df_result = df_result[df_result['start_end'] / df_result['max_wave'] > -0.85]
    df_result = df_result[df_result['max_wave'] < 0.7]

    g.df_result = df_result
    g.stock_buy = set(list(g.stock_buy) + list(df_result.index))

# 每天交易前调用
def before_trading_start(context):
    # 昨天
    g.d_yesterday = (context.current_dt    - timedelta(days = 1)).strftime("%Y-%m-%d")

    # 今天
    g.d_today = (context.current_dt).strftime("%Y-%m-%d")

# 每个单位时间(如果按天回测,则每天调用一次,如果按分钟,则每分钟调用一次)调用一次
def handle_data(context, data):
    pass
```

```
# 每天交易后调用
def after_trading_end(context):
    g.buy_flag = get_stock_risk(g.riskbench, 4)
    g.stock_buy = []
    g.stock_sell = []

    if g.buy_flag <> 0:
        get_stock_list(context)
```