



发明者量化 交易入门

从基础到实战

FROM BASIC TO PRACTICAL

- 树立对量化交易的正确认识，浅析量化与交易之间的关系
- 详解技术指标、量化择时、资金管理、策略调参等技术及交易实例
- 详解量化交易基础知识，以及Python、JavaScript、C++、可视化语言、麦语言等策略实例



目录

第一章 量化交易基础.....	1
1.1 什么是量化交易.....	1
1.2 为什么选择量化交易.....	5
1.3 量化交易需要准备哪些.....	8
1.4 一个完整的策略有哪些要素.....	12
第二章 量化工具介绍.....	17
2.1 量化工具整体介绍.....	17
2.2 如何配置发明者量化交易系统.....	21
2.3 常用的 API 讲解.....	27
2.4 如何在发明者量化系统上编写策略.....	32
第三章 简单编程语言实现交易策略.....	36
3.1 量化交易编程语言横向评测.....	36
3.2 麦语言快速入门.....	42
3.3 如何用麦语言实现策略.....	49
3.4 可视化编程快速入门.....	56
3.5 如何用可视化语言实现策略.....	66
第四章 主流编程语言实现交易策略.....	75
4.1 JavaScript 语言快速入门.....	75
4.2 如何用 JavaScript 语言实现策略交易.....	86
4.3 Python 语言快速入门.....	96
4.4 如何使用 Python 语言实现策略交易.....	109
第五章 策略回测、调试及改进.....	117
5.1 回测的意义和陷阱.....	117
5.2 如何做量化交易回测.....	124
5.3 如何读懂策略回测绩效报告.....	131
5.4 为什么需要样本外测试.....	138
5.5 交易策略优化及最佳化.....	143
5.6 建立概率思维，提升你的交易格局.....	148

第一章 量化交易基础

1.1 什么是量化交易

摘要

量化交易作为科学与机器的结合的产物，正在改变着现代金融市场的格局。如今已经有不少投资者将目光转向了这一领域。如何最大限度地降低风险并尽可能取得理想收益？也是本系列课程的目的，作为开宗明义的第一篇，先就“什么是量化交易”进行简要阐述。

概述

很多小伙伴一听到“量化交易”就会觉得高端大气、一夜暴富。人工智能时代，伴随着深度学习、大数据、云计算等先进技术等兴起，更是赋予它神秘的色彩。似乎只要运用量化交易，就能构建出“完美无缺”的交易策略。

其实，在一定程度上，量化交易已经被神话了。撇开交易，“量化”其实就是借助计算机，并利用统计学、数学等方法，通过科学的投资体系，从中找到一套正期望的交易信号系统。这个信号系统会告诉我们应该在什么时间以什么价格进行买卖。

量化交易的发展

追本溯源，最早采用量化方法来分析数据变化，并从中发现市场价格涨跌规律的人，既不是股票的发源地荷兰人，也不是将现代金融发扬光大的英国人，更不是建国就与金融共生的美国人，而是一位法国人。

早在 18 世纪，法国股票经纪人助理 Jules Regnault 就提出了股票价格变化的现代理论，随后出版了《概率计算和股票交易哲学》一书，并在书中详细阐述了自己发现的市场涨跌规律（正态分布）：“价格的偏差与时间的平方根成正比”，最后以理性量化的投资决策获取交易的成功。

现如今，在互联网+大数据+云计算+人工智能的时代大背景下，量化交易也得到

了快速发展。曾经的全球金融腹地伦敦金丝雀码头，早已变成了 IT 公司集散地。世界顶尖投行，也都在培养自己的量化团队，试图跻身到“得模型者得天下”的金融大战之中，这些开发交易模型的 IT 团队也被称为 Quant Team。从规模上看，起步较早的美国已经拥有一大批实力雄厚的量化对冲基金。

反观国内，无论是硬件设备还是投研实力，都还在起步阶段。但已经有越来越多的机构和专业投资者意识到量化交易的好处，并参与到这一领域，特别是在监管逐步趋严、市场有效性逐步提升的过程中，量化交易更具有广阔的成长空间。

量化交易的特点

科学验证：试想，当你有一个交易系统后，如果用模拟盘去测试它的有效性，可能会付出巨大的时间成本，如果直接拿实盘测试，可能损失的是真金白银。但是可以利用量化交易中的回测功能，通过大量的历史数据，以科学的方式去检验交易系统。什么有效，什么没效，让数据去说话，而非人云亦云。

客观准确：在交易中，我们真正的敌人是自己，心态管理说起来容易，做起来难。贪婪、恐惧、侥幸等人性的弱点，在交易市场中会数倍放大，量化交易则可以帮助我们克服这些弱点，在交易中做出更好的决策。

及时高效：主观交易，人的反映速度是无法快过电脑的，并且人的体力和精力也无法 24 小时运行，在机会稍纵即逝的交易市场，量化交易完全可以代替主观交易，寻找交易机会，及时快速地跟踪市场变化。

风险控制：量化交易不仅可以从历史数据中挖掘在未来可能重复的历史规律，这些历史规律都是较大概率取胜的策略。还可以构建多种不同的投资组合，降低系统性风险，平滑资金曲线。

量化交易都有哪些经典的交易策略？

开盘突破策略

开盘半小时往往能决定一天的走势，该策略以开盘之后的半小时内，价格是阳线还是阴线，作为判断日内趋势走向的标准。如果是阳线就开仓买入，如果是阴线就开仓卖出，收盘前几分钟内平掉仓位。这是一个非常简单的交易策略。

唐奇安通道策略



图 1-1 唐奇安通道策略图解

唐奇安通道策略可以称得上是日内交易的鼻祖，其规则是：如果当前价格高于前 N 根 K 线最高价的最高价时就买入，如果当前价格低于前 N 根 K 线最低价的最低价时就卖出。著名的海龟交易法则用的就是修正版的唐奇安通道策略。

跨期套利策略

跨期套利是套利交易中最普遍的一种，根据同一个交易品种，不同交割月份合约的价格为基础，如果两者价格出现了较大的价差幅度，就可以同时买卖不同时期的期货合约，进行跨期套利。假设主力合约与次主力合约的价差长期维持在 $-50 \sim 50$ 左右。如果某一天价差达到 70，我们预计价差会在未来某段时间回归到 50。那么就可以卖出主力合约，同时买入次主力合约，来做空这个价差。反之亦然。

总结

以上，我们从量化交易的定义、发展、特点以及经典的交易策略等方面，为大家简单介绍了关于量化交易的相关概念。

了解量化交易是成为宽客（Quant）道路上的一块重要的敲门砖。最后，祝大家在熊市里充实自己，早日实现认知变现！记住，你与财富自由只差一个牛市！

下节预告

量化交易与传统交易到底有哪些区别？在实战交易中，究竟选择传统交易还是量化交易？下节我们将带着这两个疑问，进一步了解量化交易。

课后习题

- 1、用一句话简述什么是量化交易？
- 2、量化交易有哪些特点？

1.2 为什么选择量化交易

摘要

很多人在探讨量化交易时会以复杂的策略编程为切入点，无意间为量化交易披上了一层神秘面纱。本节我们将尝试以通俗易懂的语言，为量化交易做一个简单“素描”，揭开他的神秘面纱，相信即便是毫无基础的小白也能轻松理解。

量化交易与主观交易的区别

主观交易更重视人为的分析和盘感，即使出现了买卖信号，也会选择性的下单交易，宁可错过行情，也不愿做错。人的感觉是复杂多变的且不可靠的，大多数交易者一旦发生连续亏损，往往就转而用另一种方法。随机性较强，容易被盈亏困扰，导致难以稳定盈利。

量化交易通过对交易的理解，制定一致性的买卖策略。在交易中，对所有的走势都一视同仁，开仓平仓全部系统化处理，宁可做错，也不愿错过。它还具有完整的评价体系，通过历史数据回测，确定策略更适合哪一类的行情和品种，并搭配多种策略和品种实现盈利。

简而言之，主观交易是量化交易的基础，量化交易是主观交易的提炼。主观交易更像是练武，最后能成功与否，天赋占大多数，有十年不悟的，也有一朝悟道的。量化交易更像是健身，只要刻苦努力，就算没有天赋，也能练出一身肌肉。

量化交易比主观交易更好？

一个成功的主观交易者，从某种意义上说，也是一个量化交易者。因为一个成功的主观交易者，必然有一套自己的规则和方法，也就是交易系统。成功的主观交易必须建立在交易纪律和交易规则之上，而交易规则的执行部分，实际上就是主观交易中的量化部分。

相反，一个成功的量化交易者，也一定是一个优秀的主观交易者，因为量化交易策略的开发，其实就是一个人交易理念的结晶。如果一个对市场的认知和理解，从一开始就是错误的，那么开发出来的交易策略，长期以来也是难以获利的。

所以，从盈利的角度讲，决定一个交易者最终能否成功，关键因素是交易理念，而不在于是主观交易还是量化交易。量化交易表面上看似高大上，其盈利的实质与主观交易没有本质的区别，它们就像是一件事物的两面性对立又统一。

但是不可否认，从交易工具上来说，量化交易确实有很多优势。

复盘更快：想要检验一个交易策略，就需要计算大量的历史数据，量化交易几分钟之内就能计算出结果。这个速度要比主观交易快许多倍。

更加科：评价一个策略是否优秀，依靠的是数据（比如：夏普比率、最大回撤率、年化收益），而不是自圆其说的神棍。

更多机会：全球有几千个交易品种，主观交易不可能同时盯盘，但是量化交易可以全市场实时盯盘，不错过任何交易机会，增加盈利能力。

量化交易一定能赚钱吗？

当然能，但长期坚持下来确是一件很难的事。赚钱与否并不取决于量化交易本身，它只是一个工具，量化交易只是把交易思想用程序化、规则化、数量化实现出来，程序代替的只是执行力。难的是长期稳定的赚钱，因为市场是博弈的、动态变化的，交易思路也要跟着市场转变。

量化交易的风险

量化交易也有风险，为什么呢？因为量化交易是在历史数据中去挖掘规律，形成交易策略。但是金融市场是一个生态体系，其规律和人性是一个相互作用的动态过程，归根到底还是人的市场。市场的规律会被人性所影响，而人性中间的贪婪、恐惧都会随着市场的变化而变化，市场上很少有一成不变的规律，再厉害的交易策略也很难应对这种突如其来的规律变化。

总结

通过上面的解释，我们可以看到，量化交易不是一种独特的交易方法，它只是一种交易工具，帮助我们分析交易逻辑，完善交易策略。无论你是价值派、技术派，无论做的是股票、债券、商品还是期权，其实都可以量化。相比于靠个人经验作决策的交易者，量化交易者手中的武器就是市场证据和理性。

下节预告

量化仅仅是一种交易方式，策略只是交易思想的载体，程序执行的是每个交易过程。下节将带你了解量化交易完整的生命周期，它将包括：策略构思、建立模型、回测调优、仿真交易、实盘交易、策略监控等。

课后习题

- 1、量化交易与主观交易最重要的区别是什么？
- 2、与主观交易相比量化交易有哪些优势？

1.3 量化交易需要准备哪些

摘要

一个完整的量化交易生命周期，不仅仅只是交易策略本身。它至少由六个环节构成，包括：策略构思、建立模型、回测调优、仿真交易、实盘交易、策略监控等。

策略构思

首先，做量化交易必须先回到交易市场，要在市场中多观察价格，理解市场波动的规律，并尝试推断每一个交易逻辑，最后总结出交易策略。这里并没有捷径，你可能需要阅读经典的投资书籍，或者不断的坚持做交易，在失败中总结经验。

对于量化交易初学者来说，刚开始开发交易策略最好的方式就是模仿。直接利用现成的技术分析指标构建策略逻辑，写入买卖规则，这样就可以得到一个简单的策略了。假如你的交易策略是这样的：如果价格高于最近 10 天的平均价格就买入，如果价格低于最近 10 天的平均价格就卖出。那么它的架构是这样的（如下图）：



图 1-2 交易策略举例

当然，随着策略经验的积累，形成自己的交易方式后，逻辑的选择会越来越多样化，再进阶到更加系统的量化交易。如果能做一个有量化思维的交易者，无论是在股票还是期货市场上，这都是一件值得庆幸的事，因为这样的人，不管在哪个交易市场都有持续稳定的获利能力。

建立模型

其次，你需要掌握一个量化交易工具，用来编写交易策略，实现你的交易想法。市面上的常用软件都可以。但是如果你想成为一名高端的量化交易者，就需要学

会一门计算机语言，这里推荐大家使用 Python，因为它是科学计算的权威语言，并且提供各种开源的分析包，文件处理，网络，数据库等。

如果你的编程能力较弱，相信这也是大多数初学者的软肋，推荐使用相对简单的可视化编程语言或麦语言，它可以提高学习量化交易的兴趣，并使你专注于策略，高效的完成策略开发。如下图：利用麦语言，开发一个上述的交易策略，双击图片可以看到策略代码中的详细注释。

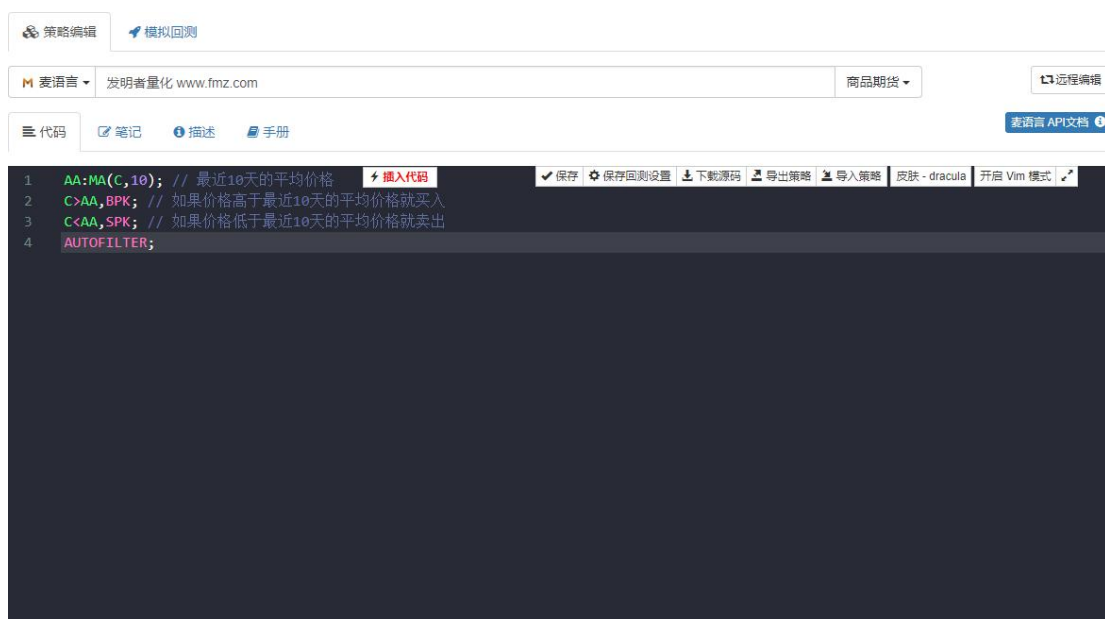


图 1-3 交易策略开发页面

上图的策略代码，是利用发明者量化工具的麦语言演示，其集成很多可直接使用的功能模块，并且支持回测和实盘交易功能，是一个不错的快速入门方式。

回测调优

然后，当编写完策略模型后，下一步就是对策略进行回测，以及参数的筛选和优化。可以利用不同的参数对策略进行回测，观察该策略的夏普比率、最大回撤、年化收益等。通过对策略的不断调试和修改，最终得到一个完善的量化交易策略。

比如，我们把 2017 年的历史数据作为样本内数据，2018 年的历史数据作为样本外数据。先用 2017 年数据优化出几组表现好的参数，再用这些参数对 2018 年的

数据回测。一般情况下，样本外的回测结果没有样本内的回测结果好，但是如果样本外与样本内的结果大相径庭，那么这个策略几乎是无效的，就要观察分析，判断策略失效的原因。

假设，发现策略失效由于是样本外数据，某几次极端行情导致的大幅亏损，那么就可以增加一个固定止损条件来规避这种风险；如果发现策略失效是由于交易次数过多，那我们可以将交易逻辑稍微收紧，降低交易频率。

需要注意的是，如果一开始交易逻辑本身就是错误的，再怎么修改也很难得到一个赚钱的策略，这个时候就需要重新审视自己的策略思路了。另外，在参数优化中，可用的参数组越多越好，说明策略的适用性广泛。在回测时，交易次数太少的策略其回测结果可能是幸存者偏差。如果回测的结果是一个超级赚钱的资金曲线，很多情况下是你的逻辑写错了。

仿真交易

接着，当你拿到一个交易逻辑正确，样本内外都赚钱的策略时，先不要急着在真实账户上交易。尤其对于初学者来说，一定要先用仿真账户运行至少 3 个月，如果是中低频隔夜策略，则需要更长的仿真交易时间。

在未来一段完全未知的仿真行情中，观察策略在仿真交易中表现，仔细核对回测信号与仿真交易信号是否吻合，下单时的价格与成交时的价格是否有偏差，如果表现与预期相符合，那么说明策略有效。

实盘交易

终于，通过一段漫长的时间检验策略之后，就可以将策略放入实战中进行交易了。当然在量化交易的过程中我们也要保持警惕，防范极端行情。在实盘中，策略的期望一般都要打折扣的，达到预期的 50%就是合格。

策略监控

最后，需要提醒大家的是，随着交易进行，我们也要观察策略的有效性，当发现策略出现超出预期的亏损时，就要重新评价这个策略。因为市场特征是会变化的，我们当下形成的策略主要是针对过去的市场特征。一旦市场特征发生变化就要对策略模型及时调整，或者暂时中止这个策略。

总结

本篇我们阐述了量化交易的完整流程。总而言之，如果你是有市场经验的投资者，阻拦你的将会是计算机语言基础，可以先从可视化语言或麦语言开始，通过在这一平台上锻炼自己，构建策略，然后逐步转向 Python 高端量化交易。

如果你是编程能力较强的理工科学生或者 IT 从业者，阻拦你的将会是市场投资经验，也千万不要小看这一点，作为合格的量化投资者，两类知识缺一不可。

下节预告

整个量化交易生命周期中，最核心的还是交易策略。下节我们将从交易策略架构的角度，详细阐述一个完整的交易策略要素有哪些？这将帮助你更加全面的搭建你的交易策略，将量化交易提升到一个新的水平！

课后习题

- 1、试着用麦语言编写本节中的交易策略。
- 2、量化交易回测中最重要的绩效指标是什么？

1.4 一个完整的策略有哪些要素

摘要

一个完整的策略，其实就是交易者给自己定的各种规则，它包括了交易的各个方面，并且不给交易者留下一点主观想象的余地，每个买卖决定，策略都会给出答案。它至少包含策略选择、品种选择、资金管理、下单交易、极端行情应对、交易心态等等。

策略选择

从对冲基金的角度讲，主流的交易策略可以分为趋势交易、配对交易、一篮子交易、事件驱动、高频交易、期权策略等等，如下图。当然，策略的分类方式不是固定的。

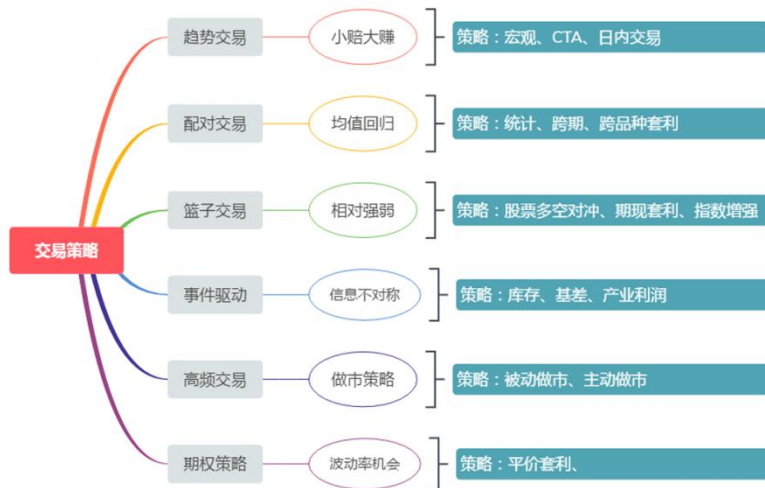


图 1-4 交易策略分类

对于刚入门的量化交易初学者来说，先不必管这么多名词概念，一步一步从最简单的开始。如果只推荐一种量化交易策略入门，那就是趋势交易，其原因是简单有效。相信即使你没有系统的学习金融知识，一样可以做好交易。并且这种策略由来已久，在早期公开的交易策略中，至今仍在多个市场中有效，因为人性是很难改变的。

买卖什么

做过交易的人应该知道，每个品种都有各自的性格。有些品种性格很“火爆”，流动性好、波动大、波动率高；有些品种性格很“温顺”，常年都在一定区间内震荡，波动率低。

所以，在选择交易品种的时候，一定要有波动率这个概念，波动率高的品种，往往很容易走出一波不错的趋势行情。对于商品期货来说，如果是趋势跟踪策略，尽量选择工业品，从品种属性上来讲，工业品往往比农产品波动率要大。

不同的策略适应不同的行情，选择好交易品种，对期货交易这项大工程来说是一个非常关键的开始。从绝对意义上来说，没有绝对好的品种也没有绝对不好的品种。根据投资风格的不同，以及风险承受力的不同，需要针对自己的标准进行相应的调整。

买卖多少

交易赔钱容易赚钱难，当账户资金亏损 50%，挽回损失则需要 100%的盈利。就算你可以赚很多次 100%，但只需要赔一次 100%就全部亏光。所以成熟的交易策略应该包含资金管理。

为了便于大家理解，这里还使用上节的均线策略讲解。事实上，很多以传统技术指标构建的交易策略，最大回撤率一般超过 50%甚至更多。但一个风险很大的策略完全不能用吗？

显然不是，最大回撤率完全可以通过资金管理控制。如果把仓位降低一半，那么整体风险也会降低一半，最大回撤率变成 30%，如果把仓位再降低一半，最大回撤率就变成 15%，最后我们得到的是一个最大回撤率控制在 15%左右的策略。这就是一个简单粗暴的资金管理方法。很多人知道不能满仓操作，但是却不知道为什么不能满仓操作，答案就在这里。

何时买卖

一个好的买点，是成功的一半，它能够让你迅速摆脱成本区。但是永远不会有什人能够告诉你在这个点开始是对的，在那个点开始是错的。开仓不是交易的核心，交易的核心是开仓之后，如何尽可能优化处理持仓。

不管是短线策略，还是长线策略，比的不是看谁持仓时间长，而是风险收益比。换言之，影响策略绩效的最终结果是如何出场，何时兑现利润。出场方法又可以分为两种：止损出场和止盈出场。而这两个部分都是任何交易系统所必需的，也是关乎于交易策略成败的重要分水岭。

如何买卖

1、委托下单类型和方式：

委托下单的类型和方式有许多种，比如：委托时用排队限价单、对手价、最新价、超价、涨停价、跌停价、买一价、买二价、卖一价、卖二价，或者先用排队价，再用超价，分批报单，或者把大单拆成一个个小单，或者干脆直接把单子全部报出去。

2、撤单

如果下单没有成交，是继续等待还是撤单，撤单条件依据是时间，比如 10 秒内还没有成交，价格已经远离下单时价格 10 跳，是继续等待、撤单还是追单。

3、追单

当下单没成交的时候，是否追单。如果追单，是按最新价去追，还是对手价，还是涨跌停价，如果追单仍未成交是否继续追单。

4、涨跌停价

当下单信号出现时，刚好是涨跌停价格时怎么办。是否在涨跌停价挂单排队成交，如果没有成交时怎么办。

5、集合竞价

开盘集合竞价要不要参与，怎么参与。

6、夜盘

有些商品期货品种夜盘是从 21:00 至次日 02:30，这段时间做不做，人工做还是让电脑来做。

7、重大节日

重大节日的超长假期之前，仓位需不需要保留。如果保留的话如何控制风险。

极端行情

1、短时间价格大幅波动

价格瞬间涨跌停、连续涨跌停、乌龙指事件、黑天鹅行情价格踩踏事件等等，这些情况如何处理。

2、流动性风险

如果一档对手盘没有你要的下单量，但你又需要及时成交，特别是非主力合约流动性很差，自己下的单子很容易对市场造成冲击，滑点很大时，如何应对。

3、品种规则变化

商品期货品种加入夜盘，保证金比例上调，手续费上调，特别是短线策略，对于

这些变化会非常敏感。

4、交易环境风险

比如：突然断电、断网、电脑故障、软件宕机、银期转账暂停、自然灾害等，出现时如何应对。

以上情况，出现的概率很小，或者说几乎不可能。但如果事情可能发生，就一定能发生。做好这些假设，并加以防范是非常有必要的。

心理建设

交易中常见的三种主要心理情绪为贪婪、恐惧和侥幸。投资者需要一个强大的交易心理体系在不同阶段对上述三种情绪加以控制甚至利用。

交易之前要有一个对未来的整体预期，包括市场预期和品种心理预期。市场预期指对市场所处的位置和未来的方向有一个较为明确的目标，品种预期是指该品种在当前位置下的交易机会和风险状况。如果没有以上心理基础，则一切都无从谈起。

实盘交易的全过程就是不断分析、修正和执行的过程，其间交易的时间不多，更多的是跟踪和忍耐。这是一个综合考察心态、考验人性的过程，交易者的各种习性在交易过程中都会被展现无遗和放大。只有不断学习和总结经验教训，不断历练，才能克服人性的思维共性和心理弱点。

总结

综上所述，所谓的交易策略，其实就是这样，有它完美的一面，也有它残缺的时候，我们衡量一个交易策略是否合理，并不能只看他完美的一面，也不能只看他残缺的一面，更应该综合的分析策略的完整性。

最后根据策略的特性，结合自身的性格和资金情况一起去衡量该策略是否适合自己，如果适合自己的话，要充分评估自己坚持下去的可能性有多大，最坏的结果要事先规划好，如果最惨的一面你都想好了，那么执行下去的可能性就相对较大。

记住，在交易中，信心源自于你发自内心的认可，信心源自于正确的交易理念！

下节预告

本篇是第一章的最后一篇，下一章节我们将围绕量化交易工具，为大家展开进一步讲解，包括：量化工具整体介绍、如何配置量化交易系统、常见的 API 讲解、如何在量化系统上编写策略。

课后习题

- 1、趋势型交易策略应该选择高波动率的品种还是低波动率的品种？
- 2、交易委托单的几种类型？

第二章 量化工具介绍

2.1 量化工具整体介绍

摘要

在前一章中，我们学习了关于量化交易的相关概念，对量化交易有了基本的了解。那么市面上都有哪些可以量化交易的工具呢？我们又该如何根据自己的需求选择呢？

开源软件和商业软件

国内量化交易工具大体上可以分为开源软件和商业软件两大类。所谓的开源软件可以理解为软件的源代码是开放的，可以直接下载源代码使用；商业软件一般泛指由商业公司维护运营的闭源软件，通常都是付费的。

开源量化软件

首先开源软件有很强大的灵活性，是完全免费的，用户基本上可以运用这个软件去实现任何功能，无论是中低频交易策略、套利策略还是期权策略，都可以通过定制化模块来实现，由于用户掌控这软件的源代码，可以了解软件里面的每一个角落，所以更为可靠安全。

尽管开源软件有很多优点，但它对量化交易初学者不是很友好，你需要系统性学习一门标准的编程语言，比如 Python、Java 或者 C++。从入门到放弃，其难度可想而知，有时候调 bug 能调到你怀疑人生。而且不像商业软件，有专门的技术客服即时答疑解惑。这时候不但没有成就感，还会打消你继续学习下去的动力。

所以，从学习的角度讲，推荐量化交易初学者一步一步，从最简单的商业软件开始，尽管它是付费的，但如果策略是盈利的，软件费用仅仅是利润的零头，再者，商业软件一般是一个团队在维护，其成熟度肯定比开源软件强很多。

商业量化软件

国内可以做量化交易的商业软件多达几十种，比如：既专业又全面产品又多的 Interactive Broker、能处理海量并发数据，适合高频交易的 APAMA、支持 C++ 的接口，执行效率不错的 SPT 盛立、侧重在交易执行和风控的掘金量化以及面向个人交易者的 MC、TB、MQ。下图中我们把国内主流的量化平台进行了综合评测，

对于量化工具的难易程度也做了一定的分类，读者可以根据自己的实际情况选择。













 <p>盈透证券 InteractiveBrokers</p>	<p>又大又全又专业，适合专家 难易程度：★★★★★</p>	 <p>掘金量化 MYQUANT.CN</p>	<p>证券高端量化，支持多种开发语言 难易程度：★★★★</p>
 <p>APAMA Community Edition BY SOFTWARE AG</p>	<p>善于处理海量高并发数据 难易程度：★★★★★</p>	 <p>Tinysoft 天软科技</p>	<p>天软特有的TSL语言开发策略模型 难易程度：★★★★</p>
 <p>赢立金融软件 WENSHU FINANCIAL SOFTWARE</p>	<p>国内高端量化交易平台，C++语言 难易程度：★★★★★</p>	 <p>MULTI CHARTS Raising The Trading Standard</p>	<p>国外TradeStation软件的国内版 难易程度：★★★★</p>
 <p>FMZ QUANT 发 明 者 量 化</p>	<p>国内中高端量化交易，支持多语言 难易程度：★★★</p>	 <p>TRADE BLAZER® 开拓者 国内高级搜索</p>	<p>国内中低端程序化软件，TBL语言 难易程度：★★★★</p>
 <p>X-Quant —专业团队级量化投资研发交易平台</p>	<p>大商所出品，Java语言开发 难易程度：★★★★</p>	 <p>webstock 文華財經</p>	<p>国内中低端程序化软件，麦语言 难易程度：★</p>
 <p>RS 北京风软技术有限公司</p>	<p>国内顶尖的期权做市商软件 难易程度：★★★★★</p>	 <p>金之塔 程序化软件提供商</p>	<p>国内中低端程序化软件，VB语言 难易程度：★</p>

图 2-1 国内主流的量化平台综合评测

以上虽然是商业软件，但其所用的也是标准编程语言或脚本语言，与其这样还不如直接使用既免费又安全的开源软件。新手这里推荐直接使用 FMZ 发明者量化平台，网站是 www.fmz.com。作为量化交易学习的敲门砖。

认识发明者量化交易工具

发明者量化工具对小白友好，即便你是零基础，也可以根据里面的工具体验量化的魅力。该工具是面向高频交易设计，在性能和安全上有严苛的要求。支持高频策略、套利策略、趋势策略。并且它集成了策略开发、测试、优化、模拟、实盘交易的完整流程。另外它既支持简单又好用的麦语言，也支持 Python、C++ 等高级量化交易语言，等于一次学习无缝切换。并且只有实盘交易才收费 0.125 元/小时，降低你学习摸索阶段软件成本，同时可以免费模拟做仿真交易。

迈出量化第一步：使用量化工具

量化工具使用非常简单，只需要进入网站点开即可设计自己的量化策略。大家可以登陆发明者量化工具的官方网站，注册并登陆，点击控制中心即可使用（如下图），类似于目前比较火的抖音，注册登录后就可以发布自己的小视频，而量化工具登录后是设计自己的量化交易策略。



图 2-2 FMZ 量化交易平台主页面

量化工具编程会有一个集中的功能区，功能区主要包括（如下图）左上角的控制中心是该量化工具的核心功能，点击之后，就可以编写交易策略和策略回测、设置交易品种的交易所、创建管理策略机器人的托管者、创建具体的量化交易机器人。至于里面的功能具体用法，我们会在后续的文章中详细介绍，当前我们只做初步。



图 2-3 FMZ 量化交易平台登陆后管理页面

首次接触量化的朋友，不用为自己不懂代码和编程而望而却步。为了降低用户的使用门槛，官方社区出品了许多视频教程，帮助量化交易初学者快速入门；同时，在策略广场中聚合了上千个官方和第三方免费开放的交易策略，方便大家复制学习。

另外，在策略编辑界面，还配置了经典的策略样例，点击就能直接使用策略代码，轻松体验整个量化交易的核心流程，即使是小白用户也能立刻学会，并跟着做起来！

在真金白银实盘之前，仿真交易也是必不可少的环节，该工具的仿真交易符合交易所规则，并且完全免费，仿真包括的时间、价格、订单量等与真实行情实时撮合，高度吻合实盘交易。大大提高策略验证效率。

总结

无论是开源软件还是商业软件，并没有优劣之分，也没有完美的量化交易工具，每个工具都有自己的侧重点，最重要的根据自己的需求选择适合自己的工具。商业软件需要付费，它在服务等方面更好一些，可能更适合刚入这个行业的初学者。如果你在这个行业做了很长时间，积累了很多经验，或者需要实现更复杂的交易策略，开源软件是个更好的选择。

下节预告

有了工具怎么用？和我们买了一台新手机，第一次开机需要做简单开机设置一样，量化工具也需要做基础的设置配置，下节我们将带你手把手配置发明者量化交易工具。开启量化交易第一扇门，包括：添加交易所、添加托管者、创建交易策略、创建量化机器人等等。完成了基础配置，就可以正式编写第一条属于自己的量化策略了。

课后习题

- 1、量化交易工具分为哪两大类？
- 2、常用的量化编程语言都有哪些？

2.2 如何配置发明者量化交易系统

摘要

对于量化交易策略开发来说，首先要做的是交易工具的配置，何为配置？其实就是设置。本节我们将带你设置交易所、创建交易策略以及创建量化交易机器人，这些都是量化交易的必要前提。

配置方面分为入门学习模拟仿真交易配和实盘交易配置，这类我们主要以国内商品期货为主，其他类别的量化投资因为国内具体情况而不做推荐和介绍，但操作流程是一样的，只是配置流程不同而已。

添加交易所

添加交易所是整个配置环节的第一步，具体流程请看下图介绍。在这个步骤中，我们需要强调一点，添加交易所并不难，对于不清楚自己所属交易所的同学。建议先模拟学习。



图 2-4 FMZ 量化交易平台注册添加交易所步骤

商品期货交易所（实盘）配置

实盘的量化交易我们主要以国内期货交易品种为主，目前发明者量化的主要服务对象也是国内期货交易所，对于做外汇的朋友，发明者量化可以作为一个学习的平台，因为外汇量化交易在 mt5 等平台已有出现，只是更为专业。

实盘配置需要注意的问题如下：由于发明者量化工具支持多种交易市场，所以配置商品期货，一定要先在第①步中选择“传统期货”；在第②步中，需要填写的

是你所开户的期货公司，给你的期货账户和密码。

发明者量化工具，采用 CTP 协议，支持国内所有的期货公司，配置实盘的时候，不会出现链接不成功，除非是账号和密码错了，因此初学者要注意账号和密码要核对清楚。

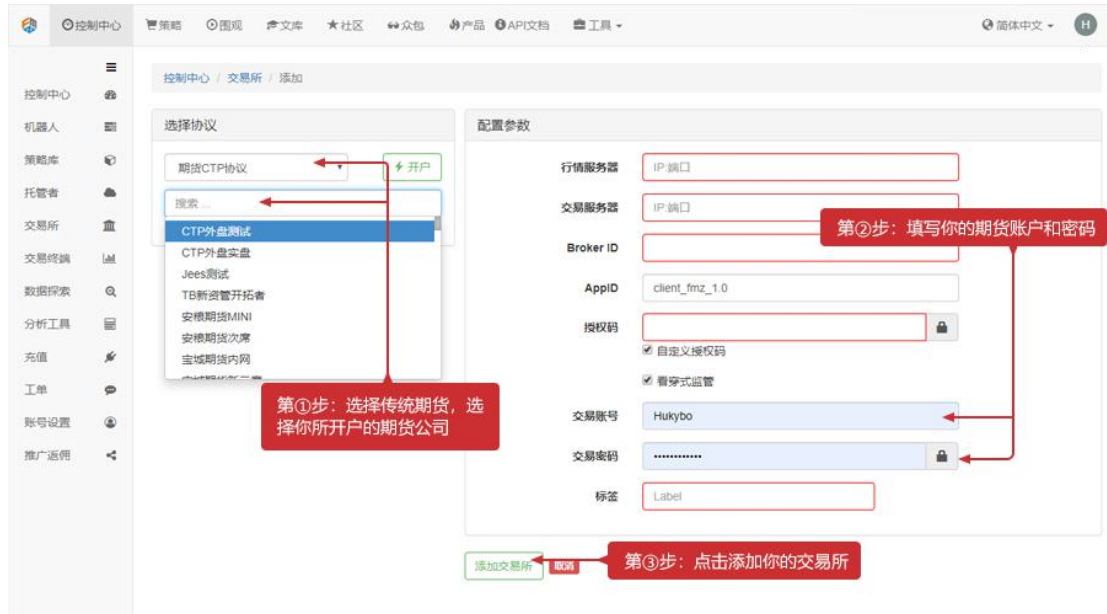


图 2-5 FMZ 量化交易平台添加期货交易所

商品期货交易所（仿真）配置

对于刚接触商品期货的朋友，我建议还是先仿真交易一段时间，因为在开发量化交易策略过程中，需要不断的检测、调试、优化。就像开车一样，刚开始肯定是在驾校摸索几个月，考核出证后，再上路。

这里推荐用 SimNow 仿真交易，SimNow 是上期技术专为投资者打造的金融模拟仿真交易平台，该产品仿真各家交易所的交易及结算规则研发，目前已经支持国内各家期货交易所的商品期货业务，具体流程请看下图介绍。



图 2-6 FMZ 量化交易平台登陆后管理页面

策略编写

策略库就是存放代码的地方，它相当于我们的量化交易策略仓库。主要分为两个功能：策略编写和模拟回测。策略编写区是我们日后开发策略主要的工作区域（如下图），很多初学者往往被各种代码给拦住，觉得非常的困难，其实只要稍微用点心，就能学会这些代码，千万不要有心理负担。模拟回测区则可以在开发策略过程中调试策略，以及策略开发完毕后检验策略，这个我们将放到之后的章节为大家详细讲解。



图 2-7 创建策略步骤

创建量化交易机器人

量化交易机器人就是交易策略的执行者，当策略创建完毕之后，创建一个机器人，它就可以自动帮你执行策略代码中的每一个交易逻辑，以及开仓、平仓、撤单等买卖操作。创建量化交易机器人具体步骤如下：首先第①步：在控制中心页面，点击“机器人”，点击“创建机器人”第②步：给机器人自定义一个名字。第③步：点击“+”号，添加交易平台。第④步：点击“创建机器人”



图 2-8 创建机器人步骤

总结

以上流程中，除了第一步选择实盘和模拟各有不同之外，后面的步骤策略编写和创建交易机器人都是统一的步骤。整个量化工具配置完毕，交易机器人也已经运行起来了，并且会根据策略的具体条件进行买卖操作。配置量化交易总共就三步：添加交易所，填上自己的期货账户密码；编写一个交易策略；创建一个实盘量化交易机器人。是不是很简单呢？

下节预告

尽管只需要简单的三步就能实现量化交易，但细心的你可能会发现，添加交易所和创建量化交易机器人尚且容易。但是，如果是要实现一个可行的交易策略，就不是那么容易了。下节我们将带大家学习量化交易中常用的 API，为编写一个可行的交易策略做准备。因为无论是哪种量化交易工具，一定离不开 API 接口，它是实现量化交易策略的重要功能。

课后习题

- 1、试着添加一个交易所。
- 2、试着编写本节中的交易策略。

2.3 常用的 API 讲解

摘要

提到编程,就一定离不开 API,对于很多非 IT 人士而言,API 到底是什么? API ≈ 听不懂。本节我们将用大白话科普,到底什么是 API,以及介绍发明者量化工具中常用的 API。

什么是 API?

如果你在网上搜索,会得到如下结果:API(Application Programming Interface, 应用程序编程接口)是一些预先定义的函数,目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力,而又无需访问源码,或理解内部工作机制的细节。那么再通俗一点来说,API 究竟是什么呢?

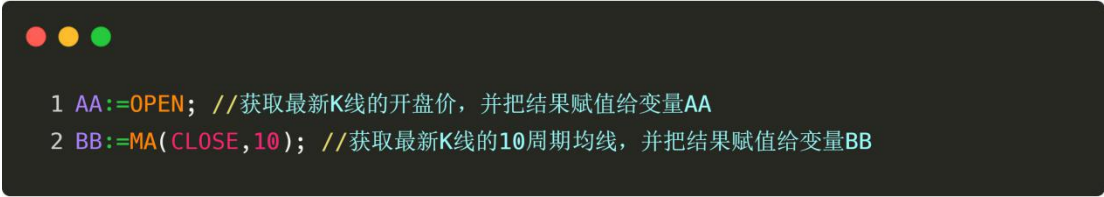
其实日常生活中,我们有很多类似 API 的场景,比如:你去一家饭店吃饭,只需要看着菜单点餐就行了,不需要知道具体是怎么做出来的。而菜单中的菜名就是具体的 API,菜单就是 API 文档。

量化交易中的 API 是什么?

假如你需要获取当前品种今天的开盘价是多少,不需要知道具体是怎么获取到的。你只需要在代码编辑器中写入“OPEN”,直接使用就行了,“OPEN”就是麦语言中开盘价的 API。

常用的麦语言 API

在讲解麦语言 API 之前,我们先来看下常用的代码结构是怎样的,以及它都有哪些功能组成,这将帮助你更好的理解 API,请看下图示例:



```
1 AA:=OPEN; //获取最新K线的开盘价,并把结果赋值给变量AA
2 BB:=MA(CLOSE,10); //获取最新K线的10周期均线,并把结果赋值给变量BB
```

图 2-9 麦语言例子

如上图所示的代码:

紫色的 AA 是变量,变量就是可以变的量,就跟我们初中学的代数是一样的。如果把开盘价赋值给 AA,那么 AA 就是开盘价;如果把最高价赋值给 AA,那么 AA 就是最高价。当然 AA 只是自定义的名字,你也可以定义为 BB。

绿色的“:=”是赋值的意思,也就是将“:=”右边的数值给左边的变量。

橘色的代码就是发明者量化工具的麦语言 API，注意看第一行中的 OPEN 是获取收盘价的 API，直接使用就可以了；第二行中的 MA 是获取均线的 API，它需要传入 2 个参数，也就是说你需要告诉发明者量化工具，你需要什么样的均线：如果你希望获取以开盘价计算的 50 周期均线，那么可以写为：MA(OPEN, 50)；注意两个参数之间有个英文逗号。

黄色的“//”是注释符，其后面的蓝色中文就是注释内容，这些都是自己看的，用来提示该行代码是什么意思。程序运行时不对注释做任何处理。注意注释符前面，每行代码都要有一个英文分号，作为该行的结尾。

有了基础的代码结构认知后，我们下面就给大家带来一些常用的麦语言，以后我们也会常用这些语言。

OPEN——获取最新 K 线的开盘价

例：AA: =OPEN; 获取最新 K 线的开盘价，并把结果赋值给 AA

HIGH——获取最新 K 线的最高价

例：AA: =HIGH; 获取最新 K 线的最高价，并把结果赋值给 AA

LOW——获取最新 K 线的最低价

例：AA: =LOW; 获取最新 K 线的最低价，并把结果赋值给 AA

CLOSE——获取最新 K 线的收盘价，当盘中 k 线没有走完的时候，取得最新价

例：AA: =CLOSE; 获取最新 K 线的收盘价，并把结果赋值给 AA

VOL——获取最新 K 线的成交量

例：AA: =VOL; 获取最新 K 线的成交量，并把结果赋值给 AA

REF(X, N)——引用 X 在 N 个周期前的值。

例：REF(CLOSE, 1); 获取上根 K 线的开盘价

MA(X, N)——求 X 在 N 个周期内的简单移动平均

例：MA(CLOSE, 10); //获取最新 K 线的 10 周期均线

CROSSUP(A, B)——表示当 A 从下方向上穿过 B，成立返回 1(Yes)，否则返回 0(No)

例：CROSSUP(CLOSE, MA(C, 10)) //收盘价上穿 10 周期均价

CROSSDOWN(A, B)——表示当 A 从上方向下穿 B，成立返回 1(Yes)，否则返回 0(No)

例：CROSSDOWN(CLOSE, MA(C, 10)) //收盘价下穿 10 周期均价

BK——买入开仓

例：CLOSE>MA(CLOSE, 5), BK; //收盘价大于 5 周期均线，买入开仓

SP——卖出平仓

例：CLOSE<MA(CLOSE, 5), SP; //收盘价小于 5 周期均线，卖出平仓

SK——卖出开仓

例：CLOSE<MA(CLOSE, 5), SK; //收盘价小于 5 周期均线，卖出开仓

BP——买入平仓

例：CLOSE>MA(CLOSE, 5), BP; //收盘价大于 5 周期均线，买入平仓

BPK——买入平仓，并买入开仓（反手做多）

例：CLOSE>MA(CLOSE, 5), BPK; //收盘价大于 5 周期均线，平掉空仓，再买开仓。

SPK——卖出平仓，并卖出开仓（反手做空）

例：CLOSE<MA(CLOSE, 5), SPK; //收盘价小于 5 周期均线，平掉多仓，再卖开仓。

CLOSEOUT——平掉所有持仓，建议在加减仓模型中使用。例：CLOSEOUT; 平掉所有方向的仓位。

常用的 JavaScript 语言 API

在讲解 JavaScript 语言 API 之前，我们先来看下常用的代码结构是怎样的，以及它都有哪些功能组成，这将帮助你更好的理解 API，请看下图示例：

```
1 var aa = exchange.GetRecords(); //获取K线数据
2 var bb = exchange.SetContractType("rb1905"); //设置交易品种为“螺纹钢1905”
```

图 2-10 JavaScript 代码例子

如上图所示的代码：

在 JavaScript 语言中创建变量通常称为“声明”变量。红色代码，我们使用 var 关键词来声明变量，变量名是橘色代码：“aa”。

在 JavaScript 语言中，用等号赋值，也就是将“=”右边的数值给左边的变量。青色代码“exchange”是交易所对象，这里的交易所指的就是你所设置的期货公司，这是一个固定的格式，也就是你在调用 JavaScript 语言的 API 时，必须指定交易所对象。

绿色代码就是 JavaScript 语言的 API 了，当我们调用它的时候，其实是调用交易所对象中的函数。注意蓝色代码后面的点，也是一个固定格式。这里的函数与我们中学时学的函数是一个意思。如果该函数不需要指定参数，就以空括号表示；如果该函数必须传入参数，就把参数写到括号里面。

通过案例，清楚代码的基本结构原理后，下面我们展示几个大家日后会常用到的 JavaScript 语言 API

SetContractType(“品种代码”)——设置合约类型，也就是你要交易哪个品种
例：exchange.SetContractType(“rb1905”); //设置交易的品种为“螺纹钢 1905 合约”

GetTicker——获取 Tick 数据

例：exchange.GetTicker(); //获取 Tick 数据

GetRecords——获取 K 线数据

例：exchange.GetRecords(); //获取 K 线数据

Buy——买入

例：exchange.Buy(5000, 1); //以 5000 元的价格买一手

Sell——卖出

例：exchange.Sell(5000, 1); //以 5000 元的价格卖一手

GetAccount——获取账户信息

例：exchange.GetAccount(); //获取账户信息

GetPosition——获取持仓信息

例：exchange.GetPosition(); //获取持仓信息

SetDirection——设置做多做空下单类型

例：

exchange.SetDirection(“buy”); //设置下单类型为买入开多仓

exchange.SetDirection(“closebuy”); //设置下单类型为卖出平多仓

exchange.SetDirection(“sell”); //设置下单类型为卖出开空仓

exchange.SetDirection(“closesell”); //设置下单类型为买入平空仓

Log——在日志中输出一条信息

例：Log(“hello, world”); //在日志中输出”hello world”

Sleep——使程序暂停一段时间

例：Sleep(1000); //使程序暂停 1 秒

可能有些小伙伴会有疑问，上面这么多 API，怎么记住呢？其实这些都不用你死

记硬背，发明者量化官方有一套详尽的 API 文档。就像查字典一样，当你用的时候，需要什么直接查就行了。不必被代码等初次认识的内容吓住，我们要的是通过这些语言去组织自己的策略，请大家记住，技术永远不是量化的门槛，是否拥有好的策略才是你能否在量化市场走的长远的关键。

总结

以上就是在量化交易中最常用到的 API 了，基本上包括了：获取数据、计算数据、下单买卖，足以应付一个简单的量化交易策略，当然如果想要编写一个更复杂的策略，就需要去发明者量化工具官网去获取了。

课后习题

- 1、试着写一个麦语言 5 周期均线上穿 10 周期均线语句。
- 2、试着用 JavaScript 语言的 GetAccount 获取你的账户信息，并用 Log 打印到日志。

下节预告

编程就像是组装乐高积木，API 就像积木的各个零件，而编程的过程就是把各个乐高零件组成一个完整的玩具。下节我将带领大家用麦语言 API，组装一个完整的量化交易策略。

2.4 如何在发明者量化系统上编写策略

摘要

经过前面几节的学习，现在终于可以上手编写量化交易策略了。这将是你的手工交易踏入量化交易最重要的一步。其实也没有那么神秘，编写策略无非是把你的

想法用代码变现出来。本节将从零开始实现一个量化交易策略，带大家熟悉如何在发明者量化系统上编写策略。

准备

首先打开发明者量化工具的官网，依次点击“策略库”、“新建策略”，需要注意的是在开始编写代码之前，需要在编程语言下拉菜单中，选择麦语言或者 JavaScript 语言，当然该平台也支持 Python、C++以及可视化语言。

策略想法

在之前的章节，曾经介绍过一个价格突破均线的策略。即：如果价格高于最近 10 天的平均价格就买入，如果价格低于最近 10 天的平均价格就卖出。但是价格虽然能直观的反映出市场状态，却会有很多假突破信号；所以我们要对这个策略升级改进。

首先，选择一个较大周期均线，来判断趋势走向，这至少已经过滤将近一半假突破信号，大周期均线虽然迟钝，但是会更加稳定；然后，为了再次提高入场的成功率，再增加一个条件，这条大周期均线至少是向上的；最后使用价格、短期均线、长期均线的相对位置关系形成一个完整的交易策略。

策略逻辑

有了以上策略想法和思路，我们就可以试着构建策略逻辑了。这里的逻辑并不是让你计算天体运行规律，它没有那么复杂。无非就是把之前的策略想法，用文字表现出来。

多头开仓：如果当前没有仓位，并且收盘价大于短期均线，并且收盘价大于长期均线，并且短期均线大于长期均线，并且长期均线是上升的。

空头开仓：如果当前没有仓位，并且收盘价小于短期均线，并且收盘价小于长期均线，并且短期均线小于长期均线，并且长期均线是下降的。

多头平仓：如果当前持有多单，并且收盘价小于长期均线，或者短期均线小于长期均线，或者长期均线是下降的。

空头平仓：如果当前持有空单，并且收盘价大于长期均线，或者短期均线大于长期均线，或者长期均线是上升的。

以上就是整个量化交易策略的逻辑部分，那么如果我们把文字版的策略逻辑转换成代码，它将包括：获取行情、计算指标、下单买卖，这三个步骤。

麦语言策略

首先是获取行情，在这个量化交易策略中，我们只需要获取收盘价就行了，那么在麦语言中，获取收盘价的 API 就是：CLOSE，也就是说你只需要带代码中，写下 CLOSE 就已经获取到了最新一根 K 线的收盘价。

然后是计算指标，在这个量化交易策略中，我们一共用到了 2 个技术，即：短期均线和长期均线，我们假设短期均线为 10 周期均线，长期均线为 50 周期均线，那么怎么用代码表示 10 周期均线和 50 周期均线呢？请看下图：

```
1 MA10:=MA(CLOSE,10); //获取最新K线的10周期均线，并把结果存到变量MA10中
2 MA50:=MA(CLOSE,50); //获取最新K线的50周期均线，并把结果存到变量MA50中
```

图 2-11 麦语言策略代码

在手工交易中，我们能一眼看出 50 周期均线是上升的还是下降的，但是我们怎么用代码表示？仔细想想，判断均线上升不就是，当前 K 线的 50 周期均线数值比上根 K 线的 50 周期均线值大，并且上根 K 线的 50 周期均线数值比上上根 K 线的 50 周期均线值大吗？反之就是判断均线下降。那么用代码表示，它应该是这样的：

```
1 MA10:=MA(CLOSE,10); //获取最新K线的10周期均线，并把结果存到变量MA10中
2 MA50:=MA(CLOSE,50); //获取最新K线的50周期均线，并把结果存到变量MA50中
3
4 MA10_1:=REF(MA10,1); //获取上根K线的10周期均线，并把结果存到变量MA10_1中
5 MA50_1:=REF(MA50,1); //获取上根K线的50周期均线，并把结果存到变量MA50_1中
6 MA10_2:=REF(MA10,2); //获取上上根K线的10周期均线，并把结果存到变量MA10_2中
7 MA50_2:=REF(MA50,2); //获取上上根K线的50周期均线，并把结果存到变量MA50_2中
8 MA50_ISUP:=MA50>MA50_1 AND MA50_1>MA50_2; //判断当前K线的50周期均线是否上升
9 MA50_ISDOWN:=MA50<MA50_1 AND MA50_1<MA50_2; //判断当前K线的50周期均线是否下降
```

图 2-12 麦语言判断均线代码

注意上图第 8 行和第 9 行，玫红色的代码“AND”，在麦语言中它代表“并且”的意思。比如：第 9 行翻译成中文就是：如果当前 K 线的 50 周期均线大于上根 K 线的 50 周期均线，并且上根 K 线的 50 周期均线大于上上根 K 线的 50 周期均线，那么就把值计算为“是”；否则就把值计算为“否”，并把结果赋值给“MA50_ISUP”。

最后一步就是下单买卖了，只需要在买卖逻辑代码的后面，调用发明者量化工具的下单 API 就可以执行买卖操作。请看下图：

```
1 MA10:=MA(CLOSE,10); //获取最新K线的10周期均线，并把结果存到变量MA10中
2 MA50:=MA(CLOSE,50); //获取最新K线的50周期均线，并把结果存到变量MA50中
3
4 MA10_1:=REF(MA10,1); //获取上根K线的10周期均线，并把结果存到变量MA10_1中
5 MA50_1:=REF(MA50,1); //获取上根K线的50周期均线，并把结果存到变量MA50_1中
6 MA10_2:=REF(MA10,2); //获取上上根K线的10周期均线，并把结果存到变量MA10_2中
7 MA50_2:=REF(MA50,2); //获取上上根K线的50周期均线，并把结果存到变量MA50_2中
8 MA50_ISUP:=MA50>MA50_1 AND MA50_1>MA50_2; //判断当前K线的50周期均线是否上升
9 MA50_ISDOWN:=MA50<MA50_1 AND MA50_1<MA50_2; //判断当前K线的50周期均线是否下降
10
11 CLOSE>MA10 AND CLOSE>MA50 AND MA10>MA50 AND MA50_ISUP,BK; //多头开仓
12 CLOSE<MA10 AND CLOSE<MA50 AND MA10<MA50 AND MA50_ISUP,SK; //空头开仓
13 CLOSE<MA50 OR MA10<MA50,SP; //多头平仓
14 CLOSE>MA50 OR MA10>MA50,BP; //空头平仓
```

图 2-13 麦语言买卖交易代码

注意上图第 13 行和第 14 行，玫红色的代码“OR”，在麦语言中它代表“或者”的意思。比如：第 13 行翻译成中文就是：如果当前 K 线的收盘价小于当前 K 线的 50 周期均线，或者当前 K 线的 10 周期均线小于当前 K 线 50 周期均线，就把值计算为“是”，并立即下单；否则计算为“否”，并什么都不做。

大家注意：“AND”和“OR”是麦语言中的逻辑运算符：

“AND”是所有条件都为“是”的时候，最终条件才为“是”；

“OR”是所有条件中，只要有任何一个条件为“是”，最终条件就为“是”。

总结

以上就是在发明者量化工具上用麦语言编写交易策略的整个过程，总共也就三步：从有一个策略想法，到策略构思并用文字把逻辑描述出来，最后用代码实现完整的交易策略。尽管这是一个简单的策略，但是具体的实现流程与复杂的策略大同小异，只是策略的算法和数据结构各有不同。因此只要理解掌握本节的量化策略流程后就可以针对所需的在发明者量化工具上用麦语言开展量化策略研究

和实践。

课后习题

- 1、自己试着实现本节中的策略。
- 2、在本节策略基础上，加入止盈止损功能。

下节预告

在量化交易策略开发中，编程语言就像武器装备，一门好的编程语言能使你事半功倍。比如在量化交易界最常用的 Python、C++、Java、C#、EasyLanguage、麦语言等等多达十几种。究竟该选择哪种武器上战场呢？下节我们就来介绍这些常见的编程语言，以及每个编程语言自身的特点。

第三章 简单编程语言实现交易策略

3.1 量化交易编程语言横向评测

摘要

在第一章和第二章中，我们学习了量化交易基础知识和发明者量化工具的使用方法，本章我们就来具体实现交易策略。工欲善其事，必先利其器。要实现交易策略，必须先掌握一门编程语言。这一节先来介绍在量化交易中主流的编程语言，以及每个编程语言自身的特点。

什么是编程语言

学习编程语言之前，首先要搞清楚“编程语言”这个概念。编程语言就是人和计算机都能看懂的语言，它是一种被标准化的交流代码，编程语言的目的是使用人类语言去控制计算机，告诉计算机我们要做的事情。计算机可以根据编程语言执行指令，我们也可以编写代码，向计算机发出指令。

就像小时候父母教我们开口说话，也教我们如何理解别人讲话的意思。经过长时间的熏陶和自我学习，我们竟然在不知不觉中学会了说话，也能听懂其他小朋友说话的意思。语言有很多种，包括中文、英文、法文等等，比如：

中文：世界你好

英文：Hello World

法文：Bonjour tout le monde

如果用编程语言，在电脑屏幕上显示“世界你好”，就是这样的：

C 语言：`puts("世界你好");`

Java 语言：`System.out.println("世界你好");`

Python 语言：`print("世界你好")`

可以看到计算机语言拥有自己特定的规则，而且语言还有很多种，而这些语言规则就是我们今天需要为大家讲解的编程语言分类，在每一种分类中我们只需要记住最基础常用的规则，就能够利用这些编程语言和电脑沟通，让电脑按照我们的指令运行相应的策略。

编程语言分类

为了便于大家参考对比，挑选适合自己的量化交易编程语言，我们将最常用的六种编程语言做一个归类说明，它们分别是 Python、Matlab/R、C++、Java/C#、EasyLanguage 以及可视化语言（如下图）。

	功能范围	运行速度	可扩展性	学习难度
Python	★★★★	★★★★	★★★★★	★★★★
Matlab/R	★★	★★	★★	★★★★
C++	★★★★★	★★★★★	★★★★	★
Java/C#	★★★	★★★★	★★★	★★
EasyLanguage	★	★	★	★★★★★
可视化	★	★★★★	★	★★★★★

图 3-1 编程语言评价

我们从功能范围、运行速度、可扩展性、学习难度分别为它们评分。分值在 1~5 之间，比如在功能范围上得了 5 分，就意味着功能强大，1 分就意味着功能较少。

（如上图）可视化语言和 EasyLanguage 语言简单易学，非常合新手；Python 功能强大扩展能力强，适合开发比较复杂的交易策略；C++ 交易速度更快，更适合高频交易者。

但是对于每个编程语言的评测，主要是针对量化交易领域中的应用，并且带有个人的主观成分。也欢迎在之后的评论区拍砖，或提出你的观点讨论。接下来，我们就开始一个个地介绍这些编程语言。

可视化语言

可视化编程由来已久，并不是新鲜事物。这种“所见即所得”的编程思想，搭载着各种控件模块，仅仅以拖拽的方式，就可以构建代码逻辑，完成交易策略设计，过程很像搭积木一样。



图 3-2 可视化编程语言界面

如上图，同样的程序，在发明者量化交易平台可视化编程中只需要几行代码就搞定。这极大的降低了编程门槛，尤其针对那些完全不懂编程的交易者，这是非常棒的操作体验。

由于该可视化语言实现策略底层是转为 C++ 的，所以对程序运行速度的影响不大。但是功能和可扩展性较弱，不能开发过于复杂、过于精细化的交易策略。

EasyLanguage 语言

所谓的 EasyLanguage 语言，指的是部分商业化量化交易软件独有的编程语言。尽管这些语言也有部分面向对象特性，但在应用中主要还是采用脚本形式。在语法上面，也非常接近我们的自然语言，对于量化交易初学者来说，使用 EasyLanguage 作为入门是个比较好的选择。比如：发明者量化交易平台中的麦语言。

这种脚本语言在它特定的软件中做策略回测和实盘是没有问题的，但是在扩展方面，往往是有限的，比如策略开发者不能调用外部 API。而且在运行速度上，这种脚本语言都是在它自己的虚拟机上运行，性能优化不如 Java/C#，速度较慢。

Python

在 Stackoverflow 上，最近几年主流的编程语言访问量基本没有太大的变动，只有 Python 是一路呈上升趋势。Python 可用于网站开发、机器学习、深度学习、数据分析等，因其灵活性和开放性已经成为了最通用的语言。量化投资领域也是如此，目前国内的量化平台，多以 Python 为主。

Python 的基本数据结构列表和字典，功能非常强大，基本上能够满足数据表示的需求。如果需要更加快捷、功能更全面的数据结构，推荐适用 NumPy 和 SciPy，这两个库基本上称为 Python 科学计算的标准库了。

对金融工程而言，更有针对性的库是 Pandas，具有 Series 和 DataFrame 两个数据结构，非常适合于处理时间序列。

在速度方面，Python 处于中游的位置，比 C++慢一些，比 EasyLanguage 语言快一些，主要因为 Python 是一种动态语言，在纯 Python 语言运行时速度一般般。但是可以用 Cython 把部分功能静态优化，就能接近 C++的速度。

作为胶水语言，在扩展性能方面，Python 是当之无愧的第一名，除了可以广泛的对接其他语言外，而且扩展 API 的设计非常易用。就学习难度方面，Python 语法简单，代码可读性高，容易入门。

Matlab/R

接着是 Matlab 和 R 语言，这两种语言主要定位于数据分析，语言作者在语法上为科学运算做了很多设计，其特点是天生支持量化交易运算。但是应用范围比较有限，一般多用于数据分析和策略回测。对于交易系统和策略算法开发，其易用性和稳定性较差些。

另外，它们的运行速度和扩展能力也相对较差，因为 Matlab 和 R 语言是在独有的语言虚拟机上运行。从性能上看，它们的虚拟机要比 Java 和 C#差很多。但由于它们的语法更接近于数学表达公式，学起来也相对容易一些。

C++

C++是一种通用程序设计语言，支持多重编程模式，例如过程化程序设计、数据抽象、面向对象程序设计、泛型程序设计和设计模式等。用 C++语言可以实现所有你想实现的功能，但是这么强大的语言有个最大的缺点就是学习难度非常高，比如模板、指针、内存泄露等等。

目前，C++仍然是大容量，高频率交易的首选编程语言，原因很简单，因为 C++语言特点更容易接近计算机底层，是开发处理大量数据的高性能回测和执行系统的最有效工具。

Java/C#

Java/C#都是在虚拟机上运行的静态语言，与 C++相比，没有数组越界、没有 coredump、抛出的异常能精准的定位到出错代码的位置、自带垃圾自动回收机制、不用担心内存泄露等等。所以在语法的学习难度上，它们也比 C++更加容易。在运行速度方面，由于它们的虚拟机都自带运行时编译的 JIT 功能，所以速度仅次于 C++。

但是在功能方面，无法实现像 C++那样，对交易系统底层进行优化。在扩展性能方面，相对于 C++会弱一些，因为它们的扩展是需要经过 C 这个桥梁的，而这两种语言本身就是在虚拟机上运行，所以在扩展功能模块的时候，就需要多穿越一层墙才能实现。

总结

不过，话说回来，量化的编程语言不重要，重要的是想法。发明者量化的麦语言和可视化语言作为量化入门的敲门砖是完全没有问题的，入门之后的提高，则是需要不断结合不同的市场状况进行尝试探索，可谓思路决定出路，眼界决定境界。

“设计你的策略，交易你的思想。”从这个角度讲，量化交易的核心仍是交易思想。作为一名量化交易者，不仅需要掌握策略编写平台的基本语法和函数，更需要在实战中体会交易理念。量化只是体现不同交易理念的工具和载体而已。

课后习题

- 1、Python 语言作为量化交易都有哪些优势？
- 2、试着用发明者的麦语言写几个常用的 API？

下节预告

相信有了以上关于编程语言的介绍，你心里一定知道该怎么选择了吧，那么接下来几个章节，我们将按照编程语言的分类，有针对性的学习量化交易策略开发。

3.2 麦语言快速入门

摘要

什么是麦语言？所谓的麦语言就是从早期的股票技术指标延伸出来的一套程序化函数库。把算法封装到一个个函数里，用户只需要像拼“积木式”一样调用一行行函数，实现策略逻辑。

它采用“小语法，大函数”的构建模式，大幅提高编写效率，其他语言 100 多句的策略，用麦语言一般 10 几句就可以编出来。配合发明者量化工具的金融统计函数库和数据结构，同样也能支持部分复杂的交易逻辑。

完整策略

为了帮助大家快速理解本节的重点知识，在介绍发明者量化麦语言快速入门之前，先对本节名词概念有个初步了解。我们还是用长期 50 日均线 and 短期 10 日均线作为基本的案例，回顾一下上一章节讲过的完整策略案例：

多头开仓：如果当前没有仓位，并且收盘价大于短期均线，并且收盘价大于长期均线，并且短期均线大于长期均线，并且长期均线是上升的。

空头开仓：如果当前没有仓位，并且收盘价小于短期均线，并且收盘价小于长期均线，并且短期均线小于长期均线，并且长期均线是下降的。

多头平仓：如果当前持有多单，并且收盘价小于长期均线，或者短期均线小于长期均线，或者长期均线是下降的。

空头平仓：如果当前持有空单，并且收盘价大于长期均线，或者短期均线大于长期均线，或者长期均线是上升的。

如果用麦语言代码编写出来，就是这样的：

```
1 MA10:=MA(CLOSE,10); //获取最新K线的10周期均线,并把结果存到变量MA10中
2 MA50:=MA(CLOSE,50); //获取最新K线的50周期均线,并把结果存到变量MA50中
3
4 MA10_1:=REF(MA10,1); //获取上根K线的10周期均线,并把结果存到变量MA10_1中
5 MA50_1:=REF(MA50,1); //获取上根K线的50周期均线,并把结果存到变量MA50_1中
6 MA10_2:=REF(MA10,2); //获取上上根K线的10周期均线,并把结果存到变量MA10_2中
7 MA50_2:=REF(MA50,2); //获取上上根K线的50周期均线,并把结果存到变量MA50_2中
8 MA50_ISUP:=MA50>MA50_1 AND MA50_1>MA50_2; //判断当前K线的50周期均线是否上升
9 MA50_ISDOWN:=MA50<MA50_1 AND MA50_1<MA50_2; //判断当前K线的50周期均线是否下降
10
11 CLOSE>MA10 AND CLOSE>MA50 AND MA10>MA50 AND MA50_ISUP,BK; //多头开仓
12 CLOSE<MA10 AND CLOSE<MA50 AND MA10<MA50 AND MA50_ISDOWN,SK; //空头开仓
13 CLOSE<MA50 OR MA10<MA50,SP; //多头平仓
14 CLOSE>MA50 OR MA10>MA50,BP; //空头平仓
```

图 3-3 麦语言完整范例

要想编写一个完整的量化交易策略，通常需要：数据获取、数据计算、逻辑计算、下单买卖等几个步骤。如上图所示，在整个代码中，只用到了一个获取基础数据的 API，就是第一行和第二行中的“CLOSE”；然后第一行至第 9 行则是数据计算部分；最后第十一行至第十四行是逻辑计算和下单部分。

注意看，紫色的代码是变量；在第一行至第九行中，绿色的“:=”是赋值符，赋值符右边的数据计算完毕后赋值给赋值符左边的变量；橘色的代码就是 API，比如在第一行中，调用 MA（均线）需要传入两个参数，传入参数你可以理解为设置，也就是在调用 MA 的时候，需要设置 MA 的类型；玫红色的“AND”、“OR”是逻辑运算符，主要用来连接多个逻辑计算等等。有了以上的基础知识概念，下面我们就开始学习详细的麦语言基础。

基本数据

基础数据（开盘价、最高价、最低价、收盘价、成交量）是量化交易中不可或缺的一部分，在策略中获取最新的基础数据，只需要调用发明者量化工具的 API 就可以了。如果要获取历史的基础数据，可以使用“REF”，如：REF(CLOSE,1) 就是获取昨天的收盘价。

变量

变量是可以变的数，变量的名字可以理解为代号，其命名支持汉字、字母、数字、划线格式命名，但长度需控制在 31 字符内。变量名称不能相互重复，不能

与参数名重复，不能与函数名（API）重复，每个语句应该以分号结束。想要在编写后，加入自己的语言注释，在结尾处用“//”表示。需在半角输入法的大写状态下进行编写。如下图所示：

```
1 INT:=2; //整数类型
2 FLOAT:=3.1; //小数类型
3 ISTRUE:=CLOSE>OPEN; //布尔类型
```

图 3-4 麦语言数据类型

变量赋值

变量赋值就是把赋值符右边的数值给左边的变量，赋值符一共有 4 种，可以控制数值是否显示在图表上，以及定义显示的位置。下图绿色的字体就是赋值符啦，分别是“:”、“:=”、“^^”、“..”，图中代码注释部分，详细解释了它们的含义。

```
1 CC1: C; //将收盘价赋值给变量CC1，并在副图中显示
2 CC2:=C; //将收盘价赋值给变量CC2，但不在状态栏和图中显示
3 CC3^^C; //将收盘价赋值给变量CC3，并在主图中显示
4 CC4..0; //将开盘价赋值给变量CC4，并在状态栏中显示，但不在图中显示
```

图 3-5 麦语言变量赋值

数据类型

在麦语言中，有多种数据类型，其中最常用的是数值类型、字符串类型、布尔类型。数值类型就是数字，包括整数、小数、正负数等，如：1、2、3、1.1234、2.23456……；字符串类型你可以理解为文字，中文、英文、数字都可以是字符串，如：'发明者量化'、'CLOSEPRICE'、'6000'，字符串类型必须用英文分号包裹；布尔类型是最简单的，它只有 2 个值“是”和“否”，如：用 1 代表 true 表示“是”，0 代表 false 表示“否”。

关系运算符

关系运算符，顾名思义是用来比较两个值的关系的运算符。分别为等于、大于、

小于、大于等于、小于等于、不等于，如下图：

```
1 CLOSE = OPEN; //当收盘价等于开盘价, 返回 1 (true), 否则返回 0 (false)
2 CLOSE > OPEN; //当收盘价大于开盘价, 返回 1 (true), 否则返回 0 (false)
3 CLOSE < OPEN; //当收盘价小于开盘价, 返回 1 (true), 否则返回 0 (false)
4 CLOSE >= OPEN; //当收盘价大于等于开盘价, 返回 1 (true), 否则返回 0 (false)
5 CLOSE <= OPEN; //当收盘价小于等于开盘价, 返回 1 (true), 否则返回 0 (false)
6 CLOSE <> OPEN; //当收盘价不等于开盘价, 返回 1 (true), 否则返回 0 (false)
```

图 3-6 麦语言运算符

逻辑运算符

逻辑运算可以把单独的布尔类型语句连接成一个整体，最常用的是“AND”（并且）和“OR”（或）。假设有两个布尔类型值，分别是“收盘价大于开盘价”和“收盘价大于均线”，我们可以将它们组成一个布尔值，比如：“收盘价大于开盘价并且（AND）收盘价大于均线”，“收盘价大于开盘价或者（OR）收盘价大于均线”。

```
1 AA:=2>1; //返回值为1, 成立
2 BB:=4>3; //返回值为1, 成立
3 CC:=6>5; //返回值为1, 成立
4 DD:=1>2; //返回值为0, 不成立
5
6 AA && BB AND CC; //返回值为1, 成立
7 AA && BB AND DD; //返回值为0, 不成立
8 AA || BB OR DD; //返回值为1, 成立
```

图 3-7 麦语言逻辑运算

大家注意：

“AND”是所有条件都为“是”的时候，最终条件才为“是”；

“OR”是所有条件中，只要有任何一个条件为“是”，最终条件就为“是”。

“AND”可以写成“&&”，“OR”可以写成“||”。

算数运算符

常用的麦语言的算数运算符（“+”、“-”、“*”、“/”）和小学学习的数学

没有任何区别，如下图：

```
1 AA:=1+1; // 计算结果为2
2 BB:=2-1; // 计算结果为1
3 CC:=2*1; // 计算结果为2
4 DD:=2/1; // 计算结果为2
```

图 3-8 麦语言算数运算

优先级

如果有一个 $100*(10-1)/(10+5)$ 表达式，程序是先计算哪一步？中学数学告诉我们：①如果是同一级运算，一般按从左往右依次进行计算。②如果既有加减、又有乘除法，先算乘除法、再算加减。③如果有括号，先算括号里面的。④如果符合运算定律，可以利用运算定律进行简算。麦语言的优先级也是如此，如下图：

```
1 100*(10-1)/(10+5) //计算结果为60
2 1>2 AND (2>3 OR 3<5) //运算结果为假
3 1>2 AND 2>3 OR 3<5 //运算结果为真
```

图 3-9 麦语言算数运算优先级

执行模式

在发明者量化工具的麦语言中，程序策略执行一共有 2 种模式，即：收盘价模式和实时价模式。收盘价模式指当前 K 线信号成立，在下根 K 线开始的时候立即执行下单交易。实时价模式指的当前 K 线信号成立，就立即执行下单交易。

日内策略

如果是日内策略，尾盘需要平仓时，就需要用到“TIME”时间函数。该函数在秒周期以上，日周期以下，显示为四位数的形式，即：HHMM（1450——14 点 50 分）。注意：使用 TIME 函数作为尾盘平仓的条件，建议开仓条件也要做相应的时间限制。如下图：


```
1 CLOSE>OPEN&&TIME<1450,BK; //如果是阳线，并且时间小于14点50分，开仓买入
2 TIME>=1450,CLOSEOUT; //如果时间大于14:50，全部平仓。
```

图 3-10 麦语言时间函数

模型分类

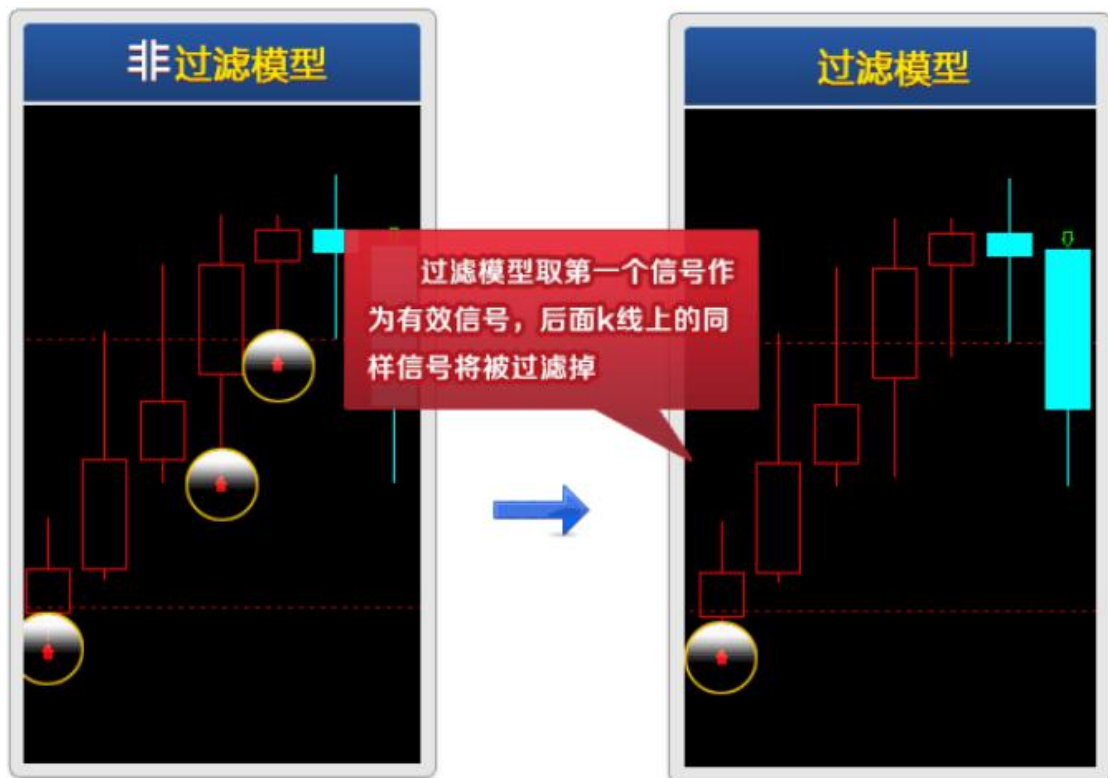


图 3-11 麦语言模型分类

麦语言中的模型分类一共有两种，即：非过滤模型和过滤模型。这个其实很好理解：非过滤模型允许连续出现开仓信号或平仓信号，可以实现加仓和减仓功能。过滤模型不允许连续出现开仓或平仓信号，也就是当开仓信号出现后，之后的开仓信号会被过滤掉，直到出现平仓信号，非过滤模型出信号的顺序是：开-平-开-平-开.....

总结

以上就是麦语言快速入门的内容，学习后就可以编程量化交易策略啦。如果需要编写更加复杂的策略，可以参考发明者量化工具麦语言 API 文档，或者直接咨询

官方客服代写量化交易策略。

下节预告

日内交易也是一种交易模式,这种方式不留仓过夜,所以受市场波动率风险较低,一旦出现不利行情,可以及时进行调整。学习了本节麦语言入门,下节我们将带大家手把手编写一个可行的日内量化交易策略。

课后习题

- 1、试着用发明者量化工具写下麦语言获取基础数据的 API。
- 2、变量赋值在图表中一共有哪些显示方式?

3.3 如何用麦语言实现策略

摘要

在上一篇中，我们从麦语言的简介、基础语法、模型执行方式、模型分类等方面为大家讲解实现交易策略的前提部分，本篇我们将继续上篇内容，从常用的策略模块、技术指标，一步一步帮助大家实现一个可行的日内量化交易策略。

策略模块

想想一下，你是怎么用乐高积木碎片拼一个机器人的？你总不可能从上到下或者从下到上，一块一块拼起来。稍微有点常识的人都知道，应该把头、手臂、腿、翅膀等等，各自拼起来，然后再组合成一个完整的机器人。写程序也是一样的，把所需要的功能都写成一个一个策略模块，然后再把一个个策略模块，组合成一个完整的量化交易策略。下面我就先列举些常用的策略模块：

阶段涨幅

阶段涨幅就是计算当根 K 线收盘价与前 N 个周期收盘价的差值的百分比。比如：计算最近 10 根 K 线阶段涨幅是多少，用代码可以写成：

```
1 CLOSE_0:=CLOSE; //获取当根K线收盘价，并把结果存到变量CLOSE_0中
2 CLOSE_10:=REF(CLOSE,10); //获取前面第10根K线的收盘价，并把结果存到变量CLOSE_10中
3
4 (CLOSE_0-CLOSE_10)/CLOSE_10*100; //计算当根K线收盘价与前面第10根K线收盘价的差值的百分比
```

图 3-12 麦语言阶段涨幅

再创新高

再创新高就是计算当根 K 线是否大于 N 个周期以来的最高价。比如：计算当根 K 线是否大于最近 10 根 K 线中的最高价，用代码可以写成：

```
1 HHV_10:=HHV(HIGH,10); //获取10个周期最高价的最大值,即10周期高点(包含当前k线)。  
2 HIGH>REF(HHV_10,1); //判断当根K线的最高价是否大于上根K线的HHV_10值
```

图 3-13 麦语言再创新高

放量上攻

放量上攻可以理解为价格上扬,成交量剧增。比如:如果当根 K 线的收盘价是前第 10 根 K 线的收盘价的 1.5 倍,即 10 天内涨了 50%;成交量超过最近 10 根 K 线平均值的 5 倍。用代码可以写成:

```
1 CLOSE_10:=REF(CLOSE,10); //获取前第10根K线的收盘价  
2 IS_CLOSE:=CLOSE/CLOSE_10>1.5; //判断当前K线收盘价与CLOSE_10的比值是否大于1.5  
3  
4 VOL_MA_10:=MA(VOL,10); //获取最近10根K线成交量的平均值  
5 IS_VOL:=VOL>VOL_MA_10*5; //判断当前K线成交量是否大于VOL_MA_10的5倍  
6  
7 IS_CLOSE AND IS_VOL; //判断IS_CLOSE条件和IS_VOL条件是否同时成立
```

图 3-14 麦语言放量上攻

窄幅整理

窄幅整理就是指近一段时期价格维持在一定幅度之内。比如:如果 10 个周期内的最高价与以 10 个周期内的最低价的差值,除以当根 K 线的收盘价,小于 0.05 左右。用代码可以写成:

```
1 HHV_10:=HHV(CLOSE,10); //获取10个周期最高价的最大值(包括当根K线)  
2 LLV_10:=LLV(CLOSE,10); //获取10个周期最低价的最小值(包括当根K线)  
3  
4 (HHV_10-LLV_10)/CLOSE<0.05; //判断HHV_10与LLV_10的差值除以当根K线收盘价是否小于0.05
```

图 3-15 麦语言窄幅整理

均线多头排列

均线多头排列分为多头排列和空头排列，K 线在 5—10—20—30—60 均线下支撑排列向上为多头排列，多头排列就是市场趋势是强势上升势。用代码可以写成：

```
1 MA_5:=MA(CLOSE,5); //获取5 周期收盘价的简单移动平均
2 MA_10:=MA(CLOSE,10); //获取10周期收盘价的简单移动平均
3 MA_20:=MA(CLOSE,20); //获取20周期收盘价的简单移动平均
4 MA_30:=MA(CLOSE,30); //获取30周期收盘价的简单移动平均
5
6 //判断MA_5是否大于MA_10, 并且MA_10是否大于MA_20, 并且MA_20是否大于MA_30
7 MA_5>MA_10 AND MA_10>MA_20 AND MA_20>MA_30;
```

图 3-16 麦语言均线多头排列

前期高点及其位置

要获取前期高点，以及这个高点所在的位置，可以直接通过发明者量化工具的 API 直接获取。用代码可以这样写：

```
1 HHV_20:=HHV(HIGH,20); //获取20个周期最高价的最大值(包括当根K线)
2 HHVBARS_20:=HHVBARS(HIGH,20); //获取20个周期最高价到当前K线的周期数
3 HHV_60_40:=REF(HHV_20,40); //获取60个周期前到40个周期前之间的最高价
```

图 3-17 麦语言前期高点

跳空缺口

跳空缺口就是两条 K 线的最高低价出现不衔接的情况，由两条 K 线组成，跳空缺口是日后支撑和压力点的参考价位。当一个跳空缺口出现时，可以假设一个沿着原来跳空方向上的趋势的加速已经开始了。用代码可以这样写：

```

1 HHV_1:=REF(H,1); //获取上根K线最高价
2 LLV_1:=REF(L,1); //获取上根K线最低价
3 HH:=L>HHV_1; //判断当前K线最低价是否大于上根K线最高价(向上跳空)
4 LL:=H<LLV_1; //判断当前K线最高价是否小于上根K线最低价(向下跳空)
5 HHH:=L/REF(H,1)>1.001; //加入辅助条件,向上跳空缺口越大,则信号越强烈
6 LLL:=H/REF(L,1)<0.999; //加入辅助条件,向下跳空缺口越大,则信号越强烈
7 JUMP_UP:HH AND HHH; //综合判断条件,是否向上跳空
8 JUMP_DOWN:LL AND LLL; //综合判断条件,是否向下跳空
    
```

图 3-18 麦语言跳空缺口

常用技术指标

移动平均线



图 3-19 移动平均线图

站在统计学的角度看，均线就是每天价格的算术平均，它是一条带有趋势性的价格轨迹。均线系统是大多分析者常用的技术工具，从技术角度看是影响技术分析者心理价位因素的，思维买卖的决策因素，是技术分析者的良好的参考工具，发明者量化工具支持多种不同类型的均线，如下图：

```

1 MA_DEMO:MA(CLOSE,5);           //计算收盘价在5周期的简单均值
2 EMA_DEMO:EMA(CLOSE,15);        //计算收盘价在15周期的指数平滑异动均值
3 EMA2_DEMO:EMA2(LOW,10);        //计算最低价在10周期的线性加权均值
4 EMAWH_DEMO:EMAWH(HIGH,50);     //计算最高价在50周期的指数加权均值
5 DMA_DEMO:DMA(OPEN,100);        //计算开盘价在100周期的动态均值
6 SMA_DEMO:SMA(CLOSE,10,3);      //计算收盘价在10周期的,权重为3的扩展指数加权均值
7 ADMA_DEMO:ADMA(CLOSE,9,2,30); //计算收盘价在9周期的,快线频率为2,慢线频率为30的考夫曼均值
    
```

图 3-20 麦语言各种指标计算

BOLL 通道



图 3-21 BOLL 通道图

BOLL 又称布林带指标，也是利用统计学原理，先根据 N 日移动平均线计算出中轨，再根据标准差，计算出上轨和下轨。当 BOLL 通道由宽变窄，说明价格逐渐向均值回归。当 BOLL 通道由窄变宽，意味着行情开始发生变化，如果价格上穿上轨，表明买力增强，如果价格下穿下轨，表明卖力增强。

在所有的技术指标中，BOLL 的计算方法是最复杂之一，其中引进了统计学中的标准差概念，涉及到中轨线 (MB)、上轨线 (UP) 和下轨线 (DN) 的计算。其计算方法如下：


```

1 MID:MA(CLOSE,100); // 计算100个周期收盘价均线，称为布林通道中轨
2 TMP2:=STD(CLOSE,100); // 计算100个周期内的收盘价的标准差
3 TOP:MID+2*TMP2; // 计算中轨加上2倍的标准差，称为布林通道上轨
4 BOTTOM:MID-2*TMP2; // 计算中轨加上2倍的标准差，称为布林通道下轨
    
```

图 3-22 麦语言布林带计算

MACD 指标



图 3-23 MACD 指标

MACD 指标是运用快速（短期）和慢速（长期）移动平均线及其聚合与分离的征兆，加以双重平滑运算。而根据移动平均线原理发展出来的 MACD，一则去除了移动平均线频繁发出假信号的缺陷，二则保留了移动平均线的效果，因此，MACD 指标具有均线趋势性、稳重性、安定性等特点，是用来研判买卖股票的时机，预测股票价格涨跌的技术分析指标。其计算方法如下：

```

1 DIFF:EMA(CLOSE,10)-EMA(CLOSE,50); //首先计算短期均线与长期均线的差值
2 DEA:EMA(DIFF,10); //最后对差值再次计算出平均值
    
```

图 3-24 麦语言 MACD 指标

以上就是在开发量化交易策略中较常用的策略模块，当然其实远不止这些，通过以上的模块例子，也可以动手实现几个你在主观交易中最常用到的交易模块，方法都是通用个的。接下来，我们就开始编写一个可行的日内量化交易策略。

策略编写

在外汇现货市场，曾经广为流传一种突破交易策略，它就是 HANS123 策略，它以其简洁的开盘后 N 根 K 线的高低点突破，作为交易信号触发的评判标准。这也是一种入场较早的交易模式。

策略逻辑

在开盘 30 分钟后准备入场；
上轨 = 开盘后 30 分钟高点；
下轨 = 开盘后 30 分钟低点；
当价格突破上轨，买入开仓；
当价格跌穿下轨，卖出开仓。
日内交易策略，收盘前平仓；

策略代码

```
1 //数据计算
2 Q:=BARSLAST(DATE<>REF( DATE,1))+1; //计算最新一个交易日的第一根K线到当前K线的周期数，并赋值给变量N
3 HH:VALUEWHEN( TIME=0930,HHV(H,Q)); //当时间等于9点30分时，取N周期内最高价的极大值，并赋值给变量HH
4 LL:VALUEWHEN( TIME=0930,LLV(L,Q)); //当时间等于9点30分时，取N周期内最低价的极小值，并赋值给变量LL
5
6 //下单交易
7 TIME>0930 AND TIME<1445 AND C>HH,BK; //如果时间大于9点30分，并且时间小于14点45分，并且收盘价是大于HH，则执行买开仓
8 TIME>0930 AND TIME<1445 AND C<LL,SK; //如果时间大于9点30分，并且时间小于14点45分，并且收盘价是小于LL，则执行卖开仓
9 TIME>=1445,CLOSEOUT; //如果时间大于等于14点45分，清空所有仓位
10
11 //信号过滤
12 AUTOFILTER; //启用一开一平信号过滤机制
```

图 3-25 麦语言策略代码

总结

以上我们学习了策略模块的概念，以及通过几个常用的策略模块案例，熟悉发明者量化工具的编程方法，可以说学好编写策略模块，提高编程逻辑思维，是进阶量化交易的关键一步。最后我们又用发明者量化工具实现了，在外汇现货交易中常用的交易策略。

下节预告

也许会有小伙伴感到困惑，密密麻麻的代码看不懂。别急，这些我们都已经替你想到了，在发明者量化工具中，还有一种编程语言，更适合小小白的用户使用，它就是可视化编程，顾名思义就是所见即所得，一起期待吧！

课后习题

- 1、试着动手实现几个你在主观交易中最常用到的交易模块。
- 2、试着用发明者量化工具中的麦语言实现 KDJ 指标算法。

3.4 可视化编程快速入门

摘要

很多主观交易者对量化交易感兴趣，刚开始信心满满，等学完传统编程语言的基础语法、数据运算、数据结构、逻辑控制等等，看着即冗长又复杂的代码后，往往又望而却步，或者浅尝辄止，这时可视化编程语言可能更适合带你入门。

完整策略

为了帮助大家快速理解本节的重点知识，在介绍发明者量化可视化编程语言快速入门之前，先看下用可视化语言写出来的策略是什么样的？以及对本节名词概念有个初步了解。我们以最简单的收盘价大于 50 周期均线做多，反之收盘价小于 50 周期均线做空为例：

多头开仓：如果当前没有仓位，并且收盘价大于 50 周期均线。

空头开仓：如果当前没有仓位，并且收盘价小于 50 周期均线。

多头平仓：如果当前持有多单，并且收盘价小于 50 周期均线。

空头平仓：如果当前持有空单，并且收盘价大于 50 周期均线。

如果用可视化语言把上面的策略编写出来，就是这样的（如下图）：

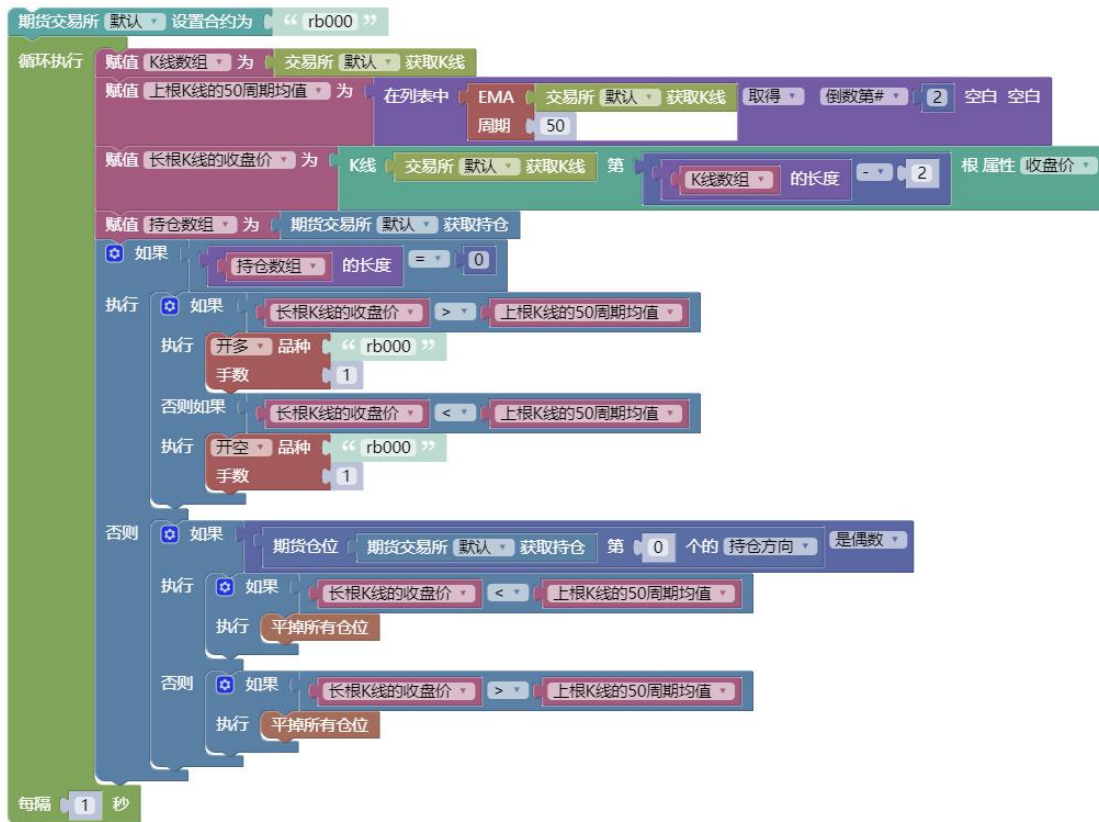


图 3-26 可视化语言界面

如上图所示，整个策略设计流程就是：设置行情品种、获取 K 线数组、获取上根

K 线的 50 周期均值、获取上根 K 线的收盘价、获取持仓数组、判断持仓状态、判断收盘价是否大于或小于均线、执行开仓或平仓。

这里需要注意“数组”这个概念，数组对于每一门编程语言来说都是重要的数据结构之一。数组就像容器一样，里面可以存放一系列的值。比如：调用获取 K 线数组的 API，它返回的结果是这样的：

```
1 //这是一个K线数组，里面有3个数据。分别是：上上根K线的数据、上根K线的数据、当根K线的数据。并且赋值给“arr”变量
2 arr = [{"Time":1540137600000,"Open":4013,"High":4116,"Low":4013,"Close":4085,"Volume":4124040},
        {"Time":1540224000000,"Open":4087,"High":4106,"Low":4068,"Close":4076,"Volume":3252216},
        {"Time":1540310400000,"Open":4064,"High":4123,"Low":4050,"Close":4120,"Volume":3642856}]
```

图 3-27 K 线数组

上图中的代码就是一个 K 线数组，该数组一共有 3 个数据，分别是上上根 K 线的数据、上根 K 线的数据、当根 K 线的数据。假如我们把这个数组赋值给一个变量“arr”，如果想要获取这个数组中，最后一个数据（当根 K 线的数据）可以这样写（如下图第 4、5 行）：

```
1 //这是一个K线数组，里面有3个数据。分别是：上上根K线的数据、上根K线的数据、当根K线的数据。并且赋值给“arr”变量
2 arr = [{"Time":1540137600000,"Open":4013,"High":4116,"Low":4013,"Close":4085,"Volume":4124040},
        {"Time":1540224000000,"Open":4087,"High":4106,"Low":4068,"Close":4076,"Volume":3252216},
        {"Time":1540310400000,"Open":4064,"High":4123,"Low":4050,"Close":4120,"Volume":3642856}]
3
4 k0_1 = arr[2]; //获取当根K线的数据，这是第一种写法
5 k0_2 = arr[arr.length - 1]; //获取当根K线的数据，这是第二种写法
6 k1 = arr[arr.length - 2]; //获取上根K线的数据
```

图 3-28 数组的引用

大家直接用第二种（第 5 行）的写法，因为现实中 K 线数据有成百上千根，而且新的 K 线是不断增加的。所以可以先获取数组的长度，“arr.length”的意思是获取该数组的长度，再减去“1”，就是最新 K 线的数据。如果想要获取上根 K 线的数据，就减去“2”。

细心的你可能会发现，这些数据都用“{}”包括起来，看英文名就大概知道里面对应的分别是：时间、开盘价、最高价、最低价、收盘价、成交量。如果想要获取上根 K 线的收盘价，直接在后面加上“.”再加上所需要的值就可以了，参照下图第 8~10 行。

```
1 //这是一个K线数组，里面有3个数据。分别是：上上根K线的数据、上根K线的数据、当根K线的数据。并且赋值给“arr”变量
2 arr = [{"Time":1540137600000,"Open":4013,"High":4116,"Low":4013,"Close":4085,"Volume":4124040},
        {"Time":1540224000000,"Open":4087,"High":4106,"Low":4068,"Close":4076,"Volume":3252216},
        {"Time":1540310400000,"Open":4064,"High":4123,"Low":4050,"Close":4120,"Volume":3642856}]
3
4 k0_1 = arr[2]; //获取当根K线的数据，这是第一种写法
5 k0_2 = arr[arr.length - 1]; //获取当根K线的数据，这是第二种写法
6 k1 = arr[arr.length - 2]; //获取上根K线的数据
7
8 k1.Close; //获取上根K线数据中的收盘价
9 k1.Time; //获取上根K线数据中的时间
10 k1.Volume; //获取上根K线数据中的成交量
```

图 3-29 数组的引用

为什么要使用可视化编程语言？

有了以上的概念，让我们先用 Java 语言写一个输出“hello, world”的程序，来感受一下传统编程，如下图：

```
1 public class HelloWorld {
2     public static void main(String[] args) {
3         System.out.println("hello world!");
4     }
5 }
```

图 3-30

仅仅只是输出一个“hello world!”字符串的程序，就写了 5 行代码。相信大多数初学者，只认识括号中的英文单词“hello, world”，其他更是无从下手。所以，比起手足无措，以可视化编程为入门，不失为更好的选择。

什么是可视化编程？

可视化编程由来已久，并不是新鲜事物。这种“所见即所得”的编程思想，搭载着各种控件模块，仅仅以拖拽的方式，就可以构建代码逻辑，完成交易策略设计，过程很像搭积木一样。

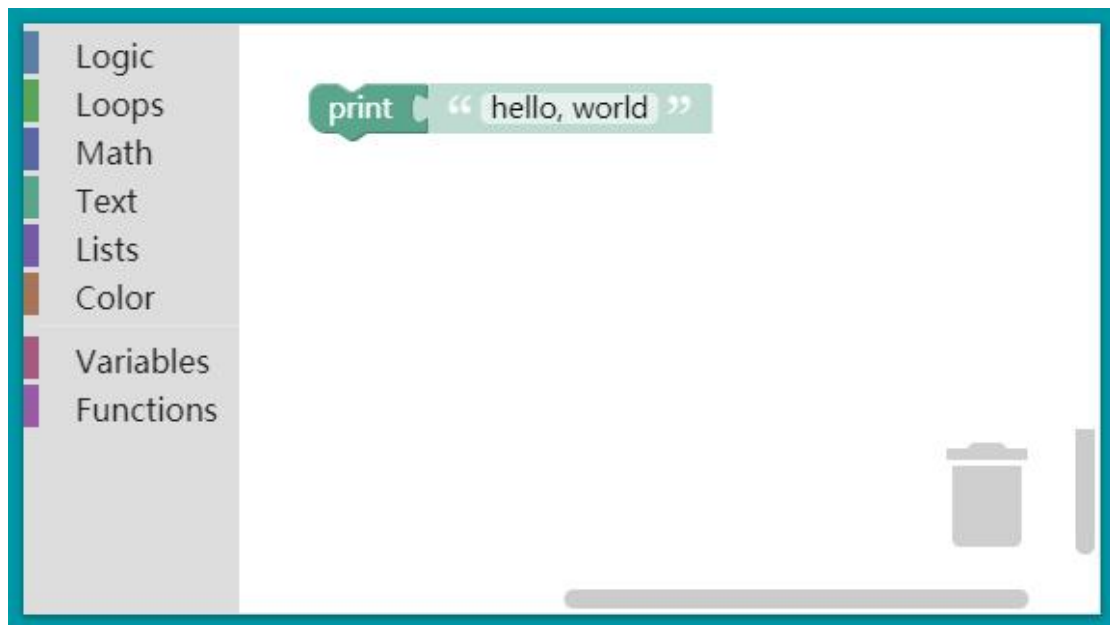


图 3-31

如上图，同样的程序，在 Blockly 可视化编程中只需要一行代码就搞定。这极大的降低了编程门槛，尤其针对那些完全不懂编程的交易者，这是非常棒的操作体验。

可视化编程语言有哪些特点

Blockly 可不是一个编程玩具，它是实诚的编辑器，而不是那种伪装成编辑器的操作系统，支持许多编程的基本元素，如：变量、函数、数组，及易于扩展自定义的块，你可以用它完成复杂的编程任务。在设计上十分符合 unix 哲学：Do one thing。

发明者量化的可视化编程，也正是借着 Google 发布的 Blockly 可视化工具来实现的。在设计上与麻省理大学推出的 Scratch 类似，真正的零门槛（如下图）。

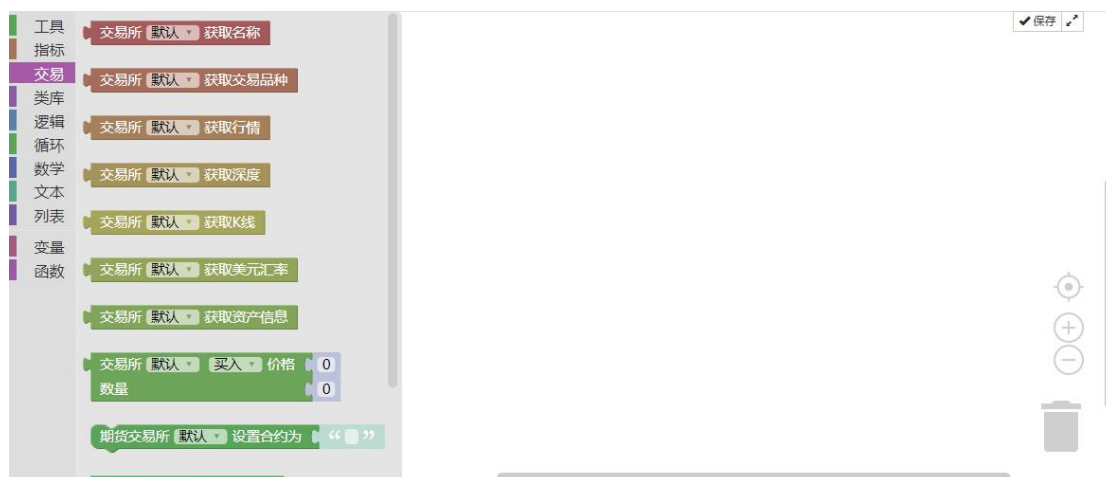


图 3-32

在发明者量化的可视化编程界面，内置上百种常用交易模块，后续会有更多的交易模块加入，来支持交易者的新思想和新应用，这些将由开发者共同开发和维护。

虽然语法简单，但又不失性能。几乎可以满足大多数简单的量化交易策略开发。无论是在功能、速度上，不输于 Python、JavaScript 等常规编程语言。未来将会支持逻辑复杂的金融应用。

如何使用



图 3-33

写一个 hello, world 程序



图 3-34

运行，把“hello, world”打印出来

第①步：模拟回测

第②步：点击下拉，选择商品期货CTP

第③步：设置必要的参数

第④步：点击“+”号

第⑤步：点击回测

回测日志 - 进度: 100.0 % - 耗时(秒): 0.044 秒 - 日志总数: 2 - 交易次数: 0 - 已完成 0.0000

账户信息

名称	定价货币	交易品种	平仓盈亏	持仓盈亏	保证金	手续费	预估收益
Futures_CTP	CNY	FUTURES	0	0	0	0	0

浮动盈亏

Futures_CTP_FUTURES_CNY

初始净值: 1000000, 累计收益: 0.000 %, 年化收益(252天): 0.000 %, 夏普比率: --, 年化波动率: -- %, 最大回撤: 0.000 % 全屏

Zoom 1h 3h 8h All

一月 '18 三月 '18 五月 '18 七月 '18 九月 '18 十一月 '18 一月 '19

一月 '18 四月 '18 七月 '18 十月 '18 一月 '19

日志信息 下载日志

时间	平台	类型	价格	数量	信息
2018-01-01 00:00:00		信息	hello world		
2018-01-01 00:00:00		信息	商品交易类库加载成功		

最后策略输出 "hello world"

图 3-35

总结

以上我们从一个完整的可视化策略开始，到可视化语言的简介以及特点，最后介绍如何在发明者量化工具上使用可视化语言，以及用写了一个“hello world”的例子。不过需要提醒大家的是，作为量化交易入门，可视化编程是一个很好的敲门砖，但是目前在发明者量化工具上，只开放了有限的 API 接口，对于量化交易来说，最好是把它当作帮助你理清策略逻辑的辅助工具。

下节预告

可视化编程与高级编程语言基础没什么区别，甚至有些地方是通用的，学会了可视化编程也就离学会高级编程更进一步。下节我们将深入可视化编程进阶学习，包括如何在发明者量化工具上用可视化语言编写常用的量化交易模块，以及如何开发一个完整的日内交易策略。

课后习题

- 1、在发明者量化可视化编程界面，使用 API 并理解他们的意思。
- 2、用可视化语言获取最新的开盘价，并把它输出到日志中。

3.5 如何用可视化语言实现策略

摘要

上篇内容我们学习了可视化编程语言的简介和特点、“hello world”例子，以及在发明者量化交易工具中策略编写等方面，为大家讲解实现交易策略的前提部分。本篇我们继续接上篇，从常用的策略模块和技术指标开始，再到策略逻辑，一步一步帮助大家实现一个完整的日内交易策略。

策略模块

阶段涨幅

阶段涨幅就是计算当前 K 线收盘价与前 N 个周期收盘价的差值的百分比。比如：计算最近 10 根 K 线阶段涨幅是多少，用代码可以写成：

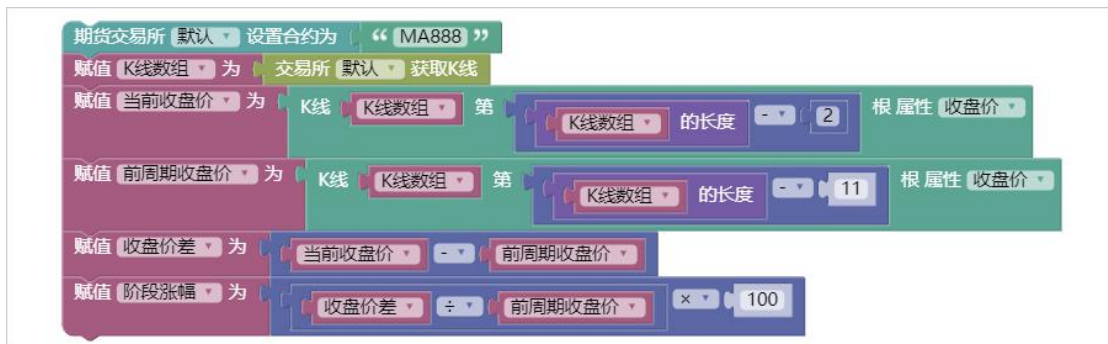


图 3-36

以上代码可以发现，计算机执行的方式是需要一个完整的逻辑循环，比如要计算最近 10 根 K 线阶段涨幅，需要拆分为以下几个步骤：

首先计算机要明确知道你要交易什么品种，比如上图案例为甲醇，那么设置合约代码为：“MA888”。设置完合约代码，就可以获取该合约的 K 线数据了。

有了 K 线数据，就可以从这些 K 线数据中，获取任何一根 K 线的详细数据。要统计阶段涨幅，就必须先获取 2 根 K 线收盘价，比如：上根 K 线收盘价和前面第 11 根 K 线的收盘价。

最后再根据这 2 根 K 线收盘价，计算出阶段涨幅比率。以下每一个策略都有这样的逻辑循环和条件属性规定的特征，看懂了这个逻辑，可视化编程也就会变得容易的多。

放量上攻

放量上攻可以理解为价格上扬，成交量剧增。比如：如果当前 K 线的收盘价是前

第 10 根 K 线的收盘价的 1.5 倍，即 10 天内涨了 50%；成交量超过最近 10 根 K 线平均值的 5 倍。用代码可以写成：

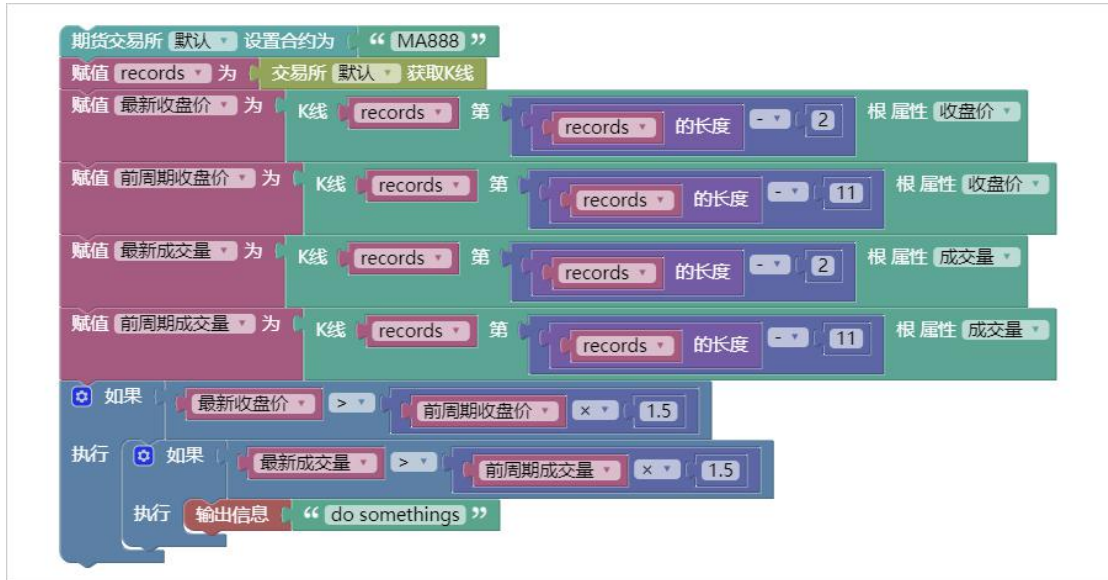


图 3-37

跳空缺口

跳空缺口就是两条 K 线的最高低价出现不衔接的情况，由两条 K 线组成，跳空缺口是日后支撑和压力点的参考价格。当一个跳空缺口出现时，可以假设一个沿着原来跳空方向上的趋势的加速已经开始了。用代码可以这样写：

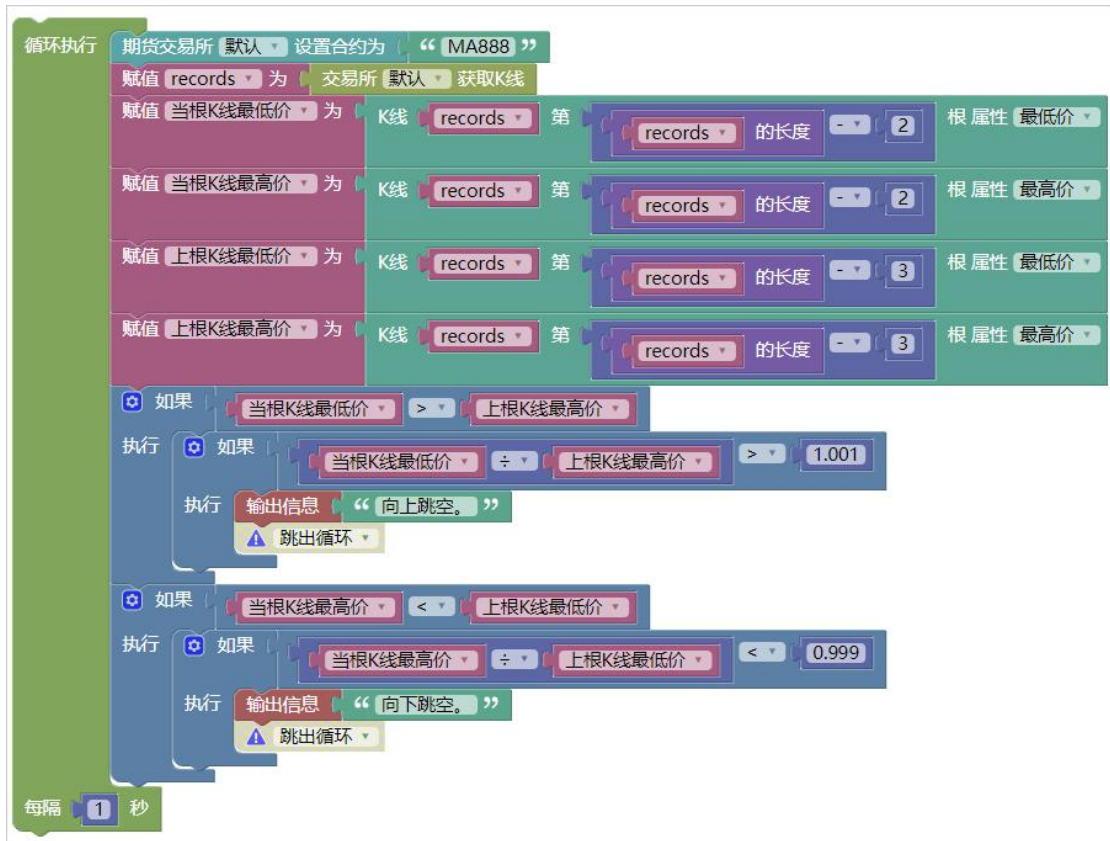


图 3-38

常用技术指标

EMA 平均线

站在统计学的角度看，均线就是每天价格的算术平均，它是一条带有趋势性的价格轨迹。均线系统是大多分析者常用的技术工具，从技术角度看是影响技术分析者心理价位因素的，思维买卖的决策因素，是技术分析者的良好的参考工具，发明者量化工具支持多种不同类型的均线，如下图：



图 3-39

MACD 指标

MACD 指标是运用快速（短期）和慢速（长期）移动均线及其聚合与分离的征兆，加以双重平滑运算。而根据移动均线原理发展出来的 MACD，一则去除了移动均线频繁发出假信号的缺陷，二则保留了移动均线的效果，因此，MACD 指标具有均线趋势性、稳重性、安定性等特点，是用来研判买卖股票的时机，预测股票价格涨跌的技术分析指标。其计算方法如下：



图 3-40

KDJ 指标

KDJ 指标综合了动量观念、强弱指标及移动平均线的优点，用来度量股价脱离价格正常范围的变异程度。考虑的不仅是收盘价，而且有近期的最高价和最低价，这避免了仅考虑收盘价而忽视真正波动幅度的弱点。其计算方法如下：



图 3-41

策略编写

沃伦 · 巴菲特的导师本杰明 · 格雷厄姆曾经在《聪明的投资者》一书中，曾经提到过一种股票债券动态平衡的交易模式。

这种交易模式非常简单：

把手中 50% 的资金投资于股票基金，剩下 50% 投资于债券基金。即股票和债券两者各占一半。

根据固定间隔时间或市场变化进行一次资产再平衡，使股票资产和债券资产的比例恢复到初始的 1:1。这就是整个策略的全部逻辑，包含了什么时候买卖，以及买卖多少。够简单吧！

在这个方法中，债券基金的波动率其实很小，远远低于股票波动率，所以债券在这里被当做『参照锚』，也就是说，用债券来衡量股票究竟是涨得太多了，还是涨得太少了。

如果，股票价格上涨，会使得股票的市值大于债券的市值，当两者市值比率超过设定的阈值时，则对总仓位进行重新调整，卖出股票，并且买入债券，使股债市值比例恢复至初始的 1:1。

反之，股票价格下跌，会使得股票的市值小于债券的市值，当两者市值比率超过设定的阈值时，则对总仓位进行重新调整，买入股票，并且卖出债券，使股债市值比例恢复至初始的 1:1。

就这样，在动态平衡股票和债券之间的比例，就够享受到股票成长的果实，并且减少了资产波动率。作为价值投资的先驱，格雷厄姆为我们提供了一个很好的思路。

策略逻辑

按照当前的 BTC 的价值，账户余额保留 ¥5000 现金和 0.1 个 BTC，即现金和 BTC 市值的初始比例是 1:1。

如果 BTC 的价格上涨至 ¥6000，即 BTC 市值大于账户余额，并且其之间的差超过设定的阈值，就卖掉 $(6000-5000)/6000/2$ 个币。说明 BTC 升值了，把钱兑换回来。

如果 BTC 的价格下跌至 ¥4000，即 BTC 市值小于账户余额，并且其之间的差超过设定的阈值，就买入 $(5000-4000)/4000/2$ 个币。说明 BTC 贬值了，把 BTC 买回来。

就这样，不管 BTC 是升值还是贬值，始终动态保持账户余额和 BTC 的市值相等。如果 BTC 贬值了就买一些，等再涨回来，就再卖一些，就好像天平一样。

买入条件： 如果当前持仓市值减去当前可用余额小于负当前可用余额 5%，就开仓买入。

卖出条件： 如果当前持仓市值减去当前可用余额大于当前可用余额的 5%，就平

仓卖出。

前提必要条件

- 当前行情
- 当前资产
- 币总市值
- 资产差

策略构建

可视化编写策略第 1 步

我们把交易策略的 4 个前提必要条件加以计算，并分别赋值给各自变量。以可视化编程，代码块是这样的。如下图



图 3-42

需要注意的是，币总市值也就是当前持仓币数的总市值，其计算方法就是，当前持仓总币数乘以当前的最新价格。资产差也就是币总市值减去当前的可用余额。

可视化编写策略第 2 步

前提必要条件赋值完成后，就需要写交易逻辑了。这个也没有想象中那么复杂。无非就是把上述的策略逻辑，用代码块的形式表现出来。

即如果资产差小于负可用余额的 5%就买入，如果资产差大于可用余额的 5%就卖出。如下图：



图 3-43

整个策略似乎已经写完，但是要知道，程序是从上到下执行的，执行完之后就停止了。但是我们的交易策略并不是把交易条件执行一次就行，而是循环往复的重复执行。

也就是说，程序需要不断的检查策略条件是否已经达成，如果是就执行买卖，否则就一直检查下去。这个时候就需要用到另一个循环语句，如下图：



图 3-44

策略回测

可视化策略跟其他编程语言写的策略没什么本质区别，同样支持多种周期、做种精度的历史数据测试，当然也支持国内外商品期货和数字货币实盘交易。以下是该策略的回测信息：

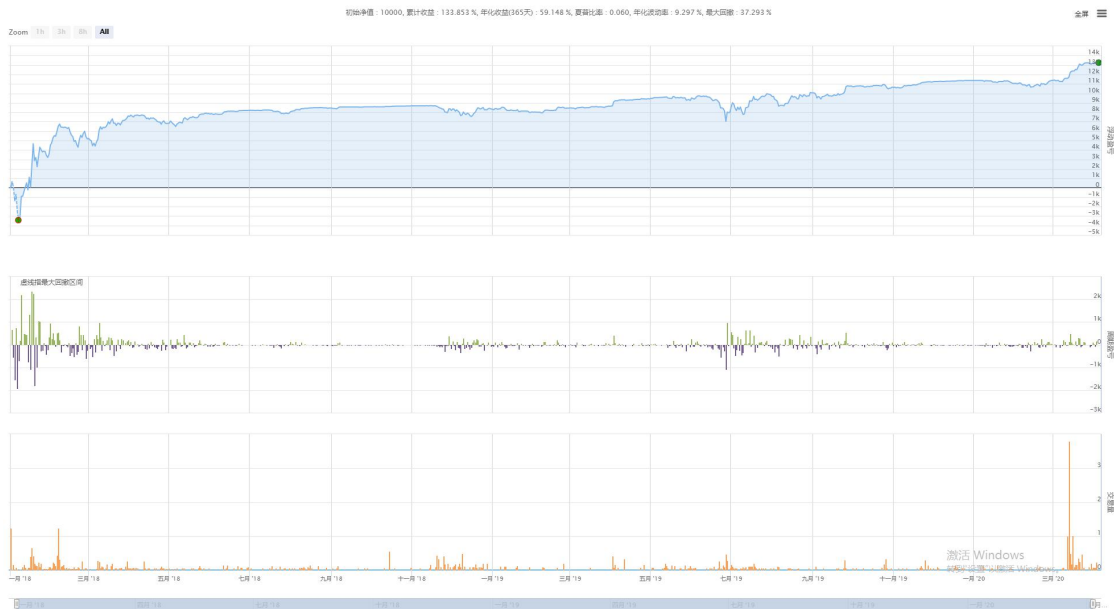


图 3-45

至此，一个完整的交易策略才算完成。为了照顾伸手党，本篇策略已经分享到策略广场中，可以直接复制研究。

结尾

一万小时定律始终存在，但是对于零基础的交易者来说，不可能花一万个小时再入行。所以你必须有一把梯子，而对于零编程基础的交易者来说，发明者量化的可视化编程就是一把快速入门的梯子。

利用可视化编程，你无需记住语法和方法名称，只需简单地浏览功能模块，从中找到你想要的即可。发明者量化的初衷也在于此，意在帮助更多的量化初学者降低准入门槛，提高量化兴趣，人人都可能成为量化交易者！

不过，话说回来，可视化编程作为量化入门的敲门砖是完全没有问题的，但也有自己的局限性，比如不能开发过于复杂、过于精细化的交易策略。但这并不影响你迈出量化交易的第一步！

下节预告

从量化交易的专业性来说，无论是麦语言还是可视化语言，都只是进入量化交易世界的过渡语言。它们的语言特点，也就决定了在量化交易策略开发中的局限性，一些复杂的策略不太可能实现出来。所以下节我们将带你学习 JavaScript，它是一门正式的高级编程语言，也是你进阶高级量化交易的必经之路。

课后习题

- 1、试着用可视化语言实现布林带指标。
- 2、试着用本节的交易模块，完成一个交易策略。

第四章 主流编程语言实现交易策略

4.1 JavaScript 语言快速入门

摘要

作为未来一名量化交易新星，你不可能只学习一门简单的语言就够了。发明者量化工具的麦语言和可视化语言，虽然能带你入门，但由于它们的语言特性，在策略开发中有很多局限性。所以，要想在量化交易中有立足之地，你必须学会一门正式的编程语言。

为什么要学习 JavaScript 语言

相比于可视化语言，JavaScript 语言有更强的性能和执行效率。并且在策略开发方面，JavaScript 语言要比可视化语言灵活许多，比如：你想开发一个套利策略，那么用可视化语言是不行的，因为它有限的模块，不支持类似套利的策略策略，而 JavaScript 语言就能轻松胜任。

另外，JavaScript 语言比可视化语言更加简洁和优雅，比如：可视化语言 10 行代码，用 JavaScript 可能 5 行就能写出来。从某些方面看，可视化语言仅仅是 JavaScript 的文字版，其代码的执行方式和逻辑与 JavaScript 几乎是一样的。如果你学会了可视化语言，那么学习 JavaScript 将是非常轻松的。

JavaScript 语言简介

JavaScript 是一门正式的高级编程语言。它既适合当作学习编程的入门语言，也适合当作日常开发的工作语言。它是目前最有希望、前途最光明的计算机语言之一，至今在浏览器端都有不可撼动的霸主地位。虽然它是作为开发 Web 页面而出名的，但是它也被用到了很多非浏览器环境中，例如：服务器、PC 端、移动端等，当然它还可以做量化交易！

完整策略

为了帮助大家快速理解本节的重点知识，在介绍发明者量化 JavaScript 语言快速入门之前，先对本节名词概念有个初步了解。我们就用最简单的双均线策略为例：

多头开仓：如果当前没有仓位，并且 5 周期均线大于 20 周期均线。

空头开仓：如果当前没有仓位，并且 5 周期均线小于 20 周期均线。
多头平仓：如果当前持有多单，并且 5 周期均线小于 20 周期均线。
空头平仓：如果当前持有空单，并且 5 周期均线大于 20 周期均线。

如果用 JavaScript 语言代码编写出来，就是这样的：

```
1 function main() {
2     $.CTA("rb000/rb888", function(st) { //设置数据合约rb指数，交易合约rb主力
3         var r = st.records //获取K线数组
4         var mp = st.position.amount //获取持仓数量(0:空仓, 1:多单, -1:空单)
5         if (r.length < 20) { //如果K线长度不够20根，无法计算均线
6             return //直接返回，等待下一根K线数据
7         }
8         var cross = _Cross(TA.MA(r, 5), TA.MA(r, 20)); //判断双均线交叉状态
9         if (mp <= 0 && cross > 2) { //如果金叉
10            return 1 * (mp < 0 ? 2 : 1) //如果有空单就反手，否则开仓做多
11        }
12        if (mp >= 0 && cross < -2) { //如果死叉
13            return -1 * (mp > 0 ? 2 : 1) //如果有多单就反手，否则开仓做空
14        }
15    })
16 }
```

图 4-1

上图中的代码就是用 JavaScript 语言写的一个完整的量化交易策略。可以实盘运行，并且自动下单交易。从代码量上看，该语言比可视化语言更简单些。整个策略的设计流程是：设置行情品种、获取 K 线数据、获取持仓信息、计算交易逻辑、下单买卖。

标识符

JavaScript 中的一切（变量、函数名和操作符）都区分大小写，也就是说变量名 test 和变量名 Test 是两个不同的变量。标识符（变量、函数、属性、函数参数的名字）的第一个字符必须是字母、下划线（_）、美元符（\$），后面的字符还可以是数字，如下图所示：

```
1 test = 1;  
2 Test = 10;  
3 test9 = test;  
4 _demo = demo();  
5 $Demo = demo(test9);
```

图 4-2

注释

注释包括单行注释和块级注释。单行注释以两个斜杠开头，块注释以一个斜杠和一个星号（*/）开头，以一个星号和一个斜杠（*/）结尾，如下图所示：

```
1 // 这是一个单行注释  
2  
3 /*  
4 * 这是一个多行  
5 *（块级）注释  
6 */
```

图 4-3

语句

每个语句都有一个分号结尾；虽然这不是必须的，但我们建议任何时候都不要省略它。加上分号，在某些情况下可以增加代码的性能，如下图所示：

```
1 a = 1; //语句以分号结束  
2 b(); //语句以分号结束  
3  
4 a = 1 //也可以省略分号，但是不建议  
5 b() //也可以省略分号，但是不建议
```

图 4-4

变量

变量可以保存任何类型的数据，创建变量的时候要使用 `var` 操作符，后面跟变量名。定义变量的时候同时还可以设置变量的值。一旦变量创建后，再次设置变量的值，就不用使用 `var` 操作符，如下图所示：

```
1 var name; //创建一个变量name
2 var myName = "发明者量化"; //创建一个变量myName，并设置myName的值为“发明者量化”
3 name = "google"; //设置name的值
4 myName = "FMZ" //设置myName的值
```

图 4-5

数据

JavaScript 一共有 5 种数据类型，分别是：未定义 (Undefined)、对象为空 (Null)、布尔值 (Boolean)、数字 (Number)、字符串 (String)，如下图所示：

```
1 str; //得到的值是：undefined，因为str没有定义
2 var str = null;
3 str; //得到的值是：null
4
5 var isTrue = false;
6 isTrue; //得到的值是：false
7
8 var num = -0.15;
9 num; //得到的值是：-0.15
10
11 var strs = "发明者量化 (FMZ) ";
12 strs; //得到的值是：发明者量化 (FMZ)
```

图 4-6

Undefined 只有一个值，即特殊的“undefined”，它代表一个还没有设置的值。比如我们只定义一个变量，不给这个变量设置值，那么该变量的值就是“undefined”。

Null 只有一个值，即特殊的“null”，它代表一个被设置为空的值。比如我们先创建一个变量，然后把变量的值设置为“null”，那么反问该变量返回的值就是“null”。

Boolean 有两个值，即“true”和“false”，“true”代表真，“false”代表

假。需要注意的是，“true”和“false”都是小写。

Number 也就是数字类型，包括：正数、负数、整数、小数等等。除此之外“NaN”也是一个特殊的数字，它专门表示未返回数值的情况，比如：1 除以 0，返回“NaN”。

String 你可以理解为文字，包含中文和英文，可以通过单引号或双引号来构造字符串。比如：“fmz”或者‘发明者量化’等。

对象

对象你可以理解为一个存放各种数据的容器，容器中属性和值都是对应的。可以通过 new 操作符先把这个容器创建出来。并且可以给创建后的对象添加属性和方法，如下图所示：

```
1 var obj = {}; //创建一个对象
2 obj.name = "google"; //给obj对象添加一个name属性，name的值是：“google”
3 obj.age = 19; //给obj对象添加一个age属性，name的值是：19
4
5 obj //获取obj对象，结果是：{name: "google", age: 19}
6 obj.name //获取obj的name属性，结果是：google
7
8 var str = "age"; //创建一个字符串
9 obj[str] //还可以通过变量的方式，获取obj的age属性，结果是：19
```

图 4-7

数组

数组也是一个存放各种数据的容器，只不过容器中的元素是从左往右有序排列的，第一位的元素是 0，第二位的元素是 1，以此类推。另外 JavaScript 的数组可以存放任何数据类型，如下图所示：

```
1 var arr = ["FMZ", 100, {name: "发明者量化"}, true]; //创建一个数组
2 arr[0]; //获取arr的第一个下标，结果是“FMZ”
3 arr[2].name; //获取arr的第二个下标对象的属性，结果是“发明者量化”
```

图 4-8

函数

JavaScript 中的函数跟我们中学学的函数没有本质的区别，你可以理解为传进去什么，通过函数的计算，输出什么，如下图所示：

```
1 //定义两个数字相加的函数
2 function add(num1, num2) {
3   return num1 + num2;
4 }
5
6 //使用这个函数，传入两个参数
7 add(1, 2); //返回的结果是3
```

图 4-9

运算符

JavaScript 有多种运算符，即算术运算符、比较运算符、逻辑运算符。其中算术运算符就是加减乘除的数学运算，比较运算符可以比较两个值是否小于或者小于，逻辑运算符主要有：逻辑与、逻辑或、逻辑非。如下图所示：

```
1 //算数运算符
2 var x = 5;
3 var y = 2;
4 var z1 = x + y; //在以上语句执行后，z1 的值是：7
5 var z2 = x - y; //在以上语句执行后，z2 的值是：3
6 var z3 = x * y; //在以上语句执行后，z3 的值是：10
7 var z4 = x / y; //在以上语句执行后，z4 的值是：2.5
8
9 //比较运算符
10 x > y; //结果是：true
11 x < y; //结果是：false
12 x != y; //结果是：true
13 x == y; //结果是：false
14
15 //逻辑运算符
16 x > y && x > y && x > y; //结果是：true
17 x < y && x > y && x > y; //结果是：false
18 x < y || x < y || x > y; //结果是：true
19 !(x==y); //结果是：true
```

图 4-10

需要注意的是：“&&”是逻辑与，代表“并且”的意思。“||”是逻辑或，代表

“或者”的意思。“!”是逻辑非，代表“否”的意思：

“&&”是所有条件都为“true”的时候，最终条件才为“true”；

“||”是所有条件中，只要有任何一个条件为“true”，最终条件就为“true”。

优先级

如果有一个 $100*(10-1)/(10+5)$ 表达式，程序是先计算哪一步？中学数学告诉我们：①如果是同一级运算，一般按从左往右依次进行计算。②如果既有加减、又有乘除法，先算乘除法、再算加减。③如果有括号，先算括号里面的。④如果符合运算定律，可以利用运算定律进行简算。JavaScript 语言的优先级也是如此，如下图：

```
1 var num = 100*(10-1)/(10+5);
2 num; //计算结果是: 60
3
4 1 > 2 && (2 > 3 || 3 < 5); //运算结果是: false
5 1 > 2 && 2 > 3 || 3 < 5; //运算结果是: true
```

图 4-11

条件语句

通常在写代码时，您总是需要为不同的决定来执行不同的动作。您可以在代码中使用条件语句来完成该任务。在 JavaScript 中，我们可使用以下条件语句：

if 语句 - 只有当指定条件为 true 时，使用该语句来执行代码

if...else 语句 - 当条件为 true 时执行代码，当条件为 false 时执行其他代码

if...else if...else 语句- 使用该语句来选择多个代码块之一来执行

switch 语句 - 使用该语句来选择多个代码块之一来执行

if 语句

只有当指定条件为 true 时，该语句才会执行代码。请使用小写的 if。使用大写字母 (IF) 会生成 JavaScript 错误！如下图所示：

```
1 //语法
2 if (condition) {
3     //当条件为 true 时执行的代码
4 }
5
6 //例子
7 if (time<20) {
8     x="Good day"; //当时间小于 20:00 时, 生成问候 "Good day"
9 }
```

图 4-12

if...else 语句

当条件为 true 时执行代码，当条件为 false 时执行其他代码，如下图所示：

```
1 //语法
2 if (condition) {
3     //当条件为 true 时执行的代码
4 } else {
5     //当条件不为 true 时执行的代码
6 }
7
8 //例子
9 if (time<20) { //如果当时间小于20:00
10     x="Good day"; //生成问候 "Good day"
11 } else { //否则
12     x="Good evening"; //生成问候 "Good evening"
13 }
```

图 4-13

for 循环

有时候我们需要获取最近几天的 K 线数据，就需要从 K 线数组中，根据 K 线数据的位置依次获取，那么使用 for 循环是很方便的，如下图所示：

```
1 function main() {
2     exchange.SetContractType("MA888"); //设置合约
3     var barArr = exchange.GetRecords(); //获取K线数组
4
5     //不使用for循环的写法
6     Log(barArr[barArr.length - 1]); //获取K线数组倒数第1根K线数据, 并输出到日志中
7     Log(barArr[barArr.length - 2]); //获取K线数组倒数第2根K线数据, 并输出到日志中
8     Log(barArr[barArr.length - 3]); //获取K线数组倒数第3根K线数据, 并输出到日志中
9     Log(barArr[barArr.length - 4]); //获取K线数组倒数第4根K线数据, 并输出到日志中
10    Log(barArr[barArr.length - 5]); //获取K线数组倒数第5根K线数据, 并输出到日志中
11
12    //使用for循环的写法
13    for (var i = 1; i < 6; i++) {
14        Log(barArr[barArr.length - i]); //依次倒数获取K线数组中的数据
15    }
16 }
```

图 4-14

while 循环

我们都知道行情是在不断变化的, 如果你想获取最新的 K 线数组, 就得不断的去一遍又一遍地运行相同的代码, 那么使用 while 循环, 只要指定条件为 true, 循环就可以一直获取最新的 K 线数组。

```
1 function main() {
2     exchange.SetContractType("MA888"); //设置合约
3     while(true) {
4         Log(exchange.GetRecords()); //获取K线数组
5     }
6 }
```

图 4-15

break 语句和 continue 语句

循环是有前提条件的, 只有这个前提条件为“true”的时候, 循环才会开始重复的做某些事, 直到这个前提条件为“false”的时候, 循环才会结束。但是 break 语句可以在循环执行的过程中立刻跳出循环; continue 语句可以中断某一次循

环，然后继续下一次循环。如下图所示：

```
1 //break语句
2 var arr = [];
3 for (i = 0; i < 10; i++) {
4     if (i == 9) {
5         break;
6     }
7     arr.push(i);
8 }
9 console.log(arr); //结果是: [0, 1, 2, 3, 4, 5, 6, 7, 8]
10
11 //continue语句
12 var arr = [];
13 for (i = 0; i < 10; i++) {
14     if (i == 3) {
15         continue;
16     }
17     arr.push(i);
18 }
19 console.log(arr); //结果是: [0, 1, 2, 4, 5, 6, 7, 8, 9]
```

图 4-16

return 语句

return 语句会终止函数的执行并返回函数的值。return 语句只能出现在函数体内，出现在代码中的其他任何地方都会造成语法错误！

```
1 //创建一个函数，这个函数计算2个参数的乘积
2 function myFunction(a, b) {
3     return a * b; //函数计算2个参数的乘积并返回
4 }
5
6 var x = myFunction(10, 10); //调用函数，函数的返回值是100，并把100设置给变量x
```

图 4-17

CTA 策略架构

在发明者量化工具中，如果用 JavaScript 语言编写策略将会非常方便，官方内置了一套标准策略框架，如下图所示：

```
1 function main() {  
2     $.CTA("商品期货品种代码", function(st) {  
3         //中间可以写你的策略逻辑  
4     })  
5 }
```

图 4-18

如上图中的代码，这是一个标准的策略框架，除了可以更改“商品期货品种代码”外，其他都是固定的格式，使用框架编写策略的最大好处是，你只需要编写策略逻辑就行了，其他的行情获取、下单处理等等一系列问题，都由框架处理。这样可以是你专注于策略开发。

总结

以上就是 JavaScript 语言快速入门的内容，学习后就可以编程量化交易策略啦。如果需要编写更加复杂的策略，可以参考发明者量化工具 JavaScript 语言 API 文档，或者直接咨询官方客服代写量化交易策略。

下节预告

日内交易也是一种交易模式，这种方式不留仓过夜，所以受市场波动率风险较低，一旦出现不利行情，可以及时进行调整。学习了本节 JavaScript 语言入门，下节我们将带大家手把手编写一个可行的日内量化交易策略。

课后习题

- 1、试着用发明者量化工具中的 JavaScript 语言获取历史 K 线数据。
- 2、试着写下本节开头的策略代码，并写上注释。

4.2 如何用 JavaScript 语言实现策略交易

摘要

在上一篇中，我们从 JavaScript 语言的简介、基础语法、CTA 策略框架等方面为大家讲解实现交易策略的前提部分，本篇我们将继续上篇内容，从常用的策略模块、技术指标，一步一步帮助大家实现一个可行的日内量化交易策略。

策略简介

布林带也称为布林通道，英文简称 BOLL。它是最常用的技术指标之一，由约翰·包宁杰（John Bollinger）在 1980 年代发明。理论上，价格总是围绕着价值在一定范围内上下波动，布林带正是根据这个理论基础，又引入了“价格通道”的概念。

其计算方式是利用统计学原理，先计算一段时间价格的“标准差”，再由均线加/减 2 倍的标准差，求出价格的“信赖区间”。其基本的型态是由三条轨道线组成的带状通道（中轨、上轨、下轨）。中轨为价格的平均成本，上轨和下轨分别代表价格的压力线和支撑线。

由于采用了标准差的概念，使得布林通道的宽度会根据近期价格的波动而做出动态调整。波动小，布林通道会变窄；波动大，布林通道会变宽。当 BOLL 通道由宽变窄，说明价格逐渐向均值回归。当 BOLL 通道由窄变宽，意味着行情开始发生变化，如果价格上穿上轨，表明买力增强，如果价格下穿下轨，表明卖力增强。

布林带指标计算方法

在所有的技术指标中，布林带的计算方法是最复杂之一，其中引进了统计学中的标准差概念，涉及到中轨线（MB）、上轨线（UP）和下轨线（DN）的计算。其计算方法如下：

中轨 = N 时间段的简单移动平均线

上轨 = 中轨 + $K \times N$ 时间段的标准差

下轨 = 中轨 - $K \times N$ 时间段的标准差

```
1 function main() { //程序入口
2     while (true) { //进入循环
3         exchange.SetContractType('MA888'); //设置合约
4         var records = exchange.GetRecords(); //获取K线数组
5         var boll = TA.BOLL(records, 50); //获取50周期BOLL指标数组
6         var top = boll[0]; //获取BOLL指标上轨数组
7         var ma = boll[1]; //获取BOLL指标中轨数组
8         var bottom = boll[2]; //获取BOLL指标下轨数组
9         Log(top); //把BOLL指标上轨数组打印到日志中
10        Log(ma); //把BOLL指标中轨数组打印到日志中
11        Log(bottom); //把BOLL指标下轨数组打印到日志中
12    }
13 }
```

图 4-19

策略逻辑

布林线的使用方法有很多，可以单独使用，也可以和其他指标结合在一起使用。本节教程我们将采用布林线一种最简单的使用方法。即：当价格自下而上突破上轨，即突破上方压力线时，我们认为多方力量正在走强，一波上涨行情已经形成，买入开仓信号产生；当价格自上而下跌破下轨，即跌破支撑线时，我们认为空方力量正在走强，一波下跌趋势已经形成，卖出开仓信号产生。



图 4-20

如果买入开仓后，价格又重新跌回到了布林线中轨，我们认为多方力量正在走弱，

或者空方力量正在加强，卖出平仓信号产生；如果卖出开仓后，价格又重新涨回到布林线中轨，我们认为空方力量正在走弱，或者多方力量正在加强，买入平仓信号产生。

买卖条件

多头开仓：如果无持仓，并且收盘价大于上轨，并且时间非 14:45

空头开仓：如果无持仓，并且收盘价小于下轨，并且时间非 14:45

多头平仓：如果持多单，并且收盘价小于中轨，或者时间是 14:45

空头平仓：如果持空单，并且收盘价大于中轨，或者时间是 14:45

策略代码实现

要想实现策略，首先需要考虑我们需要什么数据？通过什么 API 去获取？然后如何计算交易逻辑？最后通过哪些方式下单方式去交易？接下来，让我们一步一步来实现吧：

第一步：使用 CTA 策略框架

所谓的 CTA 策略框架是由发明者量化官方推出的一套标准框架，使用该框架可以不必考虑开发量化交易策略的琐碎问题，直接把精力放在编程交易逻辑上。比如，如果不使用该框架的话，在下单的时候，需要考虑换月移仓、下单买卖价格、下单不成交时撤单或追单等等问题.....

```
1 function main() {
2     $.CTA("rb000/rb888", function(st) {
3         //编写策略
4     })
5 }
```

图 4-21

上图就是使用发明者量化工具的 CTA 策略框架。这是一个固定的代码格式，所有的交易逻辑代码从第 3 行开始编写。在使用中除了需要修改品种代码外（浅黄色），其他地方不用任何修改。

需要注意的是，上图中的品种代码是“rb000/rb888”它代表的意思是信号的数据使用的是“rb000”，交易数据使用的是“rb888”，并且自动换月移仓。当然

你也可以指定具体的品种代码，比如把品种代码“rb1910”，就是信号数据和交易数据都使用“rb1910”。

第二步：获取各种数据

仔细想下，都需要哪些数据呢？从我们的策略交易逻辑中发现：首先需要获取当前的持仓状态，然后比较收盘价与布林带指标上中下轨的相互关系，最后判断行情是不是即将收盘。那么接下来让我们一以获取这些数据吧。

获取 K 线数据

首先就是获取 K 线数组和上根 K 线收盘价，因为有了 K 线数组，才能计算出布林带指标。用代码写出来是这样的：

```
1 function main() {
2   $.CTA("rb000/rb888", function(st) {
3
4     var r = st.records; //获取K线数组
5     if (r.length < 20) return; //过滤K线的长度
6     var close = r[r.length - 2].Close; //获取上根K线收盘价
7
8   })
9 }
```

图 4-22

如上图所示：

第 4 行：获取 K 线数组，这是一个固定的格式。

第 5 行：过滤 K 线的长度，因为我们计算布林带指标用的参数是 20，当 K 线小于 20 根的时候，是无法计算布林带指标的。所以这里要过滤下 K 线的长度，如果 K 线少于 20 根，就直接返回，继续等待下一根 K 线。

第 6 行：从获取的 K 线数组中，先获取上根 K 线的对象，再从该对象中获取收盘价。获取一个数组的倒数第二个元素，就是这个数组的长度减 2（`r[r.length - 2]`）；K 线数组里面的元素都是一个个对象，对象包含了开盘价、最高价、最低价、收盘价、成交量、时间，要获取收盘价就直接在后面加上“.”和属性名就可以了（`r[r.length - 2].Close`）。

获取 K 线时间数据

因为我们是日内策略，需要在收盘前平掉仓位，所以要判断当前 K 线是不是临近

收盘,如果是临近收盘的 K 线就平掉仓位,如果不是临近收盘的 K 线就可以开仓,用代码写出来是这样的:

```
1 function main() {
2   $.CTA("rb000/rb888", function(st) {
3
4     var r = st.records; //获取K线数组
5     if (r.length < 20) return; //过滤K线的长度
6     var close = r[r.length - 2].Close; //获取上根K线收盘价
7
8     var time = new Date(r[r.length - 1].Time); //根据当根K线时间戳,创建一个时间对象
9     var isClose = time.getHours() == 14 && time.getMinutes() == 45; //判断时间是否是14:45
10
11   })
12 }
```

图 4-23

如上图所示:

第 8 行: 获取当根 K 线的时间戳属性, 然后创建一个时间对象 (new Date(时间戳))。

第 9 行: 根据时间对象, 分别计算小时和分钟数, 并判断当根 K 线的时间是不是 14:45。

获取持仓数据

持仓信息是量化交易策略中一个很重要的条件, 当交易条件成立时, 还需要通过持仓状态和持仓数, 来判断是否下单。比如: 当买入开仓交易条件成立时, 如果有持仓, 就不必在重复下单了; 如果无持仓, 就可以下单。用代码写出来是这样的:

```
1 function main() {
2   $.CTA("rb000/rb888", function(st) {
3
4     var r = st.records; //获取K线数组
5     if (r.length < 20) return; //过滤K线的长度
6     var close = r[r.length - 2].Close; //获取上根K线收盘价
7
8     var time = new Date(r[r.length - 1].Time); //根据当根K线时间戳,创建一个时间对象
9     var isClose = time.getHours() == 14 && time.getMinutes() == 45; //判断时间是否是14:45
10
11     var mp = st.position.amount; //获取持仓信息
12
13   })
14 }
```

图 4-24

如上图所示：

第 11 行：获取当前的持仓状态。如果有多单，则值是 1；如果有空单，则值是 -1；如果无持仓，则值是 0。

获取布林带数据

接着就需要计算布林带指标上轨、中轨、下轨的数值了。那就要先获取布林带数组，再从数组中获取上中下轨的数值。在发明者量化工具中，获取布林带数组还很简单，直接调用布林带的 API 就可以了，难的是获取上中下轨的数值，因为布林带数组是一个二维数组。

二维数组其实很好理解，它就是数组中的数组，那么获取的顺序就是：先获取数组中指定的数组，然后在从指定的数组中获取指定的元素，如下图所示：

```
1 var arr = [[100, 200, 300], [10, 20, 30], [1, 2, 3]]; //这是一个二维数组
2 var test = arr[0]; //先获取这个二维数组里面的第一个数组，并把结果设置给test
3 var demo1 = test[0]; //再从test数组中，获取第一个数值
4 demo1; //结果是：100
5
6 var demo2 = arr[0][0]; //当然也可以这么写
7 demo2; //结果同样也是：100
```

图 4-25

如下图，第 13 行~19 行就是用代码获取布林带上轨、中轨、下轨的数值。其中第 13 行是直接使用发明者量化工具的 API，直接获取布林带数组；第 14 行~16 行是先分别获取二维数组中的上轨数组、中轨数组、下轨数组；第 17 行~19 行是分别从上轨数组、中轨数组、下轨数组中获取上根 K 线的布林带上轨、中轨、下轨数值。

```
1 function main() {
2   $.CTA("rb000/rb888", function(st) {
3
4     var r = st.records; //获取K线数组
5     if (r.length < 20) return; //过滤K线的长度
6     var close = r[r.length - 2].Close; //获取上根K线收盘价
7
8     var time = new Date(r[r.length - 1].Time); //根据当根K线时间戳，创建一个时间对象
9     var isClose = time.getHours() == 14 && time.getMinutes() == 45; //判断时间是否是14:45
10
11    var mp = st.position.amount; //获取持仓信息
12
13    var boll = TA.BOLL(r, 20, 2); //计算布林带指标
14    var upLine = boll[0]; //获取上轨数组
15    var midLine = boll[1]; //获取中轨数组
16    var downLine = boll[2]; //获取下轨数组
17    var upPrice = upLine[upLine.length - 2]; //获取上根K线上轨数值
18    var midPrice = midLine[midLine.length - 2]; //获取上根K线中轨数值
19    var downPrice = downLine[downLine.length - 2]; //获取上根K线下轨数值
20
21  })
22 }
```

图 4-26

第三步：下单交易

有了以上数据，就可以编写交易逻辑以及下单交易的代码了。格式也非常简单，最常用到的是“if 语句”，用文字可以描述为：如果条件 1 和条件 2 成立，下单；如果条件 3 或条件 4 成立，下单。如下图所示：

```
1 function main() {
2   $.CTA("rb000/rb888", function(st) {
3
4     var r = st.records; //获取K线数组
5     if (r.length < 20) return; //过滤K线的长度
6     var close = r[r.length - 2].Close; //获取上根K线收盘价
7
8     var time = new Date(r[r.length - 1].Time); //根据当前K线时间戳, 创建一个时间对象
9     var isClose = time.getHours() == 14 && time.getMinutes() == 45; //判断时间是否是14:45
10
11    var mp = st.position.amount; //获取持仓信息
12
13    var boll = TA.BOLL(r, 20, 2); //计算布林带指标
14    var upLine = boll[0]; //获取上轨数组
15    var midLine = boll[1]; //获取中轨数组
16    var downLine = boll[2]; //获取下轨数组
17    var upPrice = upLine[upLine.length - 2]; //获取上根K线上轨数值
18    var midPrice = midLine[midLine.length - 2]; //获取上根K线中轨数值
19    var downPrice = downLine[downLine.length - 2]; //获取上根K线下轨数值
20
21    if (mp == 1 && (close < midPrice || isClose)) return -1; //如果持多单, 并且收盘价小于中轨, 或者时间是14:45, 平多
22    if (mp == -1 && (close > midPrice || isClose)) return 1; //如果持空单, 并且收盘价大于中轨, 或者时间是14:45, 平空
23    if (mp == 0 && close > upPrice && !isClose) return 1; //如果无持仓, 并且收盘价大于上轨, 并且时间非14:45, 开多
24    if (mp == 0 && close < downPrice && !isClose) return -1; //如果无持仓, 并且收盘价小于下轨, 并且时间非14:45, 开空
25
26  })
27 }
```

图 4-27

上图中, 第 21 行~24 行就是交易逻辑以及下单交易的代码。从上往下分别是: 平多、平空、开多、开空。

以开多单 (第 23 行) 为例, 这是一个 “if 语句”, 在该语句中如果只执行一行代码, 花括号 “{}” 是可以省略的。该语句翻译成文字是: 如果当前持仓是 0, 并且收盘价大于上轨, 并且 K 线时间不是 14:45, 就 “return 1”

细心的你可能会发现, 这几行有 “return 1” 和 “return -1”, 这是一个固定的格式, 意思就是: 如果是买入的就写 “return 1”; 如果是卖出的就写 “return -1”。开多和平空都是买入, 所以写 “return 1”; 开空和平多都是卖出, 所以写 “return -1”。

完整的策略代码

至此一个完整的策略代码就写完了, 如果把交易框架、交易数据、交易逻辑、下单买卖等分开来写是不是很简单呢, 以下就是这个策略的整个代码:

```
1 function main() {
2   $.CTA("rb000/rb888", function(st) {
3     var r = st.records; // 获取K线数组
4     if (r.length < 20) return; // 过滤K线的长度
5     var close = r[r.length - 2].Close; // 获取上根K线收盘价
6     var time = new Date(r[r.length - 1].Time); // 根据当根K线时间戳, 创建一个时间对象
7     var isClose = time.getHours() == 14 && time.getMinutes() == 45; // 判断时间是否是14:45
8     var mp = st.position.amount; // 获取持仓信息
9     var boll = TA.BOLL(r, 20, 2); // 计算布林带指标
10    var upLine = boll[0]; // 获取上轨数组
11    var midLine = boll[1]; // 获取中轨数组
12    var downLine = boll[2]; // 获取下轨数组
13    var upPrice = upLine[upLine.length - 2]; // 获取上根K线上轨数值
14    var midPrice = midLine[midLine.length - 2]; // 获取上根K线中轨数值
15    var downPrice = downLine[downLine.length - 2]; // 获取上根K线下轨数值
16    if (mp == 1 && (close < midPrice || isClose)) return -1; // 如果持多单, 并且收盘价小于中轨, 或者时间是14:45, 平多
17    if (mp == -1 && (close > midPrice || isClose)) return 1; // 如果持空单, 并且收盘价大于中轨, 或者时间是14:45, 平空
18    if (mp == 0 && close > upPrice && !isClose) return 1; // 若无持仓, 并且收盘价大于上轨, 并且时间非14:45, 开多
19    if (mp == 0 && close < downPrice && !isClose) return -1; // 若无持仓, 并且收盘价小于下轨, 并且时间非14:45, 开空
20  })
21 }
```

图 4-28

有两个地方需要注意：尽量（但不是必须）把策略逻辑写成当根 K 线条件成立，下根 K 线发单，或者上根 K 线条件成立，当根 K 线发单，这样回测的结果与实盘的结果相差不大。不这样写也可以，但是要注意策略逻辑是否正确。一般而言，把平仓的逻辑写在开仓逻辑的前面，这样做的目的是，尽量让策略逻辑符合你的预期。比如：如果策略逻辑刚好赶上反手的时候，反手的规则是，先平仓再开新仓。而不是先开新仓，再平仓。如果我们直接把平仓逻辑写到开仓逻辑前面，就不会出现这种问题。

总结

以上我们学习了开发一个完整的日内量化交易策略的每个步骤，包括：策略简介、布林带指标计算方法、策略逻辑、买卖条件、策略代码实现等。通过这个策略案例，不仅熟悉发明者量化工具的编程方法，还可以根据这个模板改编成不同的策略。

量化交易策略无非是主观交易经验或系统的总结，如果我们在写策略之前，把主观交易中用到的经验或系统，分别写出来，然后再一条一条翻译成代码，你会发现写策略就会容易很多。试试吧！

下节预告

在量化交易策略开发中，如果只能选用一种编程语言的话，那么毫不犹豫，一定是选择 Python，从数据获取到策略回测再到交易，Python 已经覆盖了整个业务链。在金融量化投资领域占据了重要位置，下节课我们将学习 Python 语言入

门知识。

课后习题

- 1、试着用本节的知识，动手实现一个双均线策略。
- 2、试着用发明者量化工具中的 JavaScript 语言实现 KDJ 指标算法。

4.3 Python 语言快速入门

摘要

在量化交易策略开发中，如果只能选用一种编程语言的话，那么毫不犹豫，一定是选择 Python，从数据获取到策略回测再到交易，Python 已经覆盖了整个业务链。在金融量化投资领域占据了重要位置，本节课程我们将学习 Python 语言入门知识。

为什么要学习这么多编程语言

回顾之前的课程，一路走来，我们一共学习了：麦语言、可视化语言、JavaScript 语言，包括本节要学习的 Python 语言。可能有小伙伴会有疑问，我是来学习量化交易的，为什么要学习这么多编程语言呢？

其实，每个编程语言都有自己的语言特性，这些语言也没有孰优孰劣之分，更多的是要看策略更适合哪一种编程语言，以及这个编程语言是否符合你自己。所以有句话讲，只有亲自试过才知道。这也就是我们铺开这么多的篇幅，讲编程语言的原因，工欲善其事，必先利其器。

同时我们也致力于为大家打开量化的大门，普及各种编程语言知识，量化并没有我们想象中的高深莫测和遥不可及，相信未来量化会普及并平民化。

量化交易为什么要选择 Python

量化交易的流程无非是获取数据、分析计算数据、处理数据等，在数据分析方面，没有其他语言能像 Python 一样，既精于计算又保持性能。特别是在时间序列分析数据（K 线就是时间序列数据）处理，Python 有更加简单便捷的优势。另外，比起其他编程语言，Python 更简洁易学，阅读好的 Python 程序感觉就像阅读英语。

选择 Python 的五大原因

1. 量化运用广泛：

美国的 Quantopian 与国内的发明者量化都可以用 Python 语言。

2. 简单易学：

Python 的设计哲学是以用户为中心，属于方便调试的解释型语言。

3. 免费开源：

无使用成本，开源代码共享，加强学习与使用效率。

4. 丰富的库：

数据处理、数据运算、可视化、统计分析、技术分析、机器学习...

5. 应用接口：

各大平台数据获取存储调用与实时行情链接下单的接口。

完整策略

为了帮助大家快速理解本节的重点知识，在介绍发明者量化 JavaScript 语言快速入门之前，先对本节名词概念有个初步了解。我们就用最简单的双均线策略为例：

多头开仓： 如果当前没有仓位，并且 5 周期均线大于 20 周期均线。

空头开仓： 如果当前没有仓位，并且 5 周期均线小于 20 周期均线。

多头平仓： 如果当前持有多单，并且 5 周期均线小于 20 周期均线。

空头平仓： 如果当前持有空单，并且 5 周期均线大于 20 周期均线。

如果用 Python 语言代码编写出来，就是这样的：

```
1 name = 'rb1910' #合约代码
2 unit = 1 #下单数量
3
4 # 获取持仓信息函数
5 def mp():
6     positions = exchange.GetPosition() #获取持仓数组
7     if len(positions) == 0: #如果持仓数组的长度是0
8         return 0 #证明是空仓, 返回0
9     for i in range(len(positions)): #遍历持仓数组
10        if (positions[i]['Type'] == PD_LONG) or (positions[i]['Type'] == PD_LONG_YD):
11            return 1 #如果有多单, 返回1
12        elif (positions[i]['Type'] == PD_SHORT) or (positions[i]['Type'] == PD_SHORT_YD):
13            return -1 #如果有空单, 返回-1
14
15 def main():
16     while true:
17         exchange.SetContractType(name) #设置品种代码
18         records=exchange.GetRecords() #获取K线数组
19         ma5 = TA.MA(records, 5) #计算5周期均线
20         ma20 = TA.MA(records, 20) #计算20周期均线
21         isCross = _Cross(ma5, ma20) #判断均线是否交叉
22         positions = mp() #获取持仓信息函数
23         obj = ext.NewPositionManager() #使用交易类库
24         if positions > 0 and isCross < -1: #如果持多单, 并且双均线死叉
25             obj.CoverAll() #平掉所有仓位
26         if positions < 0 and isCross > 1: #如果持空单, 并且双均线金叉
27             obj.CoverAll() #平掉所有仓位
28         if positions == 0: #如果是空仓
29             if isCross > 1: #如果双均线金叉
30                 obj.OpenLong(name, unit) #买开
31             elif isCross < -1: #如果双均线死叉
32                 obj.OpenShort(name, unit) #卖开
```

图 4-29

上图中的代码就是用 Python 语言写的一个完整的量化交易策略。可以实盘运行，并且自动下单交易。从代码量上看，Python 语言比 JavaScript 语言更多一些，那是因为我们并没有使用 CTA 交易框架。

但是整个策略的设计流程几乎是一样是：设置行情品种、获取 K 线数据、获取持仓信息、计算交易逻辑、下单买卖。也就是说，虽然编程语法是不一样的，但写出来的策略逻辑是一样的，那么接下来，让我们学习 Python 的基础语法吧！

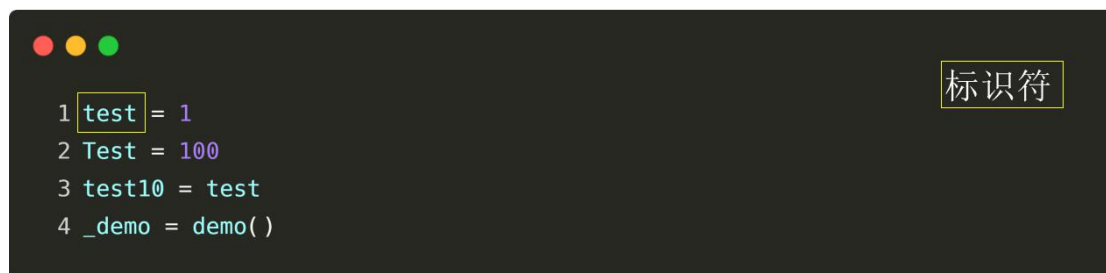
版本选择

Python 有两个版本，即：Python2 和 Python3，曾经有一个段子，说 Python 就像一个双管枪，但是每次只能用一个管发射子弹，但永远也不知道究竟哪个更准。所以如果你是 Python 新手，建议直接学习 Python3，因为它是最新的，并且 Python

社区一直在维护。我们的课程也是用 Python3 讲解的。

标识符

标识符也就是变量的名字，如下图的 `test`、`Test`、`test10`、`_demo` 等。Python 中的一切（变量、函数名和操作符）都区分大小写，也就是说变量名 `test` 和变量名 `Test` 是两个不同的变量。标识符（变量、函数、属性、函数参数的名字）的第一个字符必须是字母、下划线（`_`），后面的字符还可以是数字，如下图所示：

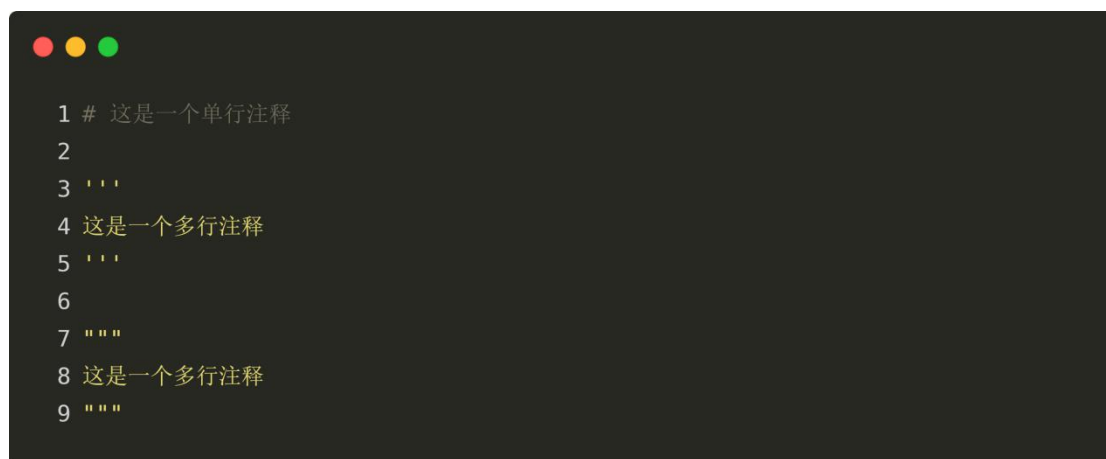


```
1 test = 1
2 Test = 100
3 test10 = test
4 _demo = demo()
```

图 4-30

注释

注释就是对一行代码的翻译或者解释，其规则非常简单，释包括单行注释和块级注释。单行注释以井号（`#`）开头，块注释以三个单引号（`'''`）或以三个双引号（`"""`）开头，以三个单引号（`'''`）或以三个双引号（`"""`）结尾，如下图所示：



```
1 # 这是一个单行注释
2
3 '''
4 这是一个多行注释
5 '''
6
7 """
8 这是一个多行注释
9 """
```

图 4-31

行与缩进

Python 最具特色的就是使用缩进来表示代码块，不需要使用大括号 `{}`。缩进

的空格数是可变的，但是同一个代码块的语句必须包含相同的缩进空格数。如下图：在这个案例中，程序会报错。就算 if 条件是成立的，也不会输出“True”，因为 Python 在代码运行之前，会自动检测代码语法是否正确，如果代码格式错误，程序将不会运行。原因是第 5 行代码并没有统一代码的缩进格式。四个空格缩进是 Python 的固定格式，需要大家多熟悉。

```
1 if True:
2     Log("True")
3 else:
4     Log("False")
5     Log("True")
```

图 4-32

变量

变量可以保存任何类型的数据，直接写变量的名字就是创建变量，但是当创建变量的时候需要同时设置变量的值，否则程序会报错。等号 (=) 运算符左边是一个变量名，等号 (=) 运算符右边是存储在变量中的值。如下图所示：name2 就是变量名，“发明者量化”就是变量的值。如果不重新给 name2 设置新的值，那么 name2 的值始终是“发明者量化”。

```
1 name1
2 name2 = "发明者量化"
3
4 Log(name1) #报错
5 Log(name2) #结果是：发明者量化
```

图 4-33

数据

Python 有六个数据类型，其中有 3 个不可变数据，和 3 个可变数据。顾名思义不可变数据一旦创建完毕，它的值是不能被改变的，在内存中的地址是唯一的；可变数据是内存中地址的引用，如果它的值改变，其内存地址不变。不可变数据（3 个）：Number（数字）、String（字符串）、Tuple（元组）；可变数据（3 个）：List（列表）、Dictionary（字典）、Set（集合）。

```
1 num = 100 #数字
2 name2 = "发明者量化" #字符串
3 tuples = ( 'abcd', 786 , 2.23, 'FMZ', 70.2 ) #元组
4
5 lists = [ 'abcd', 786 , 2.23, 'FMZ', 70.2 ] #列表
6 tinydict = {'name': 'FMZ','code':1, 'site': 'www.fmz.com'} #字典
7 student = {'Tom', 'Jim', 'Mary', 'Tom', 'Jack', 'Rose'} #集合
```

图 4-34

数字

Python 的数字类型支持 int（整型）、float（浮点型）、bool（布尔型）、complex（复数）。内置的 type() 函数可以用来查询变量所指的对象类型。如下图：

```
1 a, b, c, d = 20, 5.5, True, 4+3j #同时为多个变量赋值
2 Log((type(a), type(b), type(c), type(d))) #打印信息到日志中
3 #结果是: (<type 'int'>, <type 'float'>, <type 'bool'>, <type 'complex'>)
```

图 4-35

运算符

像大多数语言一样，Python 的数学运算都是很直观的。无论是算术运算符、比较运算符还是逻辑运算符都与我们在学校里面的知识一样。其中算术运算符就是加减乘除的数学运算，比较运算符可以比较两个值是否小于或者大于，逻辑运算符主要有：逻辑与、逻辑或、逻辑非。【交易策略中有哪些常用的字符串可以简单说一下】比如我们在交易策略中，最常用到的字符串就是品种代码了，比如：“rb1910”、“MA1910”。

```
1 #算数运算符
2 x = 5
3 y = 2
4 z1 = x + y #z1的计算结果是: 7
5 z2 = x - y #z2的计算结果是: 3
6 z3 = x * y #z3的计算结果是: 10
7 z4 = x / y #z4的计算结果是2.5
8
9 #比较运算符
10 x > y #结果是True
11 x < y #结果是false
12 x != y #结果是True
13 x == y #结果是false
14
15 #逻辑运算符
16 x > y and x > y and x > y #结果是: True
17 x < y and x > y and x > y #结果是: false
18 x < y or x < y or x > y #结果是: True
19 !(x == y) #结果是: True
```

图 4-36

需要注意的是：“and”是逻辑与，代表“并且”的意思。“or”是逻辑或，代表“或者”的意思。“!”是逻辑非，代表“否”的意思：

“and”是所有条件都为“true”的时候，最终条件才为“true”；

“or”是所有条件中，只要有任何一个条件为“true”，最终条件就为“true”。

优先级

如果有一个 $100*(10-1)/(10+5)$ 表达式，程序是先计算哪一步？中学数学告诉我们：①如果是同一级运算，一般按从左往右依次进行计算。②如果既有加减、又有乘除法，先算乘除法、再算加减。③如果有括号，先算括号里面的。④如果符合运算定律，可以利用运算定律进行简算。麦语言的优先级也是如此，如下图：

```
1 100*(10-1)/(10+5) #计算结果是: 60
2
3 1 > 2 and (2 > 3 or 3 < 5) #结果是: false
4 1 > 2 and 2 > 3 or 3 < 5 #结果是: True
```

图 4-37

布尔值

布尔型代表真假，通常用在条件判断和循环语句中。Python 定义了两个常量“True”和“False”代表真假。其实任何对象都可以转成布尔类型，也可以直接用于条件判断，如下图所示：

```
1 a = True #把a设置为True
2 b = False #把b设置为False
3
4 1 < 2 #结果是: True
5 1 > 2 #结果是: False
```

图 4-38

字符串

字符串就是文字，在设置品种代码的时候会经常用到字符串比如“if1905”。Python 中的字符串用单引号 ' 或双引号 " 括起来。加号 + 是字符串的连接符。可以根据索引值获取字符串中的某个字符，如下图：

```
1 name1 = 'google'
2 name2 = "FMZ"
3
4 name1 + " and " + name2 #结果是: 'google and FMZ'
5 name1[0] #结果是: 'g'
6 name2[0] #结果是: 'F'
```

图 4-39

列表

列表是 Python 中使用最频繁的数据类型，你可以把列表想象成一个容器，只不过容器中的元素是从左往右有序排列的，第一位的元素是 0，第二位的元素是 1，以此类推。另外 Python 的列表可以存放任何数据类型，如下图所示：

```
1 lists = ['FMZ', 100, {"name": "发明者量化"}, True]
2 Log(lists[0]) #结果是: 'FMZ'
3 Log(lists[2]["name"]) #结果是'发明者量化'
```

图 4-40

函数

Python 中的函数跟我们中学学的函数没有本质的区别，你可以理解为传进去什么，通过函数的计算，输出什么，如下图所示：

```
1 #定义两个数字相加的函数
2 def add(num1, num2):
3     return num1 + num2
4
5 #使用这个函数，传入两个参数
6 add(1, 2) #函数经过计算，返回的结果是：3
```

图 4-41

if 语句

if 语句经常出现在我们生活当中，比如：如果今天下雨，我就打伞。也就是只有当指定条件为 True 时，该语句才会执行代码。注意，注意代码的缩进格式，否则会生成 Python 错误！如下图所示：

```
1 a = 1
2 b = 2
3
4 if a < b:
5     Log("b大于a")
```

图 4-42

if...else 语句

if...else 语句也是常用的语句，比如：如果今天下雨，我就打伞；否则，我就不打伞。else 语句是 if 语句的延伸，也就是当指定条件为 False 时，else 后面的语句才会执行代码。如下图所示：

```
1 a = 1
2 b = 2
3
4 if a < b:
5     Log("b大于a")
6 else:
7     Log("a大于b")
```

图 4-43

elif 语句

由于 python 并不支持 switch 语句，所以在多个条件判断时，Python 只能用 elif 语句来实现。比如：如果是阳线，我就看多；否则如果是阴线，我就看空；否则我就观望。如下图所示：

```
1 if Close > Open:
2     Log("阳线，看多")
3 elif Close < Open:
4     Log("阴线，看空")
5 else:
6     Log("收盘价等于开盘价，观望")
```

图 4-44

for 循环

有时候我们需要获取最近几天的 K 线数据，就需要从 K 线数组中，根据 K 线数据的位置依次获取，那么使用 for 循环是很方便的，如下图所示：

```
1 def main():
2     exchange.SetContractType('MA888') #设置合约
3     barArr = exchange.GetRecords() #获取K线数组
4
5     #不使用for循环的写法
6     Log(barArr[len(barArr) - 1]) #获取K线数组倒数第1根K线数据，并输出到日志中
7     Log(barArr[len(barArr) - 2]) #获取K线数组倒数第2根K线数据，并输出到日志中
8     Log(barArr[len(barArr) - 3]) #获取K线数组倒数第3根K线数据，并输出到日志中
9     Log(barArr[len(barArr) - 4]) #获取K线数组倒数第4根K线数据，并输出到日志中
10    Log(barArr[len(barArr) - 5]) #获取K线数组倒数第5根K线数据，并输出到日志中
11    #.....
12
13    #使用for循环的写法
14    for i in range(len(barArr)):
15        Log(barArr[-i - 1])
```

图 4-45

while 循环

我们都知道行情是在不断变化的，如果你想获取最新的 K 线数组，就得不断的去一遍又一遍地运行相同的代码，那么使用 while 循环，只要指定条件为 true，循环就可以一直获取最新的 K 线数组。

```
1 def main():
2     exchange.SetContractType('MA888') #设置合约
3     while True:
4         Log(exchange.GetRecords()); #不停的获取最新的K线列表
```

图 4-46

break 语句和 continue 语句

循环是有前提条件的，只有这个前提条件为“true”的时候，循环才会开始重复的做某些事，直到这个前提条件为“false”的时候，循环才会结束。但是 break 语句可以在循环执行的过程中立刻跳出循环；continue 语句可以中断某一次循环，然后继续下一次循环。如下图所示：

```
1 #break语句
2 for i in range(10): #开始遍历一个列表
3     if i == 8: #如果遍历到值为: 8
4         break #跳出整个循环
5 print(i) #最后i的值是: 8, 因为当值为8时, 整个循环就已经结束了
6
7 #continue语句
8 for i in range(10): #开始遍历一个列表
9     if i == 8: #如果遍历到值为: 8
10        continue #跳出整个循环
11 print(i) #最后i的值是: 9, 因为当值为8时, 只是跳出当前循环, 继续进行下一个循环
```

图 4-47

return 语句

return 语句会终止函数的执行并返回函数的值。return 语句只能出现在函数体内，出现在代码中的其他任何地方都会造成语法错误！

```
1 #创建一个函数, 这个函数计算2个参数的乘积
2 def myFunction(a, b):
3     return a * b #函数计算2个参数的乘积并返回
4
5 myFunction(10, 10) #调用函数, 函数的返回值是100
```

图 4-48

策略架构

策略架构你可以理解为策略的固定格式，发明者量化工具采用轮询模式，以下是经典的商品期货策略架构。

其中第 4 行~第 7 行是整个程序的主入口函数，也就是说，计算机是从第 4 行开始执行代码的；紧接着直接执行第 5 行，就进入了无限循环；然后在无限循环里面一直执行策略逻辑函数（onTick）和休眠函数（Sleep）；onTick 函数也就是第 1 行的代码，你可以在第 2 行编写策略逻辑；我们知道，在循环中，程序的执行速度是非常快的，那么使用休眠函数（Sleep）可以让程序暂停一会，下面的代码 Sleep(500)就是每循环一次，就休眠 500 毫秒。

```
1 def onTick():
2     #在这里写策略逻辑
3
4 def main():
5     while(true):
6         onTick();
7         Sleep(500);
```

图 4-49

总结

以上就是 Python 语言快速入门的内容，虽然只是简单的基础知识，但用来写一个简单的量化交易策略还是没问题的。如果需要编写更加复杂的策略，可以参考发明者量化工具 Python 语言 API 文档，或者直接咨询官方客服代写量化交易策略。

下节预告

在技术分析领域的趋势类策略中，均线和通道突破无疑是两大门派。虽然目的都是为了抓住价格走势的趋势，但是这两类策略的交易哲学以及风险特征截然不同。学习了本节 Python 语言入门，下节我们将带大家手把手编写一个通道突破的量化交易策略。

课后习题

- 1、试着用发明者量化工具中的 Python 语言获取历史 K 线数据。
- 2、试着写下本节开头的策略代码，并写上注释。

4.4 如何使用 Python 语言实现策略交易

摘要

上篇我们学习了 Python 语言的简介、基础语法、策略框架等等。虽然内容很枯燥，但这是你实现交易策略的必备技能，也是必须要学会的。那么本篇我们将趁热打铁，继续上篇的 Python 基础知识，从一个简单的策略入手，边学边用，一步一步帮助大家实现一个可行的量化交易策略。

策略简介

在众多交易策略中，唐奇安通道策略应该是最为经典的突破类策略之一，早在 1970 年就已经大名远扬，当时国外有家公司专门对主流的程序化交易策略进行模拟测试和研究，结果表明，在所有策略测试中，唐奇安通道策略最为成功。

后来，在美国又发生了一件交易历史上最著名的“海龟”交易员培训，造就了巨大的成功。当时“海龟们”的交易方法是保密的，但过了十几年，《海龟交易法则》公之于众，人们才发现“海龟们”用的正是改进版的唐奇安通道策略。

突破型交易策略适应于走势比较流畅的交易品种，最常见的突破交易方式就是，利用价格与支撑和阻力的相对位置关系，来判断具体交易买卖点位。本节的唐奇安通道策略也正是基于这个原理。

唐奇安通道策略规则

唐奇安通道属于趋势型指标，它的外表与信号与布林带指标有点相像。但是唐奇安的价格通道是根据一定期间的最高价格与最低价格构建的。比如：在计算最近 50 根 K 线最高价的最大值，形成上轨；计算最近 50 根 K 线最低价的最小值，形成下轨。

该指标由 3 条不同颜色的曲线组成的，默认是 20 个周期内的最高价和最低价来显示市场价格的波动性，当其通道窄时表示市场波动较小，反之通道宽则表示市场波动比较大。

如果价格升破上轨时，就是买入信号；反之，如果价格跌破下轨时，就是卖出信号。由于其上轨和下轨是用最高价和最低价计算出来的，所以一般情况下，价格很少同时升破和跌破上下通道线。大多数情况下，价格是沿着上轨或下轨单边运动，或者在上轨和下轨之间运动的。

唐奇安通道计算方法

在发明者量化工具中，唐奇安通道的计算方法很简单，直接使用获取指定周期内的最高价或最低价就可以了，如下图所示：第 5 行就是获取 50 周期最高价的最大值，第 6 行就是获取 50 周期最低价的最小值。

```
1 def main(): #程序入口
2     exchange.SetContractType("rb888") #设置品种代码
3     while True: #进入循环
4         records = exchange.GetRecords() #获取K线数组
5         upper = TA.Highest(records, 50, 'High') #获取50周期最高价的最大值
6         lower = TA.Lowest(records, 50, 'Low') #获取50周期最低价的最小值
7         middle = (upper + lower) / 2 #计算上轨和下轨的均值
8         Log("上轨: ", upper) #把上轨的值打印到日志中
9         Log("下轨: ", lower) #把下轨的值打印到日志中
10        Log("中轨: ", middle) #把中轨的值打印到日志中
```

图 4-50

策略逻辑

唐奇安通道的使用方法有很多，可以单独使用，也可以和其他指标结合在一起使用。本节课程我们将采用最简单的使用方法。即：当价格自下而上突破上轨，即突破上方压力线时，我们认为多方力量正在走强，一波上涨行情已经形成，买入开仓信号产生；当价格自上而下跌破下轨，即跌破支撑线时，我们认为空方力量正在走强，一波下跌趋势已经形成，卖出开仓信号产生。

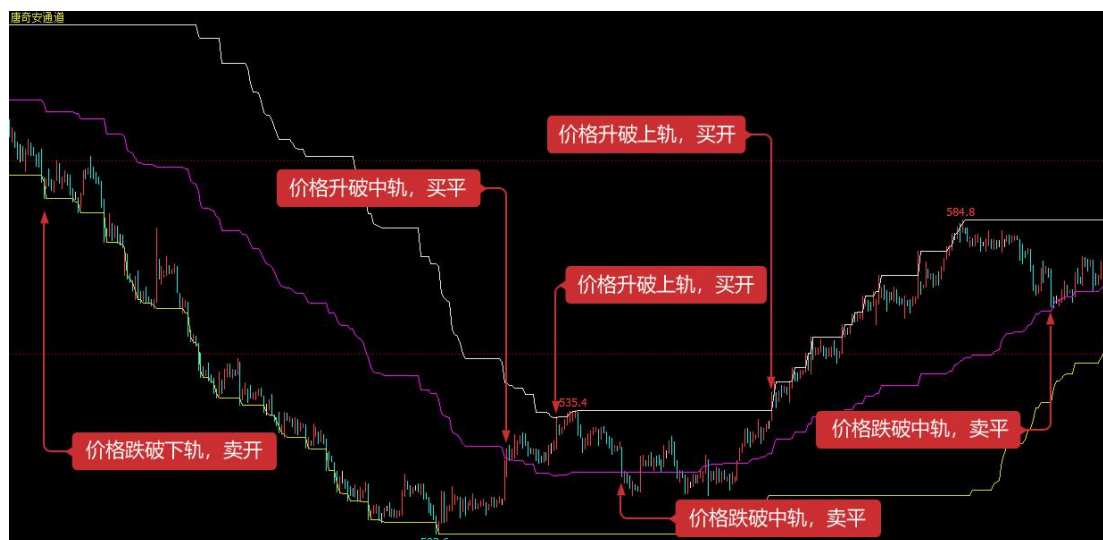


图 4-51

如果买入开仓后，价格又重新跌回到了唐奇安通道中轨，我们认为多方力量正在走弱，或者空方力量正在加强，卖出平仓信号产生；如果卖出开仓后，价格又重新涨回到唐奇安通道中轨，我们认为空方力量正在走弱，或者多方力量正在加强，买入平仓信号产生。

买卖条件

多头开仓： 如果无持仓，并且收盘价大于上轨

空头开仓： 如果无持仓，并且收盘价小于下轨

多头平仓： 如果持多单，并且收盘价小于中轨

空头平仓： 如果持空单，并且收盘价大于中轨

策略代码实现

实现策略的第一步是先获取数据，因为数据是组成交易策略的前提部分，想象一下，我们都需要哪些数据呢？以及如何获取这些数据？然后再根据这些数据计算设计交易逻辑；最后就是配合交易逻辑买卖下单交易。具体步骤如下：

第一步：使用交易类库

你可以把交易类库想象成一个功能模块，使用交易类库的好处是，可以让你把精力放到编写策略逻辑上面。举个例子：当我们使用交易类库时，开平仓的时候，直接使用交易类库中的下单 API 就行了；如果不使用交易类库，开平仓的时候，需要获取盘口价格、需要考虑报单却不成交的问题、需要考虑撤单的问题等等。

```
1 def main():
2     while true:
3         obj = ext.NewPositionManager() #使用交易类库
4         #这里写策略逻辑和下单代码
```

图 4-52

上图就是使用发明者量化工具的 CTA 策略框架。这是一个固定的代码格式，所有的交易逻辑代码从第 4 行开始编写。其他地方不用任何修改。

第二步：获取各种数据

仔细想下，都需要哪些数据呢？从我们的策略交易逻辑中发现：首先需要获取当

前的持仓状态，然后比较收盘价与布林带指标上中下轨的相互关系，最后判断行情是不是即将收盘。那么接下来让我们一以获取这些数据吧。

获取 K 线数据

首先就是获取 K 线数组和当前 K 线收盘价，因为有了 K 线数组，才能调用获取 N 周期最高价或最低价的 API。用代码写出来是这样的：

```
1 def main():      #主函数
2     exchange.SetContractType("rb888") #设置品种代码
3     while True: #进入循环
4         records = exchange.GetRecords() #获取K线数组
5         if len(records) < 50: continue #如果K线少于50根，就跳过本次循环
6         close = records[len(records) - 1].Close #获取最新K线收盘价
```

图 4-53

如上图所示：

第 4 行：获取 K 线数组，这是一个固定的格式。

第 5 行：过滤 K 线的长度，因为我们计算 N 周期最高价或最低价，用的参数是 50，当 K 线小于 50 根的时候，是无法计算的。所以这里要过滤下 K 线的长度，如果 K 线少于 50 根，就跳过本次循环，继续等待下一根 K 线。

第 6 行：我们用代码“records[len(records) - 1]”先获取到 K 线数组的最后一个数据，也就是最新的 K 线数据。这个数据是一个对象，里面包含：开盘价、最高价、最低价、收盘价、成交量、时间等数据，既然它是一个对象，那么我们就直接用“Close”就是获取最新 K 线收盘价。

获取持仓数据

持仓信息是量化交易策略中一个很重要的条件，当交易条件成立时，还需要通过持仓状态和持仓数，来判断是否下单。比如：当买入开仓交易条件成立时，如果有持仓，就不必在重复下单了；如果无持仓，就可以下单。这次我们直接把持仓信息封装成一个函数，只需要调用这个函数就可以使用了：

```
1 # 获取持仓信息函数
2 def mp():
3     positions = exchange.GetPosition() #获取持仓数组
4     if len(positions) == 0: #如果持仓数组的长度是0
5         return 0 #证明是空仓, 返回0
6     for i in range(len(positions)): #遍历持仓数组
7         if (positions[i]['Type'] == PD_LONG or (positions[i]['Type'] == PD_LONG_YD):
8             return 1 #如果有多单, 返回1
9         elif (positions[i]['Type'] == PD_SHORT or (positions[i]['Type'] == PD_SHORT_YD):
10            return -1 #如果有空单, 返回-1
11
12 def main(): #主函数
13     exchange.SetContractType("rb888") #设置品种代码
14     while True: #进入循环
15         records=exchange.GetRecords() #获取K线数组
16         if len(records) < 50: continue #如果K线少于50根, 就跳过本次循环
17         close = records[len(records) - 1].Close #获取最新K线收盘价
18         positions = mp() #获取持仓信息函数
```

图 4-54

如上图所示:

这是一个获取持仓信息的函数, 如果是空仓就返回 0; 如果持多单就返回 1; 如果持空单就返回-1。注意看上面的代码:

第 2 行: 创建一个函数, 名字是 mp, 这个函数并没有参数。

第 3 行: 获取持仓数组, 这是一个固定的格式。

第 4 行: 判断持仓数组的长度, 如果它的长度等于, 那肯定是空仓, 所以就返回 0

第 6 行: 使用 for 循环, 开始遍历这个数组, 接下来的逻辑就已经很简单了, 如果持多单, 就返回 1; 如果持空单, 就返回-1。

第 18 行: 调用刚才写的获取持仓信息函数 mp。

获取最近 50 根 K 线的最高价和最低价

在发明者量化工具中, 直接使用“TA.Highest”和“TA.Lowest”函数就能直接获取, 而不用再自己写逻辑计算了。并且“TA.Highest”和“TA.Lowest”函数返回的结果是具体的数值而不是数组。这一点非常方便, 不止如此, 官方内置了上百个指标函数。

```
1 # 获取持仓信息函数
2 def mp():
3     positions = exchange.GetPosition() #获取持仓数组
4     if len(positions) == 0: #如果持仓数组的长度是0
5         return 0 #证明是空仓, 返回0
6     for i in range(len(positions)): #遍历持仓数组
7         if (positions[i]['Type'] == PD_LONG or (positions[i]['Type'] == PD_LONG_YD):
8             return 1 #如果有单, 返回1
9         elif (positions[i]['Type'] == PD_SHORT or (positions[i]['Type'] == PD_SHORT_YD):
10            return -1 #如果有空单, 返回-1
11
12 def main(): #主函数
13     exchange.SetContractType("rb888") #设置品种代码
14     while True: #进入循环
15         records=exchange.GetRecords() #获取K线数组
16         if len(records) < 50: continue #如果K线少于50根, 就跳过本次循环
17         close = records[len(records) - 1].Close #获取最新K线收盘价
18         positions = mp() #获取持仓信息函数
19         upper = TA.Highest(records, 50, 'High') #获取50周期最高价的最大值
20         lower = TA.Lowest(records, 50, 'Low') #获取50周期最低价的最小值
21         middle = (upper + lower) / 2 #计算上轨和下轨的均值
```

图 4-55

如上图所示:

第 19 行: 调用 TA.Highest” 函数, 获取 50 周期最高价的最大值

第 20 行: 调用 “TA.Lowest” 函数, 获取 50 周期最低价的最小值

第 21 行: 根据 50 周期最高价的最大值和 50 周期最低价的最小值, 计算出平均值

第三步: 下单交易

有了以上数据, 就可以编写交易逻辑以及下单交易的代码了。格式也非常简单, 最常用到的是 “if 语句”, 用文字可以描述为: 如果条件 1 和条件 2 成立, 下单; 如果条件 3 或条件 4 成立, 下单。


```
1 # 获取持仓信息函数
2 def mp():
3     positions = exchange.GetPosition() #获取持仓数组
4     if len(positions) == 0: #如果持仓数组的长度是0
5         return 0 #证明是空仓, 返回0
6     for i in range(len(positions)): #遍历持仓数组
7         if (positions[i]['Type'] == PD_LONG) or (positions[i]['Type'] == PD_LONG_YD):
8             return 1 #如果有多单, 返回1
9         elif (positions[i]['Type'] == PD_SHORT) or (positions[i]['Type'] == PD_SHORT_YD):
10            return -1 #如果有空单, 返回-1
11
12 def main(): #主函数
13     exchange.SetContractType("rb888") #设置品种代码
14     while True: #进入循环
15         records=exchange.GetRecords() #获取K线数组
16         if len(records) < 50: continue #如果K线少于50根, 就跳过本次循环
17         close = records[len(records) - 1].Close #获取最新K线收盘价
18         positions = mp() #获取持仓信息函数
19         upper = TA.Highest(records, 50, 'High') #获取50周期最高价的最大值
20         lower = TA.Lowest(records, 50, 'Low') #获取50周期最低价的最小值
21         middle = (upper + lower) / 2 #计算上轨和下轨的均值
22         obj = ext.NewPositionManager() #使用交易类库
23         if positions > 0 and close < middle: #如果持多单, 并且收盘价跌破中轨
24             obj.CoverAll() #平掉所有仓位
25         if positions < 0 and close > middle: #如果持空单, 并且收盘价升破中轨
26             obj.CoverAll() #平掉所有仓位
27         if positions == 0: #如果是空仓
28             if close > upper: #如果收盘价升破上轨
29                 obj.OpenLong("rb888", 1) #买开
30             elif close < lower: #如果收盘价跌破下轨
31                 obj.OpenShort("rb888", 1) #卖开
```

图 4-56

如上图所示:

第 22 行: 使用交易类库, 这是一个固定的格式

第 23、24 行: 这是一个平多单的语句, 其中用到了我们之前学过的“比较运算符”和“逻辑运算符”, 意思是如果当前持多单, 并且收盘价小于中轨, 就平掉所有仓位。

第 25、26 行: 这是一个平空单的语句, 其中用到了我们之前学过的“比较运算符”和“逻辑运算符”, 意思是如果当前持空单, 并且收盘价大于中轨, 就平掉所有仓位。

第 27 行: 判断当前持仓状态, 如果持空仓, 才进行下一步。

第 28、29 行: 判断收盘价是否大于上轨, 如果收盘价升破上轨, 就买入开仓。

第 30、31 行: 判断收盘价是否小于下轨, 如果收盘价跌破下轨, 就卖出开仓。

总结

以上我们学习了用 Python 开发一个完整的量化交易策略的每个步骤，包括：策略简介、唐奇安通道的计算方法、策略逻辑、买卖条件、策略代码实现等。本节只是一个简单的策略，作为一个抛砖引玉，方法不止一个，你可以根据自己的交易系统，叠加不同的交易方法，从而形成属于自己的量化交易策略。

下节预告

在量化交易策略开发中，站在编程语言执行速度的角度看，如果说哪个语言最快，那只能是 C++ 莫属不可。特别是在衍生品和高频交易领域，C++ 独特的语言特定，C++ 在数值计算上有优势，与 JavaScript 和 Python 相比速度能提高几个量级，如果将来你想往衍生品和高频交易领域发展，这将是你不容错过的课程。

课后习题

- 1、从临摹开始，动手实现本节的策略。
- 2、试着给本节的策略增加一个均线指标，减低交易频率。

第五章 策略回测、调试及改进

5.1 回测的意义和陷阱

摘要

回测是量化交易与传统交易最不同的地方，根据历史上已经发生过的真实行情数据，快速模拟策略信号触发和撮合交易，得出一段时间内的绩效报告等数据。对国内、国外的股票、商品期货、外汇等市场都是策略开发最重要的组成部分之一。

回测的意义

前面的章节我们学习了主流编程语言的基础部分，以及教大家如何利用这些编程基础，写一些简单的交易策略，可以说万里长征已经走了一大半了。但是，一个策略编写完，肯定不是直接就可以实盘的，它还需要不断的回测——调试——回测——调试——等等，直到策略能够完整的实现模型内容，并且能够顺畅运行。

从量化交易逻辑的角度讲，策略其实就是建立在对市场上的一系列认知和假设，回测可以高效率确定这些假设是否成立和稳定。在历史不稳定时期，可能会带来什么样的损失，以及为预防这些损失辅助作出决策。

另外，从量化交易运行的角度讲，回测可以帮助检测策略逻辑中的 bug，如未来函数、偷价、多度拟合等等。为策略可以用于实盘交易提供可靠的证据。

- 验证交易信号的准确度。
- 验证交易逻辑和你的想法是否可行。
- 发现交易系统缺陷，并改进原始策略。

因此，回测的意义是通过历史数据，尽可能真实的还原实际的交易过程，对策略有效性的进行验证，避免为错误的策略付出昂贵的代价，并帮助我们筛选、改进、优化交易策略。

回测的陷阱

回测陷阱之信号闪烁：

交易策略在回测时是基于静态的历史数据。而真实的交易的数据是动态的。举个

例子：如果最高价大于昨天的收盘价就买入开仓。这个开仓条件在实盘中，如果K线还未走完，那么最高价就是动态的，交易信号就有可能来回闪烁。而在回测时，回测引擎是基于静态的历史数据是可以模拟撮合成交易的。

回测陷阱之未来函数：

未来函数是用到了未来的价格，也就是说当前的条件在未来可能会被修改，同样未来函数也能造成信号闪烁的原因。所以任何函数都具有未来函数特性，比如“之字转向函数”。

如下图：之字转向函数指示了波峰和波谷的转折点，它能根据最新的实时价格相应得调整自身取值，但是如果当前价格变化的时候，之字转向函数计算的结果也会随着改变。如果用了带有未来函数的函数，可能当前下单信号成立了并且下单，但过会儿可能这个信号又不成立了。

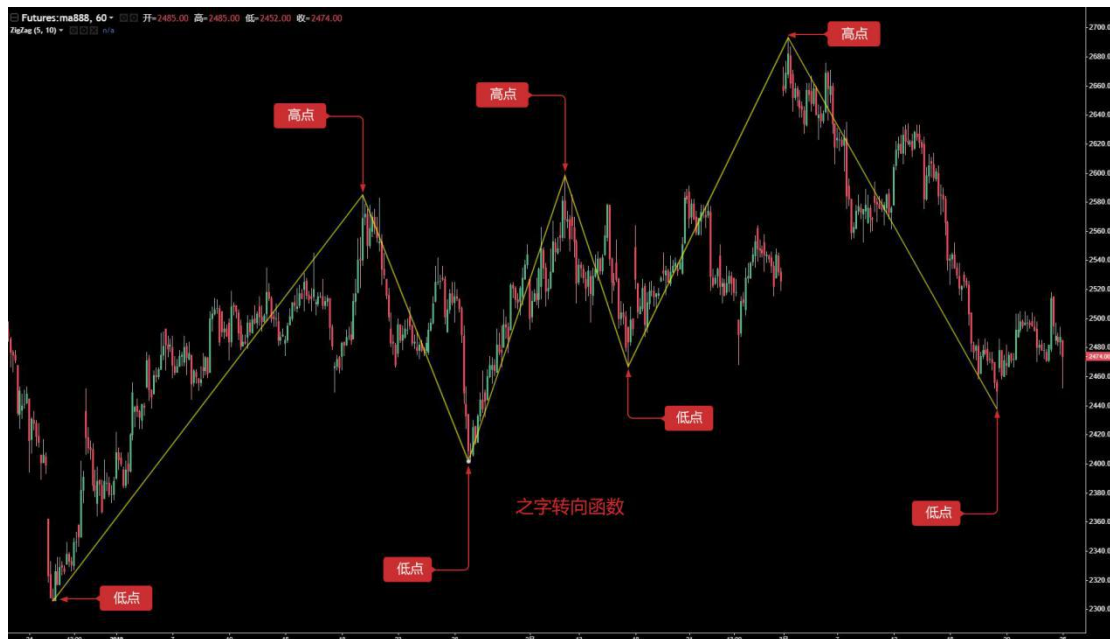


图 5-1

回测陷阱之偷价

所谓偷价行为是指利用过去的价格去交易。举个例子：如果最高价大于某个固定价位即以开盘价买入。这个条件就是在偷价格，因为在实盘中，最高价大于某个价位时，价格已经高于开盘价一定距离了，这时用开盘价是买不到的。但在回测中，是有买入信号的，并且能成交。

还有一种情况，如果价格跳空高开与策略设定的固定价格，回测时可以有固定价格成交，但是在实盘中这个固定价格显然是买不到的。

回测陷阱之不可能成交的价格

不能成交的价格分为几种情况：

第一种：在实盘中，涨停时一般情况下是买不到的，反过来跌停也是如此。但是在回测中却是可以成交的。

第二种：交易所撮合机制是：价格优先、时间优先。有些品种盘口会经常有巨量订单，实盘时如果挂单买卖，需要等待盘口厚度，才能成交甚至不能成交。但是在回测时，挂单买卖是可以成交的。

第三种：如果套利类策略，那么回测利润是很高的，因为回测时每次都已经假设了抢到了这些价差。真实的情况下，很多价差都抢不到，或者只抢到了一条腿，一般来说肯定是不利于你的方向的那条先成交，那么就需要马上去补另一条腿，这时候滑点已经不是1、2个点了，而套利策略本身就赚这几个点的价差，这种情况是回测中无法模拟的。真实利润完全不如回测。

第四种：黑天鹅事件。如下图红圈处，在外汇瑞郎黑天鹅事件中，尽管表面上看有开盘价、最高价、最低价、收盘价，其实当天的极端行情中，中间的价格是真空，大量的止损单，造成踩踏事件，流动性为零，成交难度非常大，但是在回测中却能止损。



图 5-2

回测陷阱之过度拟合

每次看到下面这张图，我的内心是：哈哈哈哈.....通过下面这张图可以看到，一个荒谬的模型，只要足够复杂，是可以完美适应数据的。

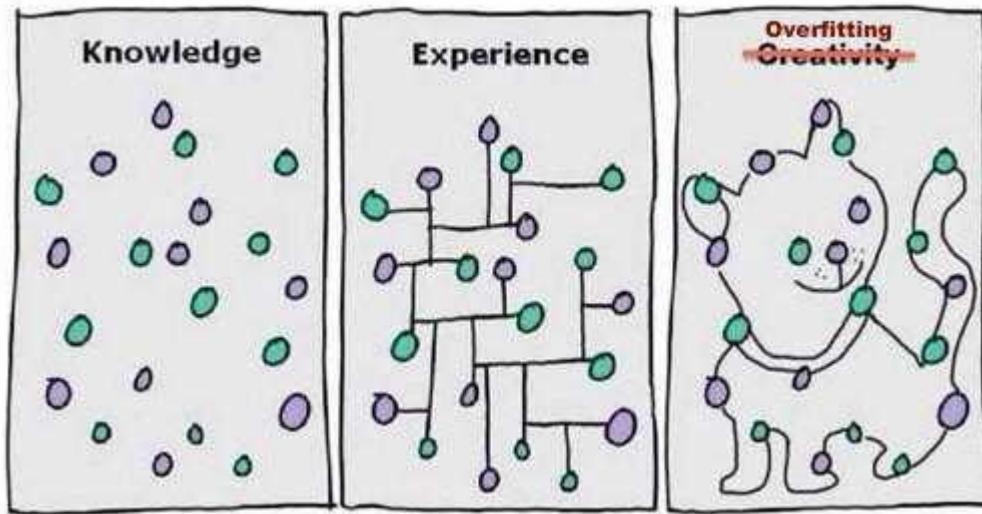


图 5-3

针对量化交易来说，回测是基于历史数据，但历史数据的样本是有限的，如果交易策略的参数过多，或者交易逻辑过于复杂，导致交易策略过多的适应历史数据。

量化策略的建模过程本质上就是一个从大量的貌似随机的数据中找寻局部非随机数据的过程，如果不借助统计学的知识，很容易落入过度拟合的陷阱。

所以，不要自欺欺人。如果发现样本外数据表现不好，又觉得丢掉模型太可惜或者不愿意承认自己这个模型不行，而对着样本外数据继续做模型优化，直到样本外数据上也表现得一样好，那最后受伤的一定是你的真金白银。

回测陷阱之幸存者偏差

华尔街流行着这样一个笑话：假设市场上有 1000 只参与投资的猴子，第一年，淘汰 500 只输给大盘的猴子。第二年再淘汰一半，剩下 250 只猴子。等到第三年末，剩下 125 只猴子。

巴菲特的猴子故事

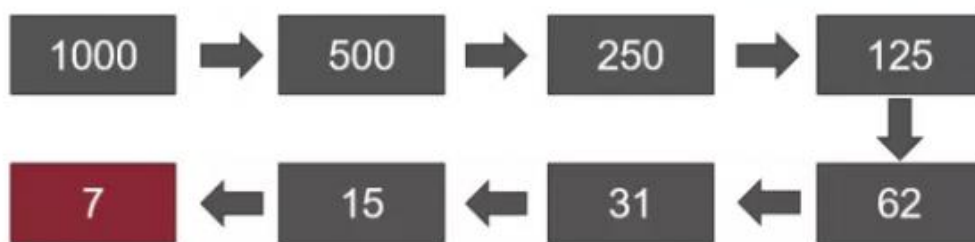
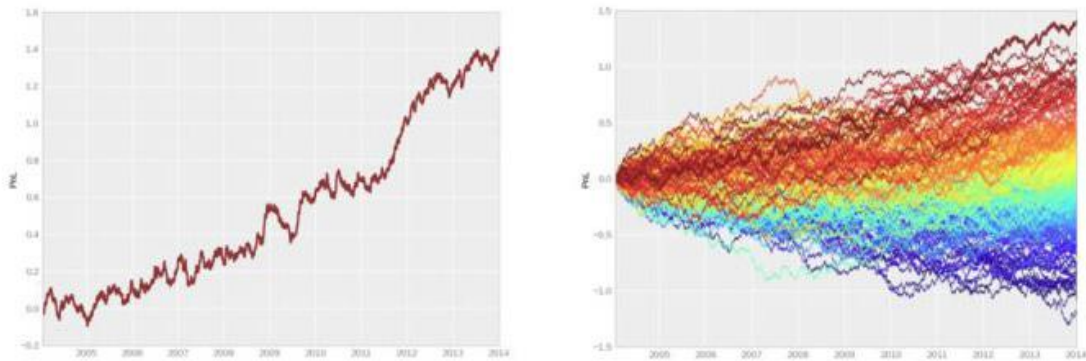


图 5-4

到了第九年，还剩下最后一个猴子。然后你看啊看，左看看右看看就觉得眼熟。最后看到财经杂志封面一下子想起来，“噢，这不就是巴菲特吗！”

当然这只是一个笑话，但是你有没有想过，如果有 1000 个基金经理，那么在 10 年后，大约有 10 个基金经理会连续 10 年跑赢大盘战胜市场。但这可能是随机和运气决定的，和基金经理们的技能没有关系。

就像下图左边那张回测绩效，相信绝大多数投资者都会眼前一亮。该投资策略有非常稳健的表现，而且几乎没有大幅度回撤。



- 左图（表象）：一个非常不错的交易策略。没有大的回撤，投资者可以获得稳定投资回报。
- 右图（真相）：这只是200次随机交易回测中表现最好的一个而已。

图 5-5

且慢，如右图所示，真实的情况就在里面。原来左边的回测曲线只是众多回测中表现最好的一只而已。也就是说在左边那个回测中，背后还有众多表现更差的情况。

回测陷阱之冲击成本

在真实的交易环境中，价格是一直在波动的，当你看好一个交易机会，下单的那一刻，可能价格就已经变化了。所以滑点问题，无论是在主观交易中，还是在量化交易中，都是不可避免的。

但是回测是基于在静态数据，很难模拟出真实的交易环境。举个例子：下单价格是 1050 买入，但实际成交价可能是 1051。造成这种现象的原理有很多，比如：极端行情时流动性真空、网络延迟、软硬件系统、服务器响应等。

不加滑点的回测

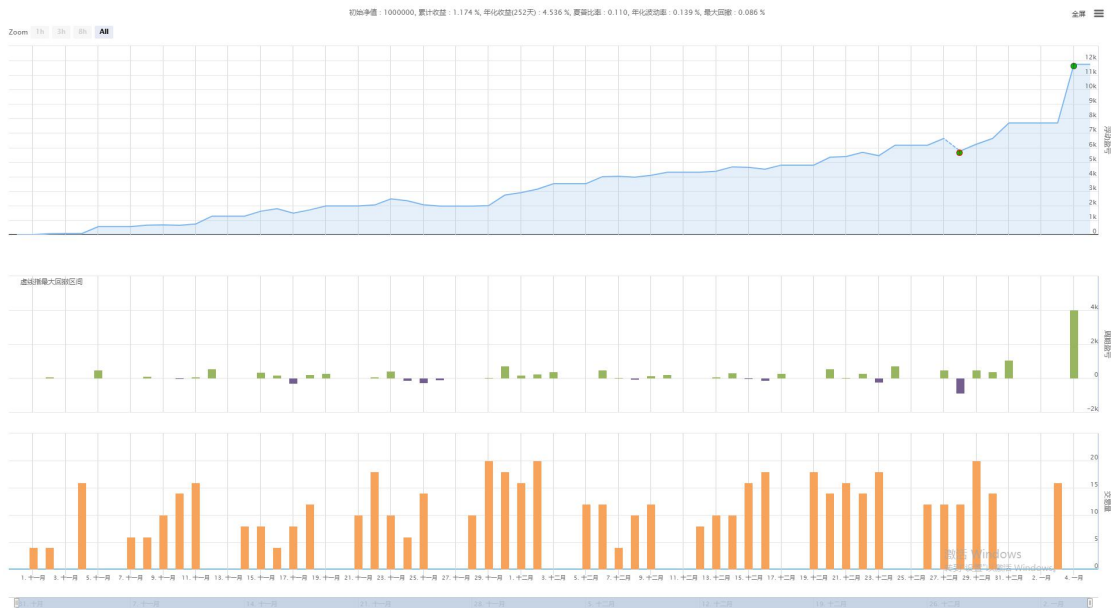


图 5-6

如上图，是一个不加滑点的回测，资金曲线比较好看，但在实盘交易中的实际成交价与策略回测的理想成交价存在差异。所以为了减小这种误差，在进行策略回测时，可以设置 2 个滑点，来提高买入价或降低卖出价。

加上滑点的回测

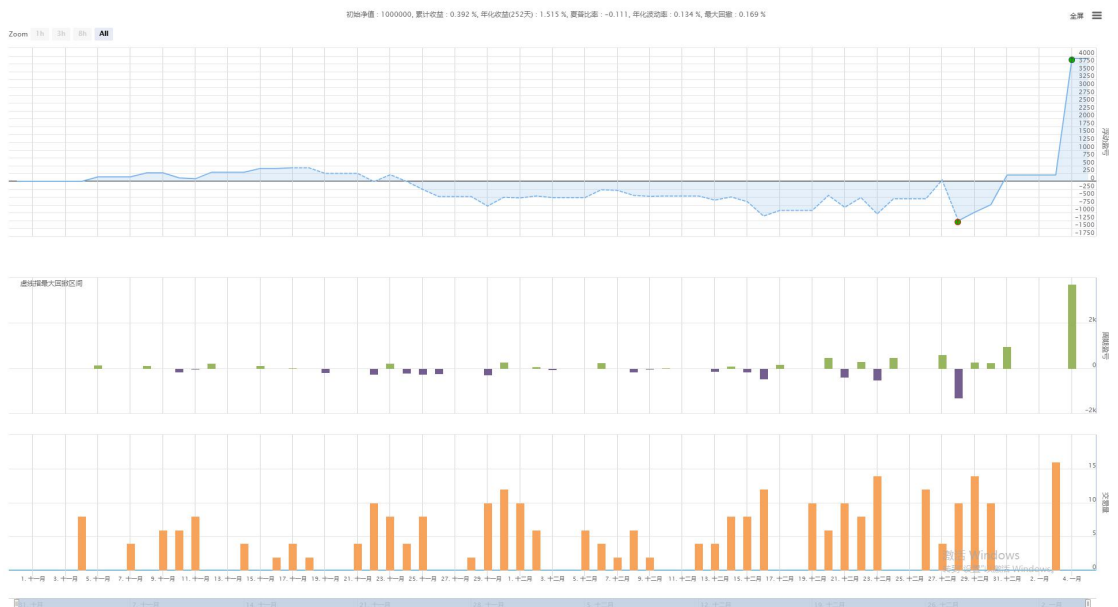


图 5-7

如上图，同样的策略，如果加上 2 跳滑点后，回测的结果与不加滑点的回测结果，相差很大，那么也就代表这个策略需要改进或者淘汰换新。特别是交易频率比较高的策略，回测时加上 1~2 跳的滑点，可以使回测更接近于真实的交易环境。

总结

可能会有小伙伴问，既然量化交易可能会出现这么多的问题，那我应该怎么证明我的策略是没有问题的呢？其实答案很简单，在策略实盘之前一定要先仿真交易一段时间，如果仿真交易的成交价格与回测时的成交价格，相差无几，那么就证明这个策略是没有问题的，至少策略逻辑是没有问题的。

不管怎么说，对于一个有经验的交易系统开发者来说，回测是必须要做的。因为它能告诉你一个策略的想法在历史交易中是否能被验证有效。但是很多时候回测并不代表未来能盈利。因为回测里面有太多坑了，不用钱买点教训，你是不会明白的。而这些教训都是用真金白银堆出来的。我想读了本篇至少能让你少走很多量化的弯路和陷阱。

课后习题

- 1、什么是过度拟合，以及如何避免？
- 2、现实生活中有哪些是幸存者偏差？

5.2 如何做量化交易回测

摘要

回测的意义和重要性已毋庸置疑，在进行量化回测时，应该尽最大可能让策略处于历史的真实环境中，如果忽略历史环境中的细节，可能会导致整个量化回测是无效的。本篇将为大家讲解如何做量化交易回测。

回测就相当于数据回放，通过回放历史 K 线数据，并进行模拟真实的交易规则进行买卖，最终汇总一个时间段内的夏普比率、最大回撤率、年化收益率、资金曲线等数据。目前有很多软件都能做到回测，比如品种很全的文华财经、可以灵活定制的 VNPY 等等。

发明者量化作为一款商用量化交易软件，自带高性能回测引擎，采用 for-loop(轮询)回测框架，进行向量化计算，速度更快。并且统一回测和实盘的代码，部分解决“回测易、实盘难”的困境。

回测界面介绍

我们以发明者量化的麦语言策略为例，打开发明者量化交易工具的官网（www.fmz.com）。依次点击控制中心、策略库、选择一个策略、模拟回测，进入到如下页面：



图 5-8

在回测配置界面，可以根据自己的实际需求定制。如：设置回测时间，K 线周期，数据种类（模拟级别数据或实盘级别数据。相比之下模拟级别数据回测速度更快，实盘级别数据回测更精准）。另外，还可以设置回测时的手续费以及账户初始资金等等。

点击麦语言交易类库，首先是交易设置标签，发明者量化交易工具中的麦语言策略有两种回测执行方式，即：收盘价模型和实时价模型。收盘价模型指当前 K 线走完才执行模型，在下根 K 线开始的时候执行交易。实时价模型指的是每次价格变动都执行一次模型，当交易信号成立时，就立即交易。如下图：

The screenshot shows the '麦语言交易类库' (Ma Language Trading Library) settings interface. It is divided into three sections:

- 交易设置 (Trading Settings):**
 - 执行方式 (Execution Mode): 收盘价模型 (Close Price Model)
 - 默认开仓手数 (Default Open Position): 1 (with a '调优' button)
 - 最大单次交易下单量 (Maximum Single Trade Order Quantity): 1000 (with a '调优' button)
 - 滑价点数(整数) (Slippage Points (Integer)): 0 (with a '调优' button)
- 期货选项 (Futures Options):**
 - 期货合约 (Futures Contract): rb000
- 实盘选项 (Real-time Options):**
 - 自动恢复进度 (Auto Recovery Progress): (勾选为True)
 - 下单重试次数 (Order Retry Count): 20 (with a '调优' button)
 - 网络轮询间隔(毫秒) (Network Polling Interval (ms)): 500 (with a '调优' button)

图 5-9

默认开仓手数是指回测时开平仓数量，最大单次交易下单量就是单笔交易委托给回测引擎的最大开平仓数量。实盘的成交价位与预设的成交价位间出现偏移，这种偏移一般是向不利于交易者的方向移动，导致交易出现额外的损失，所以加入滑点是必须的，国内商品期货一般是加入 1 跳~2 跳，甚至更多，来模拟真实的交易环境。

期货选项中填入要回测的合约品种，如 rb000 或 rb888。实盘选项主要用于实盘交易，在回测中保持默认设置即可。如果自动恢复进度点击为 true，那么当策略在实盘运行中停止机器人后，重启机器人会自动恢复之前的信号位置，而不用再重新计算信号。设置下单重试次数默认为 20，当下单失败后，会尝试重新下单。网络轮询间隔就是机器人每隔段时间去执行一次策略代码。

This is a close-up of the '实盘选项' (Real-time Options) section from the previous screenshot. It shows:

- 自动恢复进度 (Auto Recovery Progress): (勾选为True)
- 下单重试次数 (Order Retry Count): 20 (with a '调优' button)
- 网络轮询间隔(毫秒) (Network Polling Interval (ms)): 500 (with a '调优' button)

图 5-10

现货交易选项主要针对数字货币交易，在回测中保持默认设置即可。可以指定单笔交易量、最小交易量、定价货币精度、交易品种精度、手续费、账户同步时间、盈亏统计间隔等，另外对于个别数字货币交易所，还可以设置杠杆倍数以及其他

相关设置。

现货交易		
一手交易量	1	<input type="checkbox"/> 调优
最小交易量	0.01	<input type="checkbox"/> 调优
定价货币精度	1	<input type="checkbox"/> 调优
交易币种精度	4	<input type="checkbox"/> 调优
手续费	0.002	<input type="checkbox"/> 调优
账户同步时间(秒)	10	<input type="checkbox"/> 调优
盈亏统计间隔	小时	▼
杠杆倍数	10	▼
开仓后仓位同步时间(毫秒)	500	<input type="checkbox"/> 调优
失败重试(毫秒)	500	<input type="checkbox"/> 调优
使用代理	<input type="checkbox"/> (勾上为True)	
隐藏常见网络错误	<input checked="" type="checkbox"/> (勾上为True)	
切换基地址	<input type="checkbox"/> (勾上为True)	
推送通知	<input type="checkbox"/> (勾上为True)	

图 5-11

策略回测

在回测之前，先确定好你的交易策略，这里我们以恒温器 Thermostat 策略为例，这种策略会根据市场状态，在趋势行情中采用趋势策略，在震荡行情中采用震荡策略。源代码如下图（也可以到发明者量化官网的策略广场直接下载）：

```
1 // 计算CMI指标用以区分震荡市与趋势市
2 CMI:=ABS(C-REF(C,29))/(HHV(H,30)-LLV(L,30))*100;
3
4 // 定义关键价格
5 KOD:=(H+L+C)/3;
6
7 // 震荡市中收盘价大于关键价格为宜卖市，否则为宜买市
8 BE:=IFELSE(C>KOD,1,0);
9 SE:=IFELSE(C<=KOD,1,0);
10
11 // 定义10日 ATR 指标
12 TR:=MAX(MAX((HIGH-LOW),ABS(REF(CLOSE,1)-HIGH)),ABS(REF(CLOSE,1)-LOW));
13 ATR10:=MA(TR,10);
14
15 // 定义最高价与最低价3日均线
16 AVG3HI:=MA(H,3);
17 AVG3LO:=MA(L,3);
18
19 // 计算震荡市的进场价格
20 LEP:=IFELSE(C>KOD,0+ATR10*0.5,0+ATR10*0.75);
21 SEP:=IFELSE(C>KOD,0-ATR10*0.75,0-ATR10*0.5);
22 LEP1:=MAX(LEP,AVG3LO);
23 SEP1:=MIN(SEP,AVG3HI);
24
25 // 计算趋势市的进场价格
26 UPBAND:=MA(C,50)+STD(C,50)*2;
27 DNBAND:=MA(C,50)-STD(C,50)*2;
28
29 // 计算趋势市的出场价格
30 MA50:=MA(C,50);
31
32 // 震荡策略逻辑
33 CMI<20&&C>=LEP1,BK;
34 CMI<20&&C<=SEP1,SK;
35 CMI<20&&C>=AVG3HI,SP;
36 CMI<20&&C<=AVG3LO,BP;
37
38 // 趋势策略逻辑
39 CMI>=20&&C>=UPBAND,BK;
40 CMI>=20&&C<=DNBAND,SK;
41 CMI>=20&&C<=MA50,SP;
42 CMI>=20&&C>=MA50,BP;
43 AUTOFILTER; //一开一平下单模式
```

图 5-12

在模拟回测界面，配置好回测设置后，直接点击开始回测按钮，几十秒之后回测结果即刻呈现。在回测日志中，记录了回测用时多少秒，日志总数以及交易次数。其中账户信息打印了策略回测最终绩效结果：平均盈亏、持仓盈亏、保证金、手续费以及预估收益等。



图 5-13

状态信息栏中则记录了交易品种、持仓量、持仓价格、最新价格、上次信号种类、持仓后的最高价和最低价、更新次数和时间以及资金信息。另外在浮动盈亏标签中，则展示了账户的详细资金曲线，还包括常用的绩效指标：收益率、年化收益率、夏普比率、年化波动率、最大回撤率，基本可以满足绝大多数的用户需求。

其中，最为重要的绩效指标就是：夏普比率。它是同时对收益与风险加以考虑的综合指标，也是衡量一个基金产品的重要参考指标，通俗来讲，就是你每挣 1 块钱承担了多少钱的风险，所以夏普比率的值是越高越好。

年化波动率顾名思义就是日波动率 x 了每年的交易日数，它是衡量基金的风险，但绝对不是全部风险。比如策略 A 波动率较大，但一直波动向上，收益率良好，策略 B 波动率很小，但一直横着不动，我们能说策略 B 优于策略 A 吗？如下图策略 A：



图 5-14

最后，在日志信息栏中，详细记录了回测时每一笔交易的撮合情况，包括交易的具体时间、交易所、买卖以及开仓平仓类型、回测引擎撮合的成交价格、交易数量以及打印信息等等。

日志信息

时间	平台	类型	价格	数量	信息
2016-12-19 11:00:00	Futures_CTP	卖出 开空	3227	1	rb1705 Bid {"Price":3227,"Amount":1000}
2016-12-19 11:00:00		信息	SK 信号所在行数: 49 信号次数: 1		
2016-12-16 09:00:30	Futures_CTP	卖出 平多	3381	1	rb1705 平今 Bid {"Price":3381,"Amount":1000}
2016-12-16 09:00:00		信息	SP 信号所在行数: 50 信号次数: 1		
2016-12-07 11:00:00	Futures_CTP	买入 开多	3408	1	rb1705 Ask {"Price":3408,"Amount":1000}
2016-12-07 11:00:00		信息	BK 信号所在行数: 48 信号次数: 1		
2016-12-05 13:30:30	Futures_CTP	卖出 平多	3223	1	rb1705 平今 Bid {"Price":3223,"Amount":1000}
2016-12-05 13:30:00		信息	SP 信号所在行数: 44 信号次数: 1		
2016-12-05 11:00:00	Futures_CTP	买入 开多	3206	1	rb1705 Ask {"Price":3206,"Amount":1000}
2016-12-05 11:00:00		信息	BK 信号所在行数: 42 信号次数: 1		
2016-12-02 13:30:30	Futures_CTP	卖出 平多	3060	1	rb1705 平今 Bid {"Price":3060,"Amount":1000}

图 5-15

回测之后

很多时候，甚至绝大多数情况下，回测的结果都会与自己的期望，相差甚远。毕竟一个长期持续稳定盈利的策略，不是那么容易就能得到的，这需要你对市场认知能力。

如果你的策略回测结果是亏钱的，也不要灰心，这其实是很正常的。先看一下策略逻辑是不是写错了，是不是采用了极端参数，是不是开平仓条件过多等等，必要时也可以从另一个角度重新审视自己的交易策略和交易理念。

如果你的策略回测结果非常好，资金曲线非常完美，夏普比率超过 1 甚至更多。也先别急着高兴，遇到这种情况，大部分是利用了未来函数、或者偷价、或者过度拟合、或者没有设置滑点等等，可以利用样本外数据和仿真实盘交易，来排除这些问题。

总结

以上就是整个交易策略回测的整个流程介绍，可以说已经具体到每一个细节。需要注意的是，历史数据回测毕竟是一个所有风险已知的理想环境。所以策略的回测时间最好是经历一轮牛熊市，有效的交易次数应当不低于 100 次，这样可以避免部分幸存者偏差。

市场永远是在变动和进化中的，历史回测好的策略并不代表未来就一定很优秀，不能只让策略应付回测环境中已知的风险，更要应对未来的未知风险。所以增加策略的抗风险能力和普适性是非常有必要的。

课后习题

- 1、试着复制本节中的策略，并回测绩效报告
- 2、根据自己的交易经验，尝试改进并优化本节中的策略

5.3 如何读懂策略回测绩效报告

摘要

当我们的策略回测完成之后，发明者量化交易工具会在网页中输出包含各种绩效指标、收益曲线图。但可能因为我们对这些指标的释义和内容不太熟悉，导致无法准群判断策略好坏，本篇将从主要的指标概念入手，帮助大家读懂策略回测绩效报告，分辨出策略的优劣。当然大部分量化交易工具都有这种回测绩效报告，其内容也大同小异，学会了本节内容，就算换做另一个交易工具也同样适用。

客观与完整的评价

无论是实盘交易数据的记录，还是采用历史数据进行回溯（Back-Testing）的回测报告，模型的优劣都是通过对交易情况的统计来进行评价。

而问题的关键在于，到底需要通过哪些统计数据进行比较？先来看一个例子：如下图，假设在同一时间周期的测试中得到以下两组数据，我们能从中判定哪一个模型表现更优秀么？

编号	总盈利率	胜率	单笔最大盈利	最大回撤
模型A	50%	35%	10%	12%
模型B	30%	45%	20%	16%

图 5-16

答案是，不能。评价体系的片面性将导致量化交易系统走向绝境。

交易系统必须能通过历史回测才可以投入使用。无法通过历史回测的交易系统不可能长期在实际交易中获利。历史回测是交易系统投入实盘的必要前置环节。

能通过历史回测的交易系统不一定是好用的交易系统，但不能通过历史回测，则一定不是好用的交易系统。一般而言，我们需要从稳定性、可持续性、判断是否正期望等角度去分析绩效报告。

净利	已付滑价	校正后的净利/单笔最大回撤	交易总数量	最大值
毛利	已付手续费	校正后的净利/策略最大回撤	未平仓交易总数量	最大值日期
毛损	未平仓盈利/亏损	交易的平均持仓K线根数	盈利交易次数	平均值
校正后的净利	年化收益率	盈利交易的平均持仓K线根数	亏损交易次数	最大值(%)
校正后的毛利	月化收益率	亏损交易的平均持仓K线根数	胜率	最大值(%)日期
校正后的毛损	持有收益	平均空仓K线根数	单笔净利	平均值(%)
选定的净利	月平均收益	两笔盈利交易之间的平均空仓K线根数	平均盈利额	1倍标准差
选定的毛利	月平均收益标准差	两笔亏损交易之间的平均空仓K线根数	平均亏损额	单笔净利 +1倍标准差
优化毛损	潜在上涨比率	1倍标准差(单笔净利)	平均盈利/平均亏损	单笔净利 -1倍标准差
账户资金额度需求	夏普比率	单笔净利 +1倍标准差	单笔最大盈利交易	
账户资金收益比	Sortino 比率	单笔净利 -1倍标准差	单笔最大亏损交易	
初始资金收益	Fouse 指数	极端交易数量	Z 值	
策略最大潜在亏损	Calmar 比率	极端交易盈亏	信心限度	
策略最大潜在亏损(%)	Sterling 比率	交易周期	策略精确预期	
平仓交易最大亏损	净利/单笔最大亏损	策略运行时间	最大连续盈利交易次数	
平仓交易最大亏损(%)	净利/单笔最大回撤	策略运行时间(%)	最大连续亏损交易次数	
策略最大潜在亏损收益比	净利/策略最大回撤(MSDD)	最长空仓期	最大连续盈利额	
盈利因子	选定的净利/单笔最大亏损	单笔最大回撤日期	最大连续亏损额	
校正后的盈利系数	选定的净利/单笔最大回撤	单笔最大回撤日期	最大连续盈利(%)	
优化盈利因子	选定的净利/策略最大回撤(MSDD)	策略最大回撤日期	最大连续亏损(%)	
最大持有合约数量	校正后的净利/单笔最大亏损	平仓交易最大亏损日期		

图 5-17

如上图，但凡接触过量化的交易者，可能见过这些连篇冗长、晦涩难懂的回测绩效各项数据术语，在这些绩效数据中，甚至有许多数据都是互相矛盾的。好多量化初学者，反而会疑惑，到底要着重看哪些数据？

上图中的绩效指标名词，一般可以分为几大类：绩效比率、周期分析、各种曲线、

极端交易分析等等。就算严格站在基金产品的角度，大部分只是回测计算结果的展示，实际应用意义不大，比如：账户资金额度需求、持有收益、信心限度等等。甚至你只需要关注重要的几个就可以了。下面我就挑选在回测绩效指标中，最重要的几个作为详细讲解。

重要的绩效指标

最大资产回撤比率 (Max Drawdown)

$$\text{Max Drawdown} = \frac{\text{Max}(P_x - P_y)}{P_x}$$

$$P_x, P_y = \text{策略某日股票和现金的总价值}, y > x$$

最大回撤计算公式如上，对于模型而言，最大回撤 (Max Drawdown) 是一个非常重要的风险指标，这个指标甚至比波动率还要重要。在回测中看到的最大回撤也在一定意义上代表你开仓后可能出现的最糟糕的状况。

从数学角度看，资金亏损 20%则需要剩余资金盈利 25%，才可以恢复原来的资金规模，如果亏损 50%，则需要剩余资金盈利 100%，才可以恢复亏损前的资金规模。

那么毫无疑问亏损的幅度越大，恢复到初始资金规模的可能性就越小，难度也就越大。资金向上的利润空间是无限的，向下亏损的空间却是有限的，触底出局的可能性也就越大。

不管怎么定义，至少这两点是目前的主流认识：

- 1、最大回撤越小越好；
- 2、回撤和风险成正比，回撤越大，风险越大，回撤越小，风险越小。

调整后收益风险比 (RAROC)

很多人对这个概念较陌生，事实上，调整后收益风险比这个指标是专业玩家与业余玩家的分水岭。这也是投行、大型基金、职业交易员非常好的评测工具，而且是全球金融领域中通用的考核标准。

$$\begin{aligned} \text{RAROC} &= \frac{\text{风险调整后的收益}}{\text{经济资本}} \times 100\% \\ &= \frac{\text{收入} - \text{经营成本} - \text{资金成本} - \text{风险成本}}{\text{经济资本}} \times 100\% \end{aligned}$$

在投资中不光只看利润，更要看在获得这些利润的时候，付出了多大的风险。一般来说，资产的风险和收益是成正比的。这意味着当模型在收益率上傲视群雄，高歌猛进的时候，其风光的背后可能隐藏着还未爆发的风险。

例如，模型中的开平仓条件或加减仓条件，在上涨时有更高的收益，可一旦出现下跌，就会把损失成倍放大，造成巨大损失。何况，上涨和下跌具有相当大的不对称影响。

很多经验丰富的量化交易者愿意为了降低风险牺牲一部分收益，在这种情况下，经过风险调整后的收益更具有参考价值。所以在回测中，风险高、波动大的模型，即使收益较高，也不一定是好的模型。

存款安全，但年收益只有 2%。市场可以让你几天赚上 50%，也可以让你几天就亏 50%。交易这么多年，我自己有一个非常重要的理念就是：正视风险，风险和收益从来不会孤立存在，交易如同出海打渔，你想打渔，却又不想承担大海的风险，是不可能的。过于保守和过于激进，事实上是走入了两个极端。设计策略模型也是如此。

交易次数

你总不能拿着几个月的回测绩效，来证明这个模型。如果回测数据过少，那么回测结果就有可能具有偶然性，要不就是参数偶然，要不就是行情偶然等等。另外较长的历史数据，也能过滤掉部分幸存者偏差。

一般来讲，对于国内的股票、商品，应该回测 5 年以上的数据，对于新上市的品种，至少也要回测 3 年。对于上市较早的品种或国际市场的黄金、美元指数等商品，则应至少回测一个牛熊周期，一般应该在 10 年 - 15 年以上。回测的期间足够长久，回测的成绩才足够可靠。对于不能满足这个要求的品种，则应在开仓时将 R 值适当加权处理，主动降低风险暴露。

平均利润

平均利润这个指标数据，是看似普通，实则非常重要的一项。它的计算方式也非常简单： $\text{净利润} / \text{交易次数}$ 。毫不夸张的说，它是分辨那些回测绩效外表光鲜的照妖镜。如下图，如果这个策略能赚钱，那就不正常了：

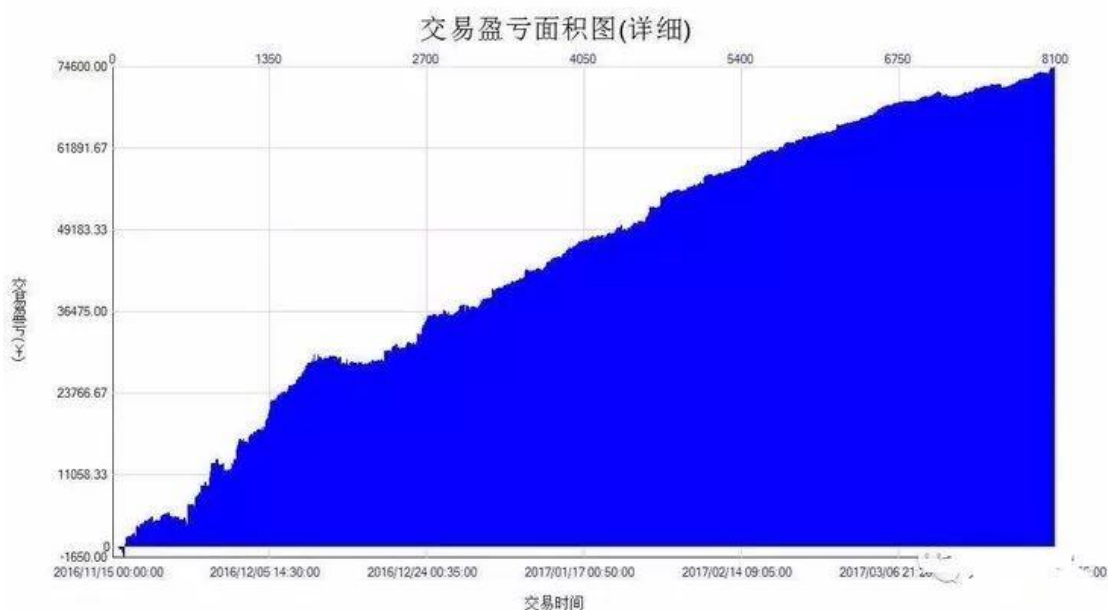


图 5-18

统计指标	性能概要		
	全部交易	多头	空头
净利润	74075.00	60575.00	13500.00
总盈利	648625.00	335925.00	312700.00
总亏损	-574550.00	-275350.00	-299200.00
总盈利/总亏损	1.13	1.22	1.05
交易手数	4122	2061	2061
盈利比率	49.39%	50.41%	48.37%
盈利手数	2036	1039	997
亏损比率	44.35%	42.99%	45.71%
亏损手数	1828	886	942
持平手数	258	136	122
平均利润	17.97	29.39	6.55
平均盈利	318.58	323.32	313.64
平均亏损	-314.31	-310.78	-317.62
平均盈利/平均亏损	1.01	1.04	0.99
最大盈利	5550.00	5550.00	4600.00
最大亏损	-5100.00	-4350.00	-5100.00
最大盈利/总盈利	0.01	0.02	0.01
最大亏损/总亏损	0.01	0.02	0.02
净利润/最大亏损	14.52	13.93	2.65
最大连续盈利手数	12	20	21
最大连续亏损手数	9	21	20
平均持仓周期	5	5	5
平均盈利周期	5	5	5
平均亏损周期	5	5	5
平均持平周期	2	1	2

图 5-19

如果你看到这个策略回测绩效，可能会有个疑问，这种近乎完美的策略，不用岂不可惜？且慢！请仔细看第二张图的平均利润，只有 17，也就是平均交易一次只赚 17 元。

就拿期货市场大多数一跳为 10 元的品种来说，但凡做过实盘交易的人就能明白什么意思。在实盘中别说一跳了，十跳八跳都有可能。两跳三跳都是家常便饭。

胜率

胜率从来都不是单独存在的，或者说单独拿胜率说问题，是不切实际的。如果你在恰好的行情用上恰好的模型，胜率达到 80% 也毫不奇怪，但这毫无意义。

价格不是涨就是跌，否则就是不动。如果时间足够长，你会发现，价格上涨和下跌的概率各是 50%。不管你用哪种类型的策略模型，如果回测时胜率超过 50%，你就要小心了。从数学和物理学的角度看，这是不可能的。

详细权益曲线(Equity Curve)

所谓一张图胜过千言万语，详细权益曲线(Equity Curve)是从第一笔进场的时间点一直到图表的最后一根 bar 的时间点结束。它是交易的实时资金曲线，说它是实时是因为它会将每根 bar 上的浮动盈亏计算在内。

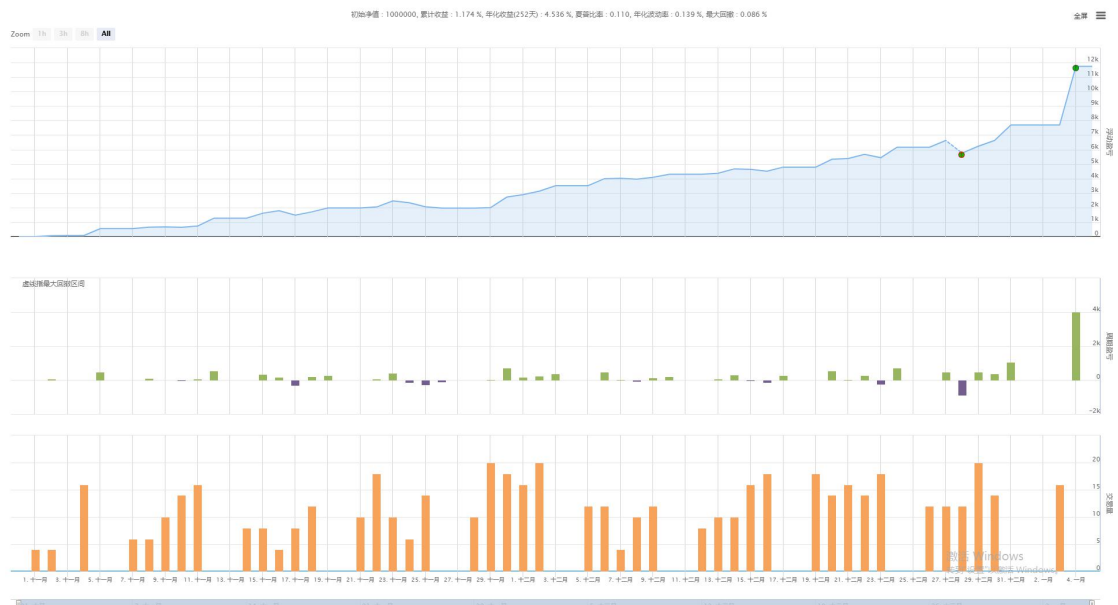


图 5-20

详细权益曲线反映的就是帐户净值的变化，是最直观的评量工具，可以一眼就对该策略亏损获利状况及损益的波动/平滑程度有概略的掌握。不过策略绩效报告这张图不仅胜过千言万语，更迷惑千万信众。另外永远不要看平仓权益曲线。

年化收益率

年化收益是一个比较争议的指标数据，有人认为它是给外行人看的，并不具备参

考意义。首先，获得盈利是模型被选用的前提，或者说模型回报本身必须是正期望值的。

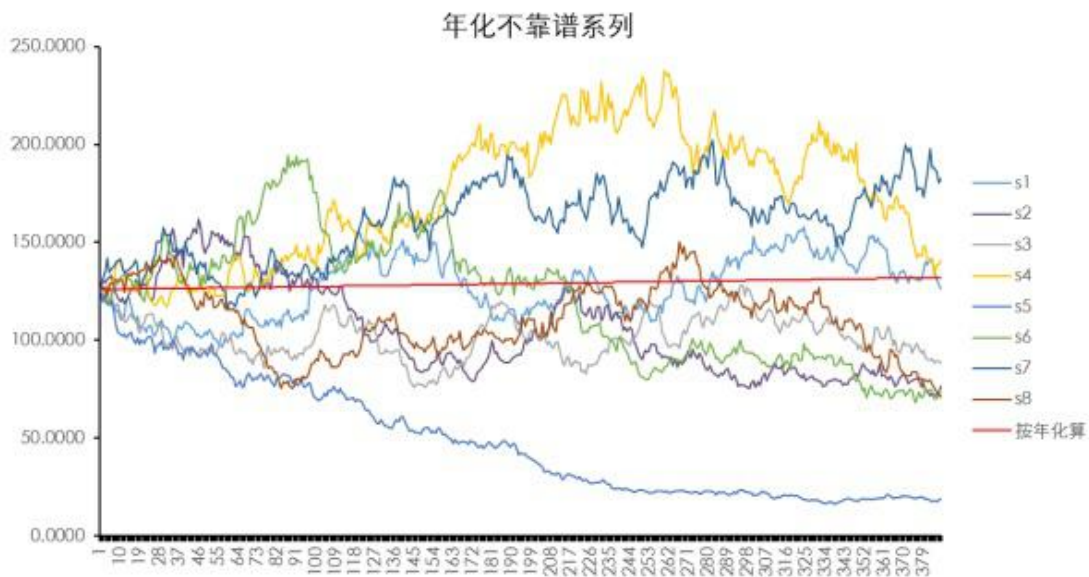


图 5-21

你可以有无数个 100% 的收益，但是你最多只能承受一个 100%。年化收益率，和真实的收益率（持有期收益率）的差距可能是很大的，有些时候大到超出我们想象。

总结

最后，有一点需要说明的是，十全十美的回测绩效并不存在，除了测试数据本身的问题以外，模型的使用方还有可能面临更多的陷阱，从参数优化到成交设计都有可能与实际运行的情况不同。

更重要的是，执行层面的情绪问题是模型投入生产的 X 因素，实盘交易不可能在“情绪真空”的环境中运行，厚尾现象是每一个程序化交易者必须时刻警惕的。

课后习题

- 1、列举出回测中，您认为最重要的绩效指标
- 2、试着计算夏普比率指标

5.4 为什么需要样本外测试

摘要

在上节中，分别围绕几个重要的绩效指标，教大家如何读懂策略回测绩效报告。其实写一个回测能赚钱的策略还不是最难的，比这更难的是，如何评价这个策略在投入实盘中是否继续有效。那么今天我将为大家讲解样本外测试，以及它的重要性。

回测不等于实盘

很多量化初学者凭借一个回测看似不错的绩效报告或资金曲线，就轻易的确信自己的交易策略，准备在市场中大展身手。诚然，这个回测结果能够完美地契合他们所观察到的某一市场状态，但是一旦这个交易策略被投入到较长时间的实战中，他们就会发现这套策略其实并不有效。

我见过许许多多交易策略，回测的时候成功率可以达到 50% 以上。在这么高胜率的前提下，还可以有 1:1 以上的盈亏比。可是，这些策略一旦付诸实盘，基本上都是亏损的。导致亏损的原因有很多，其中就有，在回测的时候，数据样本太少，导致数据迁就偏差。

然而，交易就是这样一件纠结的事情，事后回看无比清晰，但如果我们回到当初，依然不知所措。这就牵扯到量化的根源问题——历史数据的局限。那么，如果仅仅用有限的历史数据，来检验交易策略，则很难避免“看着后视镜开车”的问题。

什么是样本外测试

如何在数据有限的情况下，尽可能的充分利用有限的的数据对交易策略科学回测？答案是样本外测试法。回测的时候，将历史数据根据时间先后分为两段，前一段数据用于策略优化，称为训练集，后一段数据用于样本外测试，称为测试集。

如果你的策略始终是有效的，那么在训练集数据中优化几组最好的参数，并把这几组参数应用到测试集数据中去回测，理想情况下，得到的回测结果应该与训练集相差不大，或者相差在一个合理的范围内。那么就可以说明这个策略是相对有效的。

但如果一个策略在测试集表现很好，但是在测试集表现很差，或者变化很大，并

且选用其他参数也是如此，那么策略就有可能存在数据迁就偏差。

举一个例子，假设要回测商品期货螺纹钢，现在螺纹钢有 10 年左右的数据（2009 年~2019 年），那么可以把 2009 年~2015 年的数据作为训练集，把 2015 年~2019 年的数据作为测试集。比如一个双均线策略，在训练集中最好的几个参数组是（15 周期均线和 90 周期均线）、（5 周期均线和 50 周期均线）、（10 周期均线和 100 周期均线）..... 那么，我们把这几组参数分别放到测试集中回测，并对比训练集与测试集的回测绩效报告和资金曲线，判断它们的相差是否在一个合理的范围内。

如果不使用样本外测试，直接用 2009 年~2019 年的数据来回测策略，得到的结果就有可能因为拟合历史数据，得到一个很好的回测绩效报告和资金曲线，但是这样的回测结果对于实盘意义不大，并没有指导作用，特别是那些参数比较多的策略。

样本外测试的进阶

除了把历史数据分成两份，进行样本内和样本外回测外，其实还有一个更好的选择，那就是递推式回测和交叉式回测方法。特别是在历史数据很少的情况下，比如近年刚上市的原油期货、苹果期货，采用这两种方法可以利用有限的对模型进行全面的检验。

递推式检验的基本原理：用前一段较长的历史数据去训练模型，并用随后相对较短的数据去检验模型，然后不断地向后移动取数据的窗口，重复训练与检验的步骤。

训练数据：2000 年至 2001 年，测试数据：2002 年；

训练数据：2001 年至 2002 年，测试数据：2003 年；

训练数据：2002 年至 2003 年，测试数据：2004 年；

训练数据：2003 年至 2004 年，测试数据：2005 年；

训练数据：2004 年至 2005 年，测试数据：2006 年；

... 以此类推...

最后对（2002 年、2003 年、2004 年、2005 年、2006 年...）的测试结果进行统计，来综合评估策略表现。

如下图，可以直观的解释递推式检验的原理：

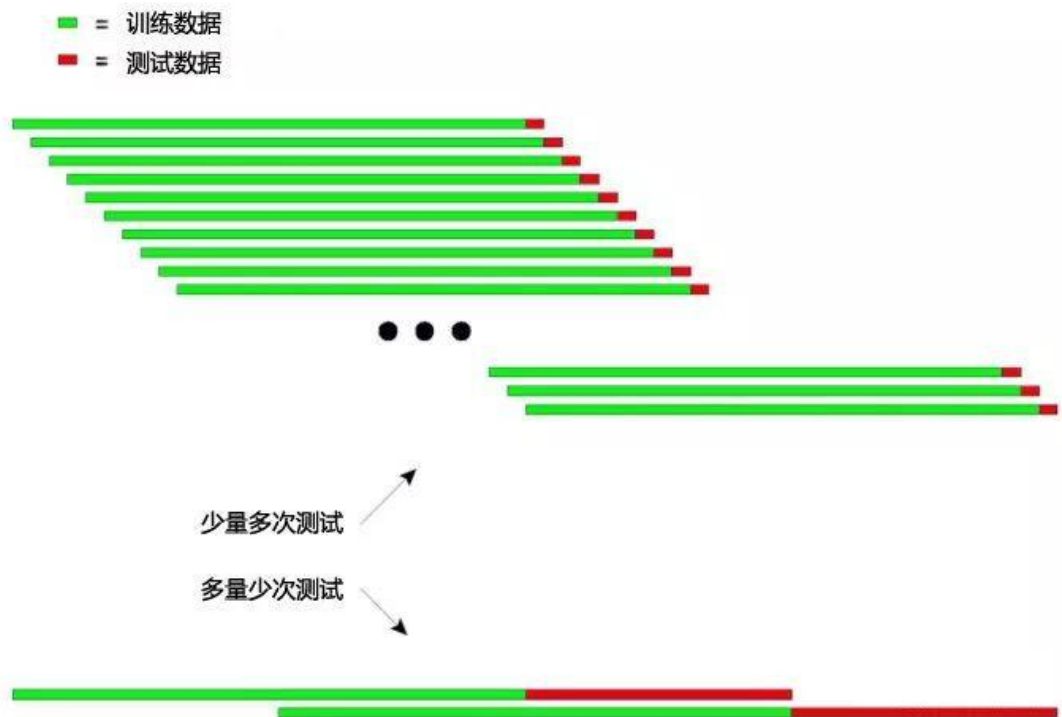


图 5-22

上图分别展示了递推式检验的两种方法。

第一种：每次检验时，测试数据比较短，测试次数较多。

第二种：每次检验时，测试数据比较长，测试次数较少。

在实际应用中，可以通过改变测试数据的长度，进行多次测试，用来判断模型在应对非平稳数据的稳定性。交叉式检验的基本原理：把全部数据等分为 N 个部分，每次用其中的 $N-1$ 个部分做训练，用剩下的部分做检验。

把 2000 年至 2003 年按照每年划分，分为 4 个部分。那交叉校验的操作过程如下：

- 1、训练数据：2001-2003，测试数据：2000；
- 2、训练数据：2000-2002，测试数据：2003；
- 3、训练数据：2000、2001、2003，测试数据：2002；
- 4、训练数据：2000、2002、2003，测试数据：2001；

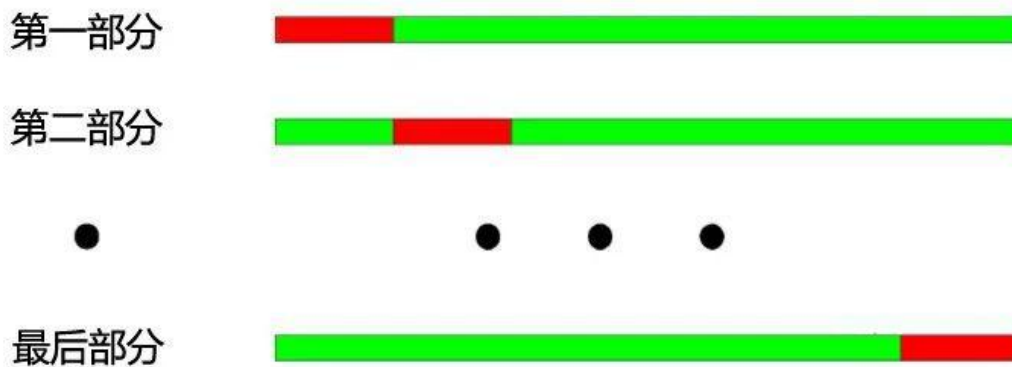


图 5-23

如上图所示：交叉式检验最大的优点就是充分的利用有限的的数据，每个训练数据同样也是测试数据。但交叉检验应用到策略模型的检验时也存在明显的缺点：

- 1、当价格数据非平稳时，模型的测试结果往往不可靠。例如，用 2008 年的数据做训练，用 2005 年的数据做测试。很有可能 2008 年的市场环境 与 2005 年相比发生了很大的变化，所以模型测试的结果不可信。
 - 2、与第一条类似，在交叉检验中，如果用最新的数据训练模型，而用较老的数据测试模型，这本身就不怎么符合逻辑。
- 另外，在对量化策略模型进行检验时，无论是递推式检验还是交叉式检验都遇到到数据重叠的问题。

在开发交易策略模型时，大部分的技术指标是基于一定长度的历史数据。例如，利用趋势性指标，计算过去 50 天的历史数据，而下一个交易日，该指标又是该交易日前 50 天的数据计算得出，那么计算这两个指标的数据有 49 天是相同的，这会导致每相邻两天该指标的变化很不明显。

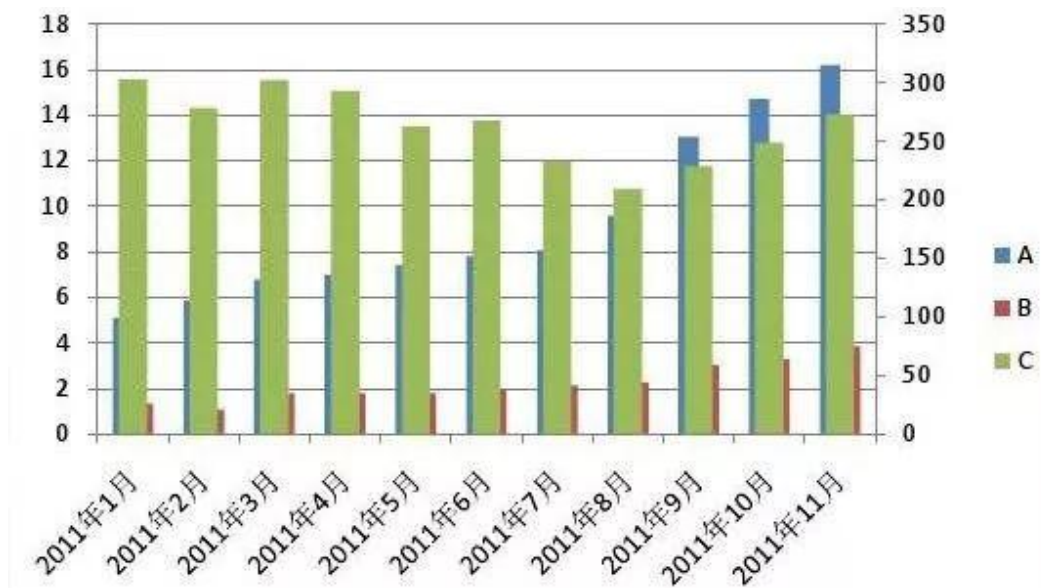


图 5-24

数据重叠会导致以下影响：

- 1、模型预测的结果变化缓慢导致持仓变化缓慢，这就是我们常说的指标的滞后性。
- 2、对模型结果检验的一些统计值不可用，由于重复数据导致的序列相关，使得一些统计检验的结果不可靠。

优秀的交易策略应该能够在未来具有获利性。样本外测试，除了能客观检测交易策略外，更能有效率节省量化交易者的时间。大部分情况下，直接采用全部样本的最优参数，投入实战是非常危险的。

如果对进行参数优化的时间点前的所有历史数据进行区分，划分为样本内数据与样本外数据，先利用样本内数据进行参数优化，再利用样本外数据进行样本外测试，则可以将这种错误排查出来，与此同时还能检验优化后的策略是否适用于未来的市场。

总结

就如同交易一样，我们永远没有办法穿越时间，为自己做一个一点错误都没有的正确决定。如果有上帝之手或者从未来穿越回来能力，那么不经过测试，直接上线实盘交易，也能赚的盆满钵满。而我等凡人，则必须在历史数据中检验我们的策略。

可是，即便拥有庞大数据的历史，但面对浩瀚无尽且不可预测的未来，历史就显得极度匮乏。所以基于历史自下而上倒推出来的交易系统，终究会随着时间而沉没。因为历史不能穷尽未来。因此一个完整的正期望交易系统必须由其内在原理、逻辑所支撑。

「信任，但要验证」——里根总统

课后习题

- 1、现实生活中有哪些现象是幸存者偏差？
- 2、利用发明者量化工具，根据样本内外回测，并比较它们的不同。

5.5 交易策略优化及最佳化

摘要

交易策略其本质就是对市场规律的概括和总结，你对市场认识的越深，用代码表达思想的能力越高，你的策略就越贴近市场。本节将继续为大家讲解如何优化交易策略，为你的实盘交易做最后的准备。

优化进出场

大部分趋势跟踪策略会利用突破或者技术指标等方法捕捉行情，通常情况下这些信号的进出场方式，时效性较低，如果策略使用的是收盘价模型，那么入场点会在下根 K 线的开盘价上，因此会错过突破这根 K 线的最佳入场时间，无形中会错过很大一笔到手的利润。

所以有效的办法是，在策略实现中使用更具优势的即时价格，当出现信号时，立即发单。这样当信号成立时可以立即入场，不会错失利润。但并非所有的即时价格都要优于收盘价，这还要根据交易策略决定。一些交易逻辑简单的策略，即时价格与收盘价效果区别较小。但收盘价模型无法处理更加细致的交易逻辑，就需要采用即时价格了。

参数优化

参数优化可以使量化交易策略更贴近历史数据，回测绩效达到更好的结果。举个例子：我们在螺纹钢合约中，使用一个双均线策略中，但到底使用哪两根均线最好呢？那么可以利用发明者量化工具中的参数调优功能，自动寻找最好的两根均线参数。

如下图，以双均线策略为例，它本身是一个多维实例，如果我们把每个参数的回测结果画成一个点（注意看下图），那么每个参数都是这个策略的一个维度，最终所有的参数组合构建了这个复杂的多维曲面形状（就像一座山峰）。

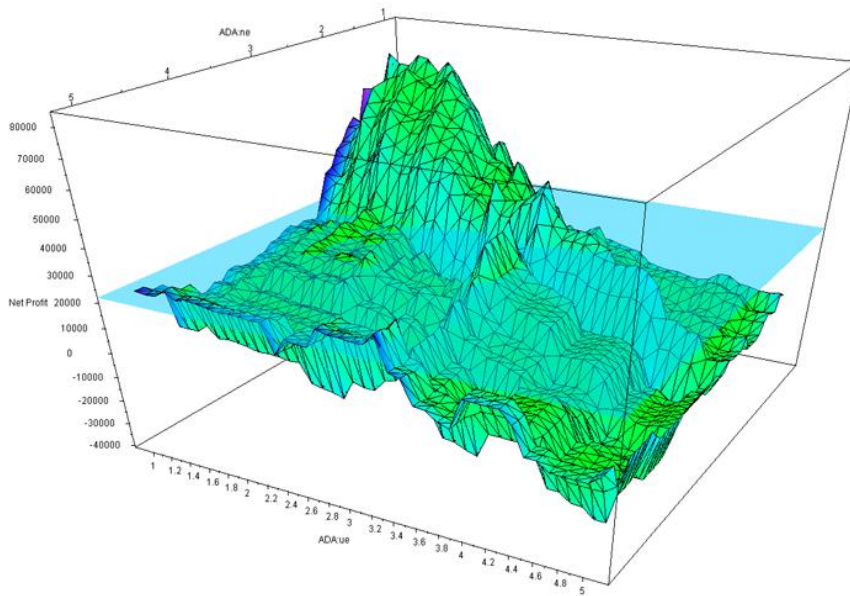


图 5-25

如上图，这是一个双参数策略绩效表现图，随着参数的不同，最终收益率也发生较大变化，曲面发生强扭曲，形成高低不同的“波峰”和“波谷”。通常优化结果收益率第一名，是全部曲面的最高点。但从参数敏感性、客观性等角度讲，有时候这个结果可能不是“最优”的。因为行情是在不断变化的。

所以，参数优化重要的原则就是要选择参数高原而不是参数孤岛。所谓参数高原，指的是存在着一个较宽泛的参数范围，策略在这个参数范围内都能取得较好的绩效。一般会以高原的中心形成类似正态分布状。而所谓参数孤岛，指的是只有参数值处于某个很小的范围内时，策略才会有较好表现，当参数偏离该值时，策略的表现就会显著变差。

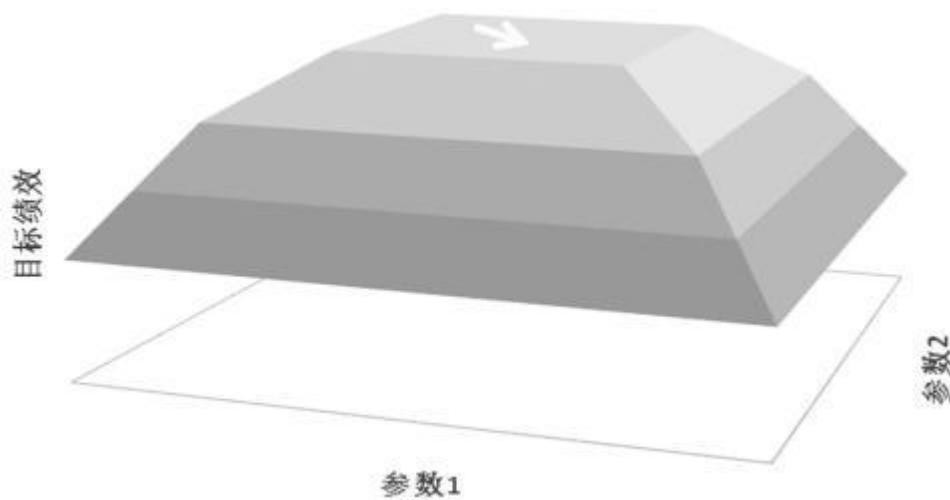


图 5-26

参数高原

以上图为例，好的策略参数分布应当是参数高原，即使当参数的设置有所偏差，策略的获利能力依然能够得到保证。这样的参数因稳定性强，可以使得策略在未来实战中遇到各类行情时，具有较强的普适性。

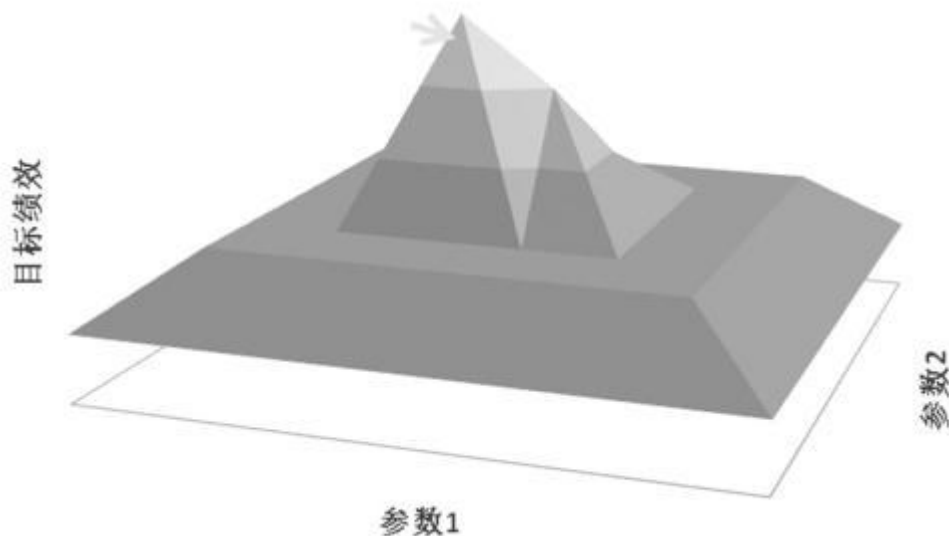


图 5-27

参数孤岛

以上图为例，如果回测绩效呈现参数孤岛，当参数发生小的偏移时，策略的获利能力将降低很多，那么这样的参数因普适性较差，往往难以应对实际交易中变化多端的市场行情。

因此，如果附近参数的性能远差于最优参数的性能，那么这个最优参数可能是一个过度拟和的结果，在数学上可以认为是奇点解，而不是所要寻找的极大值解。从数学角度来说，奇点是不稳定的，在未来的不确定行情中，一旦市场特征发生变化，最优参数可能会变为最差参数。

增加滤网

很多趋势策略，在行情出现趋势的时候，都能很好的抓住趋势，实现丰厚的回报，但是长期运行下来，最终的结果不是小赚就是亏钱，问题出在哪里？

原因在于，策略在震荡行情中不断的反复交易，而震荡的交易大部分是亏损或者小盈利的，市场有 70%左右的时间是处于震荡行情，长时间的连续小幅度亏损，

导致之前的利润全部回吐。



图 5-28

解决办法就是增加滤网，市场上的滤网有很多种，包括损益滤网、风险值滤网、走势型态滤网、技术指标滤网等等。比如增加一条大周期均线滤网，在震荡行情中能降低交易次数，过滤掉一半的错误交易。

平滑资金曲线

量化追求的是一种稳定的可持续盈利方法，这是绝大多数交易者所希望看到的，谁也不希望今年赚 50%，明年亏 30%，后年再赚 40%，宁愿接受每年 20%的收益率，但是可以持续十几年。这是量化投资可以做到的。因为量化投资是一种业绩可以持续的交易模式。

要想做到平滑的资金曲线，就需要多策略、多品种、多周期、多参数的构建投资组合。但未必是越多越好，这里有一个边际递减的效应，一开始加入组合越多，分散性越好，但是当策略达到一个数量级后就开始出现分散作用递减的效应了。组合的好处就是分散，虽然整体收益率不是最高的，但是最稳健的。

放弃寻找圣杯

究竟能不能用量化交易找到圣杯，这是很多交易者都会考虑的问题。有些交易者更是在简单的回测之后，就拿着所谓的完美策略冲进市场中。希望能够屡战屡胜，成为一路过关斩将的职业宽客。

但到底有没有圣杯？其实是非常简单的，答案就是没有。其实理解起来也并不困难，如果这个市场真的有规律，那么理应是智商越高、学历越高、越努力的人将会发现其中的规律，无论是用数学分析、信息垄断、还是其他的分析方法，最终他们将会赚到市场中大多数钱，长此以往，这些人就会垄断交易市场，直至市场无法正常运行。

总结

如果交易的时间足够长，任何人都可能在交易过程中面临各种各样的行情走势，而且这些走势也不可能完全重复过去。作为量化交易者，除了正确审视优化自己的交易策略外，还需要持续监控市场状态，针对市场的变化，不断完善策略。

同时也要意识到盈亏同源，亏损是整个交易策略中的一部分，即使是最好的交易策略也可能经历一系列回撤期，当每次交易有亏损的时候，你的交易规则和策略不应该受到质疑。至少不要轻易改变你的策略逻辑框架，除非一开始你的逻辑框架就是错误的。

课后习题

- 1、根据自己策略的特点，构建投资组合，并用发明者量化工具回测
- 2、试着根据本节的内容优化自己的量化交易策略

5.6 建立概率思维，提升你的交易格局

摘要

交易既是一门科学，也是一门艺术。交易中的方法有很多，无论是价值投资、技术分析、事件热点、套利对冲等，表面上看起来逻辑严谨，理论上也能说得通。但实际上往往相互矛盾，有时候，科学的严谨也无法解释艺术的天马行空。

虽然各种交易方法起点不相同，但条条道路通罗马。价值投资的优势是可以根据价值给价格波动划分一个安全边际；技术分析的优势是三大假设使交易具有一定的科学性。

但是，它们都有一个共同的特点，那就是：对未来的价格分析，只能做到大概预测，而不能精准预测。即使将基本面分析与技术分析相结合，也不能解决提高“精准”的问题，所以自始至终交易都是一个概率游戏。

概率游戏

其实，不仅仅交易是概率游戏。人这一生，小到过马路（绿灯了，现在过马路安全吗？），交什么样的朋友（这个朋友靠谱吗？）；大到从事什么样的事业（职业交易真的是一个好的事业吗？），跟什么人结婚（我们在一起会幸福吗？）等等，都是评估风险与回报的概率游戏。因为我们没有未卜先知的能力，每做一件事即使我们再有把握，风险都始终存在，无法做到百分之百确定性。

许多人在交易中犯错的重要原因就是缺少概率思维，在做交易时过于感性而非理性。感性其实就是我们的原始本能，在市场中，这些原始本能可以激发人的许多弱点，并且成倍放大。这也是大多数人来到市场，最终以失败而告终的原因。

交易失败的原因

原因一：因为人性

绝大多数人都有一个弱点：喜欢占小便宜，害怕吃小亏。在市场中一旦有点蝇头小利就马上兑现，获利出局；一旦有亏损，就抱着亏损的头寸不动，企图侥幸回本，结果小亏损慢慢积累成大亏损。

价格不是上涨就是下跌，否则就是不动。长期来看，在不考虑手续费和滑点等情

况下，赚钱与赔钱的概率约等于 50%，那么绝大多数人的交易方法也就变成了利润有限而风险无限的负期望策略。它们的交易结算单应该是这样的：小赚>>.....>>小赚>>大亏。

在现实生活中，这与穷人思维和富人思维很相像。穷人厌恶风险，害怕亏损。喜欢旱涝保收的工作，追求安稳。即使做一件事，没有绝对把握，也坚决不做。表面上看起来这样做并没有什么错，但是背后却承担了巨大的机会风险。

而富人更愿意承担风险，知道风险与回报始终成正比，只有风险中才孕育着机会，合理评估风险，并在风险可控的情况下，勇敢下注。

原因二：喜欢赚快钱

国外有家机构曾经做过一个统计，长期而言，大多数行业的净资产年化回报率很难超过 15%。相反，很多散户认为在市场中赚个 15%，都不好意思跟人打招呼。人们喜欢赚快钱，行动上就是重仓交易和短线交易。

重仓

重仓、高杠杆、配资都是非常诱人的，也是非常危险的。成则飞黄腾达，败则万劫不复。假如你有一个胜率为 50%的交易策略，满仓加配资操作，运气好的话，你可能连赢十几次，财富从量变到质变也是可能的。

但是只需要错一次，就全部归零。即使你只是重仓操作不配资，也有账户归零的风险，因为你无法保证，在接下来的行情中，会不会连续亏损十几次。甚至重仓交易还能使一个原本正期望的交易策略，变成了一个输赢不对等的策略。

短线

天下武功，唯快不破。在交易圈人工炒单、日内短线、量化高频一直都带有很神秘的色彩，我并不是怀疑这些看着秒表做交易的人，而是试图从另一个角度，劝你放弃短线交易。

我们判断一个方法是否可行，并不能只看用这些方法成功的那些人，更要看用这些方法失败的那些人。也就是说，你不能因为一部分人买彩票中大奖，就认为买彩票就是正期望的策略。

再者，看看私募产品排行，三年以上，排名前 100 名里面又有几个是做炒单或短线的？毫无疑问，短线成材率很低，就算成功，这种快速赚钱的方法也很难长期维持下去。如果你不是天赋异禀，慎用这类奇门绝招，毕竟西蒙斯只有一位。

原因三：偏见

如果可以，建议你花费 100 分钟，看一部电影——《十二怒汉》。一部电影由 4 个国家翻拍，1957 年美国初版，1991 年日本版、1997 年俄罗斯版、2014 年中国版。虽然这部电影并不能教你怎么去做交易，但却教你以什么样的态度看待事物，学会认识自己，这一点非常重要。

因为人的经历是有限的，所以人的认知也是有限的。每个人都或多或少，会根据自己的经历和经验，产生偏见。很多时候，偏见成为了大多数人的习惯，理所当然地带着自己的情感去评判很多事情。

回到市场，无论你对市场的判断是基于基本面分析还是技术分析，这些其实都不重要。如果你的观点与市场大多数人的观点不同，价格则更偏向于市场大多数人，市场不会以你的观点去运行。

所以，在交易中一定要牢记“判断，但不依赖判断”，最终还是要基于事实，基于价格。价格涨跌的唯一力量是大多数人对未来的预期。而你的判断在市场中并没有什么分量，千万不要让自己的判断形成自己的偏见。

原因四：追求完美。

市场的参与者有各行各业的大牛，有物理学、统计学、数学、天文学等等，很多人都试图用他们的专业知识，去解释这个市场。

但市场的参与主体是人，人本身就是有认知局限的，也就是说市场本身就是错的，是不完美的。那么又怎么用这些「完美」的方法来解释市场呢？这不就违背了市场的本质了吗？

以上列出了来到市场的绝大多数人，最终以失败告终的原因。除了上述几个主要原因外，还有很多因素，这里不再一一列出。总之，除了你必胜的信心外，其他的都是阻止你成功的绊脚石。

那些因为好运气而在市场里面赚到钱的人，也会随着时间的推移，终究会再还给市场。所以，期货市场是一个负和游戏的市场。只有转变自己的思维方式，建立自己的交易策略，才有成功的可能。

什么是概率的思维方式？

概率思维，是一个文绉绉的名字，说得通俗点就是一一赌博思维。你没有听错，交易就是赌博。听到赌博，你可能会联想的「谁谁谁赌博输得倾家荡产或欠债跑路或妻离子散」，避而远之。

社会上也确实存在一些赌红眼的赌徒。但是赌博≠赌徒。「赌博」可能是被人们误解最深的词汇之一了。如果你的策略是负期望，就是赌徒；如果你的策略是正期望，就是赌博。

如果我们把「赌博」中的贬义去掉，将之理解为承担一定风险而获得一定回报的活动，那么人生真的处处是「赌博」。上学选择哪个专业、买不买房、项目上不上马、打工还是创业等等。

甚至把钱存到银行也是赌博，因为你不确定未来是否会通货膨胀，银行是否会破产（参考希腊债务危机）。总之从摇篮到坟墓，生命的每个过程都是在赌博。

如何才能久赌必赢

有个赌博这个概念，就需要进一步解决，怎样才能久赌必赢？在研究久赌必赢策略之前，我们先来研究一下，那些久赌必赢策略的原理。除了印钞机，还有什么能久赌必赢的呢？

那就是赌场里面的：百家乐、轮盘赌、老虎机、21点等等，不管怎样变换玩法，赌场最终都会赢。这里面其实隐藏着一个赌场从来不说秘密：大数定律。

骰宝游戏的原理

三个骰子，押大小，4-10 是小，11-17 是大，押对了就赢钱。而骰宝有一种围骰，就是三个骰子点数相同，赌场庄家通杀，围骰出现的概率是 2.8%。那么出现大和出现小的概率就各是 48.6%。赌场就是靠这 2.8% 的概率，如果每个赌客每局都押 100 元，玩 100 局赌场就会赢 280 元。

$$(0.486+0.028)*100*100-0.486*100*100=280$$

但是这个赌场策略是有漏洞的，万一一个大玩家心血来潮押个几百亿，恰好又赢了，赌场就一下子破产了。所以，赌场会设置一个下注上限，本轮超过这个上限就不能再下注了。这样就算赌客可能一时运气好赢钱了，长期下去，还是会输给概率，在无限多次的骰宝游戏中，赌客就会输掉 2.8% 的钱。

大数定律

赌场老板的优势仅仅比赌客多 2%，在单次赌博中，老板可能是亏损的，甚至也可能遇到连续亏损。但是赌场老板并不会被亏损吓坏，因为他知道，自己之所以

能赚钱，正是「大数定律」在其中起作用，只要有人继续在赌，只需要 2% 的微弱优势，就能长期稳定盈利下去。

所以赌场不怕你赢钱，就怕你不来。这么多年来你甚至听说过银行倒闭的，但你什么时候听过赌场倒闭的？长期来看赌场永远都是赢家，这就是久赌必赢。

类似的久赌必赢的例子还有：各种彩票。彩票的奖池资金，自彩票上市以来是越积越多，这些钱当然来自于广大彩民。你知道双色球中 500 万的几率是多少吗？答案是 1770 万分之一。

概率的变化

假设有一个正反面一样重的硬币，抛出字（背面）和花（正面）的概率都是 50%，而且每次抛硬币与前次结果无关。连续地抛 10000 次这个硬币，那么出现正面的概率约等于 50%。

但是如果只抛 10 次，则出现正面的概率就变了，这个概率就不一定是 50% 了。所以赌场庄家必须保证触发这个正期望策略的次数足够多，这个正期望的策略才有效。这也是私募机构在开启量化交易策略时，除非特殊条件，不能停止策略的原因。

如何利用「大数定律」在金融市场中创建一个久赌必赢的策略，将是我们下个系列的课程内容，尽情期待！

总结

以上我们从概率、交易失败的原因、正确的交易思维方式、赌博中久赌必赢的原理等方面，为大家讲解如何用科学的方法看待交易。相信如果你学有所长，思维的转变将是你行为的转变，而行为的转变将是你成功的转变。

课后习题

- 1、为什么说交易是概率游戏？
- 2、交易失败的原因还有哪些？