程序化设计技术文档

目录

[软件与框架 2](#_Toc482975752)

[anaconda使用 2](#_Toc482975753)

[spyder使用 2](#_Toc482975754)

[notebook jupyter使用 2](#_Toc482975755)

[tushae使用 2](#_Toc482975756)

[talib使用 2](#_Toc482975757)

[技术指标 2](#_Toc482975758)

[K线组合 5](#_Toc482975759)

[进阶 7](#_Toc482975760)

[米筐使用 10](#_Toc482975761)

[人工智能与机器学习 10](#_Toc482975762)

[几个市场常用策略 10](#_Toc482975763)

[BM效应 11](#_Toc482975764)

[Graham number 格雷厄姆数字价值投资法 11](#_Toc482975765)

[A股市场的ETF轮动策略 11](#_Toc482975766)

[基本面选股器: Piotroski F-Score ranking system 11](#_Toc482975767)

[基于NPL对爬取的舆情数据进行情感分析 11](#_Toc482975768)

[如何筛选策略 11](#_Toc482975769)

[仓位控制与止损 16](#_Toc482975770)

# 软件与框架

## anaconda使用

## spyder使用

## notebook jupyter使用

## tushae使用

首先导入tushare库：import tushae as ts

获取股票K线数据，ts.get\_k\_data(‘share code’)

获取指数K线数据，ts.get\_k\_data(‘share code’,index=True)

例：

#获得上证指数数据

ts.get\_k\_data('000001',index=True)

## talib使用

首先导入talib库：import talib as ta

### 技术指标

例：

#获取收盘价数据，拿到的是Dataframe，转换成ndarray

dcclose=ts.get\_k\_data('000001',index=True)['close'].values

#调用talib ma均线指标，获得30日和120日ma均线

ma30 = ta.MA(dcclose, timeperiod=30, matype=0)

ma120 = ta.MA(dcclose, timeperiod=120, matype=0)

#通过matplotlib画图，画出均线

import matplotlib.pyplot as plt

plt.plot(ma30)

plt.plot(ma120)

#图如下：



常用的talib指标为：

|  |  |  |  |
| --- | --- | --- | --- |
| 指标 | 描述 | Python Api 接口或算法 |  |
|  |  |  |  |
| BIAS | 乖离率 | 乖离率=[(当日收盘价-N日平均价)/N日平均价]\*100% |  |
| BOLL BBANDS | 布林轨道 | upperband, middleband, lowerband = BBANDS(close, timeperiod=5, nbdevup=2, nbdevdn=2, matype=0) |  |
| CCI | 顺势指标 | real = CCI(high, low, close, timeperiod=14) |  |
| KDJ  STOCH | 随机指标 | slowk, slowd = STOCH(high, low, close, fastk\_period=5, slowk\_period=3, slowk\_matype=0, slowd\_period=3, slowd\_matype=0) |  |
| MA | 移动平均线 | real = MA(close, timeperiod=30, matype=0) |  |
|  |  |  |  |
| MACD | 指数平滑移动平均线指标 | macd, macdsignal, macdhist = MACD(close, fastperiod=12, slowperiod=26, signalperiod=9) |  |
| MTM  MOM | 动量指标 | real = MOM(close, timeperiod=10) |  |
|  |  |  |  |
| PSY | 情绪指标 | 无 |  |
| RSI | 相对强弱指标 | real = RSI(close, timeperiod=14) |  |
| WR  WILLR | 威廉指标 | real = WILLR(high, low, close, timeperiod=14) |  |

编程习惯：

上面列表中，都是直接API接口，实际使用中，先要导入talib库，建议使用import talib as ta ，这样每个指标使用的时候，都要加上“ta.”比如MACD指标就要写成：macd, macdsignal, macdhist = ta.MACD(close, fastperiod=12, slowperiod=26, signalperiod=9)

MA指标就要写成：real = talib.MA(close, timeperiod=30, matype=0)

意外的是，最不提倡的导入方法“from talib import \*”是无法生效的。

如前面所讲，talib需要array数据格式，大部分网上拿到的数据都是DataFrame格式，因此存在一个数据转换过程！转换可以使用.value 方法实现。

如果是在米筐等测试框架中，在handle\_bar()中使用该指标数据，那么要取矩阵最后一个数据，也就是用[-1]定位。如前面拿到MA指标的real数据，最后一位就是real[-1]

### K线组合

Talib也支持K线组合的

如：

import tushare as ts

import talib as ta

dcdata =ts.get\_k\_data('000001',index=True)

open,high,low,close = dcdata['open'].values,dcdata['high'].values,dcdata['low'].values,dcdata['close'].values

#Engulfing Pattern 吞噬模式（阴阳并线）

integer = ta.CDLENGULFING(open, high, low, close)

输出的integer是一个numpy.ndarray ,可以看到数据里有100和-100

type (integer)

Out[81]: numpy.ndarray

print(integer)

[0 0 0 ..., 0 0 0]

100 in integer

Out[83]: True

-100 in integer

Out[84]: True

常用K线形态组合列表：

|  |  |  |
| --- | --- | --- |
| K线形态组合名 | 描述 | Python接口 |
| CDL3BLACKCROWS | 三只黑乌鸦 | integer = CDL3BLACKCROWS(open, high, low, close) |
| CDL3LINESTRIKE | 三线打击 | integer = CDL3LINESTRIKE(open, high, low, close) |
| CDL3WHITESOLDIERS | 白三兵（红三兵） | integer = CDL3WHITESOLDIERS(open, high, low, close) |
| CDLADVANCEBLOCK | 上升受阻（） | integer = CDLADVANCEBLOCK(open, high, low, close) |
| CDLBELTHOLD | 捉腰带线 | integer = CDLBELTHOLD(open, high, low, close) |
| CDLENGULFING | 吞噬模式（阴阳并线） | integer = CDLENGULFING(open, high, low, close) |
| CDLLONGLEGGEDDOJI | 长十字星 | integer = CDLLONGLEGGEDDOJI(open, high, low, close) |
| CDLLONGLINE | 长实体 | integer = CDLLONGLINE(open, high, low, close) |
| CDLSHORTLINE | 短蜡烛 | integer = CDLSHORTLINE(open, high, low, close) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

参见：<https://www.ricequant.com/community/topic/2393/>

如前面一样，调用的时候建议使用import talib as ta,函数前添加“ta.”

### 进阶

可以查看talib所有的函数：

print(ta.get\_functions())

print(ta.get\_function\_groups())

每个函数可以通过help（）或“？”来获得帮助，

如：

help(ta.MA)

Help on built-in function MA in module talib.func:

MA(...)

MA(real[, timeperiod=?, matype=?])

Moving average (Overlap Studies)

Inputs:

real: (any ndarray)

Parameters:

timeperiod: 30

matype: 0 (Simple Moving Average)

Outputs:

real

ta.MA?

Docstring:

MA(real[, timeperiod=?, matype=?])

Moving average (Overlap Studies)

Inputs:

real: (any ndarray)

Parameters:

timeperiod: 30

matype: 0 (Simple Moving Average)

Outputs:

real

Type: builtin\_function\_or\_method

抽象函数abstract

talib使用熟练之后，可以使用abstract抽象API，每个函数的输入都是固定格式的arrays .

#生成一个含有价格／成交量等5项数据的数据字典inputs

import numpy as np

# note that all ndarrays must be the same length!

inputs = {

'open': np.random.random(100),

'high': np.random.random(100),

'low': np.random.random(100),

'close': np.random.random(100),

'volume': np.random.random(100)

}

#直接调用MA

output=ta.abstract.MA(inputs)

#outpu就是默认的30日均线，如图

plt.plot(output)



#还可以通过Function使用：

newma=ta.abstract.Function('sma')

newma(inputs,5)

Out[83]:

array([ nan, nan, nan, ..., 0.50493402,

0.52459283, 0.51730766])

通过字符串赋值（sma/macd/obv等），就可以更加方便的在自己的程序中自如的构建和使用指标了！

结束语

恭喜朋友们，已经学完了talib的使用，是不是从刚开始的茫然不知所措，到现在感觉这个东西很简单呢？ 通过talib来使用各种指标，没有原来想象的那么难吧？

下面请大家继续两个课题：1 通过不断的实践练习，更佳熟练使用talib 。2 集合周围投资者的优秀思路，选择1-2个指标，做量化交易，通过历史回测，判断该指标的有效性。若无效，继续拓展思路，筛选其它指标。

## 米筐使用

## RQalpha测试框架使用

安装rqalpha是很简单的，直接pip安装就行了。

安装完成后，获取回测数据：

rqalpha update\_bundle

bundle 默认存放在 ~/.rqalpha 目录里，您也可以指定 bundle 的存放位置，

$ rqalpha update\_bundle -d target\_bundle\_path

如果您使用了指定路径来存放 bundle，那么执行程序的时候也同样需要指定对应的 bundle 路径。

$ rqalpha run -d target\_bundle\_path .....

所以小伙伴们，建议直接放在默认目录里就好了。

使用方法，可以通过配置文件传参数，也可以直接在命令行里全部写出来。

参数有：

数据源路径

策略文件路径

回测起始时间

回测结束时间

起始资金

Benchmark

如果运行 RQAlpha 时不指定配置文件，那么会在 ~/.rqalpha/ 文件夹下创建 config.yml 文件作为默认配置文件。

如果您想要直接获得一份配置文件，也可以通过如下命令来获得。

$ rqalpha generate\_config

假如我们的策略存放在了 ./rqalpha/examples/buy\_and\_hold.py 路径下， 数据源存放在 ./.rqalpha/bundle/ 路径下，回测的起始时间为 2016-06-01, 结束时间为 2016-12-01，我们给策略分配的起始资金为 100000, Benchmark 设置为 000300.XSHG

那么我们通过如下命令来运行回测:

rqalpha run -f ./rqalpha/examples/buy\_and\_hold.py -d ./.rqalpha/bundle/ -s 2016-06-01 -e 2016-12-01 --stock-starting-cash 100000 --benchmark 000300.XSHG

如果我们想要以图形的方式查看回测的结果， 则增加 --plot 参数

rqalpha run -f ./rqalpha/examples/buy\_and\_hold.py -d ./.rqalpha/bundle/ -s 2016-06-01 -e 2016-12-01 --stock-starting-cash 100000 --benchmark 000300.XSHG --plot

如果想把结果保存下来，出了保存图片外，还可以通过 -o 参数将结果保存成 pkl 文件。

rqalpha run -f ./rqalpha/examples/buy\_and\_hold.py -d ./rqalpha/bundle/ -s 2016-06-01 -e 2016-12-01 --stock-starting-cash 100000 --benchmark 000300.XSHG --plot -o result.pkl

等回测结束后可以通过 pandas.read\_pickle 函数来读取数据进行之后的数据分析。

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | import pandas as pd  result\_dict = pd.read\_pickle('result.pkl')  result\_dict.keys()  # [out]dict\_keys(['total\_portfolios', 'summary', 'benchmark\_portfolios', 'benchmark\_positions', 'stock\_positions', 'trades', 'stock\_portfolios']) |

## 人工智能与机器学习

prophet

# 几个市场常用策略

## BM效应

## Graham number 格雷厄姆数字价值投资法

https://www.ricequant.com/community/topic/285/

## A股市场的ETF轮动策略

<https://www.ricequant.com/community/topic/283/>

## 基本面选股器: Piotroski F-Score ranking system

<https://www.ricequant.com/community/topic/297/>

## 基于NPL对爬取的舆情数据进行情感分析

https://www.ricequant.com/community/topic/3339//2

# 如何筛选策略

本来这一章想放在上面一章的最后一节，但是感觉这个非常重要，所以自成一章，专门讨论。

什么样的策略的是好策略呢？话说一切不赚钱的策略都是耍流氓，是否赚钱，是判断一个策略好坏的唯一标准。不过赚钱是表象，在策略选择上，不要被表面的赚钱现象蒙蔽了眼睛。想当然的认为某种方法会赚钱是不可取的。

筛选优秀的策略，有如下几点：

1 没有未来函数

未来函数的害处非常大，很多指标高手在初期，就是寻找和淘汰一个个带有未来函数的指标过程，能够不被未来函数蒙蔽，是判断一个人是否从初学者进入称为高手的条件之一。比如大家熟知的“粉进黄出”这个指标，当年就非常流行，很多初学者会把这个指标当成宝贝。“粉进黄出”，顾名思义，就是入场信号用粉色表示，出场信用用黄色表示，粉色信号都出现在最低点，黄色信号都出现在最高点！如果按照信号买卖，100%正确，能挣很多钱！很神奇对吧？但是使用一段时间后，就会发现它是骗人的，因为它已经发出的信号会随着行情而发生变化。于是人们就转而找下一个类似的带未来函数的神奇的指标。一直到某一天，自己突然醒悟了，或者被别人点醒了，原来这些指标之所以神奇，是因为它们使用了未来函数，就像一个人坐时空机器从未来到我们身边，他当然可以把股市的高点和低点一点儿不错的指出来，就像我们现在去看历史上的走势，比如看2016年的股市，也能弄清楚哪里该进场，哪里该出场。

几乎所有的带未来函数的指标表现都很完美，历史回测也很完美，但是从哲学上讲，这个世界没有完美的，拥抱完美，就是拥抱死亡。所以带未来函数的指标，直接剔除，不要有任何的犹豫，它不值得你浪费任何一秒时间！

还有一种比较容易忽略的信号消失问题，使用普通的收盘价方法，盘中信号有可能消失或变动，导致回测和实盘效果可能相差非常大！原因是最后一根K线的收盘价，实际交易时间是一直变化的，只有到了该K线的最后一刹那，两者才一致！换句话说，这是K线级别里使用了未来函数！解决的方法有三种：1 不使用那些盘中会变化的数据，而是使用不变的数据，比如开盘价。 2 等收盘价完整走出来，不会变的时候再使用它，也就是策略中取前一根K线的数据，因为那都是确定的 。3 把交易设定在早晨开盘集合竞价或下午收盘集合竞价，这样也可以使模拟和实盘的信号一致。

一般量化交易的时候，不小心就会忽略上面讲的情况，如果你想真正做一套有用的交易系统，也就是赚钱的交易系统，编程实现的时候一定要妥善解决好这个问题。实际使用中会发现，解决掉大大小小的未来函数问题后，大部分指标的收益率大幅下降，但是好处也是显而易见的，这会让回测和实盘的信号一致，让回测真正有用。

2 有逻辑或统计数据支持

经济学不是科学，证券市场的方法也没有必然的对或者错，但是一个投资方法，一次投资操作，要有逻辑或者统计数据支持才行。如果逻辑不通或者统计数据不支持，那么某一次盈利很可能是小概率事件，就像守株待兔一样，这绝不是一个好的投资方法。

最符合逻辑的分析方法，是基本面分析方法。比如用供需理论，能很好的解释股票／期货的涨跌，建立在供需理论基础上的分析投资方法，就能过逻辑这一关。大家可以多从这方面寻找突破口！

最符合统计数据的，就是“一月效应和春节效应”，即每年一月份和春节后大盘会有一定涨幅的几率较大。另外像常见的均线指标／K线组合等，也有很多是通过统计得来的。

统计数据和逻辑是相辅相成的，一方会隐含另一方，相辅相成。比如供需理论符合逻辑，同时它也符合统计规律，市场资金充裕的时候，股市一般上涨的概率高。再比如“一月效应和春节效应”，是通过统计历年来走势得出的，而为什么有这个规律呢？是因为春节后是一年中资金最充裕的时候，这也同时符合供需理论的逻辑。

所以我们的工作就是：

1 通过大数据挖掘，统计出投资规律，并落实到实际交易中

2 寻找市场投资逻辑，用统计数据进行佐证，应用到实际交易中

需要注意的是，逻辑和统计数据是有可能随着时间而变化的。比如2006-2007年牛市期间，星期一上涨的几率大，而到了2008年熊市，就变成了星期一下跌的几率大！展开来说，整个市场的投资方法都是随着时间在变化的，没有一种方法是一直有效的。所以不要幻想找到一种方法，通过编程进行程序化交易，然后就能永远的自动赚钱。大部分投资方法都是跟随市场周期在变化的，比如有些投资方法是牛市用的，有些是熊市用，还有一些是震荡市适用的。有时候市场风格转换很快，比如一段时间可能题材股涨的好，过几个星期后可能轮动到蓝筹股，相应地，投资逻辑也要跟随市场进行修改，否则就可能踏错市场节拍。

3 可描述／可实现

一个投资方法，没有未来函数，信号可靠，也符合逻辑和统计规律，但是却很难呈现出来，说不清楚，或者无法实现，那么也不是好的投资策略，因为你根本写不出来啊。比较典型的是两个例子：盘感和波浪理论。其实如果一个策略很难呈现出来，它本身就比较难去统计了，尤其是很难用回测去测试，还是那句话，因为根本就写不出来啊！

4 风险低／收益匹配

5 安慰剂型策略

6

问题策略：

符合以下条件的策略，可以直接否定，根本就不用再花时间验证的。

1 完美天使策略

拥抱完美，就是拥抱死亡。如果看到一个完美的策略，一定要绕路走。因为这个不会有完美策略，如果你看到了，那就一定是表相，它背后一定有问题！要么逻辑上有问题，要么统计数据有问题，或者它本身就是为了骗人而存在的。

2 自我毁灭策略

风险极高的策略，盈亏比极低的策略，可以直接抛弃了。因为它们就像定时炸弹一样，很可能在我们不注意的时候爆发，然后损失就是巨大的，以至于常人无法承担。我们知道生活中犯错不要紧，但不要犯大到无法挽回的地步。就想我们经常开玩笑讲的，炒股是担着卖白粉的风险，赚着卖白菜的钱，这明显就是风险和收益不匹配！而且就事论事讲，卖白粉，只要被抓到一次，后果就很严重，这就像一些偏激的策略一样，也许跑一年都不会出事，但一旦出事，就是不可挽回的大事！

3 无法量化的策略

如果一个策略无法用量化表达出来，那就不用羡慕嫉妒恨了。可以把流口水的时间和精力用在那些可以兑现的策略上。当然也许随着时间的变化，计算机技术会提升，运算能力会提升，以前无法表达或无法有效计算的策略可以实现了，那我们当然可以很开心的去使用它！

4 未验证的策略

一个策略不经过回测和实盘验证，是不建议大资金用于实盘的。首先很多策略连回测那一关都过不了，如果直接实盘，亏钱就亏的太冤枉了，因为这本来是可以提前测试并避免的。

5 过度高频策略

我们的目的是赚钱，而不是为了高频交易，所以为了高频而高频，违背了赚钱的初衷，也就导致无法赚到钱！因此所有为了高频而高频的策略，直接丢掉毫不可惜！

策略筛选原则：

1 首先一眼剔除那些显而易见的问题策略，就像上面提到的那五种策略。

2 从逻辑／统计等鉴别，保留那些符合逻辑和统计规律的策略。

3 编写程序，回测那些策略，看看在历史上它们跑的盈利情况怎样？如果回测都是亏钱，那基本就可以判断此策略无效了！

4 实盘检验，至少要经过一轮牛熊的实盘检验才有说服力。否则策略可能在牛市跑的很好，在熊市亏很多钱！实际我们对投资者以及基金经理水平的判断，很多是基于这种牛熊长周期的投资回报的考量。当然现实情况是：没有时间让我们去做那么长时间的检验，也没有必要！不管是短期还是长期经历牛熊，得出的结论都是无法100%确定的，这也说明了一个道理：投资本身就是一种不确定的事情！就像假设我们开发了一个预测六合彩的程序，非常牛，可以把中奖几率提高到10年里中一次，假设我们经过10年检验，确实中了一次，那也无法100%确定下一个10年还会中一次！只是说有较大几率罢了，而且从未来不可预测来讲，这个较大几率是多大，也无法确定。当然事实是无法开发出这种彩票系统，相对来说，金融市场的预测相对还容易点，难度大约介于彩票预测和天气预测之间！

策略优化

其实大部分策略优化，都隐含了未来函数，因为我们只要拿一个时间段的数据进行优化，就已经对整段走势提前知道了，即使测试的程序是按照时间序列走的，但我们人选择最佳优化结果的过程，就是用未来数据比较的过程。只要是回测，就是用到了历史，只要我们人去比较回测结果，就用到了未来。回测优化隐含使用了未来函数，这就是为什么大多数人花费大量时间优化某一个指标，将其优化的很完美，但一到实战的时候不管用的原因！因为前面已经提到了，只要使用到未来函数的，实战中就根本不能用！针对这个问题，推荐两个解决方法：

1 根本不进行任何优化。直接使用指标的默认参数以及默认时间切片。但是纯粹的不进行任何优化比较是不可能的，就像有了macd指标，又有人发明了kdj指标，“默认参数情况下，kdj要比macd更灵敏”这句话本身就是比较，于是“比较后决定不使用macd而使用kdj”这本身就是一种优化，所以千千万万的指标本身就是优化过的，或多或少含有那么一丝丝的未来函数成分。这就能解释为什么有些指标看起来很准，而实际用起来却不准，因为我们认为它准的时候，这个主观判断本身就已经隐含使用未来函数了。那么怎么办呢？是不是所有技术指标都不能用了呢？况且从广义上来说，基本面分析也是一种技术分析，那么基本面也没法用了，所有人类可以使用的方法都不可以用了？不要着急，记得有这样一个小故事，两个人碰到一头熊，一个人赶紧低头系鞋带，另一个人笑话他：“你难道还想跑过熊吗？”那个人回答说：“我只要跑过你就行了”这就是答案！我们不需要找到一个纯粹的独一无二的完美的指标，我们只要找到一个比对手优秀的指标就行。也就是我们只要比其他投资者少优化就行了！换句话说，越原始的指标和越原始的参数，越不会忽悠人！记住，只是不忽悠人，至于是不是赚钱，还要看大家的本事了。

2 在A环境下进行优化，在B环境下测试优化结果，而不是在A优化，在A测试优化结果。A和B两个环境要有非相关性！比如一个是牛市，一个是熊市。一个是趋势市场，一个是盘整市场。这种方法可以有效解决未来优化问题。举个例子来说，比如我们根据macd金叉买死叉卖策略进行交易，使用2006-2007年牛市那一段时间进行测试优化，发现macd的参数周期越大，策略收益率越高，因为牛市持有周期越长收益越大。于是我们将macd参数周期设长，再去到2006-2007牛市进行测试，果然赚了很多钱！我们以为自己找到了投资的金钥匙，然而我们优化后，应该用2008年熊市周期来复核该策略的表现，结果会发现，熊市里长周期策略亏钱要比默认参数还要大，这就证明这个优化并不像原来想象的那么完美！但如果只用2006-2007那段时间检测，只会迷失在完美优化里，到死也不会知道自己的优化是有问题的。

# 仓位控制与止损

# Python金融分析

统计月涨幅周涨幅

本来想直接使用asfreq(freq='M')

但是在米筐的notebook中测试了一下，感觉有问题，所以准备用自己的方法来解决：

每个月的涨幅，=月末一天收盘价-上月末收盘价

也就是取出每个月的月末收盘价，然后按序列相减少，再归一，就行了 –

<class 'pandas.core.frame.DataFrame'>

000001.XSHG 399001.XSHE 000300.XSHG

2010-01-04 3243.7600 13533.537 3535.2290

2010-01-05 3282.1790 13517.375 3564.0380

2010-01-06 3254.2150 13505.184 3541.7270

2010-01-07 3192.7760 13235.480 3471.4560

2010-01-08 3195.9970 13267.436 3480.1300

2010-01-11 3212.7500 13161.090 3482.0520

2010-01-12 3273.9660 13381.250 3534.9160

2010-01-13 3172.6580 13016.557 3421.1440

2010-01-14 3215.5500 13204.189 3469.0510

2010-01-15 3224.1520 13264.372 3482.7380

2010-01-18 3237.0980 13361.518 3500.6840

2010-01-19 3246.8740 13350.671 3507.4820

2010-01-20 3151.8500 12916.154 3394.4340

2010-01-21 3158.8630 12917.154 3408.5740

2010-01-22 3128.5880 12595.935 3366.1970

月末收盘价，

如何判断一个日期是月末？

是月末，就留下来

先按年循环

对啊，可以直接拿月K线数据啊！看了一下，还是需要自己整合月K线了。

有空再看看dataframe里的asfreq是怎么回事。

不明白为什么是这个样子：

>>> dcdata.asfreq(freq='M')

date open close high low volume code

date

2014-10-31 NaN NaN NaN NaN NaN NaN NaN

2014-11-30 NaN NaN NaN NaN NaN NaN NaN

2014-12-31 NaN NaN NaN NaN NaN NaN NaN

2015-01-31 NaN NaN NaN NaN NaN NaN NaN

# pandas进阶

pandas基本使用习惯

## 常用统计：

### 统计每个月的涨跌幅

### 统计周一到周五上涨几率和总涨跌幅

统计均线的成功率

选股与选时

君山股道100招