TURING

4

2013/02/27

码农

codeMaker()

活的信息

冯大辉：信息真正的意义

信息是神马

用编程工具实现数据可视化

量化分析：用信息讲故事

三天准备，迎战千倍用户

程序员创造的虚拟宇宙

DRM-free，不止拥抱这一天

编者的话

活的信息

编者 / 李盼

信息是什么？信息论里说，信息消除不确定性。通信系统划分为信源、编码器、信道、解码器、信宿五部分，码农们更善于和机器打交道，所以当信宿是一台台图灵机的时候，消除这样的“不确定性”是码农们的看家本事。但是，当信源和信宿都变成了一个我们所能接触到的最复杂的系统——“人”的时候，我们的 IT（信息科技）高手们有些犹豫了，因为信息消失在了神秘的黑匣子里。

信息的旅程并没有就此结束，而是以一种新的面目重生了。我们用仍在探索中的知识（科学）来解释作为信源和信宿的人。让人以人的方式接收信息，发出信息，以人为本无疑是信息时代的主旋律。

可信息时代哪里离得开编码呢？编码是砖、是瓦，是构成一切高大建筑的原子。在这里，码农作为编码的手艺人不断地精进着最巧妙的技艺，完善着最细琐的工作。本期《码农》也提供了一两块这样的磨刀石。

信息和数据并不死板，他们都是活的。当他们脱离了信息系统，就变成了莎翁笔下的哈姆雷特，在一千个人的眼中呈现出一千种姿态。甚至对于量化的数据来说，也没有绝对唯一的真理。本期，让我们跟随码农人物冯大辉的脚步来体会他眼中信息的含义。他虽然已经是数据库领域的顶级专家，但却开始了新的征程，踏上了他自己的探寻信息真谛之旅。

前路依然坎坷，希望风景越来越好。

目录

专题：活的信息

1 信息是神马——论信息的消失与重生
10 信息可视化的科学
17 用编程工具实现数据可视化的几个选择
32 用 R 语言处理大数据：课程 101
44 iGraph——图挖掘助力社会网络分析
59 量化分析：用数据、指标、信息讲述的故事

人物

67 冯大辉：信息真正的意义

践行

82 三天准备，迎战千倍用户

九卦

90 码农餐厅（一）

鲜阅

97 宇宙一种：程序员创造的虚拟宇宙

出版的未来

106　DRM-free，不止拥抱这一天

书榜

112　2012 年十大不能错过的好书

妙评

116　《图灵的秘密》
121　《程序员的数学》

成书手记

125　听《量化》译者讲“故事会”

社区动态

135　iTran 乐译 13 期
136　好运电子书第 1 期
137　“听”书计划

专题：活的信息

信息是神马

——论信息的消失与重生

作者 / 赵丹

做了十几年码农，写了百来万行程序，深感码农不是一个人道的职业。因此以实现机器全自动编程，彻底消灭码农职业作为终生奋斗的目标，在达到目标之前，继续坚定不移地以码农为职业。现在一家中等规模的技术公司担任总架构师，专门解决未曾解决过的问题，做没人

信息不是神马，只是编码

最近常看到讨论信息该如何定义的话题，搞来搞去没有大家都认可的说法。在我看来，信息论已经很好地解决了这个问题，只不过不能按照香农字面上讲的意思去看，要重新解释一番才可以。

信息论里面说，信息是消除不确定性，这句话表面上看很好懂，也大致符合一般人的经验，得到了信息，了解了情况，就没有不确定了。我想香农同学原本可能不想这么写，但是在一篇专讲信息的论文里，如果不给信息弄个简单明了的定义恐怕说不过去。他的整篇文章，除了这个地方以外，在数学上都是讲得通的，唯独这里，稍微仔细想想就会出问题。

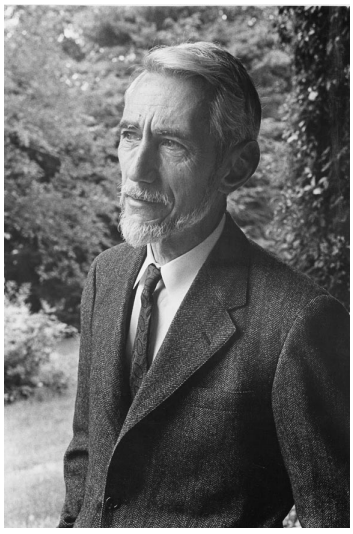
从另一个角度说，通信系统划分为信源、编码器、信道、解码器、信宿五部分，信息论解决的问题是信源之后，信宿之前那部分发生的事情，只保证发出的一串编码和被接收到的一串编码之间有某种确定的数学映射关系，而不管信源到底发了什么，信宿又到底收到了什么。这就好比一个可靠的邮差，只保证把信原封不动

做过的设计，写没人写过的程序。 喜欢各种不太剧烈的体育运动，比较擅长瑜伽、游泳、乒乓球、定向越野。喜欢徒步和骑车旅行。 喜欢讨论哲学问题，喜欢各种新思想，新观点，有人总结为年龄 70 后，外表 80 后，思想 90 后。
图灵社区 ID：赵丹 Diana Zhao

地从发信人送到收信人，而不问信里写的是不是人话，收信人认不认得字，那两个人能不能相互传达意思。如果我们承认信息论所讲的通信系统里传送的东西是信息，也知道这个信息到底有多少比特的信息量，那么这个信息到底从信宿那位同学心里消除了多少不确定性呢？答案只能是不知道。因为信息一旦到了信宿手上，它就从客观世界中消失了，进入了信宿同学的主观世界，就再也说不清楚了。

电子计算机是上世纪人类的伟大发明，是当代一切信息技术的基础，然而专门搞这个的都知道，不管多复杂的计算机，也不过就是一台外表花哨的图灵机而已，它所做的那些看起来很了不起的事情，不过就是把一串符号转换成另一串符号而已，然而很多人却认为它在做信息储存、信息加工、信息提取、信息发现乃至信息创造等一系列高深莫测的事情。居然有很多人承认计算机可以作各种高级的关于信息的事情，而不只是个符号变换器，这个事实说明什么呢？我看只能说明信息这个词被我们滥用了，早已不是信息论里面那个单纯的可以用数学度量的信息了。如果只是茶余饭后闲扯淡，那么一个名词术语无论怎么滥用都不要紧，甚至还会产生某种幽默感，但是要装模作样地当作科学来讨论，那就麻烦大了。所以要研究信息，就只能研究那个单纯的可以用数学表述的信息，既不能允许有一点模糊，也不能允许它跑到主观世界里去，因此就只能是五要素通信系统当中，砍掉两个端点之后的那条线段上流淌着的东西，也就只剩下了编码。

当然香农的理论里，常常要提到信源信宿，提到如何消除了不确定性，不过我想认真看过文章的人都知道，他所说的信源信宿在真实世界中是找不到的，是为了谈论理论而虚构出来的。虚构的信源和信宿必须对概率空间达成完全一致的认识，才能有效通信，



克劳德·艾尔伍德·香农（Claude Elwood Shannon），美国数学家、电子工程师和密码学家，被誉为信息论的创始人。1948 年，香农发表了划时代的论文通信的数学原理，奠定了现代信息理论的基础。

可是要达成这种一致认识是不是先要靠通信才能实现呢？这就掉进了循环论的陷井，再也出不来了。事实上，人类个体之间之所以能够通信，依靠的是每个个体都具有所谓的人类常识，并且假定其他人的人类常识和自己的那个是一样的，然而这种假设其实是站不住脚的，每个人的常识都和别人有所不同，都有自己独特经历的影响在里面，正是因此，人类彼此间才会有很多误解，才会很难沟通。

上文书说道，信息不是神马，只是编码。离开了信道，进入信宿，信息也就消失了。前面我们没有详细探究信源和信宿，只说道那都是香农同学杜撰出来的。然而香农同学杜撰出来的东西既然大家都能理解，那就必然有一定的现实性基础，这次我们就来好好研究一下这两个家伙。

信源和信宿要对概率空间的组成，以及先前某一时刻的状态有完全一致的认识，这是香浓理论的基础。只有这样，在后来概率空间发生变化的时候，信源得知了这种变化，并将这种变化编码发送给信宿，才能消除信宿对概率空间认识上的不确定性。

概率空间是一种抽象思维的产物，而不是自然界中客观实在之物，这本是很清楚明白的事情，但是很多人却在这个地方思想产生了混淆。就拿最简单的抛硬币试验来说，不论怎么叙述都要涉及这么几个抽象概念：质量、均匀、硬币、正面、反面、朝上，它们构成一个抽象的数学模型。现代社会中的受过一定文化教育的人理解这些概念，以及整个模型并不困难。但是如果我们设想一下，要把这整件事解释给一个原始部落中的人类，会不会是件轻松的事情呢？其实部落人类算是很容易沟通的对象了，毕竟大家都住在地球上适宜人类居住的区域，有结构和功能完全相同的大脑和

身体，共享基本的人类常识。可见，很多我们看起来理所当然的事情，其实并不那么理所当然。

在解决了通信主体对概率空间的理解问题之后，才能继续谈论信息论后面的问题。香农提出用熵来度量信息量，也就是消除掉的不确定性的量。概率空间的构造是千变万化的，对于同一事件，理解的角度不同，就可以构造不同的概率空间，不同的概率空间就意味着不确定性数量上的巨大差异。催生出邮票的发明的那个爱情故事，当事人双方约定的概率空间，只需要 1 比特的信息量就可消除不确定性，而如果是普通的爱人之间的书信，稍微多纳入一些事件的细节，为此改变一下概率空间，可能就要成千上万比特了。说句题外话，这个故事也说明：任何企图提高编码信息量的技术努力，都不如通信主体的主观能动性效果显著。

因为概率空间有无限多种可能，研究向人类这样复杂多变的智能通信主体之间的通信是很困难的，据我所知，大凡搞自然语言理解的都没有好下场，不是已经死掉了，就是快要死掉，苟延残喘。香农同学的聪明之处就在于避开了这个深不见底的大坑，引入了虚拟通信主体。对于虚拟的通信主体来说，其实并不需要考虑如何构造概率空间的问题，虚拟通信主体间的通信没有那么多猫腻，用不着考虑怎么才能赖掉邮费。虚拟主体只是傻乎乎地把一串符号从这里传到那里，决不问这些符号代表什么。这样一来，概率空间就只由所传递的符号的特点唯一确定。这是一个简单、先验的存在，可以放心地把处理算法烧进芯片里，几十年不变。于是世界清静了。

既然对于虚拟主体来说，符号没有含义，那么这些符号组成的整条信息也理所当然没有含义，这件事情显而易见，却常常被人忽略，

很多人会不自觉地把自己当成通信主体，然后对符号望文生义地解释一番，于是乎对信息论理解的很多谬误就由此而生。其实信息论只是一个关于如何传送符号的理论。不应过度解释。

信息之殇

信息熵，或者信息量，是由概率空间的性质决定的。前面说过，信息论当中的通信主体都是虚拟的，概率空间都是简单纯粹的，因此信息量就是个简单的数学问题。然而人们远远不满足于就此止步，而是把这些概念和理论扩展到了远比香农当初的设想复杂得多的领域和场合。这样做的结果就是——天下大乱！

信息本来就只是编码，和真实世界没半点关系，概率空间只能反映离散世界，对于连续性无计可施。把连续的世界变成离散的世界模型，这个可不是虚拟的信源信宿能够做到的事情，且不说这个理论出来的时候信源和信宿都只是笨拙的电子管电路，就是今天最 NB 的处理器一样也不行。因为机器就是机器，是学不会抽象思维的。

抽象思维能力，即使对人类来说，也不是理所当然，与生俱来的。人不是从有了语言那天起，才可以相互沟通的，在那之前，也可以通过表情、动作、声音、接触等进行一定的沟通。但是这种沟通没有有意识的编码解码过程，既然不存在有意识的编码，也就不是通常意义的信息沟通，因此没有必要刻意构造概率空间；事实上，这种沟通也是有概率空间的，只不过这个概率空间是人脑神经网络从生活经验中自发形成的，没有明确的定义和严格的形式，也就无法应用信息理论加以解释，沟通的结果也不可预测。

语言为沟通规定了标准的形式，尽管从工业社会的眼光来看，它还远远不够标准，但是不管怎样，人类之间的沟通从此符号化了，于是至少在理想条件下，在语言被解释出具体含义之前，它在通信系统中的传递是有保障的，噪音和畸变是可以识别出来并加以去除的。也就是说，至少可以保证信宿听到的话就是信源说出的那一句，而不用考虑信源的地方口音或者信宿的听力下降对此有什么影响（这当然指的是信源和信宿的语言概率空间完全一致的理想情况，现实中语言是不同的人类个体各自习得的，很难完全一致）。语言本身的概率空间，尽管具有复杂的结构，但至少是有限的，可以构造出来的，并且是不经常改变的。

然而信宿想要得到的，绝不仅是一串语言符号，而是理解语言的含义，从而知道自己希望知道的事实。一牵扯到语言的含义，信息论就很难应用了，因为人脑具有构造无穷多个不同概率空间的可能性。发出什么信息是由信源的通信意图，结合他对信息所涉及的概率空间的认知，以及他对信宿概率空间的猜测所决定的。而信宿从信源发出的信息中得到多少信息量，是由信息本身，自己的概率空间，以及对信源的概率空间的猜测所决定的。这样一来，就不存在一个两边一致认可的“客观”信息量，而只有信源和信宿各自的“主观”信息量，分别由信源发信时的主观心理状态和信宿收信时的主观心理状态所决定。如果我们不打算强迫香农同学去研究心理学，那么最好还是就当作信息在进入人脑的那一刹那就此消失了，而不要再去管后面发生的事情。简单的说，当信息遭遇智能，唯一的结果就是死掉。

信息幽灵

头脑中真的有信息吗？这个问题让我想起小时候常常纠结在心头，

却又找不到答案的一个问题：世界上真的有鬼吗？下面我们要开始讲鬼故事了。

虽然信息死掉了，但是由信息引发的故事才刚刚开始。信息就像一个阴魂不散的幽灵，一直在作为智能体的信宿中游荡。有的时候，他似乎又借尸还魂一般的复活了，从外人看来，一个信息被智能体吐了出来，就和他刚刚吞下去的那个看起来一模一样。当然这只是个假象，智能体不会消化不良，那个被吐出来的，此刻真正活着的，是智能体制作的被吞下去的信息的克隆兄弟，他和原来的信息长得是如此之像，以至于常常被误认为是同一个。

信息到底是什么时候死掉的，这也是个很让人纠结的问题。问题出在信息被智能体吃掉的那一刻。那一刻似乎总是把握不住，其实问题的关键在于，智能体吞下信息引发了一个复杂的过程，这个过程是一个链式反应，每一步都可能产生出来一些新的信息，然后又被吞下，直至最终没有新的信息产生为止。

如果一个智能体看到另一个智能体写的便条（在此我们不去深究“看到”的详细机理，事实上，“看到”本身就是一连串信息变换过程），就发生了以下一连串事情：

首先，便条上的笔画被辨识成文字，这时候，便条信息被接收了，消失了，文字信息产生了；

—> 然后，文字信息被识别成一个个语句，进而又识别成一个个词语，于是文字信息消失了，带有顺序和区隔的一组词语的信息出现了；

—> 再往后，词语被识别成一组对象，它们之间的关系，以及互

动的过程，或者简单说，一个场景，或者一个模型，词语信息消失，模型信息出现；

—> 当智能体再次吞下模型信息的时候，通常就会引发一些情绪的反应，如果这种反应足够强烈，就不会再有新的信息产生出来，信息的链式反应到此为止，再往后就是实实在在的生理反应了；

—> 链式反应也可以不停下来，继续向下传递，比如，模型可以再次符号化，之后再把符号作一番变换，然后再次建立模型，等等等等，这个游戏可以循环往复地玩下去，直到智能体自己厌烦了为止。

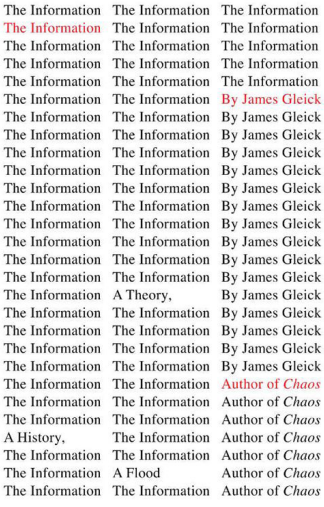
在这每一步骤当中，智能体调出一个特定的概率空间，解码出原信息，之后把这信息投入知识的熔炉当中，随即信息被熔化铸成模型，模型又可以翻印出新的信息。说句题外话，日本企业管理主张要问五个为什么，实质就是把这种铸模——翻印的过程至少搞上五次，这也部分说明了这种思维游戏的价值。

经过这么一通折腾，原信息以及他的 N 代徒子徒孙被智能体彻底消化掉，得到的是一堆模型，其中有用的会被智能体存储起来留待后用，那便是我们称为“知识”的东西。至此，信息终于可以瞑目了，那郁结不化的幽灵终于可以消散了。

信息的旅程就这么结束了，而知识的故事才刚刚开始。

后记

本文写于 2011 年 4 月，本来打算作为《知识的故事》的前篇，后来由于笔者工作繁忙，《知识的故事》一直未能着手写作，此篇



《信息简史》记录了信息发展的历史，从文字、击鼓会意、写作、语法，一直到分析机、电报和电话、ENIAC 和现在无所不在的计算机。这就是书名中说的“历史回顾”。“理论”部分则介绍了香农、维纳、图灵等曾经从事编码与解密工作的科学家们。在“洪流”部分，作者认为遗传学是生物用来作信息交换的机制 -- 小鸡正是一枚鸡蛋用来制造另一枚鸡蛋的工具。作者还讨论了具备自我复制机制的 memes，认为它是信息进化的元生命形式。

的结尾也就一直未能有所呼应。当初写作此篇短文，目的在于将信息作一明确的界定，与知识截然划分开来，以便精确阐述关于信息与知识各自的理论，夯实知识研究的理论基础。最近读了《信息简史》(*The Information: A History, a Theory, a Flood*) 一书，其中提到香农信息论将信息与其中携带的含义彻底剥离（《信息简史》428 页第三段第一句），这充分说明，研究信息的含义——也就是知识，形成严格牢固的基于数学的精确定量模型，已经成为这个时代不可回避的使命。

专题：活的信息

信息可视化的科学

作者 /Mark Smiciklas

Mark Smiciklas 是温哥华一家数字营销传播公司 Intersection Consulting 的总裁，该公司主要引导企业如何利用 Web 2.0 的活力实现经营目标。他是数字营销和社交媒体领域资深的实践者，掌握了独特的可视化思考和讲究实际的战略方法。Smiciklas 曾受邀为多所加拿大大学的本科生和成人学习者教授社交媒体战略课程。Smiciklas

信息图是什么

俗话说 " 一张图片胜过千言万语 "， 这恰恰体现了视觉传播的价值和效率。

信息图（information graphic，简写为 infographic）是一种将数据与设计结合起来的图片，有利于个人或组织简短有效地向受众传播信息（参见图 1）。

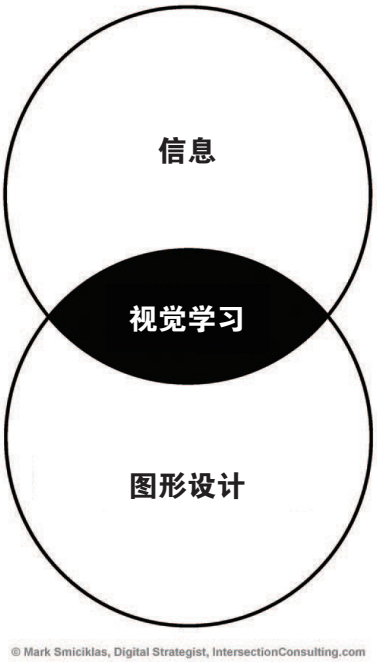


图 1　信息图的剖析

长期更新博客 http://intersectionconsulting.com/blog。

信息图的定义

更为正式地说，信息图是可视化的数据或概念，它们把复杂的信息传递给受众，使其得以快速消化和理解这些信息。

制作和发布信息图的过程被称作数据可视化、信息设计或者信息架构。

信息图将数据与设计结合，以利于视觉学习。这种传播方式使得那些复杂的信息更易被快速理解。

在商业领域有一个关于信息图的主流定义。英国图形设计师、作者、信息设计理论家奈杰尔霍姆斯（Nigel Holmes）曾经简称它们为“解释图”。

信息图的历史

在传统媒体（如报纸、杂志）和数字渠道中，你都能看到信息图的影子，其中社交媒体的发展带动了信息图的爆发式流行。

对于漫不经心的旁观者来说，信息图好像是最近才随着互联网的发展而兴起的。但正如图 1-2 所展示的，其实早在远古时代，人类就习惯于用符号、图表、图片来讲述故事、分享信息以及构建知识。

进入新千年后，信息图的应用突破了学术领域或传统媒介，变得更为大众化。

当今正处在一个信息爆炸的时代，大众对事物的关注时长也因此

公元 1510 年

伦纳德·达·芬奇（Leonardo da Vinci）混合使用插图和文字解释，制作了形象的人体解剖学指南。

公元前 3000 年

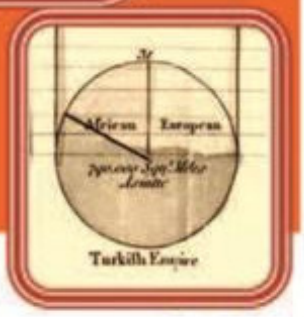
埃及的象形文字使用了符号和图标，是早期信息图的优秀典范。

公元前 30 000 年

历史上第一个信息图示例出现在**石器时代晚期**的法国南部，我们的祖先在洞穴的岩壁上绘制了动物图像。

公元 1350 年

中世纪法国哲学家**妮科尔·奥雷姆（Nicole d'Orseme）**，最早创建了一个用来解析运动物体的度量方法的信息图。



公元 1786 年

苏格兰工程师**威廉·普莱费尔（William Playfair）**是数据可视化的先驱。他的著作《商业和政治图集与统计学摘要》第一次使用线性图、饼图和柱图来表现数字类型的数据。

Source: Wikipedia.com

公元 1857 年

英格兰的护士**弗洛伦丝·南丁格尔（Florence Nightingale）**结合柱图和饼图（鸡冠花形图），来展示克里米亚战争中士兵的伤亡人数和原因，以此来说服维多利亚女王改善战地医院的条件。

公元 1970 年 –1990 年

随着**主流新闻媒体**比如《星期日泰晤士报》、《时代周刊》和《今日美国》，都开始使用信息图来简化信息，增强复杂问题和新闻故事的表现力，信息图变得越来越流行。

公元 1850 年 –1870 年

法国土木工程师**查尔斯·约瑟夫·米纳德（Charles Joseph Minard）**使用流程图来分析地理统计信息。他最有名的数据可视化图分析了拿破仑入侵俄罗斯失败的原因。该图将该时期多种复杂的数据集（地图位置、行军方向、减少的部队数量和骤降的气温）都整合在了一起。

公元 1930 年 –1940 年

现代社会由图像引领。上图由**奥托·诺伊拉特（Otto Neurath）**创建，是一套用图标和图像来传达构思与概念的可视化交流模型。

图 2　信息图简史

大大缩短，于是各机构都纷纷开始借助信息图来快速传递信息，帮助机构内外的受众更好地理解这些内容。社交媒体大大提高了资源共享性，这一切都表明信息图已经成为数字时代信息传播最有效的方式之一。（关于“共享性“这一话题，在本章后面部分将有更详尽的叙述。）

可视化的科学

首先让我们来了解一些大脑中视觉运行机制和眼部信息处理方面的知识，这有助于了解信息图在商业传播中是如何发挥作用的。

视觉中枢

大脑的一大任务就是视觉处理，几乎半个大脑都是直接或间接用来处理视觉功能的。

涉及这项活动的细胞、神经元、纤维的数量非常庞大。作为大脑的延伸部分，单单一个视网膜就由超过 1.5 亿个细胞构成。另外，大脑中与视觉相关的神经元占了很大的比重，将近 30% 的灰质即由这些神经元组成。相比之下，与触觉和听觉有关的神经元只占了 8% 和 3%。

视觉易于理解

基于这些视觉“网络”，我们不难理解比起单纯文本，大脑处理信息图会更加省力。

单词中的每一个字母都代表一个符号。在阅读文本的时候，大脑

首先需要对其解码，将这些字母与记忆中存储的形状相匹配，进一步理解这些字母如何组成单词，单词再组成句子，句子最后组成段落。尽管这一解读过程只需瞬间完成，但与大脑处理图像的方式相比，还是需要消耗更多的脑力。

我们对图像的处理要比文本快得多，这与大脑处理信息的方式密不可分。大脑对图片中数据的处理瞬间即可完成，但对文本的分析不得不遵循线性方式（如图 3 所示）。

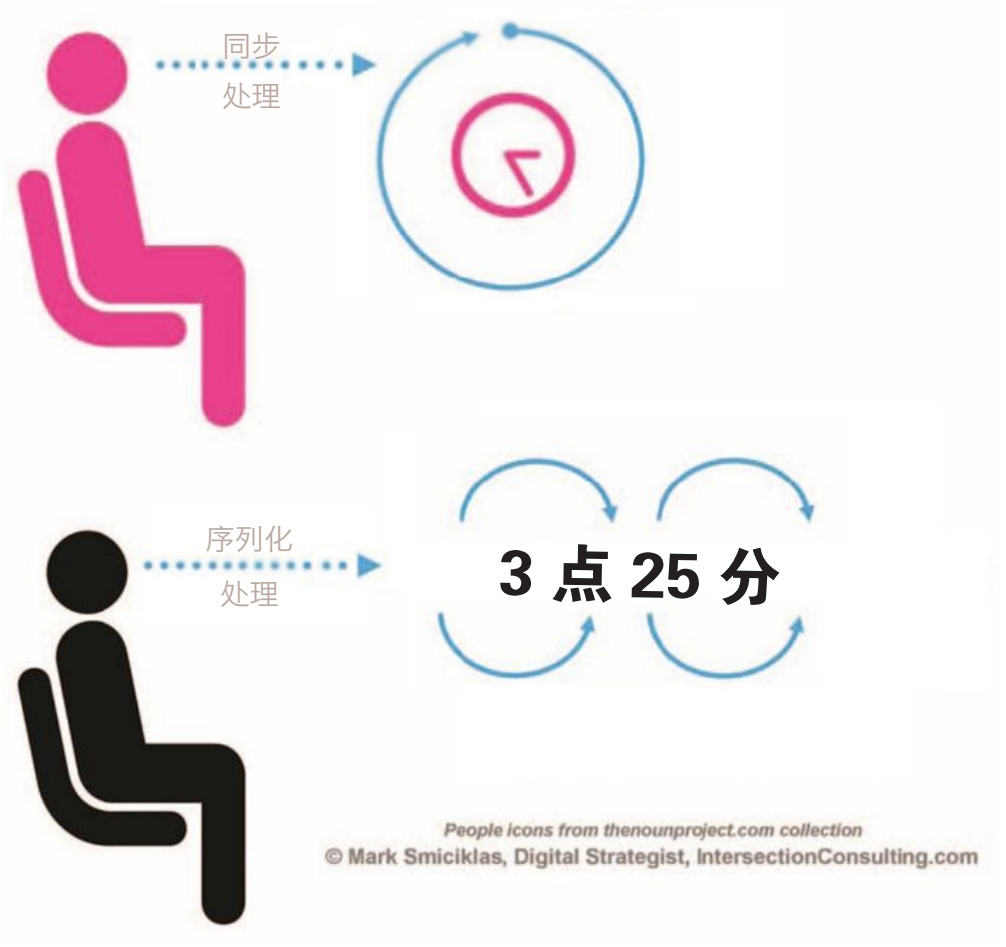
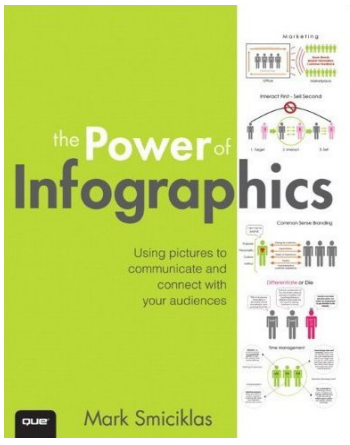


图 3　可视化学习



《信息图学的力量：巧用图形与人沟通》作者 Mark Smiciklas 用自己所熟稔的方式向大家展示了如何在信息传播过程中兼具美学形式和强大功能。如何设计出功能强大，同时也极具欣赏性的图形？什么是信息图学？它的应用为何如此广泛？如何将各式各样的信息视觉化？如何用视觉化的信息与人沟通？本书将一一解答。

译者 / 项婷婷

所以在某种程度上，利用信息图来传播，受众在生理上就更容易与所传播的信息联系起来。

在一次 TED（Technology Entertainment Design）大会上，作家和设计师戴维·麦坎德利斯（David McCandless）在《数据可视化之美》的演讲中颇为形象地解释了信息图是如何将大家从令人窒息的信息洪流中解救出来的：

“视觉化信息有种神奇的力量。它对信息的阐释毫不费力，逐字逐句地向你娓娓道来。想象一下你正穿梭在密密麻麻的信息丛林间，突然碰到赏心悦目的图表或简单明了的可视化数据，就像在密林中邂逅了一方空旷的天地，真是个巨大的解脱！”

新奇感

大脑就是被设计出来寻找不同事物的。

可以把思维想象成计算机的硬盘。为了让大脑保持灵活和高效运转，大脑存储空间就不能被填满。所以，为了维持一个最优的处理速度，大脑会过滤输入数据，并在瞬间将感知到的 99% 的数据丢弃。而这个过滤过程的核心就是判断输入数据是否和大脑习惯于收集的数据相一致。如果信息是新的或者以一种全新的方式呈现的，那么它就会引起大脑的注意。

信息图为你的组织提供了这样的机会：将你展示的信息赋予了新奇或独特的元素，从而让受众更容易注意到它们。

专题：活的信息

用编程工具实现

数据可视化的几个选择

作者 /Nathan Yau

加州大学洛杉矶分校统计学专业在读博士、超级数据迷，专注于数据可视化与个人数据收集。他曾在《纽约时报》、CNN、Mozilla 和 SyFy 工作过，认为数据和信息图不仅适用于分析，用来讲述与数

不用太紧张。掌握一点点编程技巧，你就能利用数据做更多的事情，远远超过那些开箱即用的软件。编程技巧能赋予你更加灵活的能力，而且各种类型的数据都能适应。

大多数设计新颖、令人惊艳的数据图都是通过代码或绘图软件实现的，很有可能两者兼有。有关绘图软件我们稍后也会谈到。

对于新手来说代码可能颇为神秘——我也是这样过来的。但我们可以把它当做一门新语言来看待，因为它确实如此。每一行代码都告诉计算机去做某件事情。计算机不懂我们和朋友之间所用的语言，所以你必须用它自己的语言或语法才能和它交流。

与任何语言一样，你不可能立刻就开始进行对话。要从基础开始，然后逐步建立自己的学习方式。很可能在你意识到之前，你就已经开始写代码了。关于编程最酷的事情在于，一旦你掌握了一门语言，学习其他语言就会更加容易，因为它们的逻辑思路是共通的。

可选项

看来你已经决定要卷起袖子写代码了——好样的。在这里你有很

据有关的故事也非常合适。Yau 的目标是让非专业人士读懂并用好数据。他创建了一个设计、可视化和统计方面的博，http://flowingdata.com，你可以从中欣赏到他最新的数据可视化实验作品。

多选择。针对不同的任务，各种语言之间互有优劣。有些语言善于处理大量数据，其他一些则可能偏向于更好的视觉效果，或者提供交互功能。选择何种语言取决于你对数据图的目标，或者看你对哪种语言更习惯。

有些人坚持只学一门编程语言，并把它学得很透。这没问题，而且如果你是编程新手，我强烈推荐这一策略。尽量熟悉代码的基础规范和重要概念。

我们应该选择最能满足自己需要的编程语言。不过，掌握新语言、了解新玩法也是很有乐趣的。因此在作决定之前，不妨先积累一些编程经验。

1. Python

前一章我们已经讨论了如何利用 Python 来处理数据。Python 善于处理大批量的数据，不会造成宕机。这使得该语言能够胜任繁重的计算和分析工作。

Python 干净易读的语法也很受程序员们欢迎，还可以利用很多模块来创建数据图形，例如图 1 中的这种。

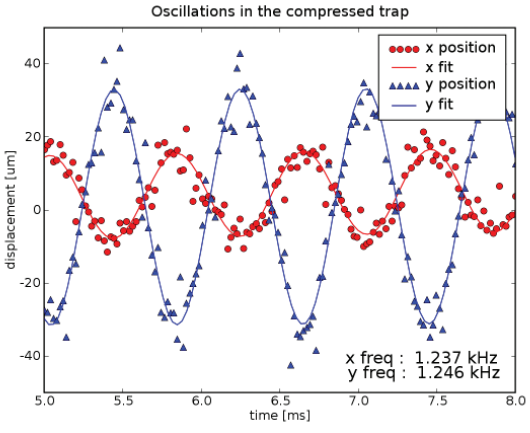


图 1　利用 Python 生成的图表

有用的 Python 资源

- Python 官方网站 http://python.org。
- NumPy 和 SciPy[①] http://numpy.scipy.org/——科学计算模块。① NumPy 是 Python 的一个数据处理的函数库，里面主要是一些矩阵的运算等。SciPy 是 Python 语言中用于科学研究的函数库，它是在 Numpy 基础上开发的。

从美学方面来看，这个图表还不够好。直接拿 Python 输出的图片用于印刷可能会比较勉强，尤其是在边缘处给人感觉比较粗糙。但不管怎样，这是数据探索阶段一个很不错的开始。或者你也可以先输出图片，然后再利用其他的图形编辑软件来润色或添加信息。

2. PHP

PHP 是我刚开始网页编程时学到的第一门语言。有些人说它很松散，确实如此，但也可以让它很有条理。大部分 Web 服务器都预安装了 PHP 的开源软件，因此要想着手写 PHP 是非常容易的。绝大多数预安装中都会包含一个叫做 GD 的图形函数库。这个库非常灵活，能让你从无到有地创建图形，或者修改已有图形。此外还有很多 PHP 图形函数库能帮助我们创建各类基本的图表。 最受欢迎的是 Sparkline（微线表）库，它能让你在文本中嵌入小字号的微型图表，或者在数字表格中添加视觉元素，如图 2 所示。

图 2　利用 PHP 图形函数库生成的微线表

有用的 PHP 资源

- PHP 官方网站（http://php.net）。
- Sparkline PHP 图形函数库（http://sparkline.org）。

一般 PHP 的出现都会伴随着 MySQL 等数据库，而不是一堆 CSV 文件。这使它能物尽其用，处理大型的数据集。

3. Processing

Processing 是一门适合于设计师及数据艺术家的开源语言。最早的 Processing 还只能算作是小品级别，可以让用户快速生成图形，但随后获得了长足的发展，完成了很多高质量的项目。比如说，第 1 章中提到的 We Feel Fine 就是用 Processing 中创建的。

Processing 很棒的一点是能很快上手：轻量级的编程环境，只需几行代码就能创建出带有动画和交互功能的图形。这款工具确实很基础，但由于它偏重于视觉思维的创造性，你很容易就能知道如何创造出更高级的作品。

虽然在一开始主要是设计师和艺术家使用 Processing，但如今它的受众群体已经越来越多样化了。你可以借助各种函数库来提升它的威力。

它的缺点之一是要使用到 Java 小应用程序，在某些计算机上载入时可能会很慢，而且并不是每个人都安装了 Java（尽管多数人都安装了）。不过这也有解决办法，Processing 在不久前发布了它的 JavaScript 版本。

无论如何，它对于新手来说是个很好的起点。即使是毫无编程经验的用户也能够做出有价值的东西。

有用的 Processing 资源

- Processing 的官方网站（http://processing.org）。

说明 虽然有很多免费、开源的 ActionScript 函数库，但是 Flash 软件和 Flash 编译器的价格比较昂贵，在你选择软件时需要考虑这一点。

① BSD 许可协议（Berkeley Software Distribution license），是自由软件（开源软件的一个子集）中使用最广泛的许可协议之一。

4. Flash 和 ActionScript

网上大多数可交互的动画数据图都是通过 Flash 和 ActionScript 开发的，尤其是在《纽约时报》这样的主流新闻网站上。我们可以直接用 Flash 来设计图形，这也是一款所见即所得的软件，但有了 ActionScript 的帮助，就能更好地控制交互行为。许多应用都是完全用 ActionScript 写的，无需用到 Flash 环境，不过这些代码还是作为 Flash 应用来进行编译。

比如说，表现沃尔玛企业成长的可交互动画演示地图 （见图 3）就是用 ActionScript 写成的。其中调用了 Modest Maps 库。这是一个用于区块拼接地图（tile-based map）的显示和交互的函数库，以 BSD 许可协议[①]发布，这表示它是免费的，大家可以随心所欲地使用。

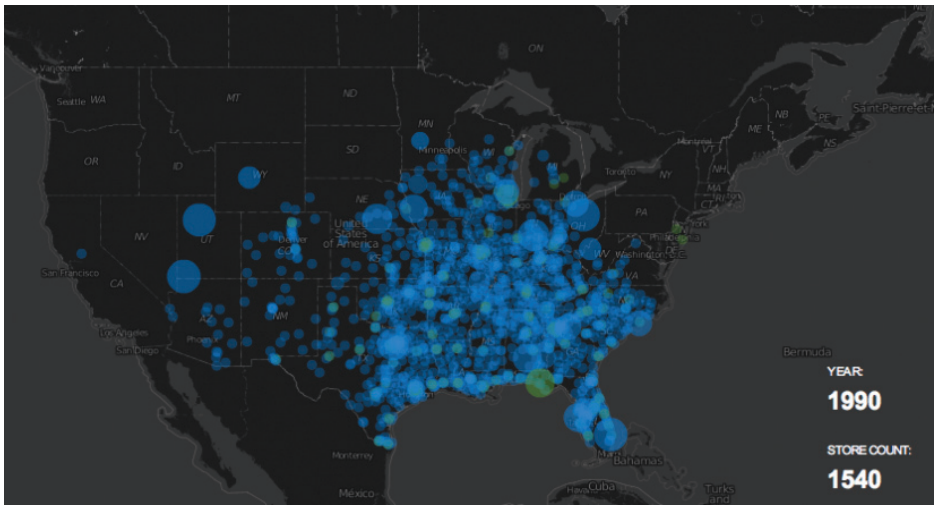


图 3 动画演示沃尔玛企业成长的地图，以 ActionScript 写成

图 4 中的可交互堆叠面积图也是由 ActionScript 写成的。它能让你搜索数年来各种消费开支的大小变化，例如住房（Housing）和交通（Transportation）。这一艰巨的任务是由加州大学伯克利分校可视化实验室开发的 Flare Action Script 库完成的。

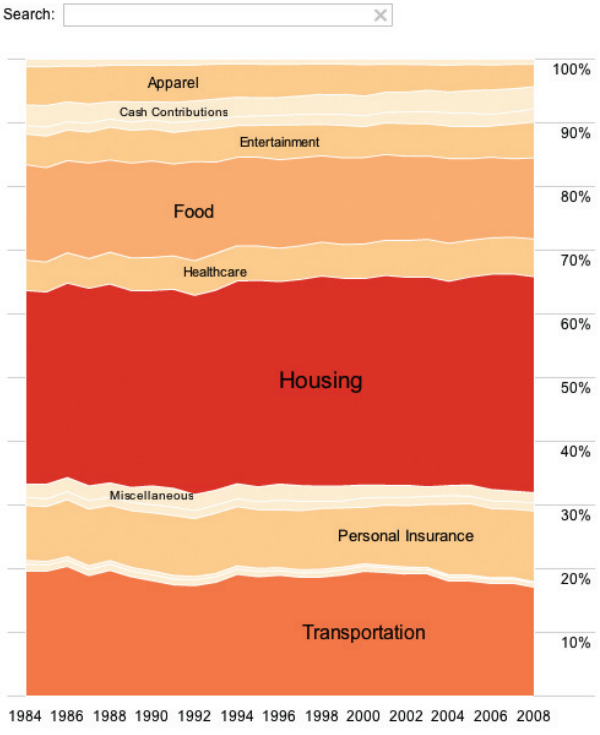


图 4　显示消费开支分类的可交互堆叠面积图，以 ActionScript 写成

如果你对网上的可交互图形感兴趣，那么 Flash 和 ActionScript 是非常好的选择。Flash 应用的下载相对较快，而且绝大多数人的计算机上都安装了 Flash 播放器。

不过 ActionScript 并不好学。虽说语法并不复杂，但是安装和代

有用的 Flash 和 ActionScript 资源

- Adobe 支持（http://www.adobe.com/products/flash/whatisflash/）——Flash 和 ActionScript（以及其他 Adobe 产品）的官方文档。
- Flare 可视化工具包（http://flare.prefuse.org）。
- Modest Maps 函数库（http://modestmaps.com）。

码的组织却可能会难倒初学者。你不可能像 Processing 那样只用几行代码就运行成功。在后面的章节中我会带领大家实践其中的基本步骤，另外由于 Flash 的广泛使用，在网上还能找到很多极有帮助的教程。

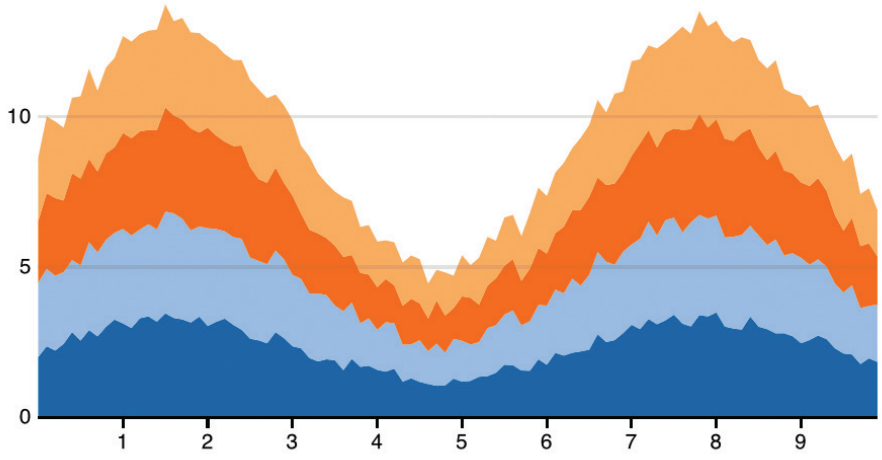
与此同时，Web 浏览器一直在飞速发展，其运行速度和效率也有了显著的提高。这让我们有了很多其他的选择。

5. HTML、JavaScript 和 CSS

Web 浏览器的运行速度越来越快，功能也越来越完善。不少人使用浏览器的时间要超过计算机上的其他任何程序。可视化在近期也有了相应的转变，开始借助 HTML、JavaScript 和 CSS 代码直接在浏览器中运行。在过去，可交互的数据图一般都是通过 Flash 和 ActionScript 来实现，而静态数据图则需要存储为图片格式。现在的情况也大抵如此，但不再只有这一种选择。

一些功能强健的工具包和函数库可以帮助我们快速创建可交互或静态的可视化图形。它们还提供了大量的选项，以便你针对数据需要进行定制。

比如说，斯坦福大学可视化团队开发的 Protovis 就是一款免费开源的可视化函数库，它能帮助你创建网页原生的可视化作品。Protovis 提供了一系列开箱即用的可视化工具，但你在创建几何图形时不会受到任何限制。图 3-15 显示的是一幅可交互的堆叠面积图。



Protovis 中内嵌了这一图表类型。我们还可以生成形式更加新颖的流线图（streamgraph），如图 3-16 所示。

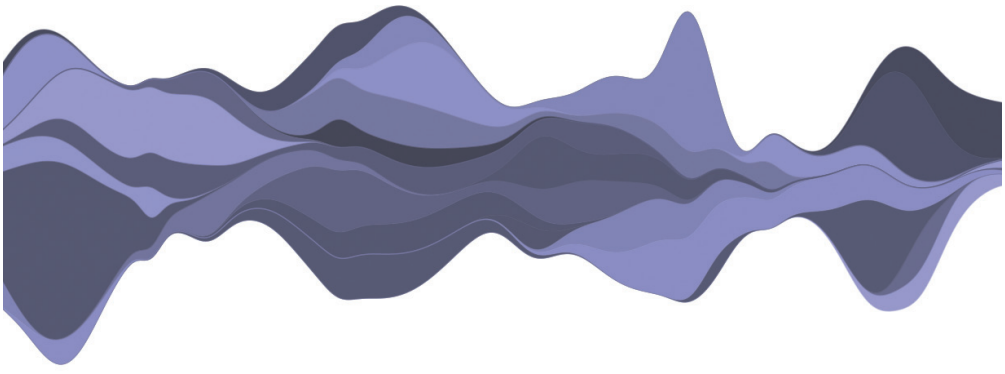


图 6　由 Protovis 自定义生成的流线图

我们还可以利用多个库来扩充功能。Flash 也能做到这一点，但 JavaScript 在代码方面的工作量会小很多。有了 jQuery 和 MooTools 等库的协助，JavaScript 也更加容易阅读和使用。这些

库并非专为可视化而开发的，但仍然能带来很大帮助。它们提供了大量基础功能，只需几行代码就能实现。如果没有这些库，我们要写的代码就会更多，而且稍不注意就可能漏洞百出。

这些库还提供了一些插件，帮助我们制作较为基础的图形。例如，你可以利用 jQuery 的 Sparklines（微线表）插件来生成微线表（见图 7）。

图 7　通过 jQuery 的 Sparklines 插件生成的微线表

用 PHP 也可以做到这一点，但这种方法具有更多优势。首先，数据图是在用户的浏览器中生成的，而非服务器端。这能缓解服务器的压力，否则在流量较大的情况下你的网站就可能会出问题。

另一个优势在于你无需在自己的服务器上安装 PHP 图形库。很多服务器上都预安装了这些图形库，但也有一些没有。如果你对 PHP 图形库不熟悉，安装过程可能会非常麻烦。

也许有人不想用插件。没关系，利用标准的网页编程也可以定制可视化的效果。图 8 就是一副可交互的日历，同时也是用户使用 your.flowingdata 工具的热度图（heatmap）。

不过还是有几点需要注意。由于相关的软件和技术还比较新，在不同浏览器中你的设计可能在显示上会有所差别。在 Internet Explorer 6 这类老旧的浏览器中，有些工具可能无法正常运行。

不过这种情况正在逐渐好转，因为绝大多数人都已经转向了

Firefox 或 Google Chrome 等更现代的浏览器。归根结底，这还是取决于你的受众群体。在 FlowingData 网站的访问者中，只有不到 5% 的人还在用低版本的 Internet Explorer，因此浏览器兼容性并不是很严重的问题。

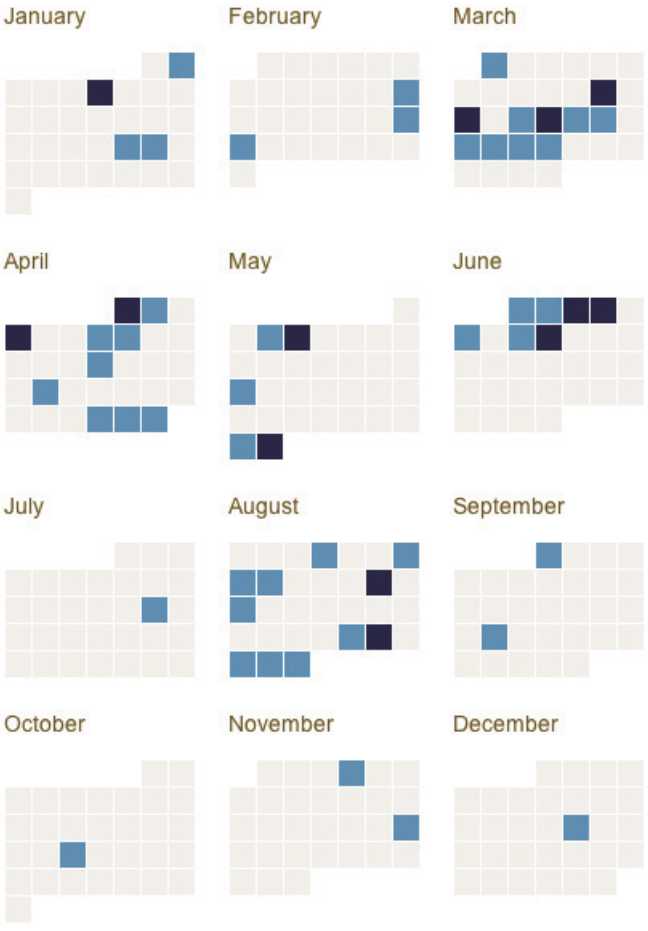


图 8　可交互日历，同时也是用户使用 your.flowingdata 的热度图

同样，由于该技术的普及率尚不够高，为 JavaScript 可视化提供

有用的 HTML、JavaScript 和 CSS 资源

- jQuery（http://jquery.com/）——一个 JavaScript 库，能让该语言的编程更加高效，而且让最终代码更加易读。
- jQuery Sparklines（http://omnipotent.net/jquery.sparkline/）——通过 JavaScript 生成静态及动画的微线图。
- Protovis（http://vis.stanford.edu/protovis/）——专用于可视化的 JavaScript 库，提供了实例以便学习。
- JavaScript InfoVis 工具包（http://datafl.ws/15f）——另一个可视化库，但不如 Protovis 成熟。
- Google Charts API（http://code.google.com/apis/chart/）——动态创建传统形式的图表，只需修改 URL 即可。

支持的函数库不像在 Flash 和 ActionScript 中那么多。这也是为什么许多主流新闻机构仍然大量使用 Flash 的原因。不过随着技术的发展这一局面终会发生改变。

6. R

如果你浏览过 FlowingData，可能就会知道我最喜欢的数据图形软件就是 R。它是一款免费且开源的统计学计算软件，图形功能也很强大。它也是绝大多数统计学家最中意的分析软件之一。此外还有一些功能近似的付费软件，例如 S-plus 和 SAS，不过它们很难比得上 R 的完全免费以及活跃的开发社区氛围。

相较于之前提到的所有软件，R 的优势之一在于它是专为数据分析而设计的。HTML 的用途主要是创建网页，而 Flash 则用于其他方面，例如视频以及动态广告。但 R 却是由统计学家开发并维护，主要的面向对象也是统计学家。至于这是好是坏，取决于你从哪种角度来看待它。

支持 R 的工具包也有很多，你只需把数据载入到 R 里面，写一两行代码就可以创建出数据图形。比如说，你可以利用 Portfolio 工具包快速创建出板块层级图（treemap），如图 9 所示。

创建热度图也同样简单，如图 10 所示。

当然，还可以创建更多传统的统计图表，例如散点图和时间序列图，我们将会在第 4 章中对此进行深入讨论。

坦白说，R 的网站看起来极为落后（见图 11），而且软件本身在引导新用户方面做得不够好。不过你要记住，R 毕竟是一门编程

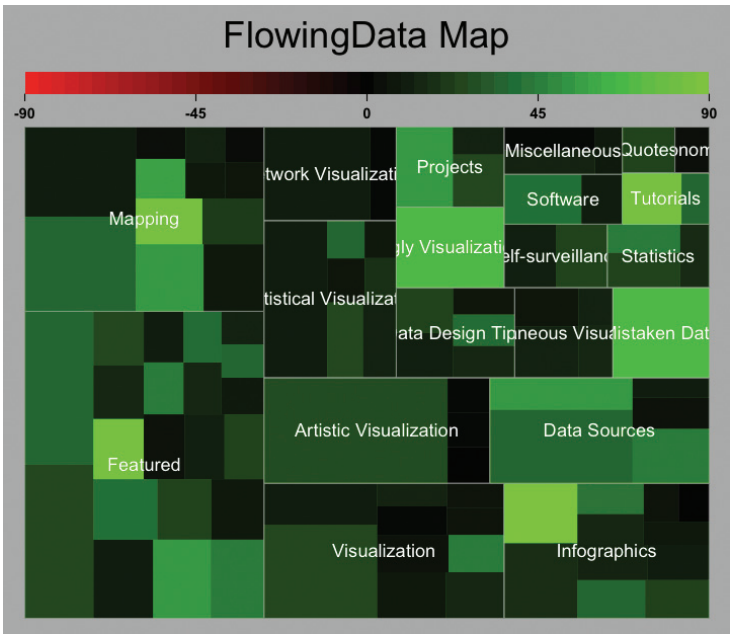


图 9

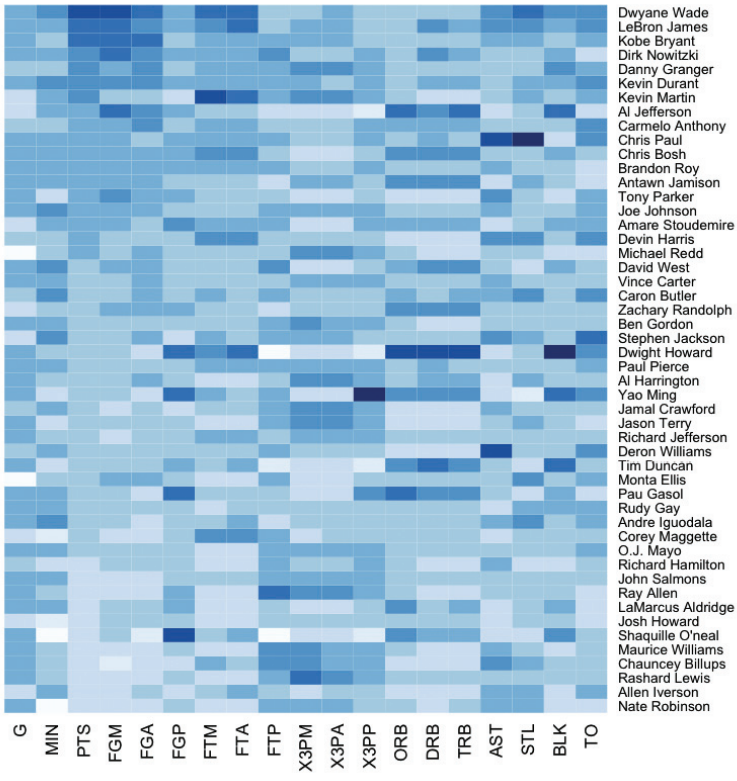


图 10　R 生成的热度图

提示

在搜索引擎中搜索有关 R 的内容时，它的本名可能会带来很多不相干的结果。可以尝试以“r-project”作为关键词，这样的搜索结果会更加精确。

有用的 R 资源

- 适用于统计运算的 R 项目（http://www.r-project.org）。

语言，学习任何语言都必须经历开头的艰难。我也读到过一些关于 R 的负面评论，但通常都是那些习惯于按钮或者鼠标操作的人在发牢骚。所以如果你打算使用 R，不要对用户界面的期待过高，否则你一定会觉得它很不友好。

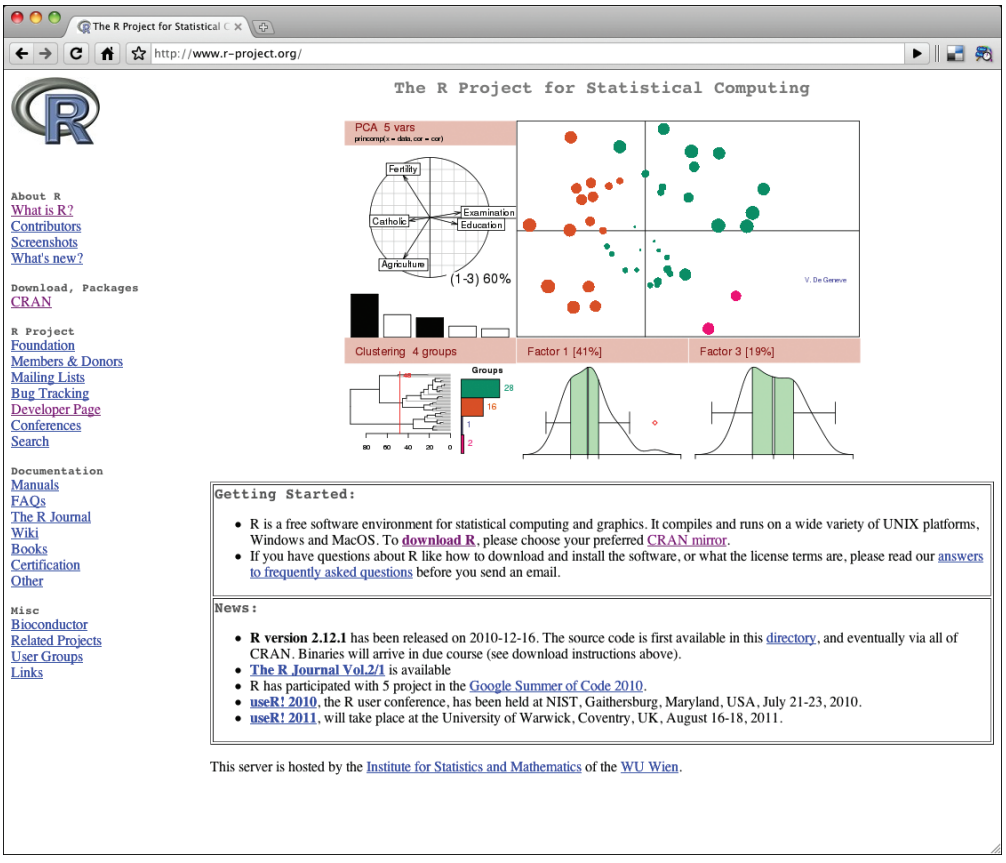
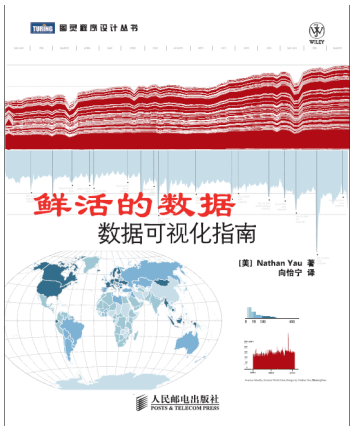


图 11　R 的网站首页 http://www.r-project.org

但如果克服了这一点，我们就能用 R 做很多事情。它能够生成达到印刷品质的图片（至少也是雏形），而且极其灵活。如果愿意，你可以写出自己的函数或者程序包，按自己想要的方式来创建图形，或者你也可以借用 R 函数库里其他人开发的成品。



《鲜活的数据：数据可视化指南》是一本系统介绍数据可视化的图书。书中主要阐述了如何将冰冷枯燥的数据转换成易于理解、生动有趣、主题清晰的图表。作者根据数据可视化的一般顺序，先后介绍了如何获取数据，将数据格式化，然后用可视化工具（如 R）生成图表，最后在图形编辑软件（如 Illustrator）中修改完善，使图表达到最佳的可视化效果。本书详细介绍了柱形图、饼图、折线图和散点图等图表的绘制方法及各自的优缺点，还用专门的一章介绍与地图相关的数据可视化技巧。本文摘自《鲜活的数据：数据可视化指南》。

R 提供了基础的绘图功能，使用它，你基本上可以随心所欲地绘制想要的图形，例如线条、形状以及框架坐标轴。所以和其他编程工具一样，能限制你的只有想象力而已。事实上，任何一种图表类型都能通过 R 或者 R 的工具包实现。

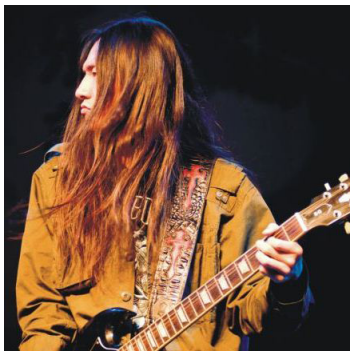
既然 R 这么强大，为什么还要学习其他工具呢？为什么不干脆用 R 来做所有事情？原因有以下几方面， R 是在你的桌面上运行的，所以它不太适合于动态网页。存储为图片然后发布到网页上并不是问题，但这一过程不会自动完成。你也可以通过网页来动态生成图片，但截至目前，R 的这一功能还不是特别强大，无法比 JavaScript 等网页原生工具。

在创建可交互图形或动画方面，R 也不是特别擅长。同样，尽管也可以用 R 来实现，但还有其他更方便的途径，比如 Flash 或者 Processing。

最后，大家也许已经注意到，图 3-19 和图 3-20 中的图形还欠缺一些雕琢。你恐怕不希望在报纸上看到这种水平的图形。当然，你也可以尝试依靠不同的选项或者添加代码来强化 R 的设计感觉，但我的策略一般是在 R 中创建基本图形，然后再利用 Adobe Illustrator 等绘图软件对其进行编辑加工（后文即将讨论）。如果是用于数据分析，那么 R 的原始输出就可以了，但如果是用于演示或印刷，则最好还是从视觉和美感上稍加调整。

取舍

学习编程就是在学一门新的语言，一门有关位和逻辑的计算机用语言。例如，当你使用 Excel 或 Tableau 时，本质上是在和程序



译者 / 向怡宁

交互和视觉设计师、摇滚乐手，同时还热衷于翻译和写作。著有《Flash 组件、游戏、SWF 加解密》及《就这么简单：Web 开发中的可用性和用户体验》，译有《奇思妙想：15 位计算机天才及其重大发现》、《瞬间之美：Web 界面设计如何让用户心动》、《网站设计解构：有效的交互设计框架和模式》、《网站搜索设计：兼顾 SEO 及可用性的网站设计心得》等书。他认为“一个不会弹吉他的设计师不是个好译者”。

的翻译器打交道。按钮和菜单是你熟悉的语言，而当你点击它们时，软件会把你的行为翻译为指令，并发送给计算机。然后计算机才会服从你的命令，例如生成图片或者处理数据。

如果你打算深入研究你的数据，而且日后可能（或者希望日后）还会接触大量与数据相关的项目，那么现在花些时间学习编程最终会节省其他项目的时间，并且作品也会给人留下更加深刻的印象。你的编程技巧会在每一次项目中获得提高，你会发现编程越来越容易。和任何一门外语一样，我们都会从基础学起，然后再逐步扩大范围，不可能从一开始就用它来著书立说。

让我们换一种思路来看这个问题。假设你被扔到一个陌生的国度，你不会说当地的语言，但有一名随身翻译。如果要和当地人交流，你会先说出来，然后让翻译去传递你要表达的信息。如果翻译不理解你的意思，或者不懂你用的某一个词，又该怎么办？他可能会干脆忽略这个词，或者如果他足够聪明，说不定去查查字典。

对于开箱即用的可视化工具来说，软件本身就是翻译。如果它不知道该怎样做一件事情，你就只能卡在那里，或者不得不寻找替代的解决办法。和现实中的翻译不同，软件通常不会立刻学会“新词”（在这里指新的图表类型或者数据处理方式）。新的功能需通过软件更新获得，也就是说你只能干等着。那么，如果你自己学会了这门语言会怎样呢？

再一次声明，我并不是说要避免使用开箱即用的工具。我一直都在使用它们。它们让很多沉闷乏味的任务变得轻松容易，这很棒。但我们不能被软件限制住，不是吗？

专题：活的信息

用 R 语言处理大数据：课程 101

作者 /Robert I. Kabacoff

Robert 是 R 语言社区著名学习网站 Quick-R http://www.statmethods.net/ 的幕后维护者，现为全球化开发与咨询公司 Management 研究集团研发副总裁。此前，Kabacoff 博士是佛罗里达诺瓦东南大学的教授，讲授定量方法和统计编程的研究生课程。Kabacoff 还是临床心理学博士、统计顾问，擅长数据分析，在健康、金融服

R 是一种区分大小写的解释型语言。你可以在命令提示符 (>) 后每次输入并执行一条命令，或者一次性执行写在脚本文件中的一组命令。R 中有多种数据类型，包括向量、矩阵、数据框（与数据集类似）以及列表（各种对象的集合）。

R 中的多数功能是由程序内置函数和用户自编函数提供的，一次交互式会话期间的所有数据对象都被保存在内存中。一些基本函数是默认直接可用的，而其他高级函数则包含于按需加载的程序包中。

R 语句由函数和赋值构成。R 使用 <-，而不是传统的 = 作为赋值符号。例如，以下语句：

```
x <- rnorm(5)
```

创建了一个名为 x 的向量对象，它包含 5 个来自标准正态分布的随机偏差。

新手上路

如果你使用的是 Windows，从开始菜单中启动 R。在 Mac 上，则

务、制造业、行为科学、政府和学术界有 20 余年的研究和统计咨询经验。

注意 R 允许使用 = 为对象赋值。但是这样写的 R 程序并不多，因为它不是标准语法，某些情况下，用等号赋值会出现问题，R 程序员可能会因此取笑你。你还可以反转赋值方向。例如，`rnorm(5) -> x` 与上面的语句等价。重申一下，使用等号赋值的做法并不常见，在本书中不推荐使用。

注释 由符号 # 开头。在 # 之后出现的任何文本都会被 R 解释器忽略。

需要双击应用程序文件夹中的 R 图标。对于 Linux，在终端窗口中的命令提示符下敲入 R 并回车。这些方式都可以启动 R（R 界面参见图 1）。

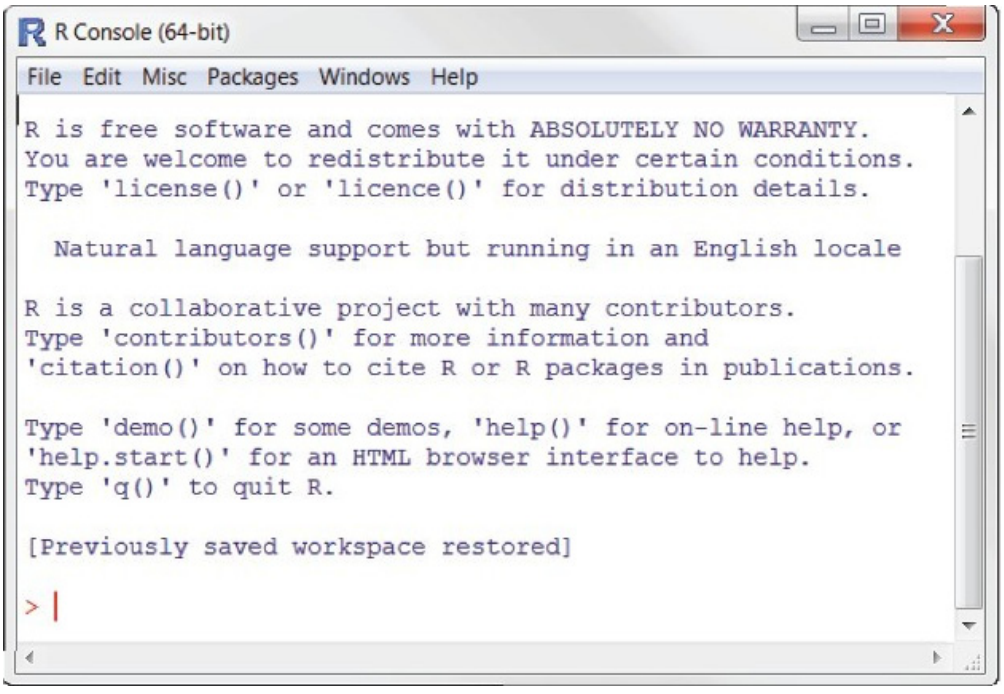


图 1　Windows 中的 R 界面

让我们通过一个简单的虚构示例来直观地感受一下这个界面。假设我们正在研究生理发育问题，并收集了 10 名婴儿在出生后一年内的月龄和体重数据（见表 1）。我们感兴趣的是体重的分布及体重和月龄的关系。

表 1　十名婴儿的月龄和体重

年龄（月）	体重（kg）
01	4.4
03	5.3
05	7.2
02	5.2
11	8.5
09	7.3
03	6.0
09	10.4
12	10.2
03	6.1

* 以上为虚构数据。

可以使用函数 c() 以向量的形式输入月龄和体重数据，此函数可将其参数组合成一个向量或列表。然后用其他函数获得体重的均值和标准差，以及月龄和体重的相关度，最后用图形展示月龄和体重的关系，这样就可以用可视化的方式检查其中可能存在的趋势。如以下代码清单所示，函数 q() 将结束会话并允许你退出 R。

代码清单 1　一个 R 会话示例

```
    > age <- c(1,3,5,2,11,9,3,9,12,3)
    > weight <- c(4.4,5.3,7.2,5.2,8.5,7.3,6.0,10.4,10.
2,6.1)
    > mean(weight)
    [1] 7.06
    > sd(weight)
    [1] 2.077498
    > cor(age,weight)
```

```
[1] 0.9075655
> plot(age,weight)
> q()
```

从代码清单 1 中可以看到，这 10 名婴儿的平均体重是 7.06 kg，标准差为 2.08 kg，月龄和体重之间存在较强的线性关系（相关度 = 0.91）。这种关系也可以从图 1-4 所示的散点图中看到。不出意料，随着月龄的增长，婴儿的体重也趋于增加。

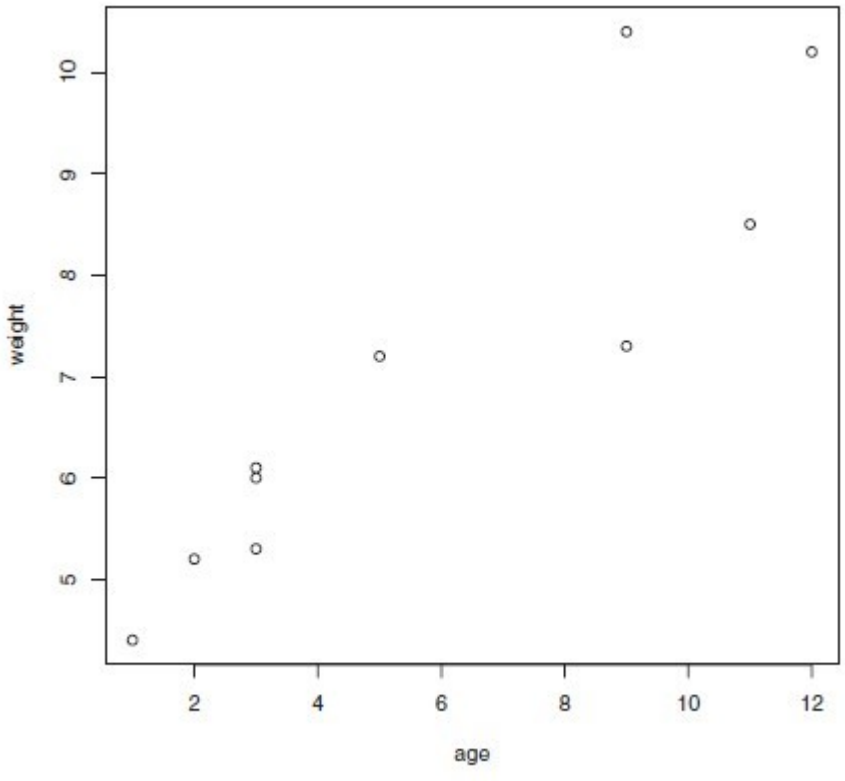


图 2　婴儿体重（千克）和年龄（月）的散点图

散点图 2 的信息量充足，但略过“功利”，也不够美观。接下来的几章里，我们会讲到如何自定义图形以契合需要。

小提示 若想大致了解R能够作出何种图形，在命令行中运行`demo(graphics)`即可。生成的部分图形如图1-5所示。其他的演示还有`demo(Hershey)`、`demo(persp)`和`demo(image)`。要看到完整的演示列表，不加参数直接运行`demo()`即可。

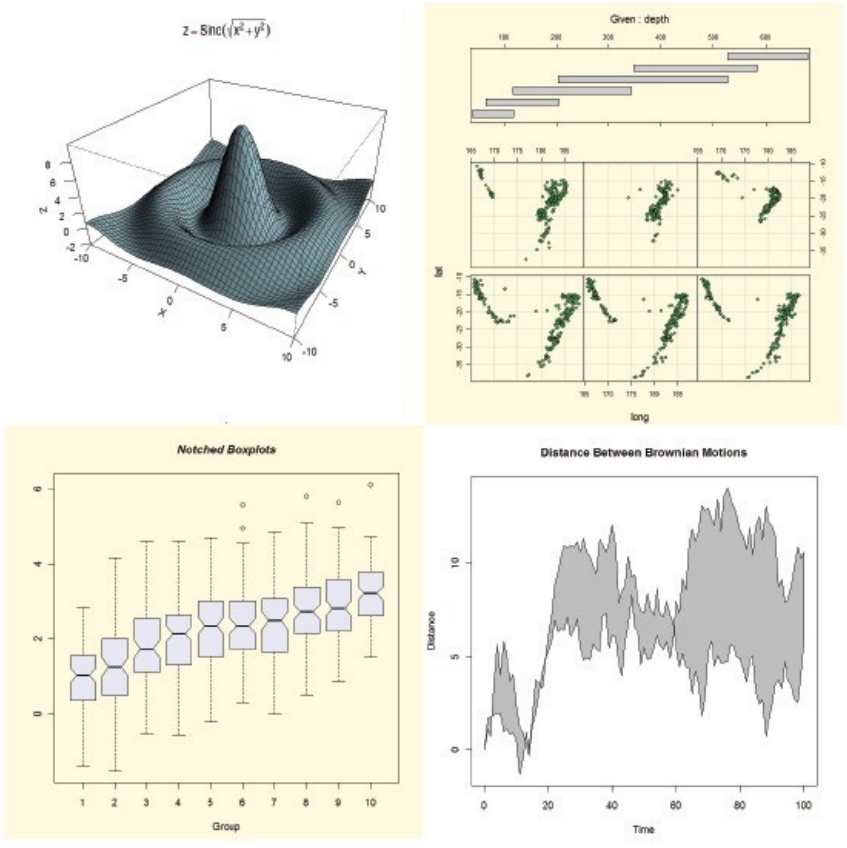


图 3　函数 `demo()` 绘制的图形示例

获取帮助

R 提供了大量的帮助功能，学会如何使用这些帮助文档可以在相当程度上助力你的编程工作。R 的内置帮助系统提供了当前已安装包中所有函数[①]的细节、参考文献以及使用示例。帮助文档可以通过表 1-2 中列出的函数进行查看。

①确切地说，这里的“所有”是指那些已导出的(exported)、对用户可见的函数。——译者注

表 2　R 中的帮助函数

函数	功能
help.start()	打开帮助文档首页
help("foo") 或 ?foo	查看函数 foo 的帮助（引号可以省略）
help.search("foo") 或 ??foo	以 foo 为关键词搜索本地帮助文档
example("foo")	函数 foo 的使用示例（引号可以省略）
RSiteSearch("foo")	以 foo 为关键词搜索在线文档和邮件列表存档
apropos("foo", mode="function")	列出名称中含有 foo 的所有可用函数
data()	列出当前已加载包中所含的所有可用示例数据集
vignette()	列出当前已安装包中所有可用的 vignette 文档
vignette("foo")	为主题 foo 显示指定的 vignette 文档

函数 help.start() 会打开一个浏览器窗口，我们可在其中查看入门和高级的帮助手册、常见问题集，以及参考材料。函数 RSiteSearch() 可在在线帮助手册和 R-Help 邮件列表的讨论存档中搜索指定主题，并在浏览器中返回结果。由函数 vignette() 函数返回的 vignette 文档一般是 PDF 格式的实用介绍性文章。不过，并非所有的包都提供了 vignette 文档。不难发现，R 提供了大量的帮助功能，学会如何使用这些帮助文档，毫无疑问地会有助于编程。我经常会使用 ? 来查看某些函数的功能（如选项或返回值）。

工作空间

工作空间（workspace）就是当前 R 的工作环境，它储存着所有用户定义的对象（向量、矩阵、函数、数据框、列表）。在一个 R 会话结束时，你可以将当前工作空间保存到一个镜像中，并在下次启动 R 时自动载入它。各种命令可在 R 命令行中交互式地输入。使用上下方向键查看已输入命令的历史记录。这样我们就可以选择一个之前输入过的命令并适当修改，最后按回车重新执行它。

当前的工作目录（working directory）是 R 用来读取文件和保存结果的默认目录。我们可以使用函数 `getwd()` 来查看当前的工作目录，或使用函数 `setwd()` 设定当前的工作目录。如果需要读入一个不在当前工作目录下的文件，则需在调用语句中写明完整的路径。记得使用引号闭合这些目录名和文件名。

用于管理工作空间的部分标准命令见表 3。

表 3　用于管理 R 工作空间的函数

函数	功能
`getwd()`	显示当前的工作目录
`setwd("mydirectory")`	修改当前的工作目录为 mydirectory
`ls()`	列出当前工作空间中的对象
`rm(objectlist)`	移除（删除）一个或多个对象

help(options)	显示可用选项的说明
options()	显示或设置当前选项
history(#)	显示最近使用过的 # 个命令（默认值为 25）
savehistory("myfile")	保存命令历史到文件 myfile 中（默认值为 .Rhistory）
loadhistory("myfile")	载入一个命令历史文件（默认值为 .Rhistory）
save.image("myfile")	保存工作空间到文件 myfile 中（默认值为 .RData）
save(objectlist, file= "myfile")	保存指定对象到一个文件中
load("myfile")	读取一个工作空间到当前会话中（默认值为 .RData）
q()	退出 R。将会询问你是否保存工作空间

要了解这些命令是如何运作的，运行代码清单 2 中的代码并查看结果。

代码清单 2　用于管理 R 工作空间的命令使用示例

```
setwd("C:/myprojects/project1")
options()
options(digits=3)
x <- runif(20)
summary(x)
hist(x)
```

```
savehistory()
save.image()
q()
```

首先，当前工作目录被设置为 C:/myprojects/project1，当前的选项设置情况将显示出来，而数字将被格式化，显示为具有小数点后三位有效数字的格式。然后，我们创建了一个包含 20 个均匀分布随机变量的向量，生成了此数据的摘要统计量和直方图。最后，命令的历史记录保存到文件 .Rhistory 中，工作空间（包含向量 x）保存到文件 .RData 中，会话结束。

注意 `setwd()` 命令的路径中使用了正斜杠。R 将反斜杠（\）作为一个转义符。即使在 Windows 平台上运行 R，在路径中也要使用正斜杠。同时注意，函数 `setwd()` 不会自动创建一个不存在的目录。如果必要的话，可以使用函数 `dir.create()` 来创建新目录，然后使用 `setwd()` 将工作目录指向这个新目录。

在独立的目录中保存项目是一个好主意。我通常会在启动一个 R 会话时使用 `setwd()` 命令指定到某一个项目的路径，后接不加选项的 `load()` 命令。这样做可以让我从上一次会话结束的地方重新开始，并保证各个项目之间的数据和设置互不干扰。在 Windows 和 Mac OS X 平台上就更简单了。跳转到项目所在目录并双击保存的镜像文件即可。这样做可以启动 R，载入保存的工作空间，并设置当前工作目录到这个文件夹中。

输入和输出

启动 R 后将默认开始一个交互式的会话，从键盘接受输入并从屏幕进行输出。不过你也可以处理写在一个脚本文件（一个包含了

R 语句的文件）中的命令集并直接将结果输出到多类目标中。

1. 输入

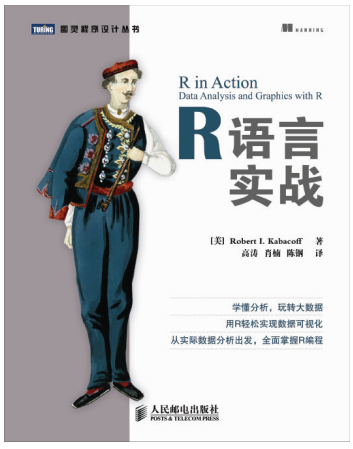
函数 source("filename") 可在当前会话中执行一个脚本。如果文件名中不包含路径，R 将假设此脚本在当前工作目录中。举例来说，source("myscript.R") 将执行包含在文件 myscript.R 中的 R 语句集合。依照惯例，脚本文件以 .R 作为扩展名，不过这并不是必需的。

2. 文本输出

函数 sink("filename") 将输出重定向到文件 *filename* 中。默认情况下，如果文件已经存在，则它的内容将被覆盖。使用参数 append=TRUE 可以将文本追加到文件后，而不是覆盖它。参数 split=TRUE 可将输出同时发送到屏幕和输出文件中。不加参数调用命令 sink() 将仅向屏幕返回输出结果。

3. 图形输出

虽然 sink() 可以重定向文本输出，但它对图形输出没有影响。要重定向图形输出，使用表 1-4 中列出的函数即可。最后使用 dev.off() 将输出返回到终端。



开源软件 R 是世界上最流行的数据分析、统计计算及制图语言，几乎能够完成任何数据处理任务，可安装并运行于所有主流平台，为我们提供了成千上万的专业模块和实用工具，是从大数据中获取有用信息的绝佳工具。《R 语言实战》从解决实际问题入手，尽量跳脱统计学的理论阐述来讨论 R 语言及其应用，讲解清晰透澈，极具实用性。作者 Robert I. Kabacoff 不仅高度概括了 R 语言的强大功能、展示了各种实用的统计示例，而且对于难以用传统方法分析的凌乱、不完整和非正态的数据也给出了完备的处理方法。本文选自《R 语言实战》。

表 4　用于保存图形输出的函数

函数	输出
`pdf("filename.pdf")`	PDF 文件
`win.metafile("filename.wmf")`	Windows 图元文件
`png("filename.png")`	PBG 文件
`jpeg("filename.jpg")`	JPEG 文件
`bmp("filename.bmp")`	BMP 文件
`postscript("filename.ps")`	PostScript 文件

让我们通过一个示例来了解整个流程。假设我们有包含 R 代码的三个脚本文件 script1.R、script2.R 和 script3.R。执行语句：

```
source("script1.R")
```

将会在当前会话中执行 script1.R 中的 R 代码，结果将出现在屏幕上。

如果执行语句：

```
sink("myoutput", append=TRUE, split=TRUE)
pdf("mygraphs.pdf")
source("script2.R")
```

文件 script2.R 中的 R 代码将执行，结果也将显示在屏幕上。除此之外，文本输出将被追加到文件 myoutput 中，图形输出将保存到文件 mygraphs.pdf 中。

译者 / 肖楠
@road2stat，中南大学统计学专业在读。useR，常年混迹于“统计之都”（http://cos.name）论坛 R 语言版。统计学习、数据可视化、渗透测试的爱好者。

最后，如果我们执行语句：

```
sink()
dev.off()
source("script3.R")
```

文件 script3.R 中的 R 代码将执行，结果将显示在屏幕上。这一次，没有文本或图形输出保存到文件中。整个流程大致如图 4 所示。

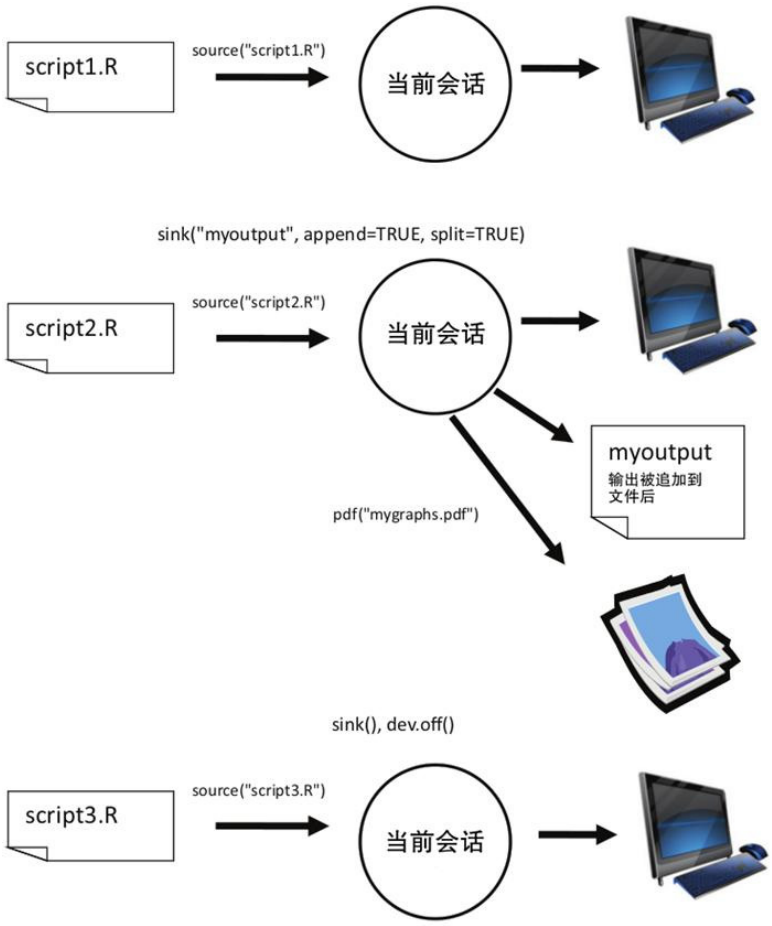


图 4　使用函数 source() 进行输入并使用函数 sink() 进行输出

R 对输入来源和输出走向的处理相当灵活，可控性很强。

专题：活的信息

iGraph——图挖掘助力社会网络分析

作者 /Ricky Ho

Ricky 是一位软件架构师和咨询师。钟情于分布式平行运算，机器智能和数据挖掘，SaaS 与云计算。现住在美国加州。个人博客：http://horicky.blogspot.com

社交网络（如 Facebook，Twitter）可以完整地表现人们的生活。人们用不同的方式与他人互动，并且这些信息都可以在社交网络中抓取到。挖掘某个站点的有用信息可以帮助一些团体增加竞争力。

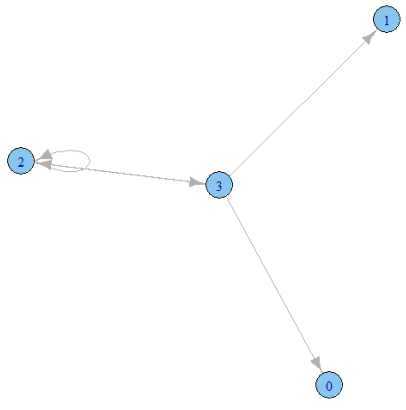
我最近无意中发现一款叫做“iGraph”的工具，它提供了一些非常有效的挖掘功能。以下列举几条我觉得有意思的：

创建图表

图表由节点和连线组成，两者都可以附上一系列属性值（键 / 值对）。此外，连线可以是有向的也可以是无向的，还可以给它加上权重。

```
> library(igraph)
> # Create a directed graph
> g <- graph(c(0,1, 0,2, 1,3, 0,3), directed=T)
> g
Vertices: 4
Edges: 4
```

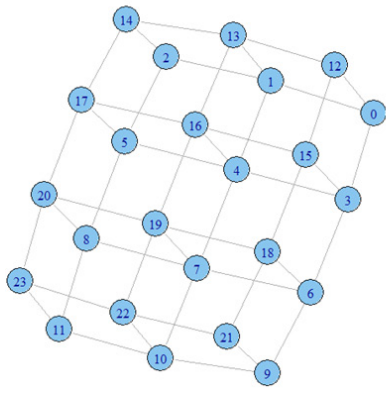
```
Directed: TRUE
Edges:

[0] 0 -> 1
[1] 0 -> 2
[2] 1 -> 3
[3] 0 -> 3
> # Create a directed graph using adjacency matrix
> m <- matrix(runif(4*4), nrow=4)
> m
[,1]      [,2]      [,3]      [,4]
[1,] 0.4086389 0.2160924 0.1557989 0.2896239
[2,] 0.4669456 0.1071071 0.1290673 0.3715809
[3,] 0.2031678 0.3911691 0.5906273 0.7417764
[4,] 0.8808119 0.7687493 0.9734323 0.4487252
> g <- graph.adjacency(m > 0.5)
> g
Vertices: 4
Edges: 5
Directed: TRUE
Edges:

[0] 2 -> 2
[1] 2 -> 3
[2] 3 -> 0
[3] 3 -> 1
[4] 3 -> 2
> plot(g, layout=layout.fruchterman.reingold)
>
```



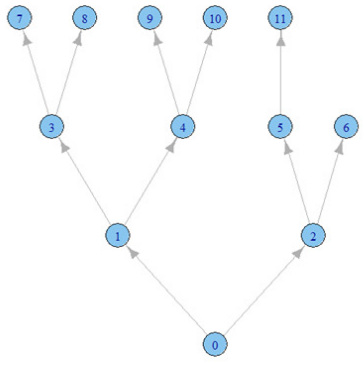
iGraph 也提供了多种创建各种图形的图表的简单方法

```
> #Create a full graph
> g1 <- graph.full(4)
> g1
Vertices: 4
Edges: 6
Directed: FALSE
Edges:

[0] 0 -- 1
[1] 0 -- 2
[2] 0 -- 3
[3] 1 -- 2
[4] 1 -- 3
[5] 2 -- 3
> #Create a ring graph
> g2 <- graph.ring(3)
> g2
Vertices: 3
Edges: 3
Directed: FALSE
Edges:

[0] 0 -- 1
[1] 1 -- 2
[2] 0 -- 2
> #Combine 2 graphs
> g <- g1 %du% g2
> g
Vertices: 7
Edges: 9
Directed: FALSE
Edges:

[0] 0 -- 1
[1] 0 -- 2
```

```
[2] 0 -- 3
[3] 1 -- 2
[4] 1 -- 3
[5] 2 -- 3
[6] 4 -- 5
[7] 5 -- 6
[8] 4 -- 6
> graph.difference(g, graph(c(0,1,0,2), directed=F))
Vertices: 7
Edges: 7
Directed: FALSE
Edges:

[0] 0 -- 3
[1] 1 -- 3
[2] 1 -- 2
[3] 2 -- 3
[4] 4 -- 6
[5] 4 -- 5
[6] 5 -- 6
> # Create a lattice
> g1 = graph.lattice(c(3,4,2))
> # Create a tree
> g2 = graph.tree(12, children=2)
> plot(g1, layout=layout.fruchterman.reingold)
> plot(g2, layout=layout.reingold.tilford)
```



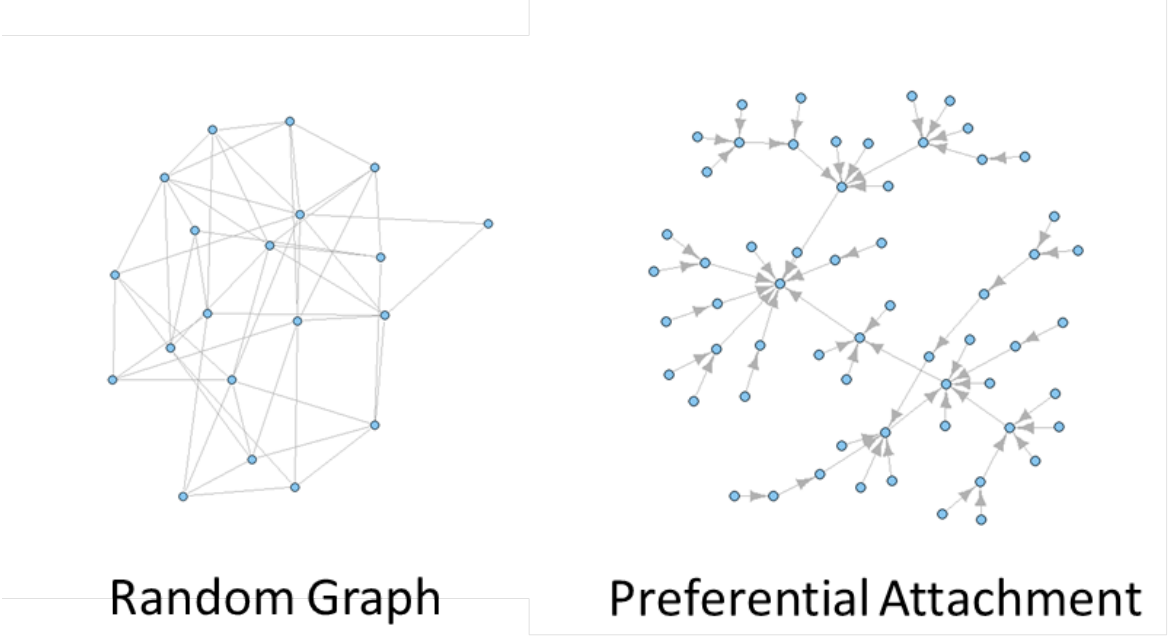
Lattice



Tree

iGraph 还提供了另外两种图表生成的机制。“随机图表”可以在任意两个节点之间进行连线。而“优先连接”会给已经拥有较大度数的节点再增加连线（也就是多者更多）。

```
# Generate random graph, fixed probability
> g <- erdos.renyi.game(20, 0.3)
> plot(g, layout=layout.fruchterman.reingold, vertex.
label=NA, vertex.size=5)

# Generate random graph, fixed number of arcs
> g <- erdos.renyi.game(20, 15, type='gnm')

# Generate preferential attachment graph
> g <- barabasi.game(60, power=1, zero.appeal=1.3)
```

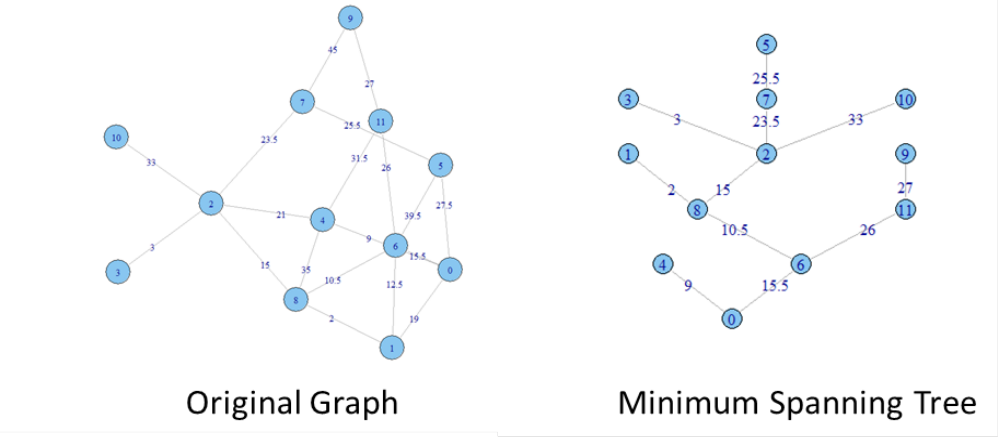


简单图表算法

这一节会介绍如何使用 iGraph 来实现一些简单的图表算法。

最小生成树算法可以在图表里连接所有的节点，并使所有的连线权重最小。

```
# Create the graph and assign random edge weights
> g <- erdos.renyi.game(12, 0.35)
> E(g)$weight <- round(runif(length(E(g))),2) * 50
> plot(g, layout=layout.fruchterman.reingold, edge.
label=E(g)$weight)
# Compute the minimum spanning tree
> mst <- minimum.spanning.tree(g)
> plot(mst, layout=layout.reingold.tilford,  edge.
label=E(mst)$weight)
```



连通分支算法可以找到会连通其他节点的连接，也就是说，两个节点之间的路径会穿过其他节点。需要注意的是，在无向图里连通是要对称的，在有向图（节点 A 指向节点 B，但节点 B 不指向

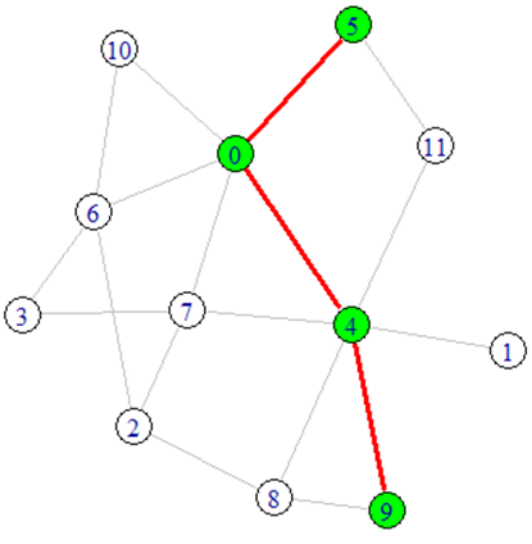
节点 A 的图表）里不是必须的。因此在有向图中存在一种连接的概念叫做“强”，也就是只有两个节点都分别指向对方才意味着它们是连通的。“弱”的连接意味着它们不是连通的。

```
> g <- graph(c(0, 1, 1, 2, 2, 0, 1, 3, 3, 4, 4, 5, 5,
3, 4, 6, 6, 7, 7, 8, 8, 6, 9, 10, 10, 11, 11, 9))
# Nodes reachable from node4
> subcomponent(g, 4, mode="out")
[1] 4 5 6 3 7 8
# Nodes who can reach node4
> subcomponent(g, 4, mode="in")
[1] 4 3 1 5 0 2

> clusters(g, mode="weak")
$membership
 [1] 0 0 0 0 0 0 0 0 0 1 1 1
$csize
[1] 9 3
$no
[1] 2

> myc <- clusters(g, mode="strong")
> myc
$membership
 [1] 1 1 1 2 2 2 3 3 3 0 0 0
$csize
[1] 3 3 3 3
$no
[1] 4

> mycolor <- c('green', 'yellow', 'red', 'skyblue')
> V(g)$color <- mycolor[myc$membership + 1]
> plot(g, layout=layout.fruchterman.reingold)
```

最短路径算法是最普遍的算法，它能找到节点 A 和节点 B 之间最短的路径。在 iGraph 里，如果图表是未加权的（也就是权重为 1

的）而且在权重为正时使用了迪杰斯特拉算法，会使用 “breath-first search” 算法。要是连线的权重是负数，则会使用 Bellman-ford 算法。

```
> g <- erdos.renyi.game(12, 0.25)
> plot(g, layout=layout.fruchterman.reingold)
> pa <- get.shortest.paths(g, 5, 9)[[1]]
> pa
[1] 5 0 4 9
> V(g)[pa]$color <- 'green'
> E(g)$color <- 'grey'
> E(g, path=pa)$color <- 'red'
> E(g, path=pa)$width <- 3
> plot(g, layout=layout.fruchterman.reingold)
```



Shortest Path

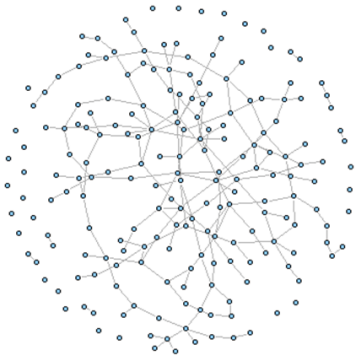
图表统计

通过大量统计信息我们可以大致看到图表的形状。在最高权限下，我们可以看到图表的各类信息，它包括：

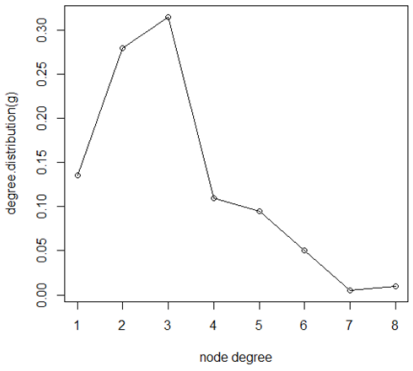
- 图表的大小（节点和连线的数量）
- 图表的密度是紧密的（|E| 与 |V| 的平方成正比）还是稀疏的（|E| 与 |V| 成正比）？
- 图表是连通的（大部分节点是互通的）还是非连通的（节点是孤立的）？
- 图表中最长的两点之间距离
- 有向图的对称性
- 出 / 入“度”的分布

```
> # Create a random graph
> g <- erdos.renyi.game(200, 0.01)
> plot(g, layout=layout.fruchterman.reingold, vertex.
label=NA, vertex.size=3)
> # No of nodes
> length(V(g))
[1] 200
> # No of edges
> length(E(g))
[1] 197
> # Density (No of edges / possible edges)
> graph.density(g)
[1] 0.009899497
```

```
> # Number of islands
> clusters(g)$no
[1] 34
> # Global cluster coefficient:
> #(close triplets/all triplets)
> transitivity(g, type="global")
[1] 0.015
> # Edge connectivity, 0 since graph is disconnected
> edge.connectivity(g)
[1] 0
> # Same as graph adhesion
> graph.adhesion(g)
[1] 0
> # Diameter of the graph
> diameter(g)
[1] 18
> # Reciprocity of the graph
> reciprocity(g)
[1] 1
> # Diameter of the graph
> diameter(g)
[1] 18
> # Reciprocity of the graph
> reciprocity(g)
[1] 1
> degree.distribution(g)
[1] 0.135 0.280 0.315 0.110 0.095 0.050 0.005 0.010
> plot(degree.distribution(g), xlab="node degree")
> lines(degree.distribution(g))
```



Random Graph



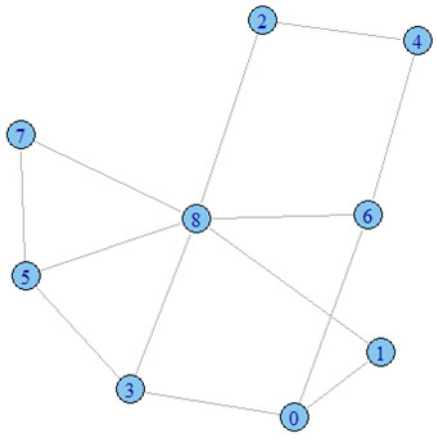
Degree distribution

往下一点，我们也可以看到每对节点的统计信息，比如：

- 计算两点之间没有公用连线的路径（也就是需要移除多少条连线可以使两节点不连通）
- 计算两点之间的最短路径
- 计算两点之间路径的数量和长度

```
> # Create a random graph
> g <- erdos.renyi.game(9, 0.5)
> plot(g, layout=layout.fruchterman.reingold)
> # Compute the shortest path matrix
> shortest.paths(g)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]013122132
 [2,]102232221
 [3,]320212221
 [4,]122031221
 [5,]231303132
 [6,]222130211
 [7,]122212021
```

```
 [8,]322231201
 [9,]211121110
> # Compute the connectivity matrix
> M <- matrix(rep(0, 81), nrow=9)
> M
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]000000000
 [2,]000000000
 [3,]000000000
 [4,]000000000
 [5,]000000000
 [6,]000000000
 [7,]000000000
 [8,]000000000
 [9,]000000000
> for (i in 0:8) {
+   for (j in 0:8) {
+     if (i == j) {
+      M[i+1, j+1] <- -1
+     } else {
+      M[i+1, j+1] <- edge.connectivity(g, i, j)
+     }
+   }
+ }
> M
       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]-122323323
 [2,]2-12222222
 [3,]22-1222222
 [4,]322   -123323
 [5,]2222   -12222
 [6,]32232   -1323
 [7,]322323   -123
 [8,]2222222   -12
 [9,]32232332   -1
>
```



中心性计算

在细节方面，我们可以看到各个节点的统计信息。根据这些数字可以测出节点的“中心性”

- 拥有较高出 / 入度数的节点也拥有较高的“度中心性”
- 与其他节点之间有短路径的节点拥有较高的“密集中心性”
- 与其他节点对之间有最短路径的节点拥有较高的“中间性”
- 连接了许多中心性较高节点的节点拥有较高的“特征向量中心性”
- 本地簇系数意味着相邻节点的互联性

```
> # Degree
> degree(g)
[1] 2 2 2 2 2 3 3 2 6
```

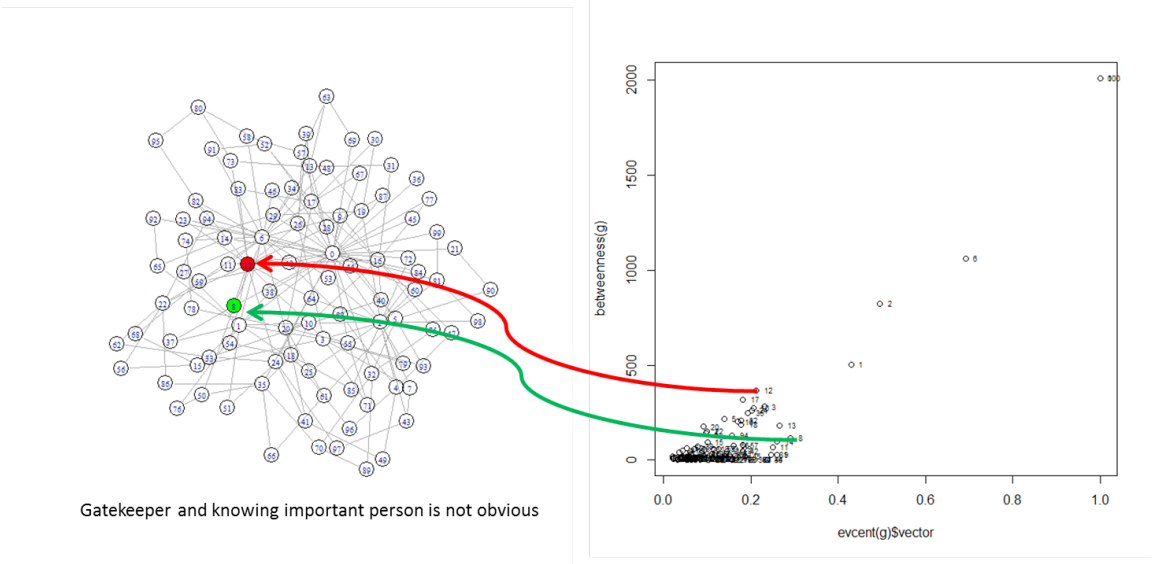
译者 / 周庆成
江西南昌人，毕业于上海海洋大学数学系。在学校的数字海洋研究所里接触了 Mac 和 iOS。从此对移动开发产生兴趣。喜欢看书。兴趣广泛。对各类知识都非常渴求。经常翻译一些文章，现在翻译另一本 Obiective—C 语言的入门书。目前居住于上海。从事游戏开发。
图灵社区 ID：周庆成

```
> # Closeness (inverse of average dist)
> closeness(g)
[1] 0.4444444 0.5333333 0.5333333 0.5000000
[5] 0.4444444 0.5333333 0.6153846 0.5000000
[9] 0.8000000
> # Betweenness
> betweenness(g)
[1]  0.8333333  2.3333333  2.3333333
[4]  0.0000000  0.8333333  0.5000000
[7]  6.3333333  0.0000000 18.8333333
> # Local cluster coefficient
> transitivity(g, type="local")
[1] 0.0000000 0.0000000 0.0000000 1.0000000
[5] 0.0000000 0.6666667 0.0000000 1.0000000
[9] 0.1333333
> # Eigenvector centrality
> evcent(g)$vector
[1] 0.3019857 0.4197153 0.4197153 0.5381294
[5] 0.3019857 0.6693142 0.5170651 0.5381294
[9] 1.0000000
> # Now rank them
> order(degree(g))
[1] 1 2 3 4 5 8 6 7 9
> order(closeness(g))
[1] 1 5 4 8 2 3 6 7 9
> order(betweenness(g))
[1] 4 8 6 1 5 2 3 7 9
> order(evcent(g)$vector)
[1] 1 5 2 3 7 4 8 6 9
```

从中 Drew Conway 发现拥有低“特征向量中心性”和高“中间性”的人是很重要的联系人，而拥有高“特征向量中心性”和低“中间性”的人与重要的人有关联。现在我们来绘制“特征向量中心性”和“中间性”的图表。

```
> # Create a graph
```

```
> g1 <- barabasi.game(100, directed=F)
> g2 <- barabasi.game(100, directed=F)
> g <- g1 %u% g2
> lay <- layout.fruchterman.reingold(g)
> # Plot the eigevector and betweenness centrality
> plot(evcent(g)$vector, betweenness(g))
> text(evcent(g)$vector, betweenness(g), 0:100,
cex=0.6, pos=4)
> V(g)[12]$color <- 'red'
> V(g)[8]$color <- 'green'
> plot(g, layout=lay, vertex.size=8, vertex.label.
cex=0.6)
```



Gatekeeper and knowing important person is not obvious

在之后的帖子里我还会介绍一些特殊的社交网络分析的例子。

查看英文原文：Basic graph analytics using igraph

专题：活的信息

量化分析：

用数据、指标、信息讲述的故事

作者 /Martin Klubeck

圣母大学的一位战略规划顾问，实用量化分析领域的知名专家。他是《量化：大数据时代的企业管理》的作者，《组织改进为什么事倍功半》的合著者，在量化分析方面也发表了很多论文。另外，他还是信息技术绩效标准构建联盟的奠基人，这个非盈利组织专注于提供业界亟需的指标标准。

如果有一种通俗易懂的语言，能让所有人（无论其人生阅历或教育经历如何）都能明白量化的好处，该有多好！我认为语言不通是事业（和生活）的最大障碍。所以，提炼总结出公用词汇至关重要，是迈向成功的第一步。

本书中的很多概念可能都比较新颖，但这并不意味着要发明新词儿，组织发展术语表已经臃肿不堪了。实际上，我用的都是常见词汇。尽量用大白话介绍那些看上去很复杂的概念，让其浅显易懂，简单直白。

数据

为了使沟通更顺畅，先来定义一些基本术语。数据（data）、指标（measure）、信息（information）、量化分析（metric）截然不同，但又彼此关联。每一个都建立在另一个的基础之上。量化分析由信息和其他量化分析组成，信息由指标组成，而指标又由数据组成。

图 1 展示了一些相互独立的实体，人们经常认为它们和量化分析

数据是最简单的信息形式，通常表示为数字或常量值。

相关，甚至认为它们就是量化分析。

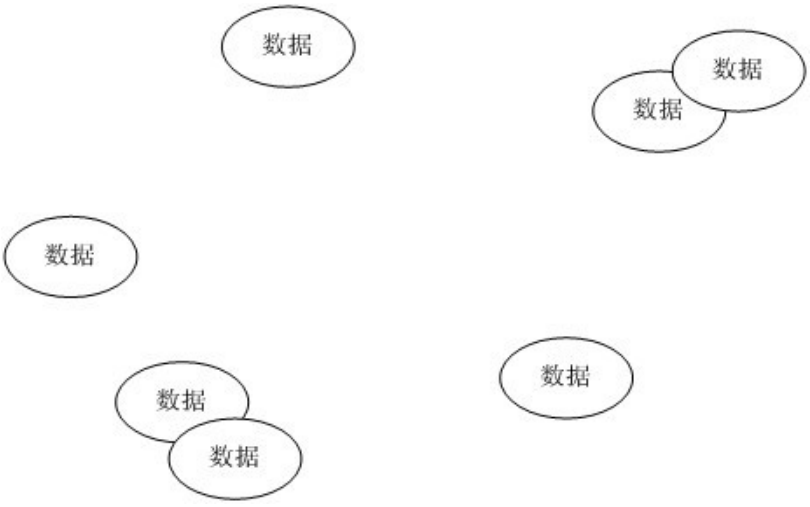


图 1　数据关系图

通常，数据被定义为“单一的事实，统计项，或信息项”。然而这个定义有些言过其实。它暗示数据是精确的，并且具有某种实用性，但其实数据本身并不具备什么实用性。按我的定义，数据就是最简单的信息，通常用数字或常量值表示，比如：6，22，70，真，假，高，低。因为没有和任何有意义的信息关联，数据本身并没什么用处。如图 1-1 所示，重叠的泡泡表示有些数据能够“联系起来”，但数据定义中并不包括这种联系。

数据之间可能毫无关联（泡泡之间的距离很远），也可能因为某个共同的目标联结在一起。在分析数据时，可以用关系图这种可视化方式表示数据之间的关系。但有时这些关系并不存在，仅仅因为来自同一数据源或采集目的相同而被误认为彼此相关。比如“响应时长”和“解决时长”，可能看似相互关联。 因为它们数

据源一致，都来自于同一问题跟踪系统。数据类型（时间）相同，也会给人造成两者关系密切的假象。实际上，无论是否来自同一数据源或采集目的是否相同，数据之间往往毫无关联。如果把无关数据误认为彼此相关的数据，就可能会得出错误结论。比如响应时长和解决时长，它们其实并不会互相影响，表示的也是不同的事情。

指标

图 2 说明了信息的下一层级：指标和数据之间的关联关系。

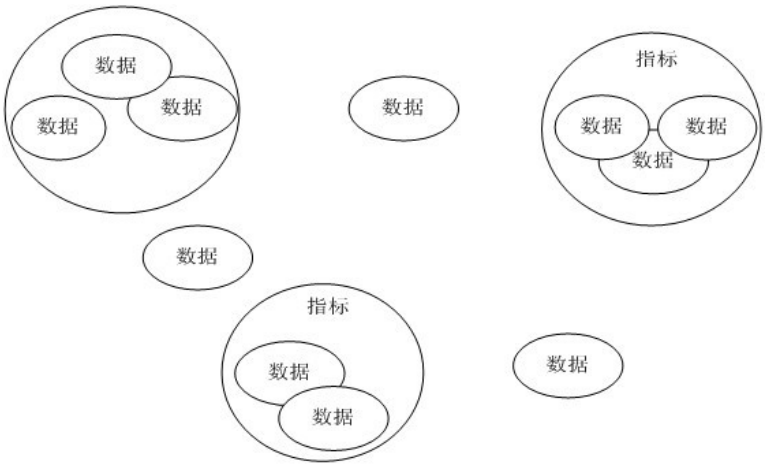


图 2　指标和数据关系图

因为有更多细节，所以指标更有价值。这种细节可能包括指标单位（以 50% 为例，指标单位是“百分比”，数据是 50），以及数据之间的关联关系。“70%”的含义比简单的“70”更丰富，如果能明确是“63 个用户中的 70%”，那就更有价值了。每个指

指标将真正有关系的数据分组并添加上下文，使数据的含义更清晰。

标都由一或多个数据组成。指标跟数据一样，彼此间可能存在不同程度的关联关系。图 1-2 左上角的泡泡表明不能归结为某个指标的一组数据。虽然这些数据彼此关联，但它们不能构成含义更丰富的指标。人口统计数据、身高、体重就是这样，每项数据都有用，但不能合在一起构成一个更大的指标。

图中还有三个孤立的数据。这些离群（“没有关联”）的数据，以后也许有用，也许毫无价值。

指标将真正有关系的数据分组，添加上下文，使数据的含义更清晰。然而，没有对应到目标或根本问题（详见下文）的指标，也仅仅是装扮起来的数据。

信息

图 3 展示了第一层可用信息——我们就称之为“信息”。信息对指标和数据（包括离群数据）分组，赋予它们明确的含义。

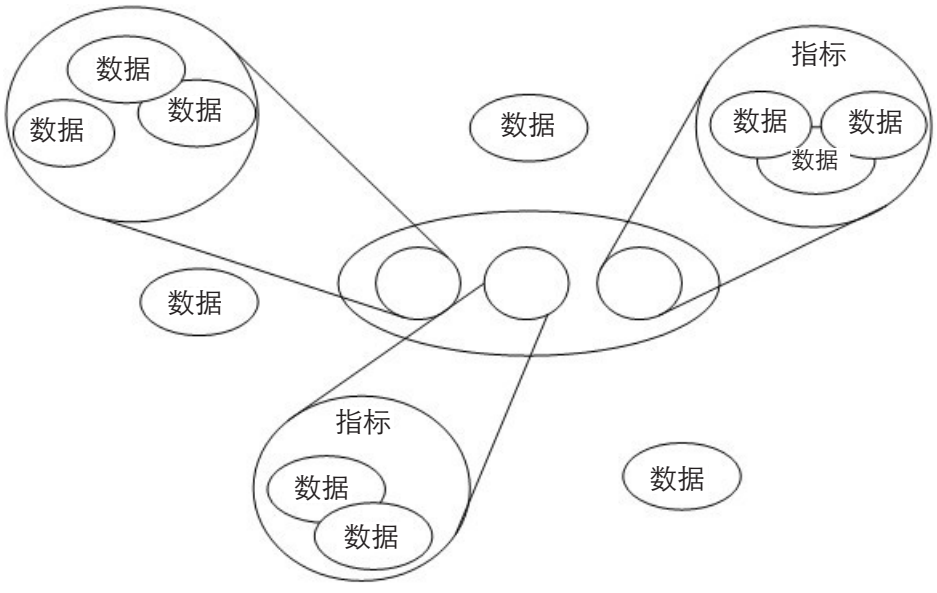


图 3　数据、指标、信息关系图

信息添加了上下文含义，使指标更好理解。

信息汇聚指标和数据，并添加上下文。上图中的信息并没有囊括所有数据，无论计划多么周密，采集的数据如何优质，总会得到一些没用的数据。最后，你会发现这些数据要么和要解决的问题格格不入，要么毫无帮助。信息只会收编需要的数据和指标。

上下文信息对于理解数据和指标中的数值非常必要。对于指标，我们只知道是在讨论一定数量用户中的百分比，而添加了上下文的信息含义更丰富，对理解指标更有帮助：“63 个用户中的 70% 更喜欢滑雪机，而不是踏步机。”

尽管上下文正确的信息非常有价值，但真正需要的可能还是量化分析。

在讨论下一个概念之前，先看个例子 ，也许会对了解信息（数据和指标）如何嵌入整张图中更有帮助。图 4 是与解决速度相关的信息。

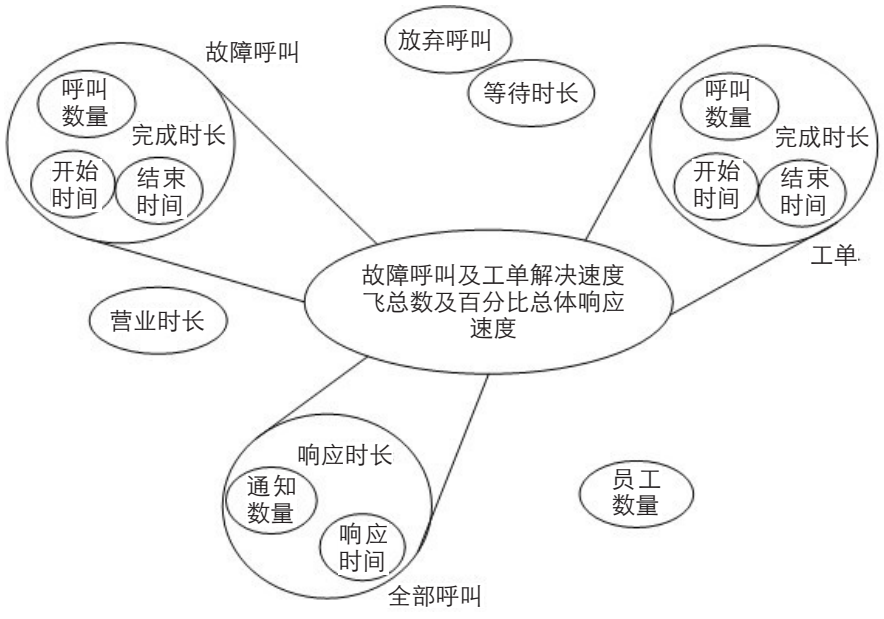


图 4　解决速度关系图

量化分析做完整描述，全面解答根本问题。

图 5 是由信息、指标和数据组成的一幅插画，展示了一个完整的故事，可比作量化分析。老话说得好：“一图抵千言。”

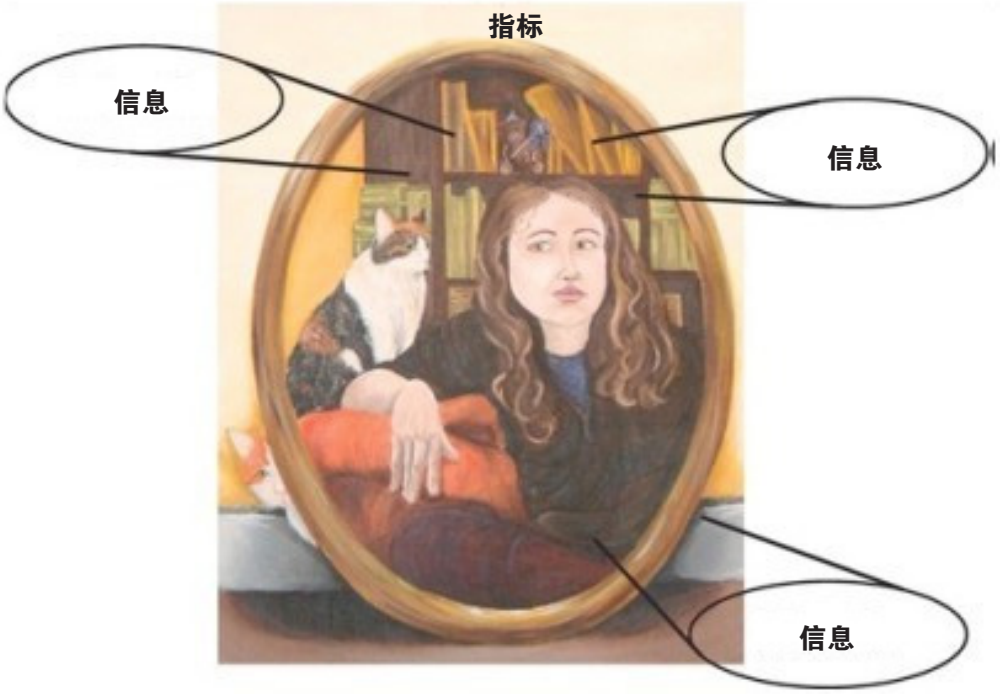
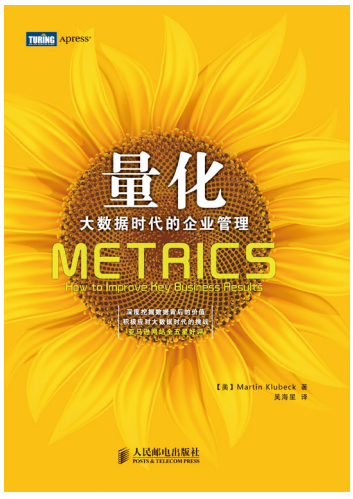


图 5　量化分析就像一幅画 插画制作：Alyssa Klubeck

终于，我们要定义最重要的“量化分析”了。量化分析不仅仅是把一堆信息分到一块。好吧，其实也差不多。

按我的定义，量化分析是由信息、指标和数据组成的，它也可以包含其他量化分析。量化分析与信息最大的区别是，量化分析讲述完整的故事，全面解答根本问题。另外，量化分析所做的描述必须是“正确的”。如果量化分析做的描述完全错误，那它就没什么价值。除了数据、指标和信息，量化分析还包括文字说明。



在企业管理领域，人们笃信这样一个原则：用数据说话。在从粗放的管理到科学地决策的转变中，数据发挥着关键的作用。进入大数据时代，人们面对着空前膨胀的海量数据，如何从其中挖掘出有用的指标和信息，帮助你进一步提升企业的绩效呢？《量化：大数据时代的企业管理》系统介绍了提升企业绩效的方法和工具，不仅阐述了企业应进行量化的必要性，而且详细讲解了如何创建量化分析体系，使其更好地反映组织健康状况和客户满意度。本文选自《量化》。

尽管一幅图可能会胜过千言万语，但面对没有任何解释的图时，往往是仁者见仁智者见智，所见所感千差万别。

如果你已经在量化分析上下足了功夫，配合图形、图表和表格将故事讲得一板一眼，那么你对它的解释就很难出现偏差，但也不能保证完全没有曲解。除非确信量化分析的观众没有自己的见解，不会曲解量化分析，持完全开放的态度，否则我强烈建议你给图画配上文字说明。

文字说明并不是什么新概念。我在读大学的女儿报了一门艺术欣赏课， 在她的教材中，每件艺术品都配上了几页文字介绍，讲述创作该作品的艺术家，其创作背景、所投入的时间、所用素材以及当时的社会大环境。我对此颇有感触，因为在我读高中的时候艺术老师就讲过，艺术品的含义就是欣赏者对它的解读，除此之外它没有任何意义。现代艺术更是如此（我一直没搞懂）。这本书没让欣赏者们玩猜猜猜，而是对隐藏在图画之后的重要信息逐一详细解说。不仅油画、雕塑和素描有，版画、雕刻和史前壁画也有。这些文字说明进一步解释了艺术家通过胜过万语千言的图像要“传达”什么。

如果艺术品的含义有必要解释，那么量化分析的含义是不是更有必要明确呢？而且，是不是最好由创作量化分析的“艺术家”自己来阐述呢？当然，对量化分析的任何解释都仅仅是一种解释，每个人对量化分析的看法都会不一样。但你肯定希望是你自己的解释与这幅图一起出现。如果量化分析使用得当，人们不会把你的解释当做唯一的“真理”，而是就把它当做解读量化分析含义的一种方式。

“在进行有氧锻炼时，63 个用户中的 70% 更喜欢用滑雪机，而

译者 / 吴海星

《量化：大数据时代的企业管理》译者。2001 年毕业于南京理工大学，现任北京北控文化体育有限公司技术总监。熟悉 Java 及 Scala 语言，熟悉 web 应用系统开发，熟悉 IC 卡应用系统建设，目前主要对商业智能方面的内容感兴趣。

图灵 ID: 海兴

不是踏步机。这里有 3 个滑雪机和 12 个踏步机，滑雪机的平均等待时间是 25 分钟，而踏步机前通常没人。”这就有点像个“好”的量化分析了。如果再配上一幅图（图形或者图表），那就更像了。量化分析的目的是解答根本问题，讲述一个完整（而有用）的故事。

应该怎么办呢？买更多的滑雪机？撤掉些踏步机？限制客户占用滑雪机的时长？开设一门踏步机的训练课程？显然我们还是不知道该怎么办。困惑的部分原因可能是因为没有提出明确的问题。为什么收集数据？为什么要分析？如果没有明确根本问题，就不可能讲述好完整的故事。

冯大辉：信息真正的意义

冯大辉，微博网名 @Fenng。生物专业毕业，被 Oracle 授予过 Oracle ACE Director 荣誉。曾就职于支付宝，负责领导数据库技术团队，现任丁香园 CTO。他热爱互联网，每天在微博上传播“小道消息”，长年维护个人博客 DBA Notes。现在和妻子生活在杭州，并养了一只猫（猫泽西）。

他既是码农，又不是码农，在微博有 20 万粉丝的他“声音”很大，可他说话的声音很小。冯大辉是数据库方面在国内数一数二的权威专家，可以说他参透了数据和信息在技术层面的含义。可是如今，技术上的纯熟已经无法满足他的视野，他开始探寻信息真正的意义。前路依然坎坷，希望风景越来越好。

圈子

> 虽然完全是野路子撞进这个圈子，但是在每个环境中，新天地都不断地打开。

图灵社区：你大学学的是什么专业？

我没有经过高考，保送上的大学。当时其实有点被忽悠了，大家都说 21 世纪是生物学的世纪。我就在离家比较近的大学保送上了一个生物学的专业，还是本硕连读那种，10 年前读硕士还是挺稀罕的事儿。但是上了学之后，我就知道，这根本不可能是我们这

代人的未来，我其实更喜欢理工类的东西。由于我没有经历高考，所以有时间在高中时读了很多人文类的书，容易胡思乱想，而生物基础课中有太多死记硬背的内容，我当时也没有体会到什么乐趣。于是，大学第一年，基本已经对整个专业很绝望了。

图灵社区：你是如何开始学习计算机的?

有两类人，一类人就是（有）苹果机或学习机的一代，我就属于（什么机器都）没有一代。我接触计算机其实非常晚，是从大学才开始的。

大学时在图书馆里，看见有个人在拿着本书看，我对那本书很感兴趣，印象很深，但是没有记住书的名字，后来就在我快把这事忘了的时候，有一天我又来到图书馆，看见有本书放在桌子上，刚巧，就是那本书。我拿起书，读了介绍，内容讲的几乎就是整个计算机发展的背景。这是一本介绍 Linux 的图书，通过学习这个操作系统，我对这个行业有了一个最初的认识。后来越了解越多，反而把开发层面的东西都丢掉了。Linux 在当时只在小圈子流行，那时候业界流行的技术是 Delphi，PB，“PB 程序员，月薪万元”，那时候月薪万元是什么概念啊，我每个月的伙食费才 2、3 百，我要是有那么多钱，不是想吃什么就吃什么了吗（笑）？

后来在实习的公司里有一些比较厉害的人，在看 TCP/IP 的书，就是 Stevens（W.Richard Stevens）那个系列（TCP/IP 系列，《UNIX 网络编程》），以及道格拉斯 • 科莫 (Douglas E. Comer) 的 TCP/IP 系列的书。我看见他们在看这些书，就偷偷买回家去，自己吭哧吭哧地啃，啃了一大段时间 。早期关于 Linux 和 Unix 的经典书，我几乎都有，都是靠伙食费买的。开始都是点的积累，然后才变成面的知识，由点到面，后来当我站在书店里面，逐渐就知道自

己喜欢读什么样的书，而哪些书又不是自己的菜。我和很多朋友聊的时候，发现这也是大家共同的经历。现在的程序员面对的信息量太大了，我那个时候的问题是找东西找不到。**如果我最开始能读到 Dennis 讲 C 语言的那本书（《C 程序设计语言》）的话，我现在肯定是个 C 程序员**，可是，我是快毕业的时候才读到这本书的，非常遗憾。当时我读完第一章就深深地觉得谭浩强那本书绝对是把人带沟里去了，看完（那本书）之后，你仍然没法知道你写这些东西是为了什么，我看的时候就一直在想，病毒是如何调用底层的呢？完全想不通啊！

我学习的方式都是铺面的，我学 Linux 和 Unix 就会把绝大部分书都买回来，学习网络的时候也是这样，但是绝大部分编程书是不能这样看的，把每本书的代码都敲出来是不可能的。

后来有一位师兄，寒假的时候带我去外面实习，这家公司的业务就是承接电信给早期互联网用户的服务，其实很多基础的事背后都有很高的技术含量，在做这些琐碎的工作的过程中，不可避免要接触到互联网、互联网协议，以及底层的东西，加上之前我对 Linux 和 Unix 也已经有了一些了解，所以相当于我在网络这方面打通了一条线。经过在这家公司的锻炼，我积累了起码能干些活儿的能力。

图灵社区：但是要放弃本来的专业，选择到 IT 这个圈子里也需要一些决心吧？

我赚的第一笔钱是帮别人做的一个浏览器上用来管理（库存信息）的程序，虽然只有 1000 块，但那是我认可自己挣的第一笔钱，那是纯粹密集的脑力劳动，用了 10 天的时间，在一个网吧里，找了台机器，现学现卖，最后把东西做出来了。我的朋友都震惊了，

纷纷表示：靠，搞 IT 真 TM 挣钱！从那之后，我决心以后也要靠这种方式工作，也要通过这个方式赚钱。

我一直觉得自己没写过程序，应该确切地说是没有进入到软件工程的那个环境中去，因为我始终都没有参与规模作战，我做的事都是单兵作战，或是几个人的小团队。我觉得如果一个人要被称为码农的话就必须要在开发团队中做事情，要把我归纳进来的话，完全是滥竽充数。但是我喜欢这个圈子，也喜欢在这里做事儿。我关注这个行业，关注码农，关注这里面有趣的人。虽然完全是野路子撞进这个圈子，但是在每个环境中，新天地都不断地打开。

图灵社区：你第一份真正的 IT 工作是在哪里呢?

当时我所理解的 IT 就是北京的中关村，我仍然不知道怎样进入到这个领域中来。后来毕业的时候歪打正着就来到了北京的一家公司。我认为这归功于我的简历。我简历的 UI 设计绝对是超前的，所以简历投递出去，用人单位很容易就会把我的简历在一大堆里面挑出来，他们也不仔细看，就会给我打电话聊几句，还是拒绝的居多。通常都会问，“你是本科生？什么专业？”“生物技术。”“那好吧。”然后就免谈，所以我当时非常伤心（笑）。

后来终于中软要我了。**我到了那以后，人家说：“是有（冯大辉）这么个人，但是一直都没联系上，我去问问总经理还要不要。”**当时我就崩溃了，后来多亏总经理大发善心，把我留下了。当时一起去的虽然都是计算机专业的，但是操作系统技术还没有我熟呢。我的工作是做工程，相当于去当地做项目实施。在中软我虽然只呆了一年，但是收获还是挺大的，中软不少优秀的同事的做事风格对我影响很深。但是更多的学习收获来源于自学，我在这里进一步接触了 Unix、Oracle。我觉得可以把 Oracle 当作工作

一直作下去，所以倾注了很大精力，还考了很多证。当时我们有四五个人，年龄都差不多，也没什么事，就每个人选一个方向，开始学。公司的计算机很多，（软件）环境也都有，我们经常都会学到晚上 11、12 点钟。当时没人琢磨加班费什么的，公司有计算机，夏天还有空调，我们就很开心了。人被局限久了，忽然遇到这样的环境，学习东西的效率是非常高的。现在圈里搞数据库的几个人聊起来，觉得经历都差不多，刚毕业的几年，每天都是要学到很晚，自己拼命啃。有朋友甚至原来是学日语的，从 0 开始学，从无到有。最后也成了专家。

后来觉得收入太低，就离开了中软，那时还年轻，的确不怎么懂事。随后我又在北京工作了四年，一直从事 Oracle 相关的工作。后来我有一个朋友去了阿里巴巴，过了一年他问我要不要来，我说我刚换了工作，不去了。后来又过了一年，他又问我，要不要来。这次我犹豫了，于是去杭州观察了一下，阿里巴巴不象当时网上流传的那样 (当时有很多负面的信息)，是个很有生气的企业，很欣欣向荣的感觉，从环境到人，给我的印象都很好。于是这次下定决心，走吧。

走吧，去阿里！

> 我要是当时知道这个工作有那么累，我就真不去了。
> 但是一旦干起来，就不能掉链子。

图灵社区：在阿里的工作是怎么样的?

当时给我的工作是维护支付宝的数据库，我对这份工作是很期待的，很多东西都是刚刚做起来。当时支付宝属于急速扩张期，很着急在招人，而整个阿里也没有几个 DBA，这些人每个人手里还都有一摊事。我刚到杭州，还一头雾水的时候，就要马上开始高负荷的工作。后来我女朋友一个月之后到了杭州，她不知道我现在忙成这个样子，结果直到她来的那天我仍然在加班，后来几天一直处于这个状态，整晚整晚的连续工作，直到早上才回家。后来有一天我早上回到家里，一敲门，她开门看见我熬了一晚上很疲惫的样子，哇一下就哭了，说：“咱们不干了！”

我从三月到五月份两个月的工作，几乎一直都是这么度过的。当时的支付宝系统要从淘宝独立出来，自己独立建立一个全新的平台，阿里也从各个子公司调了很多专家进来，要在五月份上线。当时我们手机都是 24 小时开着，短信一来就要待命。杭州的冬天很冷，接到短信就要半夜爬到电脑前准备处理情况。经常是早上 5 点多刚到家里，7 点多就被吵醒了，让我 8 点到公司来。打电话的人现在是一淘的老大，我就说，那我也得睡一会啊。人要是累成那样，还哪管什么上下级啊。但是撂下电话，想一想，还是去吧 。那几年根本没时间考虑休假这些事，要是休假去了个手机不通的地方，是要出事的。而且当时长假（五一，十一）的时候经常要做大项目迁移。**我从来没想过在阿里退休，但我想我有可能在阿里猝死**。在支付宝的第一年差点没累死我（笑）。说实话，我要是当时知道这个工作有那么累，我就真不去了。但是一旦干起来，就不能再掉链子。

我最担心的就是数据丢失，系统崩溃，数据无法恢复。里面可都

是钱啊，如果搞不定这些，是要给公司带来直接经济损失的，追究起来，我就是责任人。我当时经常睡不安稳，会惊醒。由于心理压力大，人容易暴躁，身体也处于很糟糕的状态，其实也是一种恶性循环。我觉得我很累，有人比我还累。现在支付宝的首席架构师程立，当时博士还没毕业就加入了支付宝，我们当时一起和 Sun 的工程师熬夜工作，经常是我们这边维护完的时候，他还要继续写代码。**回头想想，像我这样能力平常的人，怎么还会变成了别人眼中的行业专家呢（请允许我假装一回吧）？有些人的确有天赋，我可能靠的就是卖力工作、拼命把所有事情搞清楚、认真地去做。**

图灵社区：但是很多年轻人认为私人时间是一定要保证的。

有些人提倡八小时工作制，有些人提倡加班，其实这些在一定的场景下，或者说对某一类人来说都是合理的。新一代的程序员和我们经历的事情不一样，虽然我很反感那些所谓的“感恩”论调，但是现在的年轻人很多都需要哄着来工作，可能这也是一种一代人看不惯下一代人的心理吧。我们当时崇拜的都是像求伯君、鲍岳桥，简晶这些人，或者搞开源软件的这些牛人，当时这些人里谁赚的钱最多？没人知道，也没人关心这些不相干的事情。

所以尽管我努力尝试去理解不同时代的 IT 人，但是很多时候仍然无法完全理解。从我大学毕业到现在，大概有 15 年时间。这个社会的变化非常快，是一个大时代。我大学时候，那男女关系是多么纯洁，哪像现在这样礼坏乐崩（笑）。现在很多人的世界观已经被完全颠覆了。人们一方面很向往美好事物，但是另一方面又要在现实中挣扎，和社会去斗，为了买房而奔波赚钱。

现在学校快变成工厂了，以利益为导向，间接的导致不少学生变

得千奇百怪，歪瓜裂枣。在学习方面，一个人没有挨过饿就不会吃的特别撑。

图灵社区：贝塔咖啡也是你在阿里的时候开始的吧?

贝塔咖啡，开的时候就是想给 IT 人弄个据点，平时大家也没什么地方去，想让大家有个聚一聚的地方。我都没想到后来会有这么多人开咖啡馆，当时看北京的雕刻时光挺好的，也没想过创业人，投资人什么的，就是单纯想给和我们差不多的人一个可以聊聊的地方。当时开的时候 5 个人里有 4 个在阿里，也没有人有精力全力去做，我们当时就明确: 这个东西不会赚钱。挺坑人的其实（笑）。在开咖啡馆的过程中，我也学到了不少，认识了很多人，知道了很多事。也知道传统行业不是那么好干的。这也是一种全新的体验。人的精力是有限的，把链条拉的太长了，想做好就很难。我们做的就不算好，只算是能够运营下去，和我们最初设想的还有差距。**里面咖啡不好喝，饭也不好吃，环境也不是特别好，但是我们想刻意保持这种气氛，让人知道这也不是什么高富帅去的地方，抱着孩子抱着狗你就别来了，这里就是给大家扯扯淡，拍拍肩膀聊一聊的地方**。现在这个咖啡馆也变成了我们几个合伙人的纽带了，我们现在都不在阿里，做的事情也不一样，但是出来了都不后悔。

公众人物的公众视角

> 经常，我会对需要打击的事给予鼓励，对于需要鼓励的事给予打击，这样似乎总让我站在对立面上。

图灵社区：你是怎么开始写博客、写微博的?

我原来的工作一直都偏系统集成，后来从 03 年开始，我开始逐渐关注互联网，我的主要娱乐之一就是写博客，还不能写公司的东西。写久了你就会知道好的文案应该怎么写，一个从来不写字的人肯定是写不出来的。写博客，写着写着就会有一条脉络，里面会纪录我实践的过程。如果有人真的有耐心把我所有的东西都读一遍，就会发现我一路走来的变化，这就是一个样本。我自己也会发现自己的变化，比如以前关注的事情会比较小，比较琐碎。现在我可能会关注比较虚，大一点的事情。这就是所谓的“人的经历会影响视野吧”。**没有变化是一件很可怕的事**。业界的很多人其实都是很好的写手，你可以认为他们没有别的特长，但是至少他们很能写。比如盖茨刚出道的时候就写出了“给软件爱好者的一封信”，他绝对不是个书呆子。很多人都觉得写东西酸不啦叽的没有用，这些人肯定错过了一些好东西。

经常写东西的人就会写的越来越短，为什么说专栏很难写，就是因为其实几句话就能说清楚的事，要写的非常长才可以，这是需要技巧的（笑）。我做过这种试验，把一条微博扩展成一篇文章，啰啰唆唆说一大堆，结果不是很成功。我有的时候也有一些恶作剧，比如经常会做一些伏笔，比如把 7 号写的东西，标上 8 号，结果只有霍炬看出来了，别人都不敏感。这个有点扯远了。总之，通过写博客，我间接学习到了很多知识。

图灵社区：你在网上总是在投诉一些东西，有什么东西很让你反感吗?

我大多数投诉的都是公职类，面向公众的服务，如果是私人公司的话，我也就是调侃一下。比如 12306 或者电信的一些东西，这

些基础设施类的服务就是要做好，做不好就是要被骂，这都是天经地义的。没有监督，这些公务员就会变成滚刀肉。如果阿里没有早期的内在驱动，后来怎么会变好呢？每一次系统故障，我们都是如临大敌。第一次系统故障就是由于我的疏忽，半夜把系统搞宕机了，多亏那个时候用户量小，没人用，我当时觉得这个东西迟一些做也没关系。后来出了问题，我连话都说不出来，就想找个地缝钻进去。这都是我应该做好的，是我的责任，做不好别人骂我是应该的。

李彦宏说的“狼性”，他讨厌的东西我都可以想象出来。**大公司里有一些人就是这样，干多干少都行，干好干差都可以，整天从这个会议室窜到那个会议室，看看他对公司的贡献，几乎等于 0。**大家为什么那么喜欢考公务员什么的，可能就是因为不想承担责任，没有业绩考核。整天无所事事是常态，想做事情的人反而会被视为异类，大家都会想：你瞎折腾些什么啊？和我们一样就可以了。百度现在属于那种很疲的状态，你踢一脚也没什么反应，“我就这样，我躺着钱就从天上掉下来了”。

我总是支持你们（图灵社区）的原因也是这样，我觉得任何行业里认真做事的人都值得认真的支持。

图灵社区：你在微博上很活跃，和形形色色的人打交道，有什么发现吗？

在微博上，你会发现不同的人是怎么看你的。如果有人骂你，那一定是因为他觉得从某些方面受到了冒犯、受到了攻击。我仍然会经常看看有些人的微博，比如老赵，看看他说些什么，和一年前比有什么变化。通过这些反馈我会对我的团队有更准确的判断，这样我才能了解他们有哪些诉求其实是很正当的。

我几乎没有批评过做技术的人，我骂的都是决策层。我这个人很容易改变自己的立场，你可以认为我没有原则，但是时过境迁，有一些说法必须要改变。随着对一些事物的观察和了解，我对它们的判断也会随之改变。经常，我会对需要打击的事给予鼓励，对于需要鼓励的事给予打击，这样似乎总让我站在对立面上。

图灵社区：有很多人都反映被你拉黑过?

一条微博我就能看出来我们是不是一路人，如果不是，那就不要沟通了。我记得有一回我说阿里不好，就有人跳出来说我不应该攻击老东家，我觉得这样的人是非常龌龊的。我很反感总是要尝试改变别人的人，总是想把想法强加到别人头上。我从来就不会把自己的想法强加给别人，而有些人却一定要说服我。还有的时候我说微软，这个时候搞微软技术的人就会跳出来，我也没有一杆子把所有人打死，你干嘛对号入座呢?

拉黑对我来说根本也不是什么重要的事，他们随便怎么想，我就是随手那么一点鼠标而已。我其实就是不希望他在我说的话下面留言。就像是有人总找你说话，而你就是不想和他对话，仅此而已。这些人也需要有人来刺痛他们，我就是要告诉他们，你说这句话就是很二。很二的代价就是你看不了我的东西。如果一定有其它事情需要联系的话，可以通过其他方式，比如邮件。

有些人素未谋面，就会对我进行人身攻击，这样的人都可以归纳到脑残那一类。经常是抱团来吵架的，他们不关注我说话的内容，上来就是攻击。如果我把微博删了，他们就会说，你看他，心虚了。这些人其实都是思维上的弱者，他们觉得他们终于赢了，我认错了（笑）。我不会和这样的人有更深入的合作，我会选择和更靠谱的人互动。比如说，**如果我说创新工场好，他们就会说我拿到**

好处了，如果再过段时间，创新工场给我投资了，他们会说，“你看，我早说了”。但是我也会反思，有时我说的话的确会有负面的东西，以前就有人说“丁香园怎么找了这么个傻逼来管技术，我要是 VC 就不给他们投资”，我就觉得很好笑，多亏这样的人他不是 VC。

至于调侃阿里云的话，其实我很希望中国能出一家云计算的公司，这样很多中小公司都会受益，我希望阿里云不要再瞎折腾了，赶紧提供一些可靠的服务吧。

图灵社区：很多程序员觉得自己过得很辛苦，你有什么想说的吗？

长期做一种事情的人容易形成一种观念，只有在我这个领域牛的人才是牛人，别的领域的牛人都是狗屎，都不行，看不上。写前端的和写后端的，搞微软的和搞开放技术的，写 C++ 的，觉得 Linux 领域的牛人都不行，IT 行业里的这种隔阂非常大，所以吵架在所难免。语言之争什么的，市场的选择也不是你争论出来的。

很多程序员过得没有希望是因为他们的视野太窄了，除了看技术，就是看科幻，我建议他们多看看人文历史类的书籍，这样的书可以引导他们理解别人的内心，看看小说什么的也可以很大程度上补充他们看问题的角度。程序员整天面对的就那么几个人，经理就是监工的、客户就是傻逼，每个人的角色都已经设定好了，如果没有更多了解，圈子就会越来越窄。应该尝试开阔一下视野。

比如，Paul Graham，画画对他是一种补充，这些积累有可能在一些关键点上决定一些东西。再比如说，如果我当时没学过生物，我现在又为什么要做（丁香园）？我现在倒是经常会买一些生物学的科普书，回头想想，其实我还是喜欢生物学的某些地方的。

生命起源、进化这类东西都是很有意思的，都会帮你打开视野。

信息的意义

> 没有什么能拯救中国互联网。中国互联网自己会拯救自己。

图灵社区：你是怎么加入丁香园的？有什么特殊的理由吗？

我喜欢给自己找一些看似正向的理由，当时去杭州，我说服我自己和家人的时候都是说电子支付是能改变电子商务的事情，是对社会有价值的。但是后来做久了发现，电子商务也主要是帮人赚钱，无法解决更多的问题。**支付宝做得再好，也无法改变人的生死。但是如果信息和服务可以做得更好，就有这样的可能。**

我的病（类风湿）发作起来的时候异常痛苦，如果没有经历，是无法理解那种痛苦的。比如关节痛的时候，我甚至恨不得拿刀把手腕砍断，就希望自己昏死过去，这样就感受不到了。长期处于病痛中，难免要开始考虑生老病死的问题。后来出现了机会，我就希望如果我们能更好地给医生提供信息，是不是可以让他们的医术和待遇更好一点，这样整个医疗行业是不是会更好一点。我看到了很多医生确实因为丁香园改变了他们的工作方法。我们想通过这样的网站更好地为中国的医生服务。做到这步之后，我们也想更好地服务于象我这样的慢性病、疑难病、罕见病患者。这个事情可能赚钱很慢，可是只做一点点，也是会造福的。对于用

药的知识丰富了，对医生的理解也就会更全面，人们会发现更好更积极的治疗方法。如果我十年前最早发病的时候能了解的更多，我就可以主动和医生配合，和他交流常规疗法以外的治疗方法，就能用很系统的方式治疗了，不会造成现在这样对身体的永久损伤。

有人问我们有没有考虑把食品安全这块也做进去，其实我也有这样的打算，我们的计划很庞大，但是有可能到最后也实现不了。我们遇到了很多很多的障碍，这在事前是无法想象的。就算做不大，我们也可以赚一些小钱，也能解决 100 多人的工作，其实也相当于为社会做了贡献。**中国的人口基数这么大，将来医疗肯定是个大问题，所以我们能做的事也是越来越多。而我们这些对信息了解更多的人为什么不来做这件事呢？**我一直觉得我们做的事是半公益性质的，但是不商业化是不可能的，我们的商业化是底层的，不会影响到使用。话说回来，如果我没有学过生物，如果我没有经历阿里的工作，如果我没有得过病，也许我不会认识到我现在工作的价值。我有可能只想要份好工作，一个好 Title，一份好薪水，就够了。

当时我在阿里，似乎感觉能一眼望到底，而我现在做的事都是非常规的，都是需要加入真正的判断才能做的事。我不知道我一年后是什么角色，也不知道我三年后会在干什么。

图灵社区：你最近的微信账号“小道消息”很火热，它的口号是“只有小道消息才能拯救中国互联网”，如何理解？

关于互联网，每天都会有大量的信息，真真假假的信息都有，多数信息都是不那么可靠的，倒是很多在私底下流传的信息反而更

加准确。只有小道消息才能拯救互联网，有一点恶搞的成分，其实还有下半句，“只有小道消息才能拯救中国”… 别那么严肃，这只是一句玩笑话，没有什么能拯救中国互联网。中国互联网自己会拯救自己。也没有谁能拯救中国，也要靠我们所有人。

我现在的确花了一点业余时间在运营微信公众帐号“小道消息”，最初的目的就是想摸索学习一下微信的运营方法，看是否能够对我们的业务有帮助。倒是有些一发不可收拾了。有的时候会夹带一点私货，发表一些我对个别事情的看法，我也不知道会坚持多久，既来之，则安之吧。

三天准备，迎战千倍用户

作者 /William Hertling

William Hertling 是一位作家，他的书《Avogadro 公司：奇点近在咫尺》与《AI 启示录》都是具有现实意义的科幻小说。他的书被形容为具有“令人惊惧的可行性”。他的书曾位列亚马逊科技惊险小说的前五名，其中《Avogadro 公司》赢得了年度 Forewords 书评奖，而《AI 启示录》被普罗米修斯奖提名为年度最佳小说。

当我没在写科幻小说的时候，我为一家相当大的公司做网页应用方面的工作。这个星期让人兴奋非常：我们在周一下午得知，我们必须在周四凌晨 5:30 分之前扩容到峰值访问量 10000 人的水平。

这篇文章将讲述的是：我们所做的，我们所学到的和真正有效的方法。

一点点背景描述：我们的站点使用 Nginx、Ruby on Rails 和 MySQL 建立。我们的网页应用是一个可定制的旅游指南。它的用户部分来自 TripIt 上的路线，也有人自己选择终点并填写上旅途的细节。

他们可以在浏览器中即时预览旅游指南并订购高质量的纸质版本。

我们的网站还处在测试阶段，每天有几百个访问者。周一下午两点，我们得知将在周四早上（东部夏令时 9:30）出现在一个非常受欢迎的电视节目的专题中，根据他们之前的经验，我们那天总共将迎来 2 万到 15 万名独立访问者，峰值约为 1 万人。这大约是我们平日里服务的独立访问者数的 100 到 1000 倍。我们预计，访问量峰值将达到 1 万人同时访问的水平，相应的页面请求负载将达到每分钟约 40000 次。

William Hertling 现居于纽约，他最初的网络经历就是把七根电话线和Apple相连，制造出一个在线聊天系统。
译者 /Trockeneis

我们现有的条件：

- 我们的应用运行在亚马逊云计算服务（AWS）上，在一台负载均衡器之后使用 EC2 弹性云服务；
- 我们已经设定了自动扩展规则，在发现服务器 CPU 占用率超过某个水平后将自动将我们使用的 EC2 实例翻倍；
- 我们有可靠的部署流程（尽管这在之后成为了一个局限）；
- 我们在一个月前花了一周时间优化了网页应用的性能表现，我们把页面载入时间缩短了 30%，这很大程度上是通过减少数据库调用来实现的。

周一，我们在几台电脑上安装了 JMeter。通过 JMeter，你可以写一段脚本模拟一个典型的用户访问。在我们的案例中，我们写的脚本会载入首页、选择目的地、查看用户设置然后付账离开。尽管我们没有提交任何页面（如果能有更多时间的话，我们很想这样做），在虚拟用户访问各个用户界面的时候，数据库中确实创建了几个实体（这在之后很重要）。

在一些实验之后，我们发现不能在一个 JMeter 实例中运行超过 125 个线程。因此，如果我们需要模拟真正的高流量的话，必须在多台电脑上运行 JMeter。我们还必须保证本地网络的任一部分都不能饱和。至少，我们确定不能让所有机器都走同一台路由。我们还通过 VPN 连接到了一个完全不同的地点，远程运行另一个客户端。

我们很快发现：

- 我们需要把最初的自动扩展组从两台服务器增加到四台；
- 我们需要把 CPU 占用率的阈值从 80% 降低到 50%；
- 我们需要把保持这种负载的时间从 5 分钟降到 1 分钟。

在做出这些改变并把服务器扩展到 16 台之后，我们发现我们遇到了数据库的限制。数据库服务器的负载很快就打到了 100%。我们把 AWS 数据库大小提升到了极限（超大号，带有额外的 CPU 和内存）但数据库服务器仍然应接不暇。我们观察着 JMeter 的测试运行，把平均响应时间控制在 30 秒的水平。我们知道，如果这个问题得不到解决的话，我们的肯定就完蛋了。当时是周二中午。

我刚刚说“我们知道我们会完蛋”，但说实话，这在当时看来确实有那种意思。团队中的有些成员觉得，临近那一天的任何改变都太过冒险。我们可能会带来一些 bug。但 100% 正确的应用体验却会被负载淹没，让任何人都打不开网页。几个小时后，我们让每个人都相信，我们必须解决扩展问题。

我们简单地了解了我们可以采用的几种较大的扩展方式，但最后却发现它们太过冒险。

- 复制我们的整个站点，使用 DynDNS 或亚马逊的 Route 53 云 DNS 解析服务来分送访问请求。因为我们没有用户数量或是其他共享数据，这在理论上本来是可以做到的。如果我们做到了，我们之后必须通过一些脚本来同步一系列数据库，把所有用户数据放到一起。如果我们有一周时间或者是有一位 Ops 专家，我们也许就能这样做了。

- 通过只读副本来降低数据库负载，把一些基本上是静态的数据（比如目的地列表和对它们的描述）所带来的数据库请求转移到只读数据库上。我们没有这样做，很大程度上是因为我们不知道应该如何把某些数据库请求转到一个数据库服务器上，同时把其他请求转到另一个服务器。

- 使用 Memcache 或其他机制把数据库请求转移到内存中：这同样会要求很大的代码修改。在我们现有的时间范围内，这样做的风险太大。

我们决定这样做。

- 在每次页面请求中，都有一些容易区分的数据库请求，它们每次都会载入同样的数据。我们重写了代码，用全局变量把它们缓存在内存中，在不同请求当中不会变化。因此它们不会在每次服务器重启时都要读取，而且再也不需要这样。

这时候已经是周三上午了，我们已经能够更流畅地使用 JMeter。我们没有用来自很多客户端的数百个并发线程冲击服务器并看着请求时间起起落落，我们慢慢地提升线程数，让服务器稳定下来，然后通过每秒请求数来观察总吞吐量，并检查 CPU 占用率，然后增加线程数并再次进行检查。我们制作了一个图表记录下相关数据，每个人都能看到这些数据。我们的目标是找出在不增加响应时间的前提下的最大每分钟页面请求数。

在那一天，我们逐渐增加了代码修改量来缓存某些数据库请求和周四上电视之前的其他晚期请求。这正是我们的部署进程开始出现问题的时候。

我们的环境如下：本地电脑、开发、演示、测试、推出。通常，我们在进入开发流程之前会运行测试脚本（要花约 40 分钟）。部署脚本要花 15 到 30 分钟（80% 的时间花在等待 git clone 复制上。为什么啊？？？）。在越来越靠近推出的时候，我们要多次重复这一过程。

测试是唯一一个与推出的产品是镜像的环境。如果需要测试优化后的代码，我们必须在这一阶段花上好几个小时部署这些变化。我们可以（也确实）登录测试服务器然后手动应用这些变化（使用 SSH 连接到服务器、VI 来编辑文件，重启 Mongrel），但是我们需要在 4 台服务器上重复这些步骤，撤下自动扩展服务器上，重建自动扩展策略，然后连上这些服务器。即使是简化后的操作也要花 20 到 30 分钟。

我们这样做了好几次，认真的记录下了我们的数据。我们发现，我们努力的方向错了。我们可以维持每秒 15 个请求（每分钟 900 个），随着时间流逝，这个数字却不断下降。聪明的数据库管理员读到这些数据的话会暗笑，会知道问题究竟出在哪里。但我们的团队中没有这样优秀的数据库管理员，我的数据库技巧也荒废了不少。

我们已经安装了 NewRelic 插件。从我在数年前开发一个 Facebook 应用开始，我就爱上了 NewRelic。这是一个在 Rails 和其他网页应用开发环境下使用的性能检测插件。我真的只用它当过开发工具，它能让你看到应用每个时刻都在干什么。

它同样可以用于发布检测，当时我并不知道我们是否购买了专业版，专业版可以确定运行缓慢的页面和数据库请求，并提供已执行的请求的大量细节数据。

因此，特别想得到更多数据的我们把 NewRelic 升级到了专业版。瞬间（不到 60 秒），我们就看到了哪些数据库请求较慢和我们的应用运作时间情况的宝贵数据。

一个数据库请求占据了 95% 的数据库时间。我在看到它的时候，想起了应用性能在那天持续变差，一切都明白了，我发现我们请求的地方存在未索引的区域。因为 JMeter 测试带来了对数据库中表的请求，因为它们已经运行了好几个小时，我们的数据库又有好几十万行，我们实际上是在扫描整个表。

我查看了数据库的 schema.rb 文件，发现有人错误地创建了一个多列索引。（通常，我认为应当避免使用多列索引，除非你清楚地知道这是必须的。）在多列索引中，如果你在请求中指定了 A、B、C 列，你可以用它查询 A、A 与 B、A 与 B 与 C。但这些列与顺序有关，因此你不能略去列 A。

这正是我们碰到的问题。因此我迅速创建了一个迁移来创建我们需要的列，迁移在我们自己的服务器上进行，给它 5 分钟来建立索引，我们随后重新进行了测试。我们把每秒请求数提升到了 120 次（每分钟 7200 次），这时的数据库负载仅 16%。

这时候差不多是周三下午 3 点。我们发现数据库服务器可以应付每分钟 36000 次的页面请求。我们再次着力解决应用服务器 CPU 占用率过高的问题。

周三，我们把 EC2 实例的数量提升到了最大。我们还有另外一个小组在进行稳定性方面的测试，我们碰到了某些上限。我们关掉了不需要的 AMI，让所有人停止测试，我们通过亚马逊云计算服务提交了提升上限的要求，然后联系客服让他们这些要求。

在周三晚上离开时，我们相信我们能够应付周四早晨的负载。

周四到来了，我们每个人都在所有屏幕上盯着 Google analytics、NewRelic 和后台界面。因为当时是太平洋标准时间凌晨 5:30，我们有人在家，有人在宾馆。我们使用 Campfire 作为实时交流工具。

最初的负载出现在 5:38，高峰很快就到来了：一分钟之内访问者数就从 0 一跃超过了 1000。我们的页面响应时间稳定在 500 毫秒，在提供顾客页面方面从未感觉到延迟。

我们遇到了一个定价上的 bug，必须在服务器上在线打补丁，然后在有负载的情况下重启服务器（当然，依次解决），与此同时仍然能让用户看到页面。我们的峰值负载低于测试中的水平，但相比周一下午做出改进之前要高很多。两天半的奋战最终取得了回报。

教训如下。

- 如果我们早点购买专业版 NewRelic 的话，我们也许就能少做很多无用功，但无论如何，我都认为 NewRelic 帮了很大的忙——如果我们没在周三下午得到所需的数据，我们不可能在周四早上取得成功。

- 负载测试应当定期进行，而不是只在预期出现大事件之前做。任何时候都可能有人修改数据库，忘掉索引，或是导致其他奇怪的性能问题，这种问题在遭遇负载之前往往难以发现。

- 如果没有使用云服务的话，我们根本不可能这么快地扩展应用实例和数据库服务器。

- 让团队所有聚在一起将会大有帮助。我们一般都在不同地方工作，在没有大问题的情况下这样做没什么不好，但我不想在千里之外做这种改进。

查看英文原文：Scaling A Web App 1,000x in 3 Days

九　卦

码农餐厅（一）

作者 / 谢工
图灵公司联合创始人，图灵社区发起人之一。忽悠大牛写书，同时支持并参与技术社区的发展。要是能再开个码农餐厅，这辈子就知足了。

搞 IT 的总爱在网上吵吵，争论各种语言问题，回家看到这几天热播的电视剧《经济适用男》里被鄙视的 IT 民工，感觉他们表现的程序员生活都不正宗，不客观，所以发了个帖子：

> @ 图灵谢工：我真想开个程序员餐厅了，我当老板娘，进门时先写代码再进，一楼餐厅分 C 包间、java 包间、Linux/Unix 包间 ...，搞开源软件的就坐大厅里，搞 ruby 的上二楼 ...

现在就来聊聊这餐厅的初期设想吧。

从语言来看，光大包间有：C 和 C++ 包间、JAVA 包间、C# 与 .NET 包间、Linux/Unix 包间、数据库包间（这里面有 MySQL、Oracle、SQL、hadoop、MongoDB、Cassandra 等）、Web 前端包间，怎么都还要再给敏捷、产品经理一个包间吧，对了还有搞数学、算法的留一间。

云计算和移动开发这么热，可能得给个小厅了。开源软件的一帮

家伙也要给个大场地才行。

还需要的小包间有 Python 包间、Perl 包间、LISP 包间、Ruby 包间、Scala 包间、Erlang 包间、Haskell 包间、Go 包间等等还有很多 ....

总体规划下来，要 10 个大包间，10 个小包间，二、三个小厅，一个大厅。楼上要有露台，供开技术交流会议使用。所以总计餐厅的面积至少 300 ～ 500 平米左右。

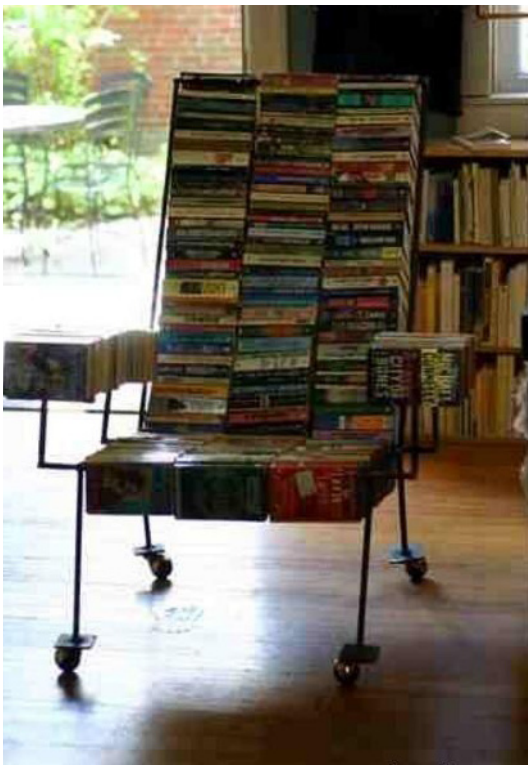
我刚找到了几张图片，看看能否满足码农餐厅的特殊需求。

进门要从这张图片下穿过：

前台设计成这样的，全是书做的，省了大钱了：

餐厅的服务员全部是美女，由各大互联网公司竞价冠名：腾讯 MM、淘宝 MM、百度 MM，收银员是支付宝 MM，形象都是这样的：

凳子，沙发什么的，都不用花钱，自己用书捆。



第一个菜名已经定了，叫：HELLO WORLD!

全国连锁店的店名有人建议：码当劳！

截止到次日上午 9 点为止，本微博帖已被转发和评论数百条了，我汇集了一下的精彩评论：

> @ 左耳朵耗子：没有问题，这个事我是一定要干的。菜价全用十六进制，这样看着便宜。
>
> @wehelpwe：如果老陈做菜单：翻页要 vi 模式，否则只能看第一页，凉菜几个凉菜边上有算法题，结出菜品免单。
>
> @clickstone：这个好！搞推荐的当跑堂的，绝对个性化服务！
>
> @gnawux：这个话题我们有次讨论过，进门要看名牌，PM 不要，还可以对猎头和 HR 开包间，设最低消费。
>
> @ 天池朋友：好主意，每个房间墙上一道编程题，做出来打八折。促销的时候放冒泡排序，黄金周放背包问题。
>
> @TV 来客开发者：嗯，基础服务的做一楼，ruby 这样搞后端的上二楼，js 的上三楼，html/css 这样的自带小马扎，上天台蹲着。
>
> @Soar- 翱翔：JavaScript 的应该可以到露天台上就餐
>
> @ 网路冷眼：Python 包间，整一个巨蟒盘旋在房间里，那才叫威武

@ 非常不小心：这个意思是美工和前端只能在厕所了。。。

@ 邓毅: 看这架势，硬件条件最豪华的得是 Objective-C 了。

@ 一只鱼向北：这个不错哦，要是只会 hello world 蹲门口么？

@deerchao：搞 Ruby 的不应该是地下挖个洞，送盏矿灯让进去找宝石么？

@ 夜雀烧：盛饭用硬盘，盛菜用光盘，饮料按流量收费。

@iamif：十六进制的菜价有创意 .…不过浮点型就不太好读了 .…哈哈。

@ 宛垚：做安全的请系统测试的吃饭，但得先破译包间智能门锁系统！

@ 左晃右过：先给我上块分布式牛排，来杯全双工咖啡，再来份栈溢出的小甜点

@ 蓝色狂想：C++ 包间的吃完饭必须要把餐具回收，可以不用安排服务员，而 Java 包间不会有这种意识，所以必须安排服务员清理

@ivancrh：在软件园开一家，菜单如下：ios 套餐、android 套餐、java 系套餐、微软系套餐、开源大餐…

@volunie：Java 的和 C 的，还有 Win 和 Linux 打起来怎么办?

@ 但丁不淡定：VB 的是不是恕不接待。。。

@fengxie83：搞 CUDA 的连个座位都没有。

@ 月儿小师妹：PHP 凭啥只能去地下室哇。。咱好歹是大网站主流。。。不服 ~~ 哈哈)

没有 .NET 的地方嘛。。有数十万企业网站是咱。。。效率杠杠滴，1 天就搭建好。)

@TV 来客开发者：嗯，菜单都弄成 O'Reilly 的封皮样式，菜名也有程序员特色：Python vs. Tomcat 龙虎斗 /

firebugs 烧烤知了。

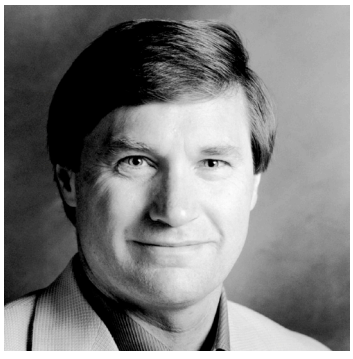
@ 尹琪 Leo：用十六进制的危险是万一某个菜价是“2B”就麻烦了。

@ 异步步： 加法溢出变负数的时候怎么算哇，嘿嘿。

餐厅怎么挣钱是个难事啊，愁死人了，反正不管，我就是要当老板娘的 .....

未完，待续 ..... ■

宇宙一种：
程序员创造的虚拟宇宙



作者 /John D. Barrow

约翰·D. 巴罗，英国剑桥大学应用数学与理论物理学系教授、皇家学会成员。研究领域涉及宇宙学、天体物理、引力理论、天体粒子物理等，发表有四百多篇学术论文。他和弗兰克·梯普勒合著的《人择宇宙原理》成为该主题的经典著作。Barrow 教授在唐宁街 10 号、温莎城

> 在其他条件相同的情况下，如果你是生活在一个虚拟世界，那么你会更不在意别人，更愿意活在当下，会让你的世界看起来更容易变得富有，并希望试图更多地参与重要事情，会让自己更富有娱乐精神，更值得称道，以及让名人乐于围着你团团转。
>
> ——罗宾·汉森（美国经济学家）[①]

一旦你认真对待所有（或几乎所有）可能的宇宙都可以（或确实）存在的想法，那么一种滑坡谬误（slippery slope）[②]就会出现在你的面前。我们已经看到，一个文明的科技水平只要比我们发达一点点，就有能力进行一场宇宙模拟实验，其中可以产生多种能够相互交流的智慧生物。[③]这个文明会拥有强大的计算机，其计算能力要比我们的计算机高一个巨大的量级。他们不但像我们这样，能够模拟天气变化或星系的形成，还能进一步看到恒星和行星系统的形成，于是通过对虚拟世界的研究，他们的行星地质学和地形学也得到了发展。接着，将生物化学的法则和天文学的模拟结

合起来以后，他们就能看到生命和意识的演化过程（根据他们的需要，这些过程都可以被任意地加速）。正如我们能观察果蝇的生命周期，他们也能跟踪生命的演化进程，观察文明如何发展，如何相互交流。他们甚至可以围观虚拟生物之间的争论：天上是否存在一个“大程序员”，他不但创造了宇宙，还可以随意挑战他们已经习惯遵从的自然法则。

一旦有谁掌握了模拟宇宙的能力，虚拟的宇宙就会如雨后春笋般冒出来，很快就会超过真实宇宙的数量。而且尼克·博斯特罗姆断言④，如果随机选出一个智慧生物作为考察对象，那它更可能生活在虚拟现实之中，而不是真正的现实之中。⑤

这个惊人的结论引发了更多讨论，如果我们极有可能生活在虚拟现实之中，自己只不过是其中的虚拟生物，那么我们该怎么办才好呢？正如在本节开头的引语中所说的，罗宾·汉森认为，你应该做的就是努力创造机会延长自己在这场模拟中的存活时间，或者寻求将来再被模拟一次，尽管这个忠告听起来像是很多人都信奉的个人奋斗策略，而不论他们认为自己生活在哪种类型的宇宙中。对此，保罗·戴维斯回应道，我们极有可能生活在虚拟现实之中，这以反证法证明了涵盖所有可能性的多重宇宙的存在——但这无疑会让我们从宇宙中获取可靠知识的愿望日渐渺茫。⑥

有些人认为宇宙是由一个“大设计师”专门为生命的存在而设计的。为了反对这种论调，一些宇宙学家明确地提到了我们所介绍过的多重宇宙学说以及暴胀宇宙理论。⑦无论生命所需的条件多么苛刻，如果在无限的空间中存在一切可能性，多重宇宙中就必然会存在一个宜居的宇宙。但在讨论中常常会被忽略的一个关键问题是，在所有的可能性中，宜居地带所占的可能性究竟有多大？没人知

堡，以及梵蒂冈都进行过公众演讲。他被授予了 2002 年的 Templeton 奖。

① R. Hanson, ‘How to Live in a Simulation’, *Journal of Evolution and Technology* 7 (2001), http://www.transhumanist.com.

② 武断地将某个可能性引申成为必然性，然后串联这些不合理的因果关系，推断出一件毫无关联的结果，这就是滑坡谬误。——译者注

③ J. D. Barrow, *Pi in the Sky: Counting, Thinking and Being*, Oxford University Press, Oxford (1992), chapter 6.

④ N. Bostrom, ‘Are You Living in a Computer Simulation?’, http://www.simulation-argument.com

道答案。因此，那些人认为，多重宇宙只不过是一种逃避微调问题的简便方法。

对于多重宇宙理论的这两种观点都希望能够避免走向形而上学。但可惜多重宇宙的事情没有这么简单。我们已经看到，一旦有意识的观测者能够随意干涉宇宙的进程，而不再是被动的“观测者”，那么结果就会是再次出现数不尽的神，他们以模拟者的面貌出现，掌管着手中虚拟现实的生杀大权。模拟者能确定虚拟世界所受的各种规律，同时也能对之作出修改。他们能设计出一种人择原理的微调方案。他们可以随时拔掉插头，也可以参与到模拟之中或者置身事外。他们可以冷眼旁观，看着虚拟生物在讨论是否存在一个掌控全局的神。他们也可以制造奇迹，或者将他们的道德准则引入到虚拟现实之中。他们永远不用担心自己会造成什么伤害，因为他们的玩具现实不是真的，不是吗？他们甚至可以观察到虚拟现实里的文明发展出了极高的科技水平，于是在虚拟现实中再模拟出一层虚拟现实出来。

面对这种错综复杂的情况，我们有可能将虚拟的现实与真正的现实区分开来吗？如果我们只不过是在虚拟现实当中进行科学观察，我们又能预期观察到什么呢？

首先，模拟者会避免在虚拟世界中运用整套自洽的自然法则，因为这会比较复杂，而且他们大可只需拼凑出接近“真实”的效果。当迪斯尼的动画片要表现湖面的反光时，他们不会利用量子电动力学和量子光学的定律去计算光的散射，因为这得动用超强的计算能力。相反，用一些合理的经验法则就能模拟光的散射过程。这比真实的散射过程简单许多，但却能营造出真实的感觉——只要人们不要离得太近就行了。如果模拟者只不过想玩玩而已，选

⑤ 但是这又回到了我们在本章前面强调过的尴尬的概率测度问题。博斯特罗姆的论述包含了一个隐含假设，即无论什么类型的世界，真实的也好，虚拟的也好，出现的概率都大致相等，至少差别不会太大。但实际上不是这么回事。

⑥ P. C. W. Davies, ‘A Brief History of the Multiverse’, *The New York Times*, 12 April 2003.

⑦ L. Susskind, *The Cosmic Landscape: String Theory and the Illusion of Cosmic Design*, Little Brown, New York (2005); A. Vilenkin, *Many Worlds in One: The Search for Other Universes*, Hill and Wang, New York (2006).

择这样经济实用的方案来模拟现实就很有必要。但如果模拟程序之中真的包含这样的不足之处，就可能偶尔会出现穿帮镜头，也就是业余剧团表演话剧时常见的“吱吱作响的布景问题”（creaking scenery problem），而且也许这种穿帮镜头在虚拟世界内部也能看到。

即使模拟者一丝不苟地模拟了自然法则，他们也无法超出自己的能力范围。虽然他们对自然法则的认识十分深刻，这种认识也不可能是**完备**的（一些科学哲学的研究者会论证说，事实绝对会是如此）。他们也许知道很多模拟宇宙所需的编程知识，但对于自然法则的了解总会存在一些空白，或更糟地，存在错误之处。这些漏洞应该是难以捉摸、不易发现的，不然我们假想的“发达”文明就配不上“发达”二字了。

这些缺点不会妨碍模拟实验开机并长时间地流畅运行，但慢慢地会有一些缺陷逐渐显现出来。最终，这些缺陷产生的效应如同一场雪崩，导致虚拟现实“死机了”。唯一的解决方案就是让程序员每发现一个缺陷，就立刻打上一个补丁。家里有电脑的人早已对这种解决方案习以为常了。我们会定期收到系统的更新提示，以便修复程序设计者原先没有预料到的漏洞，或者防止新型电脑病毒入侵。模拟开始以后，模拟者也会采取这样的临时防护措施，更新自然法则的版本，并向其中添加一些他们新学会的东西。

在这种情况下，逻辑矛盾在所难免，模拟的法则时不时地就会被破坏一次。虚拟世界中的居民（尤其是虚拟科学家们）偶尔会被他们得到的实验结果搞糊涂。而虚拟天文学家们则可能，打个比方说，从观测中发现他们所谓的自然常数正在缓慢地变化着。[8]

统治虚拟世界的法则甚至也可能突然失灵，这很可能是因为模拟

⑧ J. K. Webb, M. Murphy, V. Flambaum, V. Dzuba, J. D. Barrow, C. Churchill, J. Prochaska and A. Wolfe, ‘Further Evidence for Cosmological Evolution of the Fine Structure Constant’,*Phys. Rev. Lett*. 87, 091301 (2001).

⑨ S. Wolfram,*A New Kind of Science*, Wolfram Inc., Champaign, Ill. (2002).

⑩ K. Popper,*Brit. J. Phil. Sci*.1, 117 and 173 (1950).

⑪ D. MacKay, *The Clockwork Image*, IVP, London (1974), p.110

⑫J.D. Barrow,*Impossibility*, Oxford University Press, Oxford (1998), chapter 8.

者在应用一种在模拟其他复杂系统时已经证明行之有效的手段，即使用纠错代码，让事情都回到正轨。

以我们的遗传代码为例，如果任由这些代码运行，人类就延续不了多长时间。误差日积月累，必然会产生致命的危害。幸好存在一种纠错机制，能够识别和修正遗传代码中的错误，使我们免受退化的威胁。许多复杂的计算机系统中都有这种内置的“拼写检查”功能，以防止错误的积累。这种纠错码是由贝尔实验室的理查德·海明（Richard Hamming）在 1950 年最早提出的。

如果模拟者利用了纠错码防止整个模拟（以及更小尺寸上的我们的遗传代码）出错，那么虚拟世界的运行状态和法则就会时不时地被修正一下。这样，神秘的改变就会突然发生，违背了虚拟科学家通常观测和预测的自然法则。

我们还可以预期，虚拟现实的各处都具有最高程度的计算复杂度。虚拟生物的计算复杂度应该与最复杂的无生命系统（如天气系统）的相当，借用史蒂芬·沃尔夫拉姆的说法（虽然他是基于完全不同的原因而提出的）就是“计算等价原理”。[9]

关于从内部区分虚拟和现实的问题，一个最普遍的担心就是，模拟者能够隐隐预感到不妙，因而会预先调整模拟的过程，防止故障的发生。这套新的虚拟现实可能又会与真正现实产生新的不一致，但他们又可以通过再次预判而防止故障的发生。

问题在于，把其推至极致时，这种先见之明是否可能。这个问题很像卡尔·波普尔所考虑的计算机的自指能力极限（self-referential limits）[10]。最近，唐纳德·麦凯又在另一种背景下利用了同样的论证。麦凯借此否认存在这样的逻辑可能性，即如果事先将一个人的命

⑬ 不过，赫伯特·西蒙有一篇著名的高引用率的文章提出了相反的观点，但其论证是错误的，参见：Herbert Simon, ‘Bandwagon and Underdog Effects in Election Predictions’, *Public Opinion Quarterly* 18, Fall, 245 (1954); reprinted in S. Brams, *Paradoxes in Politics*, Free Press, New York (1976), pp. 70–77. 文中的错误在于，作者不恰当地在离散变量，而不是连续变量中应用了布劳威尔不动点定理，更详细的说明参见：K. Aubert, ‘Spurious Mathematical Modelling’, *The Mathematical Intelligencer* 6, 59 (1984).

⑭ J. D. Barrow, *The Constants of Nature: From Alpha to Omega*, Jonathan Cape, London (2002).

⑮ 1965 年，英特尔公司的创始人之一戈登·摩尔发

运告诉他，他未来的行为仍然是可预测的。⑪

只有不把预测结果告诉你时，关于你未来一切活动的预测才可能是正确的。一旦我的预测结果被你得知，你就有可能改变未来的计划。⑫因此，关于你未来的活动，不可能作出没有条件限制的预测。世俗事物中也会出现这样的例子，例如预测选举的结果，对选举结果的公开预测不可能不考虑预测本身对选民的影响。⑬这种类型的不确定性在原则上是不可约的（irreducible）。但如果不公开预测结果的话，其正确率可以达到 100%。

这一切都说明，如果我们生活在虚拟现实中，我们就会偶尔遇到自然法则的突然失灵，会发现自然法则和自然常数随着时间的推移而发生了微小的变动。⑭这也让我们第一次意识到，对我们理解真正的现实来说，自然的缺陷就同自然法则一样重要。

这些略有跑题的讨论已经进入了通常由科幻作家占据的地盘，但我只是想指出，如果我们认真对待存在无数个可能的世界（它们穷尽了所有的可能性）的想法，那么也许会得出许多异乎寻常的推论。我们可以设想，在将来，现有某些科学和技术的进步可以让我们的后人对这些推论进行检验。我们甚至不用发明新的科学，技术进步本身就可以把这搞定。⑮我们所见宇宙的本质及其可能的虚幻性让我们触目惊心，甚至令人忧心忡忡，这不禁让我们想起了英国哲学家大卫·休谟（1711 ～ 1776）在 18 世纪末时所说的话。

休谟在《自然宗教对话录》中驳斥了许多在当时非常流行的关于上帝存在性的论证，将矛头直指这些论证当中对于神创论完美本质以及神的唯一性等假设。⑯这里摘录了休谟关于“多重世界”及其可能存在的缺陷所作出的讨论⑰：

现，每过两年，每平方英寸集成电路上的晶体管都会进一步微型化，数量翻倍而成本减半。这个论断在一定精度下可以预测技术的发展趋势。其他文明的技术微型化趋势也可能存在某种类似的规律。摩尔定律对计算机产业的发展非常重要，因为软件公司和硬件公司可以根据这个规律调整自身的发展战略，以跟上技术发展的脚步。

⑯ 关于这些宗教论证以及休谟的回应（及其对康德的影响）的深入讨论参见：Barrow and Tipler,*The Anthropic Cosmological Principle*, chapter 2.

⑰ 摘自陈修斋、曹棉之的译文（[英] 休谟，《自然宗教对话录》，北京：商务印书馆，1962 年，页 39，页 41）。——译者注

⑱ David Hume,*Dialogues Concerning Natural Religion* (1779), in Thomas

> 你必须承认，从我们狭小的观点，我们是不可能说出，这个宇宙体系比起其他可能的、甚至真实的宇宙体系来，是否包含有任何巨大的错失，或值得承当任何巨大的赞美……假如我们考查一只船，对于那个制造如此复杂、有用、而美观的船的木匠的智巧，必然会有何等赞叹的意思？而当我们发现他原来只是一个愚笨的工匠，只是模仿其他工匠，照抄一种技术，而这种技术在长时期之内，经过许多的试验、错误、纠正、研究和争辩，逐渐才被改进的，我们必然又会何等惊异？在这个世界构成之前，可能有许多的世界在永恒之中经过了缀补和修改；耗费了许多劳力；做过了许多没有结果的试验；而在无限的年代里，世界构成的技术缓慢而不断地在进步……

就他所知道的，这个世界，比起一个高的标准来，是非常错失，非常不完全的；这个世界只是某个幼稚的神的初次的尝试作品，后来他抛弃了它，并对于他的恶劣的工作感到羞愧；它只是某个不独立的，低级的神的作品；对于较高级的神，它是嘲笑的对象；它是某个老迈的神的衰朽期的作品；自从他死了之后，它就依靠着它从神取得的起始的冲动和动力，往前乱撞乱碰。⑱

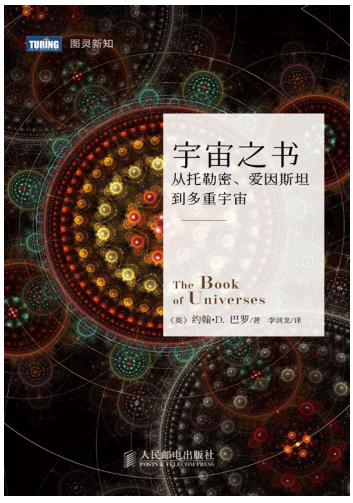
休谟设想了一幅多神论的情境，这些神的资质良莠不齐，创造出来的宇宙质量高下不等，就像徒弟试图模仿师傅的作品。但如果我们将“幼稚的神”和“老迈的神”代之以各种模拟者的话，那么他所预想的就是一个遍布虚拟宇宙的世界，其中有好的、前景光明的，也有次的、难逃厄运的。

如果所有可能的世界都存在，而我们活在一个自然法则与众不同的虚拟世界中，这和只有一个真实的世界有什么区别吗？实际上，

Hill Green and Thomas Hodge Grose (eds.),*David Hume: The Philosophical Works*, vol. 2, London, 1886, pp. 412–416.

⑲ 这些问题与瑞·库兹韦尔（Ray Kurzweil）在《灵魂机器的时代》一书 [*The Age of Spiritual Machines*, Viking, New York (1999)] 中讨论的主题密切相关。库兹韦尔思考了虚拟现实和人工智能的产物拥有心灵和审美能力的可能性。

⑳ Hanson, ‘How to Live in a Simulation’.



浩渺的星空，自古以来就激发着人类无尽的好奇和想象。为了解释我们所见的宇宙，从亚里士多德、托勒密、哥白尼、牛顿、康德到爱因斯坦，再到现代学者对爱因斯坦宇宙学方程组的艰难求解，人们提出了各式各样的理论，描述了种种面貌不一的宇宙。身为知名的宇宙学家和科普作家，John D. Barrow 将在《宇宙之书：从托勒密、爱因斯坦到多重宇宙》中带领我们回顾历史，追踪现代天文学的前沿进展，纵览“奇异得超乎我们想象”的万千宇宙。本文摘自《宇宙之书》。

这应该有什么区别吗？[19]如果你是一个想要研究世界运转规律的（虚拟）科学家，那么这会让你感觉相当失落，毕竟任何事情都可能无缘无故地发生。所以毫不意外，在科学世界观里，虚拟现实并不受欢迎，因为它们的存在会动摇科学世界观的根基。哲学家们倒是更能严肃地考虑这样的问题，有些甚至会以此为背景讨论伦理学问题。他们提出的一些问题发人深省。

罗宾·汉森指出，如果存在虚拟现实，那它就会以一种独特的方式影响你的行为。[20]虚拟的经历，无论看起来多么真实，都比真实的经历更可能遭遇无法预料的戛然而止。这让汉森推断出，“在其他条件相同的情况下，你会更不在意别人，更愿意活在当下”。我们很熟悉电影和舞台剧中这样的场景，明星的周围簇拥着大批优秀演员，他们和明星之间有互动；再往远处，就是临时演员和打零工的低薪演员，他们以低成本充当着背景。同理，在虚拟现实之中，离你很远的人物可能不过是一些虚假角色，你不必太在意他们。汉森指出，如果你处于他人的模拟之中，那么最重要的是，你要让自己更富有娱乐精神！要出名！要变成重要角色！这会增加你在虚拟现实中的存活几率以及将来别人再次把你模拟出来的机会。而如果你没能拥有这些特质，就会像肥皂剧中的角色一样，被编剧发往到符拉迪沃斯托克度长假，然后再也不会回来了。

如果我们看看新闻里那些人的行为举止，再想想汉森的话，也许就很容易得出，我们一定生活在模拟之中。然而，以上这些都不太有说服力。你应该如何举动完全依赖于模拟者的道德立场。如果他们喜欢娱乐，你就会变得富有娱乐精神。而如果他们追求一种崇高的目的，那么通过为正义事业牺牲殉道，你就非常有可能被再次模拟出来。虽然我们并不认为这些结论应该成为你生活中的行为规范，但它们确实凸显了道德哲学及其实践的核心问题。

如果虚拟现实很常见，而我们又处于其中之一，而且它又是我们熟知的那种模拟的话，麻烦就大了。但是凭什么虚拟现实就得是我们熟知的那种模拟呢？如果我们总是用“模拟”这个词来描述上帝创造一次性宇宙的行为结果，那么我们就生活在一场非常类似的模拟之中，尽管模拟者属于另一种类型。

也有些人认为，从我们可能生活在虚拟现实中的想法得出的这些异乎寻常的推论，反而成为了否定其他世界存在的强有力论据。如果绝大多数世界是虚拟的，那么这些世界之中就会存在虚假的物理定律，而我们就会犯下另一种滑坡谬误：因为不存在任何可靠的知识，所以我们什么都不知道。这是与唯我论[21]恰好相对的一种论调，也会导致类似的、使思想瘫痪的后果。如果可能性有无限多种且都是存在的，那么现实无疑远远超出了我们的承受能力。

译者 / 李剑龙

@ 松鼠会 Sheldon，浙江大学理论物理学博士，“科学松鼠会”成员，与一群年轻人一起利用业余时间致力于“让科学流行起来”的愿景。

㉑ 唯我论是指，除了我和我的思想之外，其他一切事物都不可信。——译者注

DRM-free，不止拥抱这一天

作者 /Mike Hendrickson

Mike Hendrickson， 是 O'Reilly 负责内容战略的副总裁。同时在出版行业他也有很多头衔：产品发展总监、编辑、主编，以及联合出版人。他最喜欢的事就是和前沿技术极客们聊天。

在您更加深入地阅读本文之前，请认识到支持 DRM-free 内容并不意味着 O'Reilly 支持剽窃、盗版，或者其他任何形式的盗窃。您应该知道我们严肃地对待有知识产权保护的材料，但是我们也明白在某些情况下，您可能不能或者不愿意停止盗版。本文的主旨是探讨后一种情况。

似乎许多初次写作者、写作过很多书的作者、编辑、出版者，以及出版技术人员，都受到 DRM 和数字内容在网络上随处可以下载

的情况，这种盗版让人困扰到夜不能寐。我想告诉他们，冷静下来，放心去睡吧！这不是一种昙花一现的情况，有几种非常简单的办法，只要做到就可以成功。这个问题已经出现很长时间了，并将继续持续很长时间。

几年以前，David Pogue 曾经友好地参加一项与 O'Reilly 合作的在自然条件下进行的关于 DRM-free 的实验。你可从这里读到结论，但基本观点如下——

> 结果？结果是真实的。盗版满天飞。现在网络上到处都是盗版，随处可以免费下载，这简直是太荒谬了。荒唐的事情是：书籍的销售额没有下降，甚至在当年还有轻微的上升。这不是一个完美的、所有变量相等的实验，当然，任何参数的数值可能解释这个结果。但是，这肯定不是我曾经担心过的灾难。

我认为这是一个非常重要的启示。那一年中 Pogue 的作品被故意泄露到 DRM-free，这反而增加了销售。我设想如果在实验过程中，能在海盗湾打一块图标广告，表示任何读者都可以通过海盗湾优惠券以我们的纸质书零售价来同时获得该图书的纸质本及其数字版本，效果会更好。我的观点是，面对盗版我们需要更加有创造力，并和盗版共存，而不是考虑 DRM、聘请律师或者让搜索引擎来屏蔽问题。

最近，Eric Freeman 和 Beth Robson，也就是 *Head First Design Patterns*, *Head First HTML* 和 *Head First HTML5 Programming* 这三本书的作者，和 O'Reilly 共同重新发起了一个旧话题，这是一个大多数作者都感同身受的话题，你可以从这里读到 4 年前我们关于它的讨论。

这次 Beth 展开了他的讨论：

> 我已经在非法文件共享网站上看到了 *Head First HTML5 Programming*，今天早上，就在 http://bit.ly/lM7I84。O'Reilly 知道吗？你们知道他们做了什么，我意识到除了好奇之外我们无法阻止盗版。

Eric 也有同感：

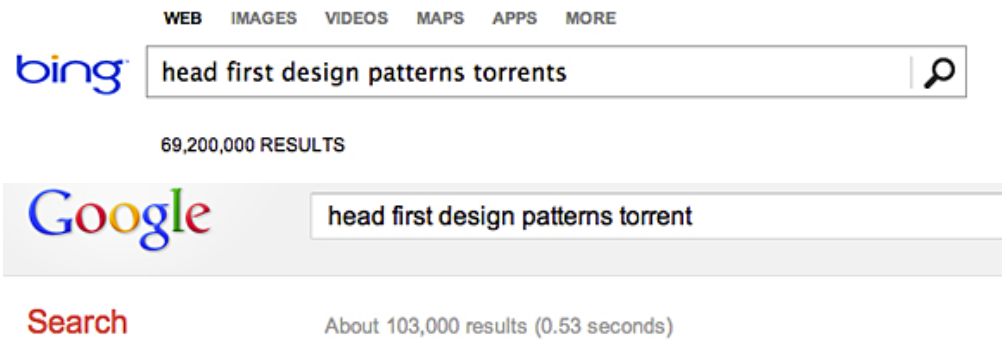
> 一天之后，我就在网上发现盗版已经出来了，并且出现在了 Google 搜索引擎的首位。可能你无法阻止海盗湾，但是你应该可以让 Google 删除那些链接。

基于我在 O'Reilly 和其他机构的出版发行经历，我回复道：

> 我不同意你的观点，我认为这有助于推广你的书，*Head First Design Patterns* 这本书已经在大多数 P2P 网站上出现了，但它一直销售得很好。难道你没有听说过 Tim 的咆哮：盗版不是作者的敌人，没有知名度才是作者的敌人。那些习惯盗版的人，不论是出于什么原因，都会继续盗版，并且将想方设法得偿所愿。

让我们深入挖掘一下我的回复，来看一看为什么我会做出如上声明。首先，让我们看一组数据，当我搜索书籍 torrent 文件的时候，Bing 从 utorrent 客户端返回了 77 800 000 个结果，而 Google 返回了 12 000 000 个结果。当我搜索技术书籍 torrent 文件的时候，从 Bing 返回了 20 300 000 个结果，而从 Google 返回了 5 180 000 个结果。更精确地定义了我的搜索之后，当我搜寻 O'Reilly 的时候从 Bing 得到了 937 000 个结果，而从 Google 得到了 23 200

000 个结果，这颠覆了之前从 Bing 和 Google 获得的结果数值。再深入一点，搜索 *Head First Design Patterns*，返回如下结果：



请注意，通过 Bing 搜索引擎搜索 utorrent，可以获得超过 69 000 000 个结果。一个有趣的补充说明是：当我开始点击进入搜索页面，比如点击进入搜索结果的第 10 页，马上就会发现结果变为 191 000 个结果中的第 211-220，这可是和 69 000 000 相差甚远。但是链接还是有效的，直到第 54 页，在这一页我收到了这样的消息：32 600 个结果中的第 531-540 没有发现。因此长活短说，网上有大量的种子，但并不像初次搜索结果显示的那么多，虽然在之后的一段时间内我可以找到并下载到文件。

摆脱掉野草的束缚，回到原来的观点，对于 *Head First Design Patterns*，网上有大量可获得的种子。但是，是否那些种子文件的出现减缓了我们图书的纸质本和数字版的销售？自从 2004 年开始，根据 Nielsen Bookscan 技术类书籍报告，O'Reilly 的 Head First 系列图书中，有三本书进入在累积排行的前 15 名。如果我们只统计标价在 100 美元以下的图书的话，Head First 有三本书进入累积销售的前十名。我上一次查看的时候，在 Safaribooksonline.com，我们的 Head First 系列图书占据了全部

前 10 名。*Design Patterns*，*Java* 和 *HTML* 的领先地位奠定了 O'Reilly 的销售收入，不仅仅在实体书店，也体现在 Safari 获得的收入和网络上的盗版书数量上。这仅仅是一个巧合吗，还是这就是做生意的代价?

我相信那些买不起 50 美元一本书的人不大可能放弃其他必需品来购书。如果你对那些从网上获取种子文件和未授权内容的人们做个人口统计，我相信他们中的大多数人的经济条件不足以支付产品的价格。还可以这么想，你读大学的时候，买书是否也像啤酒、美食、约会、衣服一样吸引你，你是否是用额外的钱才去购买图书? 因此存在着一个有趣的轮回：那些大学生工作挣钱之后，会记得那些传授他们知识的图书吗？当然会。他们未来有固定收入之后，会更愿意购买那家出版社的图书吗？当然会。对出版社而言，那些大学生是成长期的市场投资吗？当然是。

这里有一个困难：一些出版社如果发现他们的图书没有遍布 P2P 网络，无法找到种子文件或者 DRM-free 版本的话，他们为此感到高兴。但，真的，请考虑一下: 你的内容不足以吸引人们去尝试得到，那你是否还有必要出版它们呢？没有知名度是比盗版更大的敌人。第二个困难是：如果你的内容是免费的，并出现在 P2P 网络上，有种子文件，等等，人们不下载，这有任何好处吗？严肃地说，大多数网站在网页上显示出下载次数，这样其他人可以看到是否有许多人喜欢你的内容。我认为如果免费也没有人下载我的作品的话，将是令我尴尬的。作为一个出版者，那时候我们需要马上重新评估我们的出版计划。再重申一次，我把这视为做生意的代价，类似于在缴纳市场税。

给内容增加 DRM 来杜绝盗版，你开玩笑吧？严肃地说，我考虑

译者 / 沧海

过这个问题。**一个好的程序员用一个小时就能够破解目前已知的绝大部分 DRM 文件**，或者世界各地的手工店铺都可以把书撕开，逐页扫描纸质图书并加以传播。DRM 有点像穴居人宁愿用前肢握拳支撑身体前行，却不愿用他发达的大脑和肌肉来直立行走和用手传递工具，这样做是无法完成物种进化的。作为一种行业，我们需要进化那古老的、迟滞我们行业发展和革新的 DRM。新的 DRM 技术没有创新，它们是类似穴居人的反应。我们需要发布技术的创新，我们需要学习科学的创新，我们需要产品和制造的创新，我们需要把所有专注力集中于内容创新。

目前，出版业正面对着黑洞洞的枪管，如果我们不能想出新奇的方法前进，这支枪很快就会主宰出版业的经营方式、定价、内容创造和发布的行为。我们能否使用 P2P 网络和种子文件来帮助我们，为我们的内容和服务来促销和打广告？我们能否认为个人到个人的发布方式和付费网络可以服务于我们？我们认为，为内容嵌入链接的方式驱动人们返回我们的网站，在这里我们可以基于他们的经济条件给他们提供更多的适合他们的产品和服务。你可以从这些人那里学到更多，听到他们为什么下载未授权文件的原因。可能这会创造一个后续产品或衍生产品的机会。

我希望你们看到了令人足够信服的原因来实施 DRM-free。因为对我们出版业来说，DRM-free 不是权宜之计，而是应该永远拥抱的。

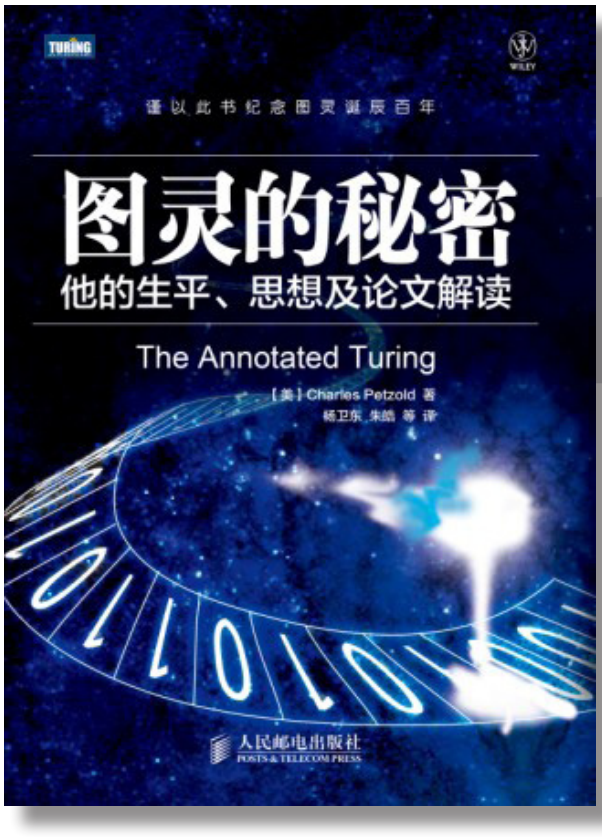
睡个好觉，我的朋友们。

Photo: Eliminate DRM by YayAdrian, on Flickr

查看英文原文：DRM-Free Day, forever

书　榜

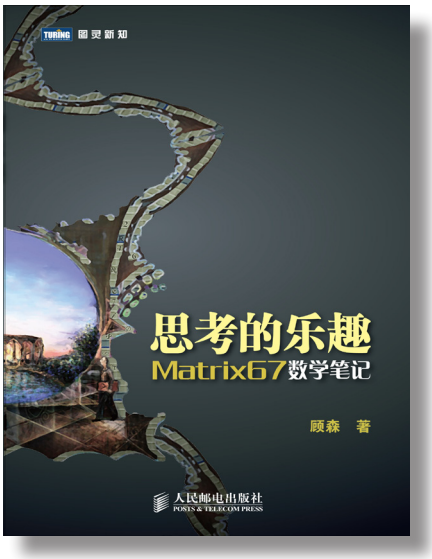
2012 年
十大不能错过的好书



图灵的秘密

作者：Charles Petzold
译者：杨卫东，朱皓
书号：978-7-115-28214-9
图灵社区推荐：

由 Windows 编程大师 Charles Petzold 耗时多年编写的这本书剖析了现代计算机原理开山之作、阿兰·图灵流芳百世的论文。图灵在其中描述了一种假想的计算机器，探索了其功能和内在的局限性，由此建立了现代程序设计和可计算性的基础。这本书也像是一本小说，行文间穿插讲述了图灵的成长经历和教育背景，以及他跌宕起伏的一生，包括破解德国恩尼格密码的传奇经历，他对人工智能的探索，他的性取向，以及最终因同性恋的罪名而在 41 岁时自杀的悲惨结局。全书完整揭示了阿兰·图灵非凡、传奇而悲剧的一生，是了解图灵的思想和生平的极好著作。



思考的乐趣：Matrix67 数学笔记

作者：顾森

书号：978-7-115-27586-8

图灵社区推荐：21

“事实上顾森的每篇文章都在向读者展示数学确实好玩。数学好玩这个命题不仅对懂得数学奥妙的数学大师成立，对于广大数学爱好者同样成立。”

——汤涛，《数学文化》期刊联合主编，香港浸会大学数学讲座教授

大数据：互联网大规模数据挖掘与分布式处理

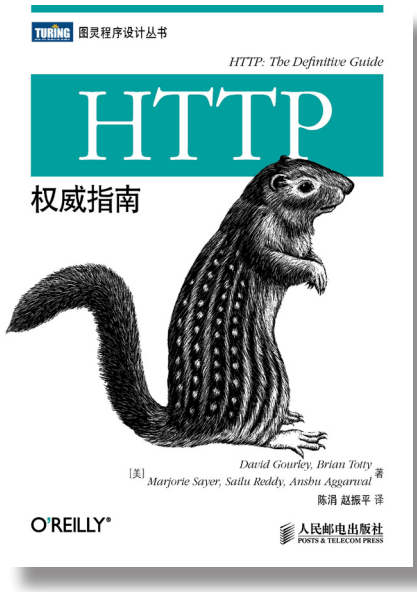
作者：Anand Rajaraman, Jeffrey D. Ullman

译者：王斌

书号：978-7-115-29131-8

图灵社区推荐：15

本书由斯坦福大学的“Web 挖掘”课程的内容总结而成，主要关注极大规模数据的挖掘。主要内容包括分布式文件系统、相似性搜索、搜索引擎技术、频繁项集挖掘、聚类算法、广告管理及推荐系统。其中相关章节有对应的习题，以巩固所讲解的内容。读者更可以从网上获取相关拓展材料。



HTTP 权威指南

作者：David Gourley, Brian Totty

译者：陈涓，赵振平

书号：978-7-115-28148-7

图灵社区推荐：35

要想高效地进行 Web 开发，所有 Web 程序员、管理员和应用程序开发者都应该熟悉 HTTP。本书由具有多年实践经验的专家编写，通过简洁、精确的语言和大量翔实的细节图解帮助读者形象地理解 Web 幕后所发生的事情，详细说明了 Web 上每条请求的实际运行情况。

30 天自制操作系统

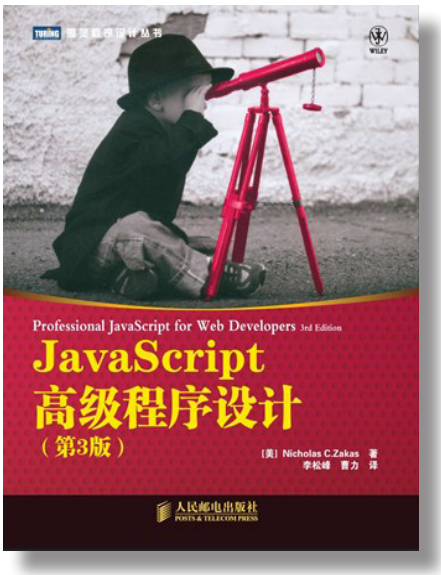
作者：川合秀实

译者：周自恒，李黎明，曾祥江，张文旭

书号：978-7-115-28796-0

图灵社区推荐：23

自己编写一个操作系统，是许多程序员的梦想。也许有人曾经挑战过，但因为太难而放弃了。其实你错了，你的失败并不是因为编写操作系统太难，而是因为没有人告诉你那其实是一件很简单的事。那么，你想不想再挑战一次呢?



JavaScript 高级程序设计（第 3 版）

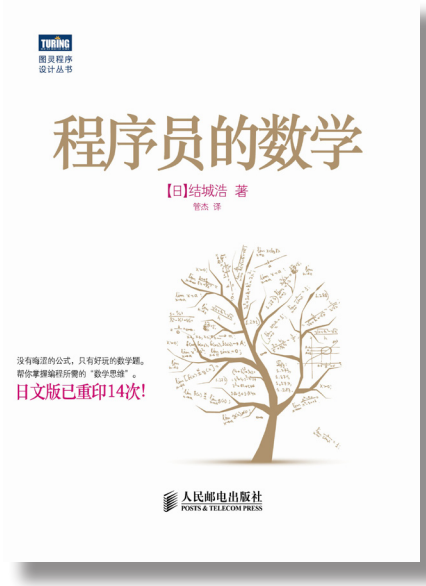
作者：NIcholas C. Zakas

译者：李松峰，曹力

书号：978-7-115-27579-0

图灵社区推荐：19

本书是 JavaScript 超级畅销书的最新版。ECMAScript 5 和 HTML5 在标准之争中双双胜出，使大量专有实现和客户端扩展正式进入规范，同时也为 JavaScript 增添了很多适应未来发展的新特性。



程序员的数学

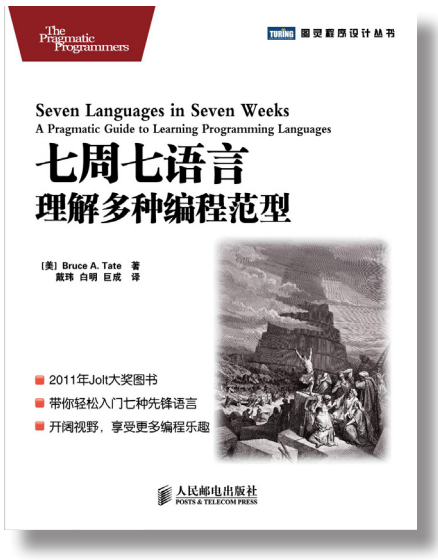
作者：结城浩

译者：管杰

书号：978-7-115-29368-8

图灵社区推荐：20

如果数学不好，是否可以成为一名程序员呢？答案是肯定的。本书最适合：数学糟糕但又想学习编程的你。书中讲解了二进制计数法、逻辑、余数、排列组合、递归、指数爆炸、不可解问题等许多与编程密切相关的数学方法，分析了哥尼斯堡七桥问题、高斯求和方法、汉诺塔、斐波那契数列等经典问题和算法。



七周七语言：理解多种编程范型

作者：Bruce A.Tate

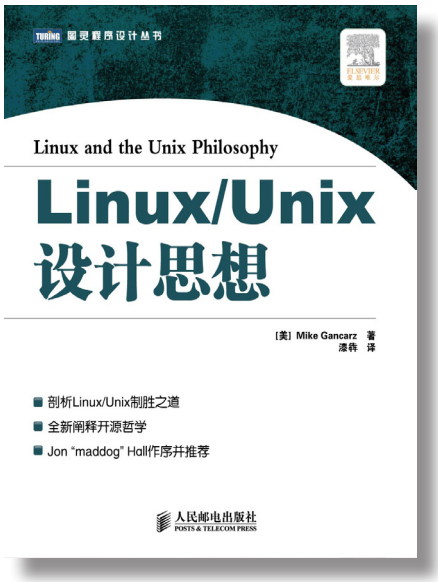
译者：巨成，戴玮，白明

书号：978-7-115-27611-7

图灵社区推荐：24

“为了了解更多的范例，提高编程水平，我读了很多相关的技术书，其中不乏让人拍案叫绝之作，本书即是一例。Bruce 有丰富的学习经历和使用多种语言的经验，他把一些杰出的范例很好地组合到了一起，让读者从他的这些经验中有所斩获。我极力推荐！”

——Vankat Subramaniam 博士，Agile Developer 公司创始人，敏捷开发权威



Linux/Unix 设计思想

作者：Mike Gancarz

译者：漆犇

书号：978-7-115-26692-7

图灵社区推荐：29

"Linux 和 GNU 项目的理念表面上是 Unix 哲学的下一个发展阶段，实际上它只是生生不息的 Unix 的强势回归。*The Unix Philosophy* 第一版中阐述的准则至今仍确切无误，甚至得到更多的佐证。开源除了可以让你清楚地了解到这些编程大师们创建系统的方式，还可以激励你去创建更快、更强大的系统。"

——Jon "maddog" Hall Linux 国际协会，执行理事

推荐系统实践

作者：项亮

书号：978-7-115-28158-6

图灵社区推荐：20

“从大家经常使用的相关搜索、话题推荐、电子商务的各种产品推荐，到社交网络上的交友推荐等，推荐系统在今天互联网的产品和应用中被广泛采用。但是，至今还没有一本书系统地从理论上对此进行分析和论述。《推荐系统实践》恰恰弥补了这个空白。”

——吴军，腾讯副总裁，《数学之美》和《浪潮之巅》作者

妙　评

《图灵的秘密》

在数字计算机出现之前，阿兰·图灵就预想了它们的功能和通用性……也证明了哪些事是计算机永远做不了的。由 Windows 编程大师 Charles Petzold 耗时多年编写的这本书剖析了现代计算机原理开山之作、阿兰·图灵流芳百世的论文。图灵在其中描述了一种假想的计算

真希望我能写出这样的书

作者：J. Tauber

一些书供消遣，一些书能获得信息；一些书让你明确了自己已经知道的事情，一些书改变了你对某事的看法。还有一些书只是让你想到：“哇！真希望我能写出这样的书。”

对我而言，Charles Petzold 的这本书就属于后者（当然也能从中获得信息）。不只是它的主题内容，而且就其呈现的方式来说，这本书都很值得一读。

在阐述图灵在 1936 年发表的论文 "On computable numbers, with an application to the Entscheidungsproblem" 之前，Petzold 给出了必要的背景知识，每个阶段都有丰富的注解，包括图灵的生平。

如果你对基础数学、可计算性、图灵的论文，甚至历史背景下解释数学的方式感兴趣，我强烈推荐你读读这本书。

机器，探索了其功能和内在的局限性，由此建立了现代程序设计和可计算性的基础。

译者 / 傅志红
图灵教育副总经理，《图灵的秘密》责任编辑。

应该放在每个有抱负的数学家的书架上

作者：Peter De Croos

这本书太奇妙了！它简洁、清晰地解释了图灵里程碑式的论文，并提供了全面的基础数学知识。我在大学里学习了一些集合论的知识，不过我相当确信，即使是一个有点几何证明能力、专注于此的高中生也能理解并领会这本书。当然，我要说的是，这本书写得极好，没有一个地方会让我感觉无聊。你上一次发现一本特别吸引人的数学书是什么时候了？

丰富，极易理解

作者：Trevor Burnham

别被书名骗了。这不是简单地添加了一些脚注就来讲述阿兰·图灵开创性论文 "On computable numbers, with an application to the Entscheidungsproblem" 的书。虽然整篇论文都囊括在了书中，但平均来说，是对图灵论文的每一行都做了解释。在进行解释之前，作者还引入了大量的重要概念，从有理数、无理数、代数数、超越数、可计算数之间的区别开始，任何聪明的大学生都可以理解这些内容。所谓没有数学背景是指超出了代数的知识。

这本书甚至超越了最好的本科教材，清晰、简洁地解释了这些概念，而且是设置在了图灵所生活的历史背景下。在讲述一个有关希尔伯特或罗素的有趣故事时，讲述了这篇论文。（罗素的生活相当迷人，这是漫画书《罗素的故事》的主题。）那些拥有大量数学背景的人是会粗粗略过前面的章节的，但其实不应该完全跳过。

侯世达（Douglas Hofstadter）的《哥德尔、艾舍尔、巴赫： 集异璧之大成》一书讲述的是哥德尔的不完备定理，而 Petzold 的这本书讲述的是图灵的通用机。如果你对计算理论有兴趣，这应该是你读的第一本书。

理解图灵论文，这 10 年之久的愿望终于如愿了

作者：Sytelus

我第一次在网上发现图灵的论文是在 10 年前了，我当时认为这篇论文并不难以理解（毕竟是“纯”计算机科学）。从那时起，我就有相当多次、断断续续地想要吃透它，但是一直都不能如愿。大多数时候，我都卡在一些具体细节上，不能继续下去。我甚至购买或是借来了几乎所有有关这一主题的“大众科学”类的书。不用说，像这类中的许多书一样，它们从不深入细节，实际上对所有实用的目的几乎都无用。

可以想象，我读到“The Annotated Turing”这个书名时是多么地吃惊，作者不是别人，正是我熟悉的、通常写编程书的 Charles Petzold。这份吃惊只是个开头。当我打开书时简直是震惊。就好像有人知道了你的梦想，并让它绝对精准、不差分毫地成为了现实。如果对于所有里程碑式的论文都有这样的一本书，那将是一场怎样的学习变革啊。

我列举一些这本书之所以完美的地方。不是空泛的“近乎完美”，我说的是极其完美。

- 首先，这本书对图灵论文的每一行都做了解释，逐字地。这本

书以醒目的方式复述了图灵的论文，又对这些内容分段做了解释和讨论。作者提到，去掉所有解释可以重新组成带有原始页码的图灵论文。那就是我所说的精准。

- 书中还包含了所有背景概述、历史情境、重要事件、结果等内容，给读者的这次旅程做好了准备。

- 这本书读起来津津有味，我经常会忘记自己是在读一本讲述计算机科学核心内容的技术类图书。读这本书，就像是最好的计算机科学教授帮你逐行解读论文，并回答你提出的所有问题，即使这些问题很傻。

- 就像我多次怀疑的那样，图灵的原始论文中有很多错误。令人惊讶的是，这本书指出并纠正了这些错误，哪怕只是很小的印刷错误。我只是惊讶，作者如何知道这么多这些微不足道的印刷错误的“内幕”的。

- 图灵的论文里充满了晦涩、奇怪的符号，（你看到那些古老的哥特式德国字体了吗？）它们在今天的科学文献常见。作者解释了所有这些符号，它们的含义、从哪儿来的、细微的差异等等。太神奇了。

- 图灵的论文中有许多省略的解释和步骤，很可能是他忽视了“留给读者操作”，而让论文保持简短。有时候，你可能会卡在这些操作上，而且如果你不在学术界，很可能得不到外界帮助。这本书论及了所有这些省略内容，并做了极其漂亮的扩展，我无法想象还能有更好的方法来描述它们。

- 除了省略，还有很多内容简化了，图灵做了相当快速的解释，

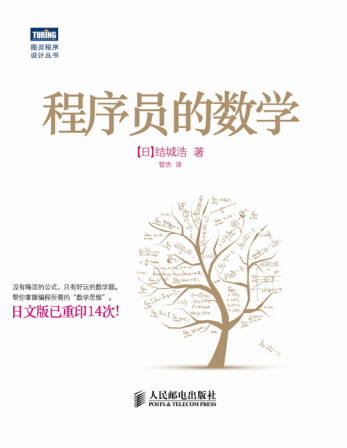
有时就直接略过了。这本书 100% 涵盖了这些内容。

- 实际上，要理解图灵论文，主要是在心里针对他给出的例子一步步运行他的机器。这本书实际上就是在解释这一步步的运行，让论文更容易阅读，而且能快速理解。

不管怎样，一些人总会想，为什么有必要读这篇古老的计算机科学论文？答案是，我们观察宇宙的方式已经发生了巨变。Seth Lloyd 的 *Programming the Universe* 一书确实很好地解释了这一思想，大概就是宇宙本身可以看作是一台计算机器，而不是物理领域中的粒子和能量。甚至有一篇论文提出，一个由空间中互动的台球组成的简单系统也是图灵完备的！这意味着通过设置台球在空间中的某些初始点和速度，理论上可以计算任何你的笔记本电脑可以计算的东西。要理解这个领域的进展，你必须充分了解什么是图灵机、图灵完备意味着什么，以及如何证明某个系统是图灵完备的。这本书就这样出现了。教科书是做不到的。

妙　评

《程序员的数学》



如果数学不好，是否可以成为一名程序员呢？答案是肯定的。本书最适合：数学糟糕但又想学习编程的你。读者无需精通编程，也无需精通数学，只需具备四则运算和乘方等基础知识，就可以阅读本书。书中讲解了二进制计数法、逻辑、余数、排列组合、递归、指数爆炸、不可解问题等许多与编程密

引导你数学再入

作者：つね

这本入门书对初学者容易失误的地方进行了细致的解说。

可以说，结城老师的所有著作都有这一特点。

我们常常会在程序业务逻辑中同递归、排序和二分法查找等算法打交道。

以数学的观点进行思考，也是对平日所写的代码有个重新认识的机会。

虽然书中的示例代码是用 C 语言写的，但只要有一些 Java、PHP、JavaScript 等类 C 语言的基础知识就可以阅读本书。

非常有趣

作者：herbest

切相关的数学方法，分析了哥尼斯堡七桥问题、高斯求和方法、汉诺塔、斐波那契数列等经典问题和算法。

译者 / 管杰
《程序员的数学》译者，图灵社区 ID：flashjerry

非常轻松愉快地读完了这本书。

令我不禁感到数学真奇妙啊！

虽然阅读过程中也遇到了对我来说比较难的地方，但最终还算顺利地读到了最后。

在阅读本书的同时，我也开始了编程的学习。一捧上这本书，我就会不由自主地产生想写程序的冲动。

这本书带给了我一段“美妙的旅行”。

文科生亦适用

作者：happy-yuki

虽然我是学文科出身的，但我觉得这本数学书很好理解，感觉很亲切。

目前尚未读完全书，不过我会用从书上学到的“简化规则”方法将多年不解的问题在脑海中理清脉络。

还有，书中的图示也非常简单明了，让文科生豁然开朗。

强烈推荐想要战胜数学的文科同学们也来阅读本书。

是本好书，难者亦难

作者：Yazin

有些读者可能会认为这本书“非常简单”或“深度不够”，这些人想必受过很好的数学教育或具有数学天分吧！

这本书恐怕不是面向他们而写的。

我在学生时期对数学最感头痛了，与数学有关的科目一概敷衍了事。

我高中能够毕业全靠数学老师的手下留情。

大学学的也是理科，能够毕业也只因数学不是必修课而已。

另一方面，我从学生时期就开始自学编程，现在是一名程序员。

随着在编程领域不断深入，我对其背后的计算理论产生了兴趣。

但我能够理解的仅仅是四则运算、逻辑运算以及二进制而已。

数论方面的书对我来讲高深莫测，难以理解。

而这本书就像是我的“救命稻草”。

我读完了整本书。果然不擅长数学的人也能一口气读完，就这一点而言这本书是很有价值的。

试想，如果阅读第一本书就遇到困难，丧失信心，那就会对数学产生排斥情绪。

不过，也并不是说任何人都能完全理解透彻。

无论书中的文字多么平易近人，对于完全没有数学细胞的人来说也是难者亦难。

我个人在理解上感到有困难的是思考题的解答部分。

的确，解答内容本身写得很有条理，不难理解。

但对于“如何思考才会迸出这个想法”的问题似乎没有具体的说明。

这样一来，因为作者是有数学功底的，所以有可能不太清楚“不懂数学的人哪里不明白”。

虽说如此，我对这本书的整体评价就是“如果你想战胜你那糟糕的数学，那么这本书将是不错的选择”。

据我所知，目前还没有类似的书籍。这本书可以作为你数学再入门的第一本书！

听《量化》译者讲“故事会”

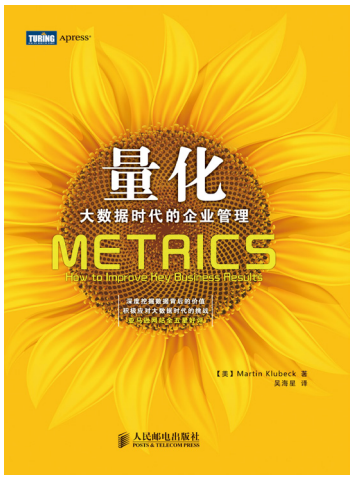
作者 / 吴海星
2001 年毕业于南京理工大学，现任北京北控文化体育有限公司技术总监。熟悉 Java 及 Scala 语言，熟悉 web 应用系统开发，熟悉 IC 卡应用系统建设，目前主要对商业智能方面的内容感兴趣。《量化：大数据时代的企业管理》译者。
图灵 ID: 海兴

1963 年 7 月，故事会创刊。那时候杂志很少，除了老奶奶口口相传的狼来了，想看故事就全靠他了。其中那些短小精炼的故事，娓娓道来，或轻松诙谐，或荒谬怪诞，或凝重感人，告诉我一个又一个人世间的道理。现在还能在报刊亭琳琅满目的期刊杂志之中看到他，简单质朴，一如往昔。

人世间有很多道理，能弄明白这些道理并有能力践行的人，会变成一个纯粹的人，一个脱离了低级趣味的人，一个成熟的人。在人生的不同时期，我们要掌握的道理是不一样的。“身体是革命的本钱”是绝对真理，所有人都应该懂。但如果只掌握了这些基本的人生真理就停下来，不再追寻新的真理，那一个人的灵魂也就定了形，不再长大。哪怕活到 70 岁，也只是一个夭折的毛头小子，或者姑娘。

对不起，亲，让你误会了。这些文字不是讲故事会的，是给《量化：大数据时代的企业管理》做的广告。

继续聊故事。故事对于我们来说至关重要，小时候听爸爸妈妈讲睡前故事，长大后为了卖东西给客户讲故事，为了融资给投资人讲故事，似乎我们的一生都在和故事打交道，直到我们自己变成



在企业管理领域，人们笃信这样一个原则：用数据说话。在从粗放的管理到科学地决策的转变中，数据发挥着关键的作用。进入大数据时代，人们面对着空前膨胀的海量数据，如何从其中挖掘出有用的指标和信息，帮助你进一步提升企业的绩效呢？《量化：大数据时代的企业管理》系统介绍了提升企业绩效的方法和工具，不仅阐述了企业应进行量化的必要性，而且详细讲解了如何创建量化分析体系，使其更好地反映组织健康状况和客户满意度。

了一段故事。哦，还有给女朋友讲的鬼故事。所以我借用了故事会的名头，给大家讲量化的故事。既然扯起了虎皮，就得对得起这面大旗，先来个小段：

> 讲的是某所大学招生办公室里的一位职员，她这两年净和大一新生打交道了。
>
> 有一次，有位大一新生跟她说，他作为客户，理应得到更好的服务，并大声跟她嚷嚷，让她想尽一切办法，直到自己满意为止，因为离了学生学校就无法生存。
>
> “你提供的服务在哪？”他最后喊了一句，享受着其他同学的注目礼。
>
> 她安安静静地等他喊完，希望他能听清楚自己要说的话。
>
> “亲，你搞错了。你不是客户，是产品。”

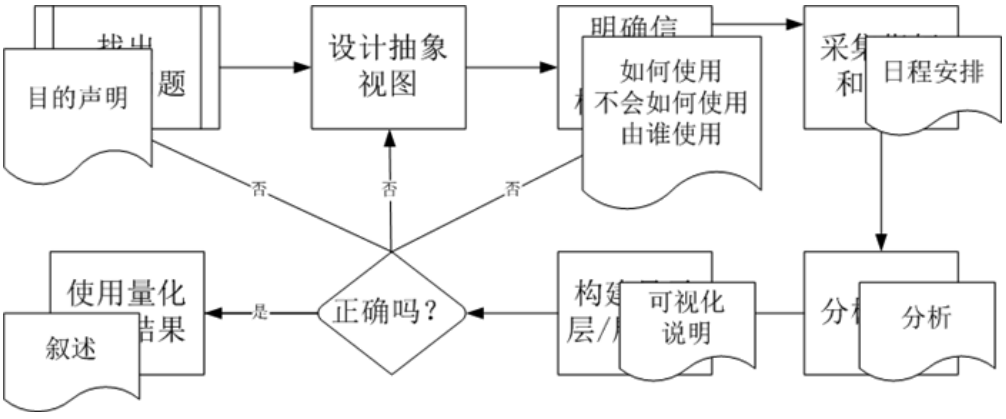
这个段子出自书中的第六章，貌似嘲笑学生，实则批评这位聪明的职员。**做量化，一定要知道自己的客户是谁**。

前戏结束，正题开始。列位，请上眼。

这本书是作者根据个人经历整理而成，讲述了一个对量化一无所知的懵懂“少年”，历尽艰难困苦，一路披荆斩棘，克服种种困难，最终成长为叱咤风云的数林盟主的励志故事。

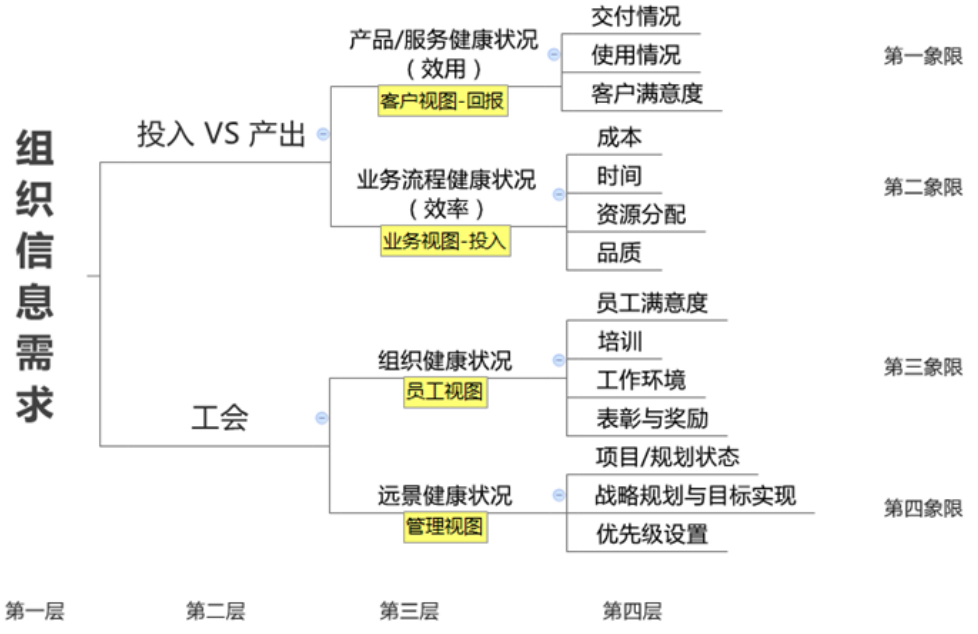
该书以量化分析体系的建立过程为主线，穿插介绍每个环节中要

注意的问题以及应该采取什么样的策略和原则。总体来说，这是一本讲道理的书。其核心是下面这张路线图：



这是由 7 幅图拼成的，每一幅在书中都有详尽的描述。

上面的过程是量化的经线，而下面这幅图是量化的纬线：



这是作者根据自己的经验总结出的量化分析体系框架，或者说模板，但这个模板如何使用，还请参见书中说明。作者用自己血的教训告诉我们，量化有风险，使用须谨慎！切忌乱动，一招不慎，可能被宫！

还有一点值得一提，本书在内容安排上也非常新颖有趣。作者很体贴，几乎每一章都有一节被称之为 小灶，用以印证、强调该章的内容。之所以叫小灶，我猜是因为其中那些字没跟出版社算钱。要没要钱咱不管，总之名符其实，都很美味。

下面这个故事就来自小灶：

> 我有个六岁的女儿，她们学校通过举办募捐活动来补充资金。最近，学校以 5 美元的低价销售本地快餐店的优惠册。这生意不错，一张优惠券就能让我轻松收回投资。
>
> 我喜欢学校的这个活动。在我看来，它卖的东西物美价廉，学校的资金也得到了丰厚的回报。校长也喜欢，但他还是想确认计划的进展是否像表面上看起来那么好，以便决定是继续开展活动，还是另辟蹊径。
>
> 有一天，我送女儿上学，她下车后，校长朝我走了过来。“Marty，我们收集了一些数据……”他知道我是量化分析专家。实际上他已经“在收集数据”了，说到这里，他停顿了一下。
>
> “真的？”我的语气尽量保持平淡。我喜欢谈论量化分析，和大多数人一样，我也喜欢讲自己的专业，但我还从没遇

到过喜欢我的答案的人。**大多数人都不知道根本问题，不喜欢创造性。他们只想用数据证明自己是正确的，并且“比想象得简单”**。

“嗯。我们收集了优惠册活动今年的数据。”

“嗯嗯。”我咬了自己的舌头。

“我想知道它是否成功，所以我们统计了销售册数。”

没有定义“成功”，没有找出**根本问题**，他本可以避免这几件事。好在他们没投入太多精力去收集这些数据，他只是清点了本年剩下的优惠册，然后和学校当初购买的册数对比。

他甚至将每周销量占总销量的百分比做成了图形。

“图形不错。”我说。

“你也看到了，我们提前一周卖光了所有优惠册，所以明年我想把采购数量翻一番。”

“真的？”

“当然，这个活动非常成功！”

“好吧……”我忍不住皱了下眉头。

接下来是短暂的沉默，实际上我已经在想象自己结束谈论转身逃跑的情景了。

“我知道你是专家，肯定知道我们哪里做错了。没事儿，告诉我吧。 ”

我极其耐心、温和地向他解释哪里可能会有问题。

“John，你没做错，只是可能漏掉了一些事情。我建议你在决定之前最起码做些调查。在继续深入下去之前，我想第一件事就是首先定义一下你所说的‘成功’，就是买了多少本优惠册这么简单吗？”

“当然。难道不是吗？”

“我真不知道。你们为什么卖优惠册？目的是什么？”他转了转眼睛，感觉我问的是显而易见的事情。

“为了赚钱。”

“好，你在这方面取得了成功，对不对？”

“是啊。”

“你就想知道这些吗？”

“当然。”

“真的吗？你为什么要把明年采购的优惠册翻倍？看起来你想知道的是明年该买多少本优惠册……”

“哦，好。如果它成功了，我想我们应该增加采购量。”

“如果没有全卖光，你会做什么呢？假如你只卖了一半？”

“不知道。可能明年只会买今年的四分之三吧，并且努力多卖点儿，也可能搞个别的募捐活动？”

“John，我建议客户做的事情之一，就是在采集任何数据之前先决定如何使用问题的答案。所以，我想问您，如果答案是‘很成功’，你怎么办？”

“非常成功！”他微笑着说。

“好，非常成功。关键是，如果你告诉我想把订单翻倍，我就能帮你找出真正的问题，听起来可能像‘明年我应该买多少本优惠册’，而可能的答案是：没有答案，放弃活动，订单翻倍。”

“有啥区别？”

“哦，如果我们要回答新问题，你看待数据的方式就会不一样，并且你也得收集些新数据，比如：

……

好吧，这个故事的确没讲完，我残忍地把它砍了，就是为了吊你的胃口。虽然这个故事不完整，但我砍的时候也是掌握好了分寸的，主题还在，**找出真正的问题**，是量化分析的关键所在。

如果你觉得不爽，觉得人生不完整，那就找本书看完整的故事吧。

不过为了不让你带伤前行，最后再来一个故事，权作药用。如果你够坚强，请直接跳过这个故事，毕竟这只是个搞笑小段。

下面这个故事告诉我们一个道理：**拿到量化分析结果的第一反应应该是展开调研，毕竟量化分析结果不是事实，只是指示器。**

> 我有个朋友喜欢讲一个分析师的故事。有位仁兄结一个案子所用的时间是其他人的三到五倍。“事实”很明显——他效率太低了。他结的案子连其他同事的一半都不到，而且处理每个案子花的时间都很长。“数值”惨不忍睹。
>
> 经理看到这个“事实”后决定痛下杀手。不用管他的思想斗争过程，这位“慢羊羊”先生是年纪最长、资格最老的分析师。我们要讨论的是这位经理犯的错误，他把自己看到的数据当成了“事实”，可实际上那只是个指示器。而且他没有做调查研究，就匆忙采取了行动。
>
> 他把这位弱弱先生叫到自己的办公室，开始折磨他。在叽里呱啦一通狠批之后，他抛给这位仁兄一大串充满威胁意味的问题：“那，你准备怎么办？想怎么缩短解决问题的时间？我希望你做得又快又多。”
>
> 就在他觉得耐心快要达到极限时，这位老哥说话了：“我想先问问，我工作的质量如何？”
>
> “少扯淡！我就告诉你，你是所有分析师里最慢的！”
>
> “那只是速度问题，不是质量。最近有人跟您抱怨过吗？”

“这个，没有。”

“客户投诉过我吗？”

“没有。”

“那其他同事呢？有人埋怨我吗？”

“也没有，”经理说，“但数据不会说谎。”

“您说得对，它不说谎。但它反映的不是全部，所以这不是真相。”

“啥？你是想说你不是最慢的？公司花钱是让你结案的。行不行啊你？”经理言下之意是他干完不结案。

“是，我最慢，”最慢帝痛快地承认了，“但我不是不行，相反，我很专业。你问过其他人我为什么慢吗？”

“没问，我这不正在问你吗？”

“但实际上你从没问过我为什么慢。你只是把数据拿出来证明我‘慢’，我‘效率低’，现在又变成我‘没能力’了。”

球被踢回来了，经理很生气。最慢帝还没完：“您检查过我结得案子都是什么类型的吗？实际上我比他们大多数人都快。如果您看我处理的简单案子，就知道我是最快的了。”

“没那种数据，”经理说，“我怎么可能知道你处理的案子都是什么类型的？”

最慢帝应道：“问问？”他沉默了一会儿，说，“如果您问问我或者其他任何同事，为什么我结案时间长，为什么我结的案子少，您可能会找出点眉目。我结的比较少，是因为我的案子结起来时间长。其他同事把他们处理不了的案子都交给我了，因为我经验最丰富，所以我接的都是最难的案子。不是我慢，也不是效率差，更不是不称职。恰恰相反，我是这层楼里最好的分析师。”

经理看起来有点挂不住了。

最慢帝又开始了：“要不，您告诉我该怎么办，我该怎么改进。要是您同意，我就不从其他人那里接案子了，客户那些最刁钻古怪的问题我都放着。您让咋办我就咋办，您是领导。”

本文到了这里就该结束了，一本书好与不好，不同的人会有不同的标准，就像现在我对故事会的看法一样，你可能觉得里面的内容太浅显，太幼稚了。但一本书是不是有趣，大家的看法应该是一样的。所以我可以负责任地的告诉你，《量化》很有趣，笑果很明显！

在跟你说再见之前，有一句临别赠言：**量化有风险，使用须谨慎！**另外，由于生命可贵，时间有限，请慎重选择准备花时间阅读的书。■

社区动态

好文章要给更多人看，用你的笔传递最新 IT 好文。阅读量前 5 名的译人可任选图灵出版图书一本。

iTran 乐译 13 期

- 开火前进 ——识破微软的龌龊伎俩
- 开火，移动 - 大神 Joel 也浮躁
- 微型出版——你我都是变革的见证人
- Markdown 的未来
- Strategy Letter V 开源软件的经济学
- 策略书之 V：开源经济学
- 如何教新手编程
- 对用户记录产生的思考
- 奇思妙想 II 前言
- 奇思妙想 II：16 位天才揭秘未来的计算
- 计算机算法：Strassen 矩阵相乘算法
- 模仿生命的艺术

好运电子书！购电子书抽奖送纸版。

好运电子书第 1 期

抽奖规则

- 电子版图书销售每满 50 份，从中随机抽出 3 位读者；
- 赠送与纸书定价等值社区银子，可以自由兑换任何纸质书。

第一期开奖图书

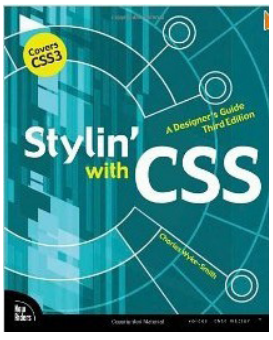
只要能把本书读懂、吃透，自称“CSS 老手”绝对没人敢说你吹牛皮。要是能将书中给出的建议和提示融会贯通地运用到自己的项目当中，那“CSS 高手”这顶桂冠也将非你莫属！

恭喜以下读者：

eason007、mattdamon、zhangdi_china、cisolarix、yinzixin、

宇宙一片小囧中奖。

观看开奖视频



《CSS 设计指南（第 3 版）》

听书
Reader

读你爱的书，传播一份能量。为了有阅读障碍的人士，为了在运动时也能获得信息，为了自己的好心情！

“听”书计划

挑一本书（或者其中某个章节）朗读、录制，并把文件分享出来。

阅读推荐：

- 黑客与画家：硅谷创业之父 Paul Graham 文集
- 编程大师访谈录
- 编程人生：15 位软件先驱访谈录
- 软件随想录：程序员部落酋长 Joel 谈软件
- 宇宙之书：从托勒密、爱因斯坦到多重宇宙
- 程序员的职业素养
- 演讲的艺术：演说大师教你如何打动听众
- 苹果狂潮：iPhone 开启永远在线的时代
- 短！微讯息时代写作的艺术
- 罗素的故事（可以尝试分角色朗读）
- 卓越程序员密码
- 创意不是想出来的

了解活动详情

欢迎加入

图灵社区

最前沿的IT类电子书发售平台

电子出版的时代已经来临。在许多出版界同行还在犹豫彷徨的时候，图灵社区已经采取实际行动拥抱这个出版业巨变。作为国内第一家发售电子图书的IT类出版商，图灵社区目前为读者提供两种DRM-free的阅读体验：在线阅读和PDF。

相比纸质书，电子书具有许多明显的优势。它不仅发布快，更新容易，而且尽可能采用了彩色图片（即使有的书纸质版是黑白印刷的）。读者还可以方便地进行搜索、剪贴、复制和打印。

图灵社区进一步把传统出版流程与电子书出版业务紧密结合，目前已实现作译者网上交稿、编辑网上审稿、按章发布的电子出版模式。这种新的出版模式，我们称之为“敏捷出版”，它可以让读者以较快的速度了解到国外最新技术图书的内容，弥补以往翻译版技术书“出版即过时”的缺憾。同时，敏捷出版使得作、译、编、读的交流更为方便，可以提前消灭书稿中的错误，最大程度地保证图书出版的质量。

最方便的开放出版平台

图灵社区向读者开放在线写作功能，协助你实现自出版和开源出版的梦想。利用“合集”功能，你就能联合二三好友共同创作一部技术参考书，以免费或收费的形式提供给读者。（收费形式须经过图灵社区立项评审。）这极大地降低了出版的门槛。只要你有写作的意愿，图灵社区就能帮助你实现这个梦想。成熟的书稿，有机会入选出版计划，同时出版纸质书。

图灵社区引进出版的外文图书，都将在立项后马上在社区公布。如果你有意翻译哪本图书，欢迎你来社区申请。只要你通过试译的考验，即可签约成为图灵的译者。当然，要想成功地完成一本书的翻译工作，是需要有坚强的毅力的。

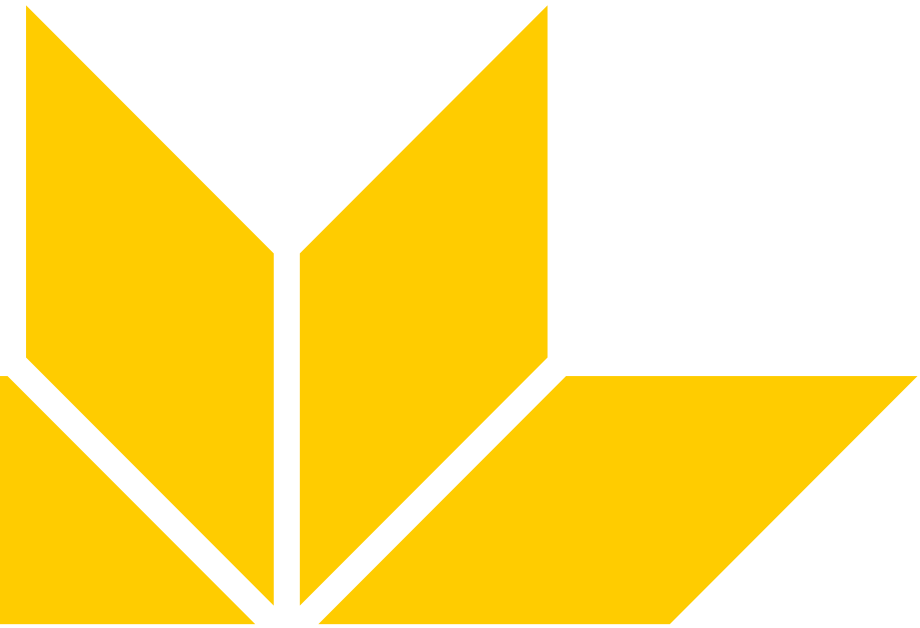
最直接的读者交流平台

在图灵社区，你可以十分方便地写作文章、提交勘误、发表评论，以各种方式与作译者、编辑人员和其他读者进行交流互动。提交勘误还能够获赠社区银子。

你可以积极参与社区经常开展的访谈、乐译、评选等多种活动，赢取积分和银子，积累个人声望。

图灵社区 出品

出版人：武卫东
执行主编：杨帆
编辑：李盼
顾问：谢工、傅志红、李松峰
设计：大胖



地址：北京市朝阳区北苑路13号院领地OFFICE C座603室
电话：010-51095181
微博：weibo.com/ituring
Email: ebook@turingbook.com