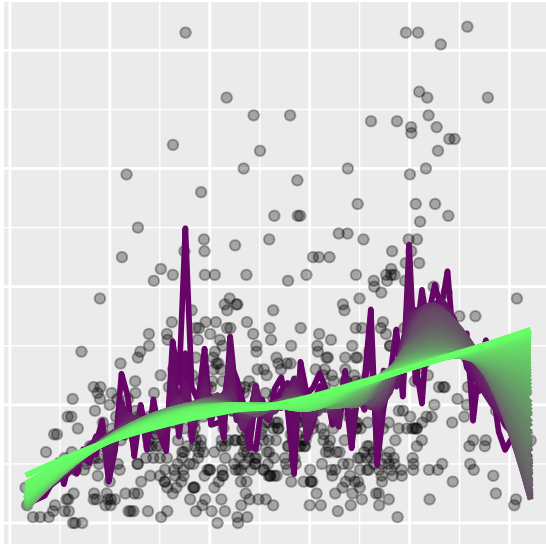
现代统计图形

赵鹏　　谢益辉　　黄湘云

2021-03-16



目录

插图

自序

2019 年底，陈兴璐编辑给我写了封邮件，问我是否有兴趣写一本中文书。这问题可以说是问到我心坎上了。*2018* 年我在给赵鹏的《学 *R*》一书写推荐序时就表达了写中文书的强烈愿望，只不过写书这种工作最好是用大块连续的时间去做，而如今多数人的时间都已经严重碎片化，我也不例外（尽管还在假装抗争），所以想要徒手写出一本新书的可能性已经微乎其微，于是我想到这本书稿。

赵鹏在本书后序《古统新修记》中提到了这本书的历史。从我 *2007* 年开始写这本书起，就不断有人问我何时出版。当年我把这本书的信息放在博客的某个网页上，后来我把这个网页藏了起来，但现在仍然可以看见当年很多读者在下面的留言。有些读者甚至说“银子都准备好了”。一晃十三年过去，我也从一个不靠谱的小青年成长为一个不靠谱的小中年，唯一不变的是这书仍然没有出版，惊喜不惊喜……我不知道以前当大家在谈论拖延症时大家在谈论些什么，反正以后大概可以谈谈这本被我拖成《古代统计图形》的《现代统计图形》了。

这些年来，我曾想过找人帮忙把这本书稿重新整理一下，也有不少人提出要给我帮忙，但我开发了几年 *R Markdown* 之后，实在不想再打开这本书的 *LaTeX* 原稿件，所以也没和这些志愿者一起推进。恰好在兴璐编辑问我的三个月之前，我终于咬咬牙把这本书的旧稿子全盘交给了黄湘云（他之前也问过我好几次了），请他帮忙把原始的 *LaTeX* 格式转化为更简单的 *R Markdown* 格式，结果他三下五除二很快就把这事搞定了，这解决了出版的两大障碍之一。另一大障碍是，由于此书写于十几年前，彼时 *ggplot* 还不像现在这样一统江湖，于是这本书的旧稿将重点放在 *R* 的基础图形上（第九章），而

对 *ggplot* 的介绍则很简略。这个大坑，则被赵鹏卷起袖子给填了，书中凡是能用 *ggplot* 作的图，他都用 *ggplot* 重写了一遍，好歹算是把《古代统计图形》拉上了《近代统计图形》的台阶。我依然觉得“现代”二字有点名过其实，但现在总算不像一年前那么心虚了。

话说回来，我们当然不能说用什么工具作图才称得上“现代”。“现代”与否，不应该与工具挂上钩，否则我们一来容易陷入工具崇拜，二来也很容易变成拿着锤子找钉子的人，也就是查理芒格说的“铁锤人倾向”。如果不注意作图的目的和原则，那么优秀的作图工具也一样可能制造图形垃圾，就如同 *LaTeX* 也绝对可以排出把人丑哭的版式一样。也许有读者还记得，我在最早版本的序言中引用过顾炎武在《日知录》中引用《易经》中的一句话：“形而上者谓之道，形而下者谓之器。”那时候年少轻狂，引这种话有卖弄之嫌（装作很厉害的样子），不过也有一份真心实意在里面，也就是期待读者能得“道”。如今不敢说什么“道”不“道”的大话，只能说要是本书对读者有所启发的话，则善莫大焉。

我本人当然不算图形领域的专家，而且这些年的工作重心离这个领域越来越远，但我也斗胆讲我的两个观察，不知是否确切：

一是随着数据科学的浪潮，数据可视化也被推上了浪尖，图形成了数据科学不可或缺的一个组成成分，但我们似乎越来越依赖现成的作图工具和系统（哪怕自己写代码作图，也是用现成的库），难以见到新颖的数据展示方式，而那些新颖的图形，往往需要用更原始和底层的方式创造出来。在第二章中，我提到用写代码的方式作图能提供高度的定制性，意即越高层的工具，定制性相应也会更弱，所以也更容易束缚创新。有鉴于此，时至今日，我仍然认为读者不妨了解一下 *R* 的基础作图系统，而不必随大流、认准了 *ggplot* 不撒手。基础作图系统用起来当然是繁琐一些，但它提供了所有的图形元素供你调遣，而且对数据形式也没有任何假设（不必非得是整齐的数据框），有时候可能会更方便和自由。注意，我是学基础作图系统长大的，所以我的这个观念可能有偏差。当年我玩这些点线面的时候，感觉就像玩画笔，可以说乐在其中、不能

自拔，比如图 *9.6* 让我兴奋地发现原来渐进色可以这样创造（虽然很低效），图 *6.4* 是我受那幅著名的拿破仑远征图（图 *1.5*）启发而创造出来的，图 *3.11* 中的篮球场地则是我按照场地标准尺寸“一笔一划”地用点线圈画出来的（可见曾经有多闲）。第九章最后那个画温度计的练习，可能是我当年沉迷基础作图系统的最好例证。打个比方，用 *ggplot* 或其它高层作图系统就像是上帝捉住你的手在画图，而用基础图形系统则需要你捉住上帝的手来画。

二是如今画图似乎朝着美观方面一边倒，而难以见到把数学原理与图形结合表达的例子。我不太明白这是果真成了看脸的时代，还是说统计理论与图形的隔阂更深了。坦言之，我读研究生之后开始不太喜欢数学，但有时候看到一种数学方法以图形的方式巧妙表达出来时，还是觉得很惊喜的。例如第 *7.8* 小节中的四瓣图，和第 *6.12* 小节中的调和曲线图。对于后者，我本科大四时还仔细验证过那个欧氏距离（当然现在恐怕三角函数的积分都忘了），并感叹这家伙是怎么想出这么绝妙的方法来的。不知这些“古代统计图形”，能否启发我们创造出更多有数学灵魂的图形？其实也未必一定要追求这些看起来高端的东西，有时候一个简单的想法也许就能启发我们，例如图 *5.3* 中的向日葵散点图，它的想法很简单，而名字又多有诗意。我想表达的意思是，一幅图不管用什么形式表达，只要你注入了特别的心意，它自然会萌发出生命力而动人。正如中岛美雪一首歌所唱的：生命的别名就是心。

所以就算这本书稿拖了十三年，我自问仍是有一定的出版价值的；内容方面依旧有一些闪光点，只不过以我如今的文字标准，有些地方文绉绉的表达我自己也看不惯了。我猜这本书至少会有两类购买者：一类是等它等了十几年的，不为别的，就图买个情怀，以纪念逝去的青春，也许买回去最终只是吃火锅的时候垫桌脚；另一类是冲着鄙人的虚名来的（我为你们赐名“冲虚道长”）。对第一类读者，我只想说，吃火锅的时候请叫上我；对第二类读者，你们来就来，还买什么东西嘛，非要买的话，我也只好第无数次重复我的告诫：读书的时候自己多判断，不要被我一面之词忽悠入坑。

本书每章开头都挑选了一段《福尔摩斯探案集》中的文

字，其内容与各章内容有一定关联（有些关联需要一定的脑洞才能理解），这也是由于我个人在上高中时就喜欢看福尔摩斯，并且我认为统计图形也可以看作是一种小小的“探案”。探案集中我最喜欢的一篇是《血字的研究》，尤其欣赏该篇的第二部分中大篇的景色描写，以及对主人公杰弗逊 · 霍普坚韧不拔性格的刻画，这种波澜壮阔的笔法，令我着实艳羡不已，只可惜我没这种文字功夫能把书写得如此吸引人，于是只能寄希望于“一图胜千言”了。

最后，我要感谢在写作过程中给我提供过帮助的人们，包括我在中国人民大学本硕期间的导师赵彦云老师、人大学弟学妹和统计之都的朋友们（如魏太云、邱怡轩、郑冰、李皞、方莹、李丰、王晓伟、李承文、肖楠、姜晓东等等）、爱荷华州立大学的师友们（如殷腾飞、我的导师 *Di Cook* 和 *Heike Hofmann*）。本书修订过程中也收到了来自张列弛、*Song Li*、*JackieMe*、*Yang Cao*、*Jonie Yao*、*tiansworld* 等人的贡献。这本书先后得到了多位编辑的付出，包括周筠老师（约十年前）、卢鹏翔编辑、陈兴璐编辑、王军花编辑等，其间我掉过链子，很不好意思，在此觍着脸一并感谢。当然，这本书最终的出版，离不开我的两位苦力合作者黄湘云和赵鹏；要不是他们玩命推，我估计这本书稿可能真的要留给未来的考古队来发掘了。

——谢益辉　于　美国奥马哈

导读

内容结构

本书根据统计图形制作的需要，将所有内容分为三部分：

第一部分（一至三章）侧重趣味性，介绍从古至今的一些统计图形案例。首先，在欣赏前人智慧的基础上，说明统计图形在社会生活的各个方面所能体现的价值；接着，简单介绍了一下本书作图使用的工具——R 语言；然后，从数据的角度对各种统计图形给出一些应用实例。介绍案例的同时，提供了可供运行的源代码。

第二部分（四至八章）侧重知识性，集中介绍讲解现有的统计图形种类，如常见的条形图、直方图、箱线图、散点图等，此外还会引入若干较特殊和不太常见的图形种类和数据图示方法，并且配以相应的统计数据分析实例，深入说明统计图形的用法和含义。介绍图形的同时，提供了 R 基础作图和 **ggplot2** 作图两种解决方案的源代码。最后，总结分析了绘制统计图形的一些指导原则，以便让读者清楚区分统计图形运用的条件和场合。

第三部分（九至十一章）侧重技术性，讲解 R 语言中的作图系统，包括基础函数、**ggplot2**、**lattice**、**rgl** 等多种作图系统，详细介绍了点、线、多边形、颜色和文本等图形元素的使用细节。本部分会给那些期望能自定义统计图形的读者提供方便的解决方案。

阅读方案

本书可以从任意章节开始读，各章节之间没有严格的逻辑依赖关系。不过，为了方便不同的读者，我们推荐三种阅读方案。

方案 1: 对统计图形的解读感兴趣的读者，我们推荐优先阅读第一部分和第二部分。这些内容介绍了图形实例和统计分析。读者不妨在阅读的同时，考虑如何用自己熟悉的作图工具把这些图形作出来。如

第一章　经典图形

> “这易如反掌，”他说，“我看到你左脚穿的那只鞋的内侧，也就是炉火刚好照到的地方，皮面上有六道几乎平行的划痕。显然，这些划痕是有人为了去掉沾在鞋跟上的泥疙瘩，极其粗心大意地顺着鞋跟刮泥而造成的。因此，现在你就明白了我得出的这两个推断：其一，你曾经在恶劣的天气外出过；其二，你穿的皮靴上面的特别难看的划痕是伦敦的女佣所为。至于你开业行医，这么说吧，如果一位先生走进我的房间，身上带有碘的气味，右手食指上有硝酸银腐蚀的黑斑，高顶黑色大礼帽的右侧鼓起一块，那里面藏着听诊器，而我不断言他是医务界的一位活跃分子，那我不是太迟钝了吗？”
>
> ——柯南·道尔《波希米亚丑闻》

统计图形的意义在于引导我们观察到统计数据中的信息。用著名统计学家 John Tukey 的话来讲，就是“图形的最大价值就是使我们注意到我们从来没有料到过的信息”（The greatest value of a picture is when it forces us to notice what we never expected to see）。从这个意义上讲，统计图形的重要性毋庸赘述。

在统计图形历史上，能够达到“揭示人们不曾料到的信息”这种高度的图形并不多，那么这里我们首先欣赏几幅前人创造出的名垂青史的统计图形。

1.1　饼图和线图的起源

饼图和线图都是当今社会中常用的统计图形，它们是由有着“统计图形奠基人”之称的苏格兰工程师兼政治经济学家 William Playfair 发明的。在 *The Commercial and Political Atlas* (Playfair, 1786) 一书中，

他用线图展示了英格兰自 1700 年至 1780 年间的进出口数据（如图 1.1），从图中可以很清楚看出对英格兰有利和不利（即顺差、逆差）的年份；而在 *The Statistical Breviary* (Playfair, 1801) 一书中，他第一次使用了饼图来展示一些欧洲国家的领土比例，图 1.2 即为史上第一例饼图。从下方的饼图中我们可以清楚看出当时的土耳其帝国分别在亚洲、欧洲和非洲的领土面积比例。

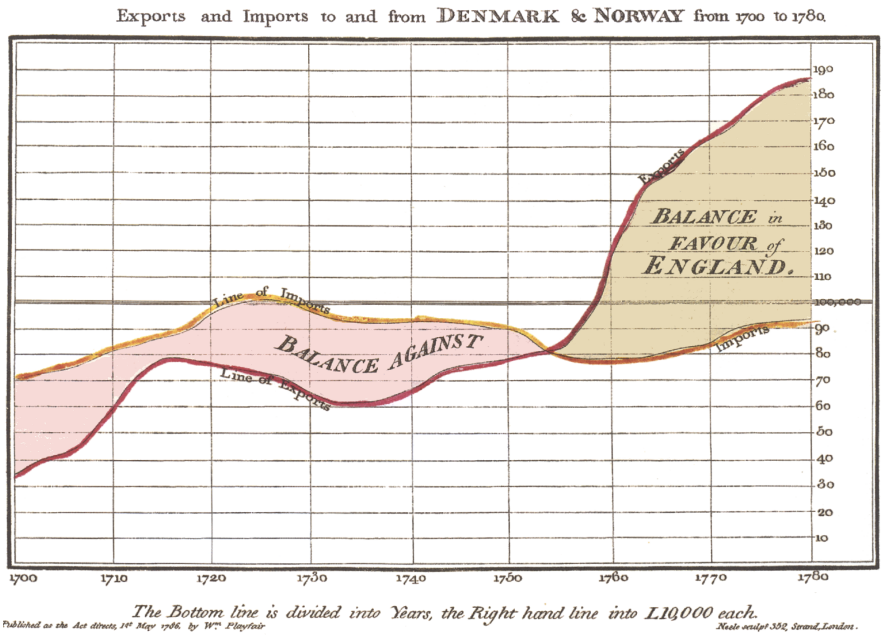
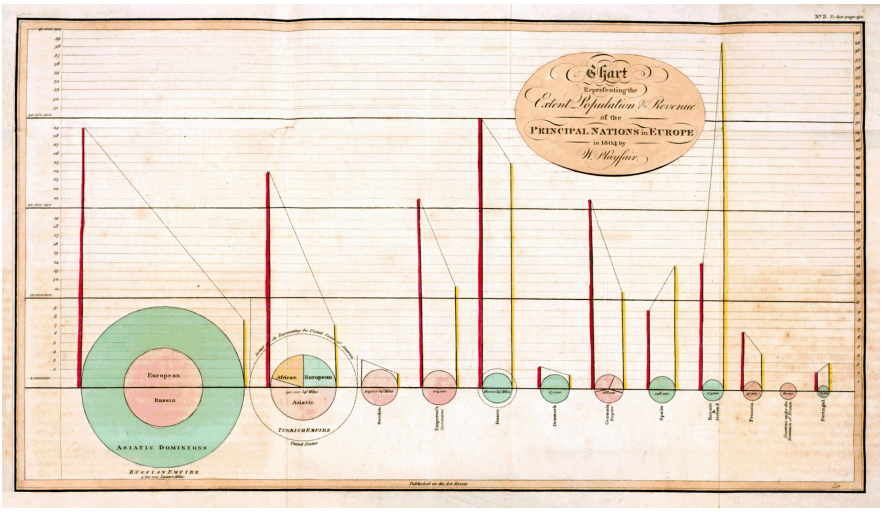


图 *1.1: Playfair (1786)* 绘制的线图。这幅图主要展示了 *1700* 年至 *1780* 年间英格兰的进出口时序数据，左边表明了对外贸易对英格兰不利，而随着时间发展，大约 *1752* 年后，对外贸易逐渐变得有利。图片来自维基百科

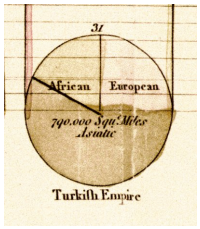
这两幅图在今天看来似乎没有什么惊世骇俗之处，但在当时统计图形种类极为稀少的年代，能以这种方式清晰展示数据结构，也实属难能可贵。除了这两种图形之外，他还发明了条形图和圆环图。

1.2 霍乱传染之谜

袭击欧洲大城市最严重的天灾要数 19 世纪的霍乱。由于垃圾没有得到及时清理，清洁水源的缺少，以及下水管道系统的不足，伦敦成为



(a)



(b)

图 1.2: *Playfair (1801)* 绘制的饼图。这是历史上最早出现的饼图，描述了法国大革命前后一些欧洲国家的统计数据。上方的大图展示了各个国家的领土面积（和圆圈成比例）以及人口（左垂线）、税收（右垂线）、国土在各大洲分布比例等数据，两条垂线连线的斜率可表示税负的轻重（这一点颇有争议，因为斜率与圆的半径有关）。下方的饼图展示了土耳其帝国在三大洲的国土面积分布。图片来自维基百科

无药可医的流行病滋生的温床。公众一致认为霍乱是由空气传播的，如果呼吸到了“瘴气”或者接触到霍乱患者，就会染上这种病。医生兼自学成才的科学家 John Snow 对这个观点颇为怀疑，他决心通过彻底调查这种致命疾病的根源来证实他的怀疑。

通过和当地居民交谈，他确定了霍乱爆发的源头是位于 Broad 大街的公共水泵。他对这种疾病类型的研究看起来很可信，因此他成功说服了当地政府废弃那个水泵。他所利用的主要证据就是图 1.3：死亡发生的地点有明显的地理规律，在这种规律的指引和相关调查证据的支持下，

第二章　作图工具

“好了，好了，我的好伙计，就这么办吧。我们在这房子里共同生活了好几年，如果再蹲在同一座牢房里就更有意思了。华生，我跟你说实话。我一直有个想法：我要是当罪犯，一定是超一流的。这是我在这方面难得的一次机会。看这儿！”他从抽屉里拿出一个整洁的皮制小袋，打开来亮出里面几件闪亮的工具。“这是最新最好的盗窃工具，镀镍的撬棒，镶着金刚石的玻璃刀，万能钥匙，以及对付现代文明所需要的各种新玩意儿。我这儿还有在黑暗中使用的灯。一切都准备好了。你有走路不出声的鞋吗？”

——柯南·道尔《查尔斯·密尔沃顿》

2.1　选择准则

传统的作图工具，无非就是笔、尺、纸一类。随着计算机科技的发展，现代的工具和作图软件层出不穷。面对这么多工具，我们应该如何选择呢？我们认为主要的准则有三点：一是统计计算功能齐全，二是统计元素易于控制，三是图形类型丰富多样。

在人们通常的观念中，图形往往代表着简单，然而直观与简单是两个不同的概念。图形的首要作用的确是直观展示信息，而这里的信息未必是简单的。一幅优秀的统计图形背后也许隐藏着重要的统计量，而统计量是统计图形的最关键构成因素。

我们日常见到的所谓统计图形，常常是 Microsoft Excel 的产物。事实上，Excel 的图形总体看来只有三种：第一种是表现绝对数值大小，如条形图、柱形图、折线图等；第二种是表现比例，如饼图；第三种则是表示二维平面上的变量关系，如 X-Y 散点图。从更广泛的意义上来说，

Excel 展示的几乎都是原始数据，基于数据的统计推断的意味比较单薄。而统计学的核心研究对象是什么？答案应该是分布（Distribution）。注意：分布不仅包含一元变量的分布，而且更重要的是多元变量的分布，诸如“均值”、“方差”、“相关”和“概率”等概念都可以归为分布的范畴。无论 Excel 的图形如何搭配色彩、怎样变得立体化，都跳不出上述三种类型的限制，而不能全面妥善表达统计学的要义。可能正是因为这样的原因，统计图形界内的一位大家 Leland Wilkinson 才说，给统计刊物投稿时永远不要用 Excel 作图。相比之下，本书所采用的 R 语言能表达的统计量种类极其丰富。毫不夸张地说，任何理论上可以计算出来的统计量都能在 R 中很方便地以图形的方式表达出来。

除了统计量之外，我们也应对图形本身的组成元素给予足够的重视。这些元素包括点、线（直线、曲线、线段和箭头等）、多边形、文本、图例、颜色等，往往这部分工作都由常见的统计软件替我们做了——我们不必自己设计点、线等基本元素，但同时也就意味着我们几乎无法灵活运用这些元素（比如在图中添加点、线、文本标注等）。表面上看似计算机软件给我们省了不少麻烦，实际上，这种限制的弊端要大于那一点微不足道的好处。我们在实际工作中遇到不少这样的例子，比如往散点图中添加若干条不同的回归直线（根据某自变量不同分类的回归、或者是不同分位数的分位回归 (Koenker, 2020) 直线等）、在图中添加箭头或者文本标注甚至添加包含希腊字母、微积分符号、上下标的数学公式等等，不一而足。对于这些简单问题，用传统的（商业）统计软件恐怕太难解决，原因就在于，它们让计算机替代了太多本可以由用户完成的工作。当然，总体来说，自动化是好事，但如果为了自动化而大幅度牺牲可定制性，则可能得不偿失了。

相比之下，R 语言汇集统计计算与统计图示两种功能于一身，灵活的面向对象（Object-Oriented，OO）的编程方式让我们可以很方便地控制图形输出，从而制作出既精美又专业的统计图形。我们说图形并不意味着简单，指的是统计图形的构造可以很细致入微（包括统计量的选定和图形元素的设计），而不是指 R 作图的过程或程序很复杂。同样，如果我们总是需要在图形细节上耗费大量的精力，那么 R 也将不是一个好工具。在第二部分和第三部分，我们会介绍若干作图的细节问题，读者需要斟酌美观与效率之间的平衡，避免陷入细节泥潭；实际上，**ggplot2** 是一个可以大幅减少细节设置的图形系统，我们将会在第十章详细介绍。

要想真正精通统计图形，首先要练好基本功（例如对图形基础元素或构造作深入的了解），然后在此基础上通过各种抽象、提炼和认识，最

终上升到理性认识的境界（自如地表达统计量）。

2.2 R 语言简介

R (R Core Team, 2020) 是一款优秀的统计软件，同时也是一门统计计算与作图的语言，它最初由奥克兰大学统计学系的 Ross Ihaka 和 Robert Gentleman 编写 (Ihaka and Gentleman, 1996)。自 1997 年起, R 开始由一个核心团队（R Core Team）开发，这个团队的成员大部分来自大学机构（统计及相关院系），包括牛津大学、华盛顿大学、威斯康星大学、爱荷华大学、奥克兰大学等。除了这些作者之外，R 还拥有一大批贡献者（来自哈佛大学、加州大学洛杉矶分校、麻省理工大学等），他们为 R 编写代码、修正程序缺陷和撰写文档。迄今为止，R 中的程序包（Package）已经是数以万计，各种统计前沿理论方法的相应计算机程序都会在短时间内以软件包的形式得以实现，这种速度是其它统计软件无法比拟的。除此之外，R 还有一个重要的特点，那就是它是自由、开源的。由于 R 背后的强大技术支持力量和它在统计理论及应用上的优势，加上目前国内对 R 的了解相对较少，我们期望能够通过本书引进并推动它在国内的广泛使用。

R 的功能概括起来可以分为两方面：一是统计计算（Statistical Computation），二是统计图示（Graphics）。它由一个语言系统（R 语言）和运行环境构成，后者包括图形、调试器（Debugger）、对某些系统函数的调用和运行脚本文件的能力。R 的设计原型是基于两种已有的语言：S 语言 (Becker et al., 1988) 以及 Sussman 的 Scheme，因此它在外观上很像 S，而背后的执行方式和语义是来自 Scheme。R 的核心是一种解释性计算机语言，指令以函数的形式书写和运行，大部分用户可见的函数都是用 R 语言编写的，而用户也可以调用 C、C++ 或者 FORTRAN 程序以提高运算效率。正式发行的 R 版本中默认包括了 **base**（R 基础包）、**stats**（统计函数包）、**graphics**（图形包）、**grDevices**（图形设备包）、**datasets**（数据集包）等基础程序包，其中包含了大量的统计模型函数，如：线性模型/广义线性模型、非线性回归模型、时间序列分析、经典的参数/非参数检验、聚类和光滑方法等，还有大批灵活的作图程序。此外，附加程序包（add-on packages）中也提供了各式各样的程序用于特殊的统计学方法，但这些附加包都必须先安装到 R 的系统中才能够使用 (Hornik, 2018)。

R 的官方网站中对 R 有详细介绍，我们也可以从它在世界各地的镜

像（CRAN，全称 Comprehensive R Archive Network）下载 R 的安装程序和附加包。可能令读者感到不便的一点是，R 不像别的统计软件那样，图形用户界面（GUI，即 Graphical User Interface）提供琳琅满目的菜单。R 的界面常常是命令行界面（CLI，即 Command Line Interface），即：输入一些代码，R 就会输出相应的运算结果或其它输出结果。因此，在正式使用 R 之前，我们有必要大致了解一下 R 的运作方式。

计算机科学家 Nikiklaus Wirth 曾经提出，程序语言的经典构成是“数据结构 + 算法”：数据结构是程序要处理的对象，算法则是程序的灵魂。R 也不例外，它有自己独特的数据结构，这些数据结构尤其适应统计分析的需要，它们包括：向量（vector）、矩阵（matrix）、数据框（data frame）、列表（list）、数组（array）、因子（factor）和时间序列（ts）等。当然，数值、文本和逻辑数据都可以在 R 中灵活使用；至于算法，我们暂时可以不去过多了解，因为 R 中已经包含了大量设计好的函数，对于一般的用户来讲都不必自行设计算法，除非有特殊或者自定义的算法，那么也可以根据 R 的语法规则编写程序代码。

对 R 的运作方式粗略了解之后，我们再回头看看 GUI。一个软件的菜单、按钮、对话框等 GUI 组件不可能无限增多，否则软件会变得无比庞大臃肿，而繁杂的操作顺序的难度将很可能会超过使用程序命令行的难度。而且，非开源的 GUI 隐藏了计算原理，除了程序的原始编写者，无人知道输出结果究竟是以怎样的方式计算出来。随着统计学的发展，各种新方法、模型必将不断涌现，我们现在不妨试想未来的统计软件用户界面将会变成什么样子。看看 R 的发展，可以体会到这种思路：菜单不可能无限增加，但是程序、函数都是可以无限增加的——道理很简单，因为它们不受计算机屏幕的限制。迄今为止，R 的仓库（repository）中附加包的数量已经超过 1.6 万个，这还只是 CRAN 上的仓库，不算另外几个站点如 Bioconductor、R-Forge、Omegahat 以及 GitHub。在这样的大仓库中，我们可以找到最前沿的统计理论方法的实现，所需做的仅仅就是下载一个附加包，通常只有几十 K 到几百 K。值得注意的是，R 的主安装程序大小只有几十 M。相比之下，SPSS 的安装程序已达六七百 M，而 SAS 的基本安装程序也有数百 M，从程序大小角度便可知 R 语言的精炼。

关于 R 更深入的介绍，请参考官方发行的若干手册和网站上的大量学习材料，如官方的 7 本手册（均可从 R 的安装目录或者官方网站上找到）中，R-intro 手册附录 A 中给了一个很好的代码入门演示，推荐读者通过这个演示初步熟悉 R 语言；其它手册都比较偏重 R 的底层介绍，

第三章　现代实例

> “哦，你还不知道吧？”他笑了，大声说道。“很惭愧，我写了几篇专论，都是技术方面的。比方说，有一篇叫《论各种烟灰的辨别》。此文列举了一百四十种雪茄烟。卷烟、烟斗烟丝的烟灰，并附有彩色插图，以说明烟灰的区别。这是刑事审判中常常出现的重要证据，有时还是案件的重要线索。举例说，如果你断定某一谋杀案系一个抽印度雪茄的男人所为，显然缩小了侦查范围。在训练有素的人看来，印度雪茄的黑灰与‘鸟眼’牌的白灰的区别，正如白菜和土豆的区别一样大。”
>
> ——柯南·道尔《四签名》

本章以一些实际数据案例来说明统计图形的应用，这些数据都尽量取自生活，以保证足够的新颖性；然后，我们利用统计模拟生成的数据来说明统计图形的另类价值，即它在解释统计模型中的独特地位。为了增加阅读的趣味性，我们以探案的形式来介绍统计图形的创建过程，以及应用中可能存在的问题。

首先我们看一个最简单的实例：猪肉价格。

3.1　猪肉价格

案情回放

最近几年，我们都比较关心物价问题，民间也有不少描述物价上涨的方式，比如笔者曾经在邮件中收到一幅“猪肉价格走势图”（图 3.1）。

这幅图非常生动有趣，但莞尔之余我们不妨细看一下这幅图的横坐标——前四根柱子对应的单位是一年，而后面五根柱子的单位却变成了

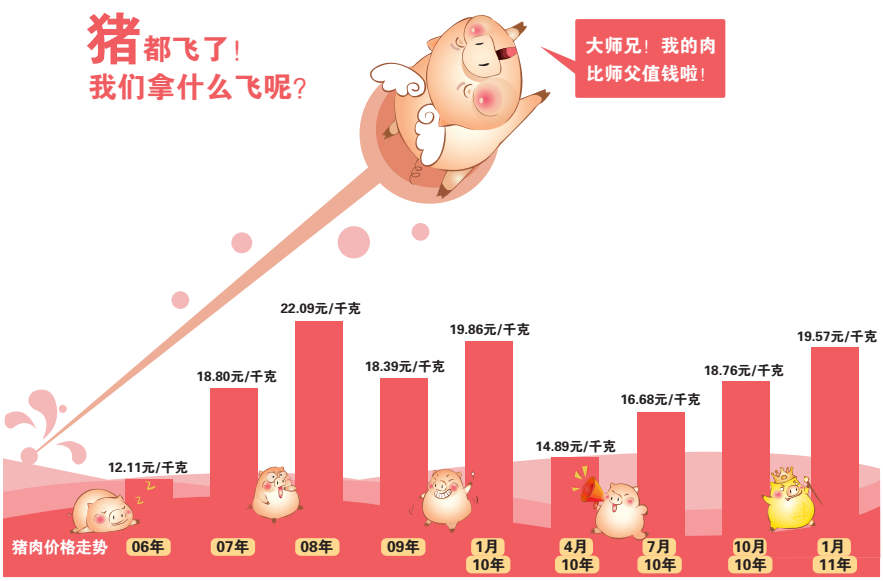


图 *3.1: 自 2006 年以来猪肉价格的走势图*

表 3.1: 2006 到 2011 年猪肉价格

年月	2006	2007	2008	2009	2010-01	2010-04	2010-07	2010-10	2011-01
价格（元/千克）	12.11	18.8	22.09	18.39	19.86	14.89	16.68	18.76	19.57

三个月。虽然图中的小猪作腾飞状，但这幅图本身似乎看不出那么夸张的上涨效果，柱状图整体上比较平淡。这是为什么呢？

探案过程

表 3.1 列出了图中的数据。这是一个一维时间序列数据。我们按照真实的横坐标单位来重画这幅图，会看到另一番景象。

对于时间序列，我们通常会采用折线图来表达数据的波动起伏，观察的核心是数据的大小随着时间的变化，但原图使用条形图实际上也有一定道理。这批数据一共只有 9 个数字，如果只是用折线画图，则整幅图形显得单薄，加上该图的装饰物太多，所以更容易埋没数据。相比之下，矩形条的视觉冲击力较强，更能让读者的注意力集中在数据上。本例中，选取折线图或是条形图取决于具体的应用场景。

我们在图 3.2 中同时给出了两种图形。横坐标被调整到正确的位置之后，2010 年的价格上涨，真的如原图中的小猪一样直飞而上。道理很简单，原本三个月的上涨如果被“稀释”到一年中，那么我们当然不会感觉到强烈的上升趋势。图 3.2 给我们另外一点启示是，条形图容易观察差异大小（比较条的长短），而折线图更容易观察斜率大小（升降速度）。

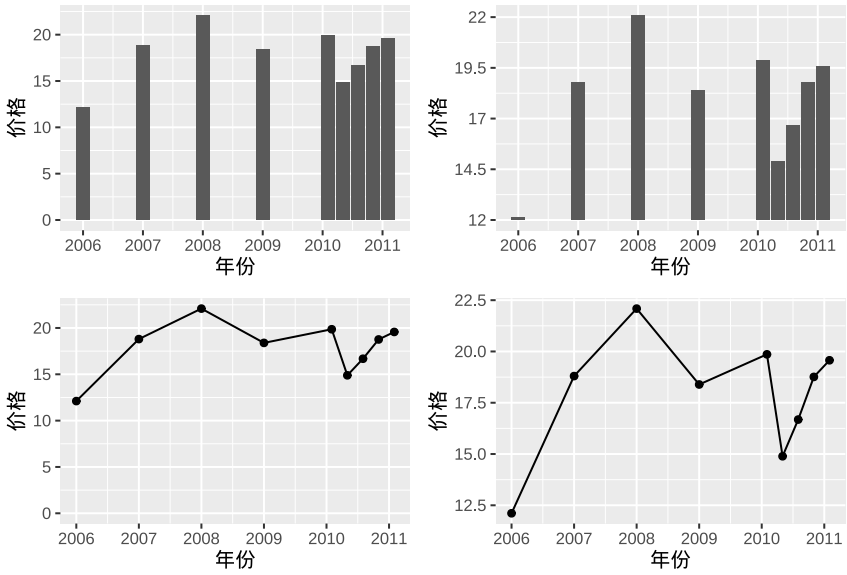


图 3.2: R 语言重新绘制的猪肉价格走势图：上方为条形图，下方为折线图，左侧图形的 y 轴从 0 开始，右侧图形的 y 轴从数据的最小值开始

当我们用 R 重画出图形的时候，我们意识到纵坐标可能也存在问题：原图的纵坐标的零点对应的价格是多少？可以肯定并不是 0，这一点通过简单地选取参照物就能判断出来，例如 06 和 07 年价格条的高度差大约是 6.7，这个高度差与 2010 年 4 月的价格条的高度相接近，这说明零点可能在 8 元/千克左右。这个零点选择似乎非常随意，通常我们画图会以真实的 0 为零点，或者以数据的最小值为零点，很少选取其它位置为零点。关于零点的选取，我们在 8.2.1 小节中会继续解释。这里我们简要说明一下：如果将零点选在真实的 0 上，那么我们更容易计算真实的比率（高度之比）；若选在最小值上，那么更容易看出绝对变化值（因为差异被“放大”了）。

探案工具

本案例使用了条形图（详见 4.1 小节）和折线图（详见 5.1 小节）。图 3.2 可以用 **ggplot2** 包绘制，也可以用 R 基础函数绘制。读者只需用 R 来运行下面的代码即可：

```
# 基础函数绘制更新后的猪肉价格走势图
year = c(2006, 2007, 2008, 2009, 2010 + c(1, 4, 7, 10, 13) / 12)
price = c(12.11, 18.80, 22.09, 18.39,
          19.86, 14.89, 16.68, 18.76, 19.57)
par(mfrow = c(2, 2), mar = c(4, 4, 0.5, 0.5))
plot(year, price, type = "h", lwd = 5, lend = 1,
     ylim = c(0, max(price) + 1), yaxs = "i")
plot(year, price, type = "h", lwd = 5, lend = 1)
plot(year, price, type = "o",
     ylim = c(0, max(price) + 1), yaxs = "i")
plot(year, price, type = "o")
```

```
# ggplot2 绘制更新后的猪肉价格走势图
library(ggplot2)
library(patchwork)
pork_price = data.frame(
  year = c(2006, 2007, 2008, 2009, 2010 + c(1, 4, 7, 10, 13) / 12),
  price = c(12.11, 18.80, 22.09, 18.39,
            19.86, 14.89, 16.68, 18.76, 19.57)
)
p = ggplot(pork_price, mapping = aes(year, price)) +
  labs(x = "年份", y = "价格")
p1 = p + geom_bar(stat = "identity")
p2 = p +
  geom_bar(mapping = aes(year, price - 12), stat = "identity") +
  scale_y_continuous(labels = function(y) y + 12)
p3 = p + geom_line() + geom_point()
p4 = p3 + ylim(c(0, max(pork_price$price)))
print((p1 | p2) / (p4 | p3))
```

3.2 末日狂奔

第四章　单变量图

福尔摩斯接着说："案发后第一时间内，有且仅有一条至关重要的线索摆在面前，可惜你们没能把握住，所以才会觉得我所做的一切莫名其妙。本人有幸抓住了这个线索，之后发生的每件事都证实了最初的假设，完全符合我的推理逻辑。那些令你们疑惑不解的问题，那些令案子扑朔迷离的细节，反而擦亮了我的眼睛，坚定了我的结论。就犯罪案件来说，奇怪不等于难解，两者不能混为一谈。其实，最平淡无奇的案子往往最难解，因为它缺乏新奇和特别之处，很难挖掘到推理的依据。就拿这桩谋杀案来说吧，如果凶手只是简单地抛尸街头，没有制造出那些奇怪的迹象，想要侦破此案恐怕难上加难。正是由于异乎寻常的细节，案子的难度系数不仅没有增加，反而降低了。"

——柯南·道尔《血字的研究》

单变量图，指的是用来展示单一变量的统计图形，最常见的就是直方图和箱线图。当然，我们会经常将多个箱线图、小提琴图、带状图等单变量图放在一起比较，严格来讲其实包含了两个或多个分类变量（如 4.1 小节使用的数据，包含了死亡率、性别和年龄组三个变量），但是为了方便描述，也列在本章。

4.1　条形图/柱状图

概述

如同 1.5 节中曾经提到的，条形图目前在各种统计图形中应用得最为广泛，但条形图所能展示的统计量比较贫乏：它只能以矩形条的长度

表 4.1: 弗吉尼亚州死亡数据

	Rural Male	Rural Female	Urban Male	Urban Female
50-54	11.7	8.7	15.4	8.4
55-59	18.1	11.7	24.3	13.6
60-64	26.9	20.3	37.0	19.3
65-69	41.0	30.9	54.6	35.1
70-74	66.0	54.3	71.1	50.0

Rural Male - 农村男性, Rural Femal - 农村女性, Urban Male - 城市男性, Urban Female - 城市女性

展示原始数值，对数据没有任何概括或推断。

示例

这里以 1940 年弗吉尼亚州分年龄组、分地区和分性别死亡率数据（表 4.1）为例，介绍一下条形图。

图 4.1 展示了各组之间死亡率的差异。其中，堆砌的条形图容易比较各年龄组总死亡率的大小。显然，年龄越高，死亡率越大。而并列的条形图容易比较组内的城乡和性别差异。一般说来，男性死亡率高于女性，农村男性死亡率低于城市男性，但女性的城乡差异没有明显规律。由于人眼对长度比比例更敏感（例如在区分城乡和性别差异时，图 4.1 的上图就不如下图直观），所以我们制图时要考虑清楚我们想展示的是数据的哪一方面，将最关键的信息用最能激发视觉感知的形式表现出来。

绘制方法

R 中条形图的基础作图函数为 `barplot()`，用法如下：

```
## Default S3 method:
barplot(height, width = 1, space = NULL, names.arg = NULL,
    legend.text = NULL, beside = FALSE, horiz = FALSE, density = NULL,
    angle = 45, col = NULL, border = par("fg"), main = NULL,
    sub = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL,
    xpd = TRUE, log = "", axes = TRUE, axisnames = TRUE,
    cex.axis = par("cex.axis"), cex.names = par("cex.axis"),
    inside = TRUE, plot = TRUE, axis.lty = 0, offset = 0, add = FALSE,
    ann = !add && par("ann"), args.legend = NULL, ...)
```

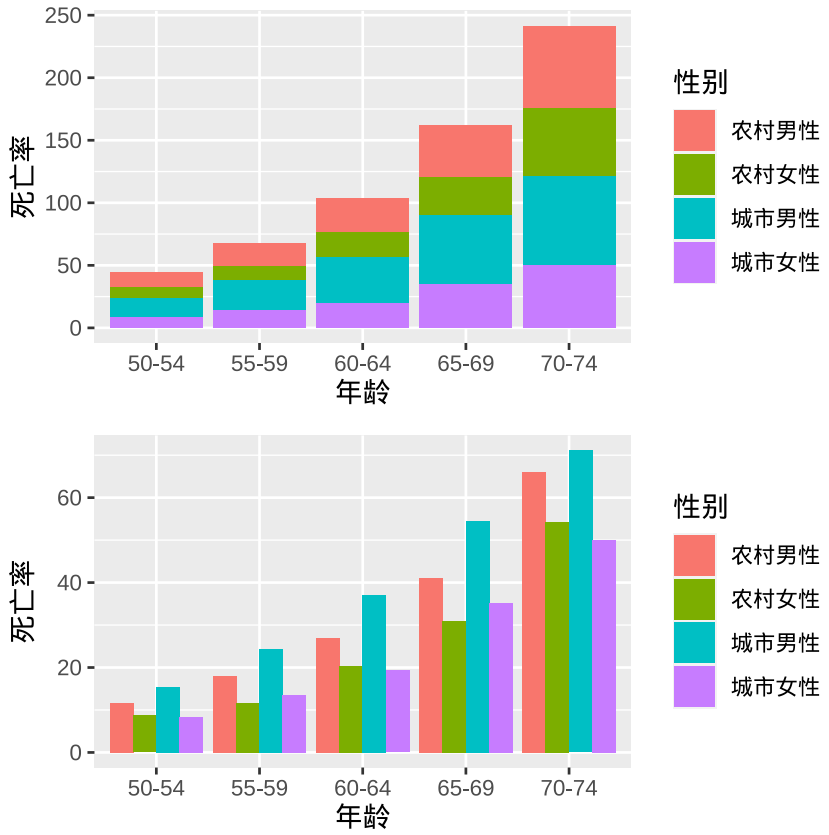


图 *4.1: 弗吉尼亚死亡率数据条形图：堆砌和并列的条形图效果*

参数含义：

height: 主要参数，指定了长条的长度。这个参数可以接受一个数值向量或者一个数值矩阵作为参数。前者容易理解，后者稍有些复杂：当传入一个矩阵时，条形图针对矩阵的每一列画图。若 `beside` 为 `FALSE`，则矩阵每一列占据一条的位置，该条由若干矩形堆砌而成，这些矩形的高度对应着矩阵的行数据；若 `beside` 为 `TRUE`，这些矩形则并排排列而非堆砌。

width: 设置条的宽度。

space: 设置条之间的间距。

names.arg: 条形图的标签，即每一条的名称。

legend.text: 用来添加图例，在 `height` 为矩阵时比较有用。

horiz: 设置条形图的方向（水平或垂直）。
plot: 逻辑值，决定是否将条形图添加到现有图形上。

density、angle 等参数可以参考 9.2.5 小节中关于绘制矩形的介绍。

我们可以用下面的代码绘制弗吉尼亚死亡数据条形图，这里展示了参数 beside 和 legend.text 的效果：

```
# 基础作图法绘制弗吉尼亚死亡率数据条形图
data(VADeaths)
library(RColorBrewer) # 用分类调色板
par(mfrow = c(2, 1), mar = c(3, 2.5, 0.5, 0.1))
death = t(VADeaths)[, 5:1]
barplot(death, col = brewer.pal(4, "Set1"))
barplot(death, col = brewer.pal(4, "Set1"),
        beside = TRUE, legend.text = TRUE)
```

也可以用 **ggplot2** 包绘制，使用的是 geom_col() 函数：

```
# ggplot2 绘制弗吉尼亚死亡率数据条形图
library(ggplot2)
library(patchwork)
data(VADeaths)
reshape_VADeaths = transform(
  expand.grid(sex = colnames(VADeaths), age = rownames(VADeaths)),
  rates = as.vector(t(VADeaths))
)
p = ggplot(data = reshape_VADeaths,
           aes(x = age, y = rates, fill = sex)) +
  labs(x = "年龄", y = "死亡率", fill = "性别") +
  scale_fill_discrete(labels = c("农村男性", "农村女性",
                                 "城市男性", "城市女性"))
p1 = p + geom_col(position = "stack")
p2 = p + geom_col(position = "dodge")
print(p1 / p2)
```

4.2 Cleveland 点图

第五章　双变量图

> “他的职位独一无二，是他自创的，以前没有过，以后也不会有。他头脑清晰，极富条理，储存信息的能力超强，世上无人能及。我们拥有一样杰出的才能，我用来探案，他则用到这份特殊工作上。每个部门的文件都会送到他手上，他就像中心交易点，或者说票据交换所，起到清算账目的作用。其他人都是专家，只有他是全才。假如某位大臣需要了解某个信息，同时涉及海军、印度、加拿大和金银复本位制问题，虽然可以从不同部门分别获取相关意见，但只有麦考夫一人集中了所有信息，立刻就能将一个因素和另一个因素联系起来。最初他只是他们走捷径、图方便的工具，现在俨然成了必需品。他那颗超强大脑分类储存了各种信息，随时可以输出。”
>
> ——柯南·道尔《布鲁斯-帕廷顿设计图》

本章介绍反映两个变量之间关系的统计图形，即双变量图。

双变量图以我们熟悉的 X-Y 散点图为代表，此外还有一些基于普通散点图的变种。如果两个变量中的一个是分类变量，我们常常使用的是条形图、箱线图、点图等类型，在第四章已有介绍，此处不再赘述。而等高图需要基于散点图计算而得到二维核密度矩阵，因此在第七章介绍。

5.1　散点图

概述

散点图通常用来展示两个变量之间的关系，这种关系可能是线性或非线性的。图中每一个点的横纵坐标都分别对应两个变量各自的观测值。因此，散点所反映出来的趋势也就是两个变量之间的关系。

表 5.1: 一组人造的散点图数据（部分。原数据为 20000 行 2 列）

V1	V2
0.889	-1.764
0.072	-0.495
0.123	-0.180
-0.499	0.030
0.252	0.432

示例

我们设计了 2 万个样本，其中有 1 万个样本点来自于两个独立的标准正态分布，另 1 万个样本点的坐标落在半径为 0.5 的圆上，最后将这 2 万个样本拼起来并打乱顺序（表 5.1）。图 5.1 展示了这组人造数据的散点图。

该数据收录在 R 语言的 **MSG** 包中，名为 `BinormCircle`。虽然数据只有两个变量，但我们用普通的统计模型和数值分析几乎无法找出数据的特征，例如，我们在 R 里运行下面的代码来加载数据，并计算线性回归系数和 P 值：

```
data(BinormCircle, package = "MSG")
# 回归系数以及 P 值（不显著）
coef(summary(lm(V2 ~ V1, BinormCircle)))
```

```
##                Estimate  Std. Error    t value  Pr(>|t|)
## (Intercept) -0.006085156 0.005258164 -1.1572776 0.2471728
## V1           0.003988851 0.007015317  0.5685917 0.5696397
```

结果显示两个变量 `V1` 和 `V2` 的回归系数非常不显著。换用高阶回归的结果也类似：无论回归阶数为多少，系数均不显著，这一点从数据的构造上就可以知道（理论上两个变量的相关系数为 0）。

图 5.1 左图给出了这两个变量之间的散点图。由于样本量太大，普通的散点图上点与点之间严重重叠，所以也很难看出散点图有何异常。右图则使用了半透明色，很容易看出，在大量的数据点中，还隐藏着一个圆圈，说明有相当一部分数据分布有特殊规律。

我们在网页（见笔者博客）上给出了其它五种不同的解决方案，都可以从图形的角度反映出这种规律。本章 5.4 小节也将以平滑散点图的方式回顾这批数据。

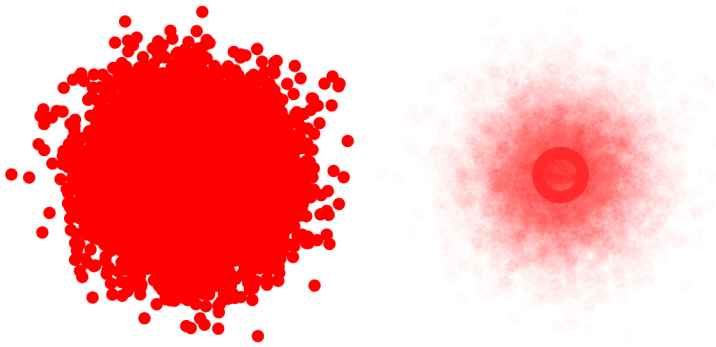


图 5.1: 半透明散点图中的规律：左图是一幅普通的散点图，图中几乎看不出数据有任何异常特征；右图中对点使用了透明度为 0.01 的红色，图中立即显示出一个深色的圆圈，表明该圆圈上集中了大量数据点

绘制方法

R 中散点图的基础绘图函数为 `plot.default()`，但是很少用。由于 `plot()` 是泛型函数（参见 9.1.1 小节），通常我们只需要提供两个数值型向量给 `plot()` 即可画散点图，或者提供一个两列的矩阵或数据框。

函数 `plot.default()` 的用法如下：

```
## Default S3 method:
plot(x, y = NULL, type = "p", xlim = NULL, ylim = NULL, log = "",
    main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
    ann = par("ann"), axes = TRUE, frame.plot = axes,
    panel.first = NULL, panel.last = NULL, asp = NA, xgap.axis = NA,
    ygap.axis = NA, ...)
```

参数含义：若 `x` 是一个两列的矩阵或数据框，则无需再提供 `y`，否则 `x` 和 `y` 都必须是数值型向量；其它参数在 9.1.1 小节中介绍（例如设置 `type = "o"` 就会得到折线图）。

使用基础函数绘制图 5.1 的代码如下：

```
# 基础作图法绘制半透明散点图中
data(BinormCircle, package = "MSG")
par(mfrow = c(1, 2), pch = 20, ann = FALSE, mar = rep(.05, 4))
plot(BinormCircle, col = rgb(1, 0, 0), axes = FALSE)
box()
```

```
plot(BinormCircle, col = rgb(1, 0, 0, alpha = .01), axes = FALSE)
box()
```

ggplot2 包中绘制散点图的函数是 `geom_point()`。下面是绘制图 5.1 的代码：

```
# ggplot2 绘制半透明散点图中
data(BinormCircle, package = "MSG")
library(ggplot2)
library(patchwork)
p = ggplot(BinormCircle, aes(V1, V2)) +
  theme(axis.ticks = element_blank(), axis.text = element_blank(),
        axis.title = element_blank())
p1 = p + geom_point(color = rgb(1,0,0)) + theme_void()
p2 = p + geom_point(color = rgb(1,0,0), alpha = 0.01) + theme_void()
print(p1 | p2)
```

5.2 一元函数曲线图

概述

一元函数曲线图指的是给定函数关系 $y = f(x)$ 的曲线图。这可以通过给 x 赋值后计算 y 值，接着绘制散点图得到。

示例

图 5.2 给出了函数 $f(x) = \sin(\cos(x) \cdot \exp(-x/2))$ 的曲线以及均匀分布 $U(-1, 1)$ 的特征函数曲线作为示例，其中特征函数为概率论中定义的 $\varphi_X(t) = \mathrm{E}[\exp(itX)]$。

绘制方法

R 专门提供了一个函数 `curve()`，可以方便地对任何一元函数作出它在某段定义域上的曲线。我们当然可以用 `plot()` 函数绘制散点图来得到这条曲线，而使用 `curve()` 则可以节省我们去使用低层作图函数（如 `lines()`）的精力和时间。

`curve()` 函数用法如下：

第六章　多变量图

福尔摩斯小心地打开字条，在腿上摊平，用放大镜仔细查看，不漏掉一个微小细节。

“从纸质看，是印度当地生产的纸张，以前被钉在板子上。”他说，“纸上画的是大房子的局部建筑图，有许多厅房、走廊和过道。图上有个小红叉，上方用铅笔写着‘左 3.37’，字迹模糊不清。左角有个奇怪的象形文字，像四个十字架连成一排，旁边写着‘四签名—— 乔纳森·斯莫尔、穆罕默德·辛格、阿卜杜拉·克安、多斯特·阿克巴’，字迹粗糙潦草。坦白说，我也看不出字条和你的案子有什么关系，但肯定是重要资料。字条两面都干净光洁，想必一直保存在皮夹里。”

——柯南·道尔《四签名》

本章介绍展示三个或三个以上变量关系的统计图形。其中，比较常见的是散点图矩阵、条件分割图、三维透视图和地图。其它不常见的图形也略作介绍。此外，矩阵和二维联表在形式上是二维结构，但实际上有三个变量，理应归入本章。但是，考虑到矩阵和联表的特殊形式，我们将在第七章再集中介绍。

6.1　散点图矩阵

概述

散点图矩阵（Scatterplot Matrices）是散点图的高维扩展。它的基本构成是普通散点图（5.1 小节），只是将多个变量的两两散点图以矩阵的形式排列起来，就构成了所谓的散点图矩阵，通常包含 $p \times p$ 个窗格（p

为变量个数）。散点图矩阵从一定程度上克服了在平面上展示高维数据的困难，对于我们查看变量之间的两两关系非常有用。

示例

图 6.1 是对鸢尾花数据（表 5.2）所作的散点图矩阵。除了展示任意两个变量之间的关系外。对角线窗格显示的是直方图，从中我们可以看到四个变量各自的分布情况。上三角窗格中用不同样式的点标记出了鸢尾花的不同类型（参考图 9.8 ）；下三角窗格中简化了点的样式，但是添加了一条平滑曲线，对鸢尾花的四个变量两两之间的关系作出了一种非参数概括（散点图平滑技术，参见 Cleveland (1979)）。

绘制方法

R 语言绘制散点图矩阵的基础函数为 `pairs()` ，其用法如下：

```
## Default S3 method:
pairs(x, labels, panel = points, ..., horInd = 1:nc, verInd = 1:nc,
    lower.panel = panel, upper.panel = panel, diag.panel = NULL,
    text.panel = textPanel, label.pos = 0.5 + has.diag/3,
    line.main = 3, cex.labels = NULL, font.labels = 1,
    row1attop = TRUE, gap = 1, log = "", horOdd = !row1attop,
    verOdd = !row1attop)
```

```
## S3 method for class 'formula'
pairs(formula, data = NULL, ..., subset, na.action = stats::na.pass)
```

散点图矩阵函数是泛型函数，可以直接接受数据矩阵或者公式作为参数。

参数含义：

x: 一个矩阵或数据框，包含了要作散点图的那些变量。

labels: 变量名称（标签）。

panel: 给定一个画散点图的函数，这个函数将应用在每一格图形中。有时候我们并不需要统一的散点图函数，这时可以利用 `lower.panel` 和 `upper.panel` 来分别指定上三角窗格和下三角窗格中的作图函数，也就意味着上三角和下三角窗格中的图形（不一定非得是散点图）可以不一样；`diag.panel` 和 `text.panel` 分别指定对角线窗格上的作图函数和添加文本标签的函数。

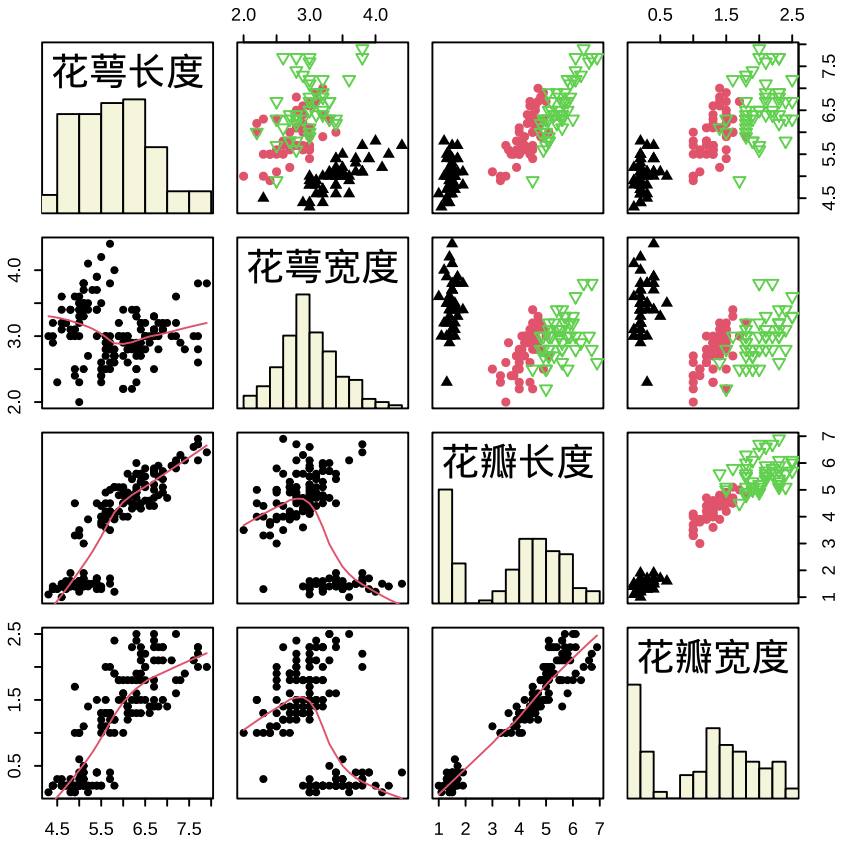


图 *6.1:* 鸢尾花数据的散点图矩阵：上三角区域为不同样式的点，对应着不同种类的鸢尾花，对角线的直方图展示了花瓣花萼长宽的一维分布，下三角区域用平滑曲线显示了变量之间的关系

label.pos: 指定文本标签的位置。

cex.labels: 指定标签的缩放倍数。

font.labels: 指定标签的字体样式。

row1attop: 逻辑值，指定散点图的第 1 行出现在顶部还是底部（按常规讲，前者是矩阵的形式，后者是图的形式，因为矩阵通常是从上至下、从左至右，而图的坐标是从下至上、从左至右）。

gap: 设定窗格之间的间距大小。

图 6.1 的绘制代码如下（注意其中的上三角和下三角作图函数是如何定义的）：

```
# 基础函数作图法绘制鸢尾花数据的散点图矩阵
# 观察如何使用 hist() 做计算并用 rect() 画图
data("iris")
panel.hist = function(x, ...) {
  usr = par("usr")
  on.exit(par(usr))
  par(usr = c(usr[1:2], 0, 1.5))
  h = hist(x, plot = FALSE)
  breaks = h$breaks
  nB = length(breaks)
  y = h$counts / max(h$counts)
  rect(breaks[-nB], 0, breaks[-1], y, col = "beige")
}
idx = as.integer(iris[["Species"]])
names(iris)[1:4] = c("花萼长度", "花萼宽度", "花瓣长度", "花瓣宽度")
pairs(iris[1:4],
      upper.panel = function(x, y, ...)
        points(x, y, pch = c(17, 16, 6)[idx], col = idx),
      pch = 20, oma = c(2, 2, 2, 2),
      lower.panel = panel.smooth, diag.panel = panel.hist
)
```

ggplot2 包所支持的 **GGally** 包给散点图矩阵里添加了更为复杂的信息，下面是一个代码示例。你能看懂该段代码绘制的图 6.2 吗？

```
# ggplot2 绘制鸢尾花数据的散点图矩阵
library(ggplot2)
library(GGally)
data("iris")
names(iris) = c("花萼长度", "花萼宽度", "花瓣长度", "花瓣宽度", "种类")
p = ggpairs(iris, aes_string(colour="种类", alpha=0.5))
print(p)
```

在变量数目较多时，我们不妨将散点图矩阵作为一种探索变量之间相关关系的工具，它比起相关系数矩阵等统计指标来优势在于：散点图矩阵展示了所有原始数据，这样我们可以看到变量之间的任何关系（线性或非线性、离群点），从而避免被单一的统计指标所误导。

6.2 条件分割图

图 *6.2:* 鸢尾花数据的散点图矩阵增强版：散点图、直方图、密度曲线图、箱线图的结合

概述

条件分割图（Conditioning Plot）的思想源自于统计学中的条件分布，即：给定某一个（或几个）变量之后看我们所关心的变量的分布情况。在条件分割图中，这种“分布”主要指的是两个变量之间的关系，通常以散点图表示。

条件分割图可以看作是对散点图的进一步深入发掘，它可以以一个或者两个条件变量作为所有数据的划分条件，条件变量在图形的边缘用灰色矩形条标记出变量的取值范围，每个矩形条对应着一幅散点图（严格来说此时应该称作“条件散点图”），这就是条件分割图的基本做法。

第七章　矩阵图形

> 长长的紫杉小道昏暗无光，两边的树篱修剪整齐，像是两堵高墙，路和树篱间还有两条狭长的草地。小道顶头有座破旧的凉亭，中段有扇偏门通向荒原，正是老人留下烟灰的地方。白色木门上装着门闩，外面是一望无际的荒原。我记得你的推理，试着在现场还原当时的情形。老人站在门旁，看见什么东西从荒原过来，吓得失去理智，撒腿就跑，最后因为极度恐惧和心脏衰竭身亡，逃跑路线正是眼前那条阴森的长隧道。问题是，他在躲什么呢？荒原上的牧羊犬？还是暗如黑夜、悄如幽灵的魔鬼猎犬？有没有人为因素？那个面色苍白的巴里莫尔格外谨慎，会不会隐瞒了什么？一切都模糊不清，背后始终有一层罪恶的阴影。
>
> ——柯南·道尔《巴斯克维尔的猎犬》

矩阵数据在形式上属于二维数据（行 × 列），实际上有三个变量，即每个网格的数值、该网格对应的行数和列数。由于其形式上的特点，我们在本章专门介绍这类数据的统计图形。

二维联表是用矩阵形式来描述统计量的表格。为此，我们把二维联表与矩阵数据在本章一并介绍。等高图的绘制基于散点图，需要经过计算而得到二维核密度矩阵，所以也放在本章介绍。

7.1　等高图/等高线

概述

等高图（Contour Plot）和等高线（Contour Line，或称为等值线）表面上看起来是二维形式，但实际上展示的是三维数据。我们知道，三

维图形往往比二维图形看起来更具有吸引力，然而在平面上展示三维图形也有其缺陷，最主要的就是视角问题：一幅三维图形可以有无数种视角，正视、侧视、俯视可能都会看到不同的信息，而且各种角度下可能都有一部分数据被前面的数据挡住而不能被看到。当然，这些问题都可以通过更灵活的图形设备克服，如 **rgl** 包 (Adler et al., 2020)，但是在更多的情况下，我们的图形都必须展示在静态介质上（如书籍、论文等），我们不可能在纸面上拖动鼠标对图形进行交互式操作。因此，我们需要等高图这样一种以二维形式展示三维数据的工具。

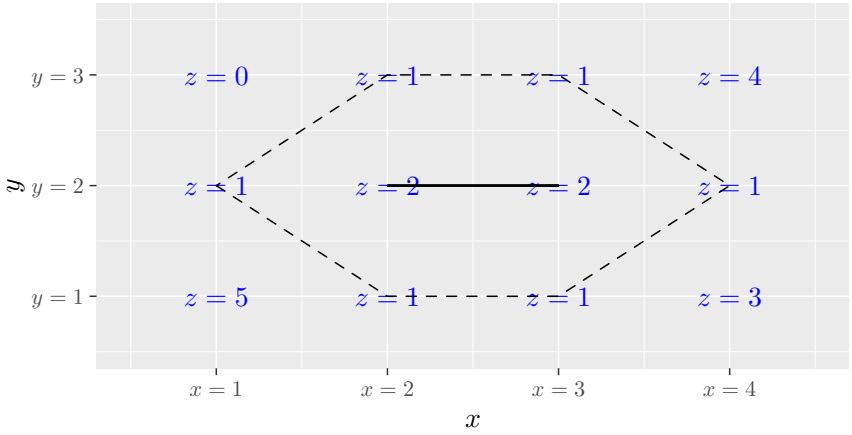


图 7.1: 网格数据的示意图：体会 x、y 与 z 的对应关系

首先，我们需要理解等高图所展示数据的形式，因为它与我们想象的三维数据有所不同：并非三个数值向量，而是两个数值向量 x、y 和一个相应的矩阵 z。我们不妨将数据的形式想象为一座山峰，两个数值向量分别是横向和纵向的位置（如经纬度），第三维数据是每一种横纵向位置点组合上的高度，而横纵交叉组合之后形成的是一个网格，矩阵 z 则是这个网格上的高度数值，用数学式子表示这种关系就是 $z_{ij} = f(x_i, y_j)$。图 7.1 为这种网格数据的示意图，请读者自行体会。

所谓等高线，就是将平面上对应的 z 值（高度）相等的点连接起来形成的线。同样，我们可以以一座山峰来想象：在同一海拔高度上围绕山峰一圈的线就是一条等高线。图 7.1 中的连线即等高线，如实线表示的是高度为 2 的点，而虚线表示高度为 1 的点。注意等高线之间不可能相交，因为同一点不可能同时有两种高度。

等高线上通常会有数字表示高度，从这些数字我们不难想象出三维

表 7.1: 2005 年我国人口预期寿命和高学历人口数量（部分。原数据为 31 行 2 列）

Life.Expectancy	High.Edu.NO
76.10	48001
74.91	18601
72.54	40036
71.65	23197
69.87	23660

Life.Expectancy - 预期寿命, High.Edu.NO - 高学历人口数量

的山峰的形状，从这个意义上来说，等高图本质上也是一种三维图示方法。

示例

表 7.1 是一组我国人口预期寿命和高学历人口数量（定义为大专以上学历人数）的数据，数据来源于 2005 年中国统计年鉴（实际数据来自 2000 年）。首先，我们对这组二维变量利用 R 语言的 **KernSmooth** 包 (Wand, 2020) 进行核密度估计，得到二维核密度值（一个矩阵）。然后，用两个原始变量以及这个密度值矩阵作等高图（图 7.2）。

由于密度值反映的是某个位置上数据的密集程度，图 7.2 所能揭示的现象是：中国 31 省市自治区在人口预期寿命和高学历人口数量上呈现出聚类的特征。图中密度值大的区域主要有中部、右上和左下三个，东中西格局比较明显，即：东部地区分布在图中右上角，中部省市分布在图中中部，西部地区集中在图中的左下角。对照图 6.4 可以知道聚类的具体地区名称，就更能理解这里 “聚类” 的含义了。这批数据更为详细的分析见 6.3 小节。

绘制方法

R 中等高图的函数为 `contour()`，同时 **grDevices** 包中也提供了等高线的计算函数 `contourLines()`，用法分别如下：

```
## Default S3 method:
contour(x = seq(0, 1, length.out = nrow(z)),
```

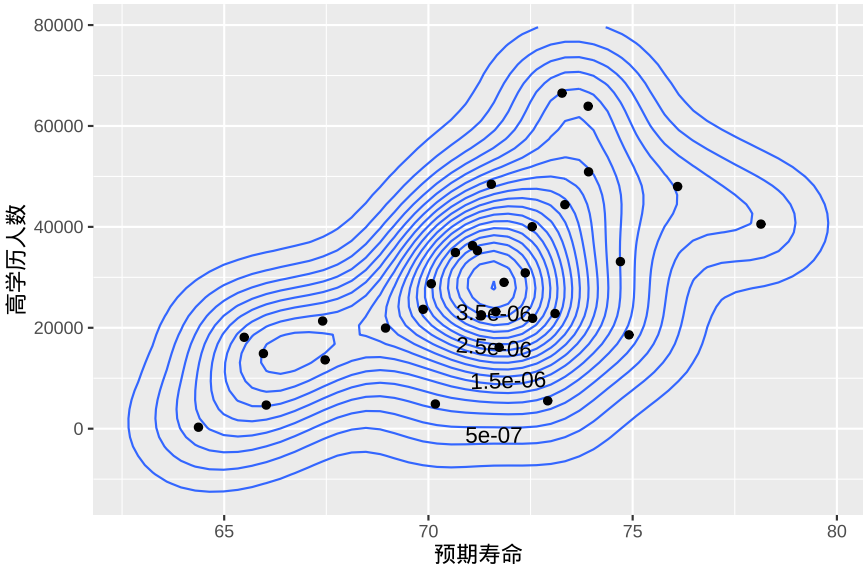


图 7.2: 2005 年中国 31 地区国民预期寿命和高学历人数密度等高图

```
    y = seq(0, 1, length.out = ncol(z)), z, nlevels = 10,
    levels = pretty(zlim, nlevels), labels = NULL,
    xlim = range(x, finite = TRUE), ylim = range(y, finite = TRUE),
    zlim = range(z, finite = TRUE), labcex = 0.6, drawlabels = TRUE,
    method = "flattest", vfont, axes = TRUE, frame.plot = axes,
    col = par("fg"), lty = par("lty"), lwd = par("lwd"), add = FALSE,
    ...)
```

```
contourLines(x = seq(0, 1, length.out = nrow(z)),
    y = seq(0, 1, length.out = ncol(z)), z, nlevels = 10,
    levels = pretty(range(z, na.rm = TRUE), nlevels))
```

参数含义：

nlevels: 设定等高线的条数、调整等高线的疏密。

levels: 设定一系列等高线的 z 值，只有这些值或者这些值附近的点才会被连起来。

labels: 等高线上的标记字符串，默认是高度的数值。

xlim 、ylim 和 zlim: 设定分别设定 x 、y 与 z 的范围，默认从数据中获得。

method: 设定等高线的画法，有三种取值："simple"（在等高线的末端加标签、标签与等高线重叠）、"edge"（在等高线的末端加标签、标签嵌在等高线内）或 "flattest"（在等高线最平缓的地方加标签、嵌在等高线内）。

其它参数用来调整等高图的外观，此处略去不介绍。

contour() 函数用来绘制图 7.2 的代码如下：

```
# 基础作图法绘制中国 31 地区国民预期寿命和高学历人数密度等高图
library(KernSmooth)
data(ChinaLifeEdu, package = "MSG")
par(mar = c(4, 4, 0.2, 0.2))
est = bkde2D(ChinaLifeEdu, apply(ChinaLifeEdu, 2, dpik))
contour(est$x1, est$x2, est$fhat, nlevels = 15, col = "darkgreen",
        vfont = c("sans serif", "plain"),
        xlab = "预期寿命", ylab = "高学历人数")
points(ChinaLifeEdu, pch = 20)
```

也可以用 **ggplot2** 包的 geom_contour() 函数来绘制，并且配合使用 **metR** 包中的 geom_text_contour() 函数来给等高线添加数字：

```
# ggplot2 绘制中国 31 地区国民预期寿命和高学历人数密度等高图
library(KernSmooth)
library(metR)
data(ChinaLifeEdu, package = "MSG")
est = bkde2D(ChinaLifeEdu, apply(ChinaLifeEdu, 2, dpik))
est_tidy = data.frame(
  life = rep(est$x1, length(est$x2)),
  edu = rep(est$x2, each = length(est$x1)),
  z = as.vector(est$fhat)
)
levels = pretty(range(est_tidy$z, finite = TRUE), 15)
p = ggplot(est_tidy, aes(life, edu)) +
  geom_contour(aes(z = z), breaks = levels) +
  geom_text_contour(aes(z = z)) +
  geom_point(aes(Life.Expectancy, High.Edu.NO), data = ChinaLifeEdu) +
  labs(x ="预期寿命", y = "高学历人数")
print(p)
```

我们在绘图前先计算了二维核密度值。实际上，**ggplot2** 包的 geom_density2d() 函数可以更方便地绘制等高线，无需事先计算二维核密度：

表 7.2: 新西兰 *Maunga Whau* 火山高度矩阵数据（部分。原数据为 87 行 61 列）

100	100	101	101	101	101	101	100	100	100	101	101
101	101	102	102	102	102	102	101	101	101	102	102
102	102	103	103	103	103	103	102	102	102	103	103
103	103	104	104	104	104	104	103	103	103	103	104
104	104	105	105	105	105	105	104	104	103	104	104

```
ggplot(ChinaLifeEdu, aes(Life.Expectancy, High.Edu.NO)) +
  geom_density2d() +
  geom_point()
```

在 **graphics** 包中还有一个类似的等高图函数 `filled.contour()`，而 **ggplot2** 包中也有另一个函数 `geom_density_2d_filled()`, 原理完全类似，只是用颜色来区分高度值的大小并且有颜色图例，看起来可能更美观一些，7.2 小节中我们会详细介绍。

7.2 颜色等高图/层次图

概述

颜色等高图，Cleveland (1993) 又称之为层次图（Level Plot），与等高图的原理完全类似，只是颜色等高图用不同颜色表示不同高度，并配有颜色图例，用以说明图中的颜色与高度值的对应关系。读者可以参考 7.1 小节关于等高图的介绍。

示例

我们举两个颜色等高图的例子。

第一个例子是 7.1 小节介绍的我国国民预期寿命和高学历人数密度等高图，可以用颜色等高图来展示，如图 7.3 所示。该图与等高线图 7.2 含义相同，只是改用颜色来表示二维核密度，看上去更直观。

第二个例子，我们使用新西兰 Maunga Whau 火山高度矩阵数据（表 7.2）。这份数据包含了在 10 m × 10 m 的地理网格上测得的火山高度，是一个 87 × 61 的矩阵。

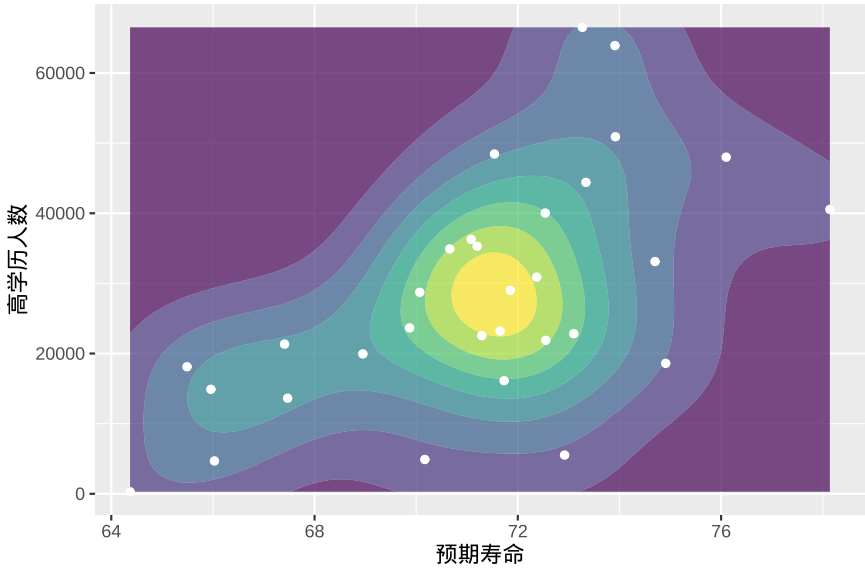


图 7.3: 2005 年中国 31 地区国民预期寿命和高学历人数密度颜色等高图

我们将这个矩阵数据绘制成图 7.4。仔细观察这幅图，由于火山口的存在，颜色等高图的中部（偏左）有一小块区域的颜色并非白色，意即此处的高度比周围一圈要低。这种情况在三维图中有时未必能够迅速看出来，必须将视角调整为略向下俯视才能看到火山口。注意本图的调色板用的是“绿黄棕白”调色板，如 9.2.1 小节所介绍的，这种调色板比较适合展示地理数据。图 7.8 提供了真实的火山立体图形。

绘制方法

R 中的颜色等高图函数为 `filled.contour()`，其用法如下：

```
filled.contour(x = seq(0, 1, length.out = nrow(z)),
    y = seq(0, 1, length.out = ncol(z)), z,
    xlim = range(x, finite = TRUE), ylim = range(y, finite = TRUE),
    zlim = range(z, finite = TRUE), levels = pretty(zlim, nlevels),
    nlevels = 20,
    color.palette = function(n) hcl.colors(n, "YlOrRd", rev = TRUE),
    col = color.palette(length(levels) - 1), plot.title, plot.axes,
    key.title, key.axes, asp = NA, xaxs = "i", yaxs = "i", las = 1,
```

第八章　作图经验

“情况的确如此，我在这方面有直觉。偶尔也会出现一些较复杂的案子，那我就得忙碌一阵子，亲自去查访一番。要知道，我有许多特殊知识，可以用来解开这些谜团，而且能轻易地解决问题。那篇文章中讨论的推理的原则，让你很鄙视，但对我的实际工作却是无价之宝。敏锐的观察力是我的第二天性。我们俩第一次见面时，我说起你是从阿富汗来的，你那时似乎很惊讶哩。”

——柯南·道尔《血字的研究》

在本部分的前几个章节，我们介绍了大量的统计图形。从技术角度来说，作图是一件很容易的事情，但作一幅好的统计图形则并非易事，它需要一些指导原则。什么样的数据适合用什么样的统计图形展示？数据中的信息应以怎样的方式表达？那么图形优劣的评判标准是什么？

最直接的标准就是，读者能否通过图形清楚地了解数据中的信息。这涉及对读者群体的心理学研究和对数据的反复思考。比如，饼图和条形图分别用角度和长度来表达数值大小，那么人眼对角度和长度的感知精度是一样的吗？心理学调查结果显示并非如此：人眼对角度的感知较差。迄今为止，专门做过统计图形方面的心理学研究的统计学家寥寥无几，其中成果最显著的当属 Cleveland (1985)。

本章首先从数据的角度对前文所叙述的统计图形进行简单的梳理，然后基于 Cleveland 的一些观点进行总结与展开，最后针对这些统计图形绘制方法的特点进行总结。

表 8.1: 各种类型的数据对应的统计图形概览

	一维	二维	高维	矩阵
分类数据	条形图	马赛克图、关联图、四瓣图	马赛克图	
连续数据	直方图、箱线图、Cleveland 点图、一维散点图	散点图	平行坐标图、散点图矩阵、三维散点图、三维透视图、平滑散点图、星状图、符号图、脸谱图	颜色图、热图、等高图
混合数据		条件密度图、棘状图	条件分割图	

8.1 图形选择

我们知道，统计数据可以分为离散型和连续型两种。所谓离散数据，又称分类数据（categorical data），就是数据的取值范围只是有限个元素的集合，或是可以一一列举的元素的集合，例如性别、民族、国籍等；所谓连续数据，也就是取值范围为一段连续的区间，例如气温、速度、体重等。如本书第二章所提到的，统计图形要刻画的核心对象是统计分布。对分类数据来说，和分布联系最密切的概念是频数；对连续数据来说，则有很多种可能性，可以是分位数、密度曲线、相关系数等等。

下面，我们对每种数据类型在不同维度下的图形类型做一个简单概括，见表 8.1。注意：该表格并非一个完整的作图指引；面对实际数据时，我们仍然需要具体问题具体分析。

8.1.1 分类数据

对于分类数据，我们关心的往往是每个分类的频数或者比例是多少，这样的问题通常也都很简单，阅读图形仅仅是用眼睛排序的工作。一维分类数据几乎没有别的选择，我们最常用的是条形图（4.1 小节）；多维数据情况下，马赛克图（6.7 小节）能清晰表达列联表中各个单元格的频数大小，同时也能观察表中的边际概率和条件概率。除了描述性质的图形，我们还可以使用具有推断性质的关联图（7.7 小节）和四瓣图（7.8 小节），前者可以让我们清楚看到如果列联表的行列变量不独立，那么哪些单元格对这个“不独立”的结论贡献大；后者可以让我们很快读出列联表的行列变量是否独立，即扇形环是否有重叠部分。

8.1.2 连续数据

相比之下，连续数据的表达方式则要宽广得多：一维情况下，我们可以用直方图和密度曲线（4.3 小节）展示数据的概率分布，用箱线图（4.5 小节）以刻画四分位数的方式展示数据的概要（这是粗略的分布表达方式），用 Cleveland 点图（4.2 小节）表达原始数值的大小，或者用带状图（4.8 小节）同时表达原始数值及其粗略的分布；二维情况下，最常用的是散点图（5.1 小节），通常用来表达两个连续变量之间的线性或非线性关系，而散点图又常常和其它图形元素结合使用，例如回归直线等；三维情况下，我们可以画三维的散点图（3.5 小节）；对于特殊的三维数据我们还可以画三元图（6.6 小节）；更高维情况下，我们有以下选择：

寻找载体 在二维平面上寻找其它维度的“载体”，这些“载体”有很多可能性，但都是用图形元素的某些属性来附着高维数据，例如符号图中的符号长宽高等（6.3 小节）、脸谱图中的脸部特征（6.5 小节）

更改坐标系 笛卡尔坐标系理论上只能放二维变量，因此使用其它坐标系也是扩展到高维的自然选择，例如星状图使用的是星状的坐标系，从中心向外的每个分支都是一个坐标（6.4 小节）；平行坐标图也是常见的表达高维数据的工具，它将垂直的坐标系改为平行的，而平面上理论上可以容纳无穷多根平行线，所以平行坐标图理论上也可以放置任意多的变量（6.11 小节）

重复二维图形 二维的重复也可以达到表达高维数据的目的，例如散点图矩阵就是对所有变量两两重复画散点图，这样所有变量组合的散点图都可以表达在平面上了（6.1 小节）

降维 把高维数据降为二维数据也不失为一种办法，例如对数据做主成分分析，然后仅仅画前两个成分的散点图

8.1.3 混合数据

专门针对混合数据的图形并不多，前面介绍过的条件密度图（5.7 小节）和棘状图（5.8 小节）是两个少见的例子。在绝大多数情况下，我们都推荐使用“条件分割”的办法，利用分类变量的各个取值水平分别画二维图形，这样让我们很容易比较分类变量不同取值水平下的二维变量之间关系的差别。基础图形系统中的条件分割图（6.2 小节）只是一个引子，**ggplot2** 包中的切片功能更灵活易用，图 4.3 便是一例。

一定程度上，这种针对数据类型总结图形类型的做法有些死板，例如两个分类变量一般情况下是无法画散点图的，因为分类变量只取有限的几个值，所以两个分类变量之间的散点图通常只是若干个网格点，而这些点本身并不能反映出该位置上真正的频数。我们在本书第二部分提到过一些分类变量的图示方法，包括关联图（7.7 节）、四瓣图（7.8 节）和马赛克图（6.7 节）等，不过它们都不是最直接的散点图，而是将频数表达在其它图形元素中。那么分类变量是否一定不能画散点图呢？当然不是，向日葵散点图和随机打散的散点图就是两种可能。

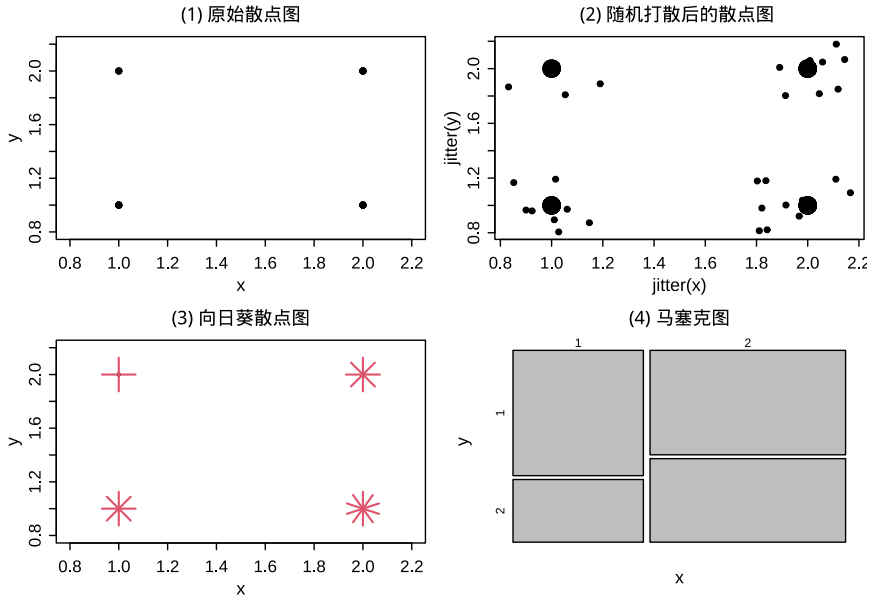


图 *8.1:* 分类变量的散点图示方法示例：原始散点图、打散方法、向日葵散点图和马赛克图

关于向日葵散点图，在 5.3 小节中已经有详细介绍，这里我们再次强调一下它在展示分类变量散点图上的功效。如 5.3 小节中讲到的，向日葵散点图用向日葵的花瓣表示该处有多少个重复的数据点，而分类变量的散点图大多数情况下都会有重叠的数据点，因此分类变量尤其适合用向日葵散点图来表示。图 8.1(3) 给出了一个用向日葵散点图表示分类变量的示例。

由于分类变量散点图的关键问题是重叠问题，因此我们不妨将重叠的数据稍微“打散”一些，然后再作散点图。关于打散方法，我们曾经在 4.8 小节中用到过，即 `jitter()` 函数。注意打散过的散点图不能严格按

照点的坐标来解读，而是应该按聚集在一处的点的数目来解读频数。图 8.1(2) 给出了一个打散之后的分类变量散点图示例。当然，最自然的选择可能还是马赛克图。我们画图不必墨守成规，按照需要去考虑应该使用的图形类型即可。

8.2 作图原则

8.2.1 数据至上

数据是宝贵的，它们也许来自艰辛的问卷调查，或是繁琐的实验测量，因此我们应该尽量珍惜，但现实状况是我们经常有意或无意糟蹋数据，这样的情形包括：表达数据的元素被次要图形元素遮挡，数据的特征无法在图中凸显，或者数据经过了不恰当的人工处理等。

分清主次

对于一幅图形而言，显然并非所有的图形元素都同等重要。例如，散点图中的点应该是最重要的元素，等高线图中的线更重要，等等。因此，我们不能让次要的图形元素干涉数据的表达，要让图形显得干净、清晰。主要元素的外观要仔细选择，使数据在图中占有最重要的视觉地位，而不会被标签等元素遮挡或干涉。用 Cleveland 的话说，就是要让数据“站出来”（stand out）。

R 基础图形系统中点的默认的样式是空心点。在很多情况下，这并不是一个好的选择，因为空心点在图中看起来太不起眼，尤其是数据点较少的时候。我们延续 3.4 小节中的美剧演员收入数据，在这里讨论如何突出主要元素。

图 8.2 是演员收入与电视剧评分的散点图，左图使用了默认的空心点，右图使用了实心点。这批数据的样本量只有 72，左图中的点所用的墨水可能和坐标轴等次要元素差不多，所以数据在图中也显得不够突出，而实心点则很明显占据了一幅图的视觉重心。

这两幅图实际上有个共同的问题，就是图中空白区域太大。这是由最高收入的那位演员引起的。这种情况也从一定程度上降低了图形元素的表达效率，因为我们放眼望去，一幅图的一半区域都是空白，绝大部分点都集中在图的下半部分。当然，在这个问题上我们也不能绝对化，因为有时候这种大片空白能反衬处离群点——取决于我们要显示的重点是什么。

第九章　R 基础作图系统

他从信封里拿出一页四折叠的半张 13×17 英寸的信纸。他把信纸打开铺在桌上，中间有一行铅字拼贴成的句子：

若你看重自己生命的价值或还有理性，那就远离沼地。

只有“沼地”两个字是用墨水写的。

“现在，”亨利·巴斯克维尔爵士说，“福尔摩斯先生，也许您可以告诉我，这是什么意思，到底是谁对我的事这么感兴趣呢？”

……“可是，亲爱的华生，两者之间的联系非常紧密，短信中的每个单字都是从这个长句中抽出来的。例如：‘你’、‘你的’、‘生’、‘命’、‘理性’、‘价值’、‘远离’等，你现在还看不出这些字是从那里面弄来的吗？”

“天啊！太对了！唉呀，您可聪明绝顶！”亨利爵士喊了起来。

——柯南·道尔《巴斯克维尔的猎犬》

R 语言绘制的统计图形都是通过相应的图形函数生成的。很多图形函数都包含了默认的图形细节设置，这些细节对不太苛刻的用户来说大致可以满足需要，但是往往由于其它方面的要求（如排版、强调某一部分），我们可能要对图形作一些细节性的微调，比如字体、字号、图形边距、点线样式等等。本章介绍 R 语言基础图形系统中的细节参数设置。

另外，我们也在这里介绍一些统计作图上的技巧。这些技巧对于数据分析来说也许没有显著的作用，但它们可以帮我们进一步调整、组织好我们的图形输出。这些内容包括：数学公式的表示、一页多图的方法、离散变量的散点图示和各种图形设备的使用方法。

9.1 作图函数

在本书第二部分，我们介绍了很多 R 基础作图函数，如条形图函数 `barplot()`（4.1 小节）、直方图函数 `hist()`（4.3 小节）、箱线图函数 `boxplot()`（4.5 小节）、向日葵散点图函数 `sunflowerplot()`（6.1 小节）、等高图函数 `contour()`（7.1 小节）、颜色图函数 `image()`（7.3 小节）等。

R 中最普通的作图函数是 `plot()` 函数。R 的很多图形参数可以用 `par()` 函数来查看和设置。本节重点讲述一下这两个函数的用法，相信读者可以举一反三，在深入了解这两个函数之后，能够理解其它作图函数的用法规律。

9.1.1 `plot()` 函数

`plot()` 是一个泛型函数，可以接受很多不同类的对象作为它的作图对象参数。我们这里要解释的只是其中的图形参数，而非作图对象参数。

`plot()` 的通用参数有：

`type` 图形样式类型，有九种可能的取值，分别代表不同的样式：

- `"p"`⇒ 画点；
- `"l"`⇒ 画线[1]；
- `"b"`⇒ 同时画点和线，但点线不相交；
- `"c"`⇒ 将 `type = "b"` 中的点去掉，只剩下相应的线条部分；
- `"o"`⇒ 同时画点和线，且相互重叠，这是它与 `type = "b"` 的区别；
- `"h"`⇒ 画铅垂线；
- `"s"`⇒ 画阶梯线，从一点到下一点时，先画水平线，再画垂直线；
- `"S"`⇒ 也是画阶梯线，但从一点到下一点是先画垂直线，再画水平线；
- `"n"`⇒ 作一幅空图，没有任何内容，但坐标轴、标题等其它元素都照样显示（除非用别的设置特意隐藏了）。

图 9.1 的九幅图清楚说明了这九种类型。笔者开发的 **beginr** (Zhao, 2019) 包提供了函数 `plottype()`，可以方便地查询这九种图形样式，安装后运行 `beginr::plottype()` 即可。

[1]注意是字母 `l`（表示 “line”），不是数字 `1`！同样后面的字母 `o`（表示 “overplotted”）也不要误认为是数字 0

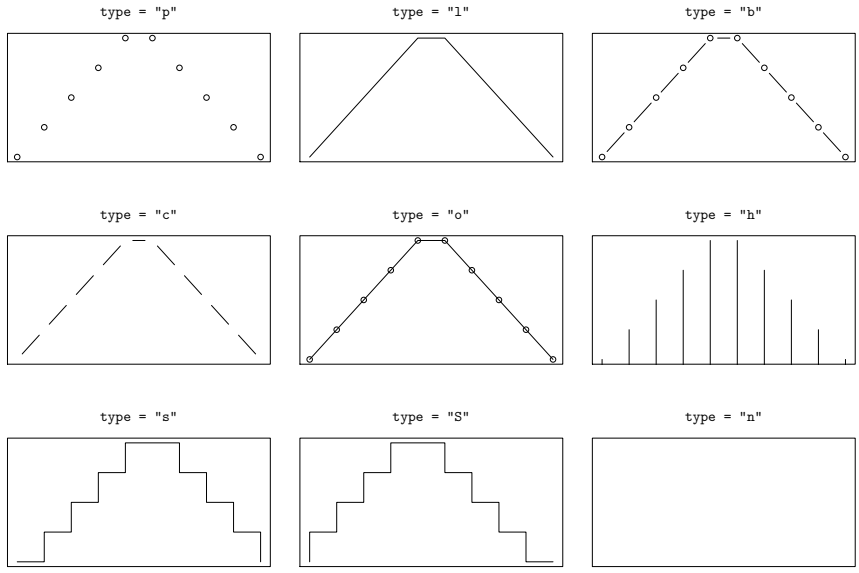


图 *9.1: plot()* 作图的九种样式类型：点、线、点线（不相接）、擦掉点的线、点线（相接）、垂线、阶梯（水平起步）、阶梯（垂直起步）、无

main、sub、xlab、ylab: 主标题、副标题、x 轴标题、y 轴标题。也可以在作图之后用函数 `title()` 添加，参见 9.2.7 节。

asp 图形纵横比，即 y 轴上的 1 单位长度和 x 轴上 1 单位长度的比率。通常情况下，这个比率不是 1。有些情况下需要设置以显示更好的图形效果，例如需要从角度表现直线的斜率。若 `asp` 不等于 1，那么 45° 的角可能看起来并不像真实的 45°。

然后，我们看看默认的散点图函数 `plot.default()`。对于一般的散点图（两个数值变量之间），我们只需要调用 `plot()` 即可，如 `plot(x, y)`，而不必写明 `plot.default(x, y)`，原因就是 `plot()` 是泛型函数，它会自动判断传给它的数据类型从而采取不同的作图方式。`plot.default()` 的参数当然包含了前面介绍的 `plot()` 中那些参数，此外还有：

x, y 欲作散点图的两个向量。如果 y 缺失，那么就用 x 对它的元素位置（`1:n` 的整数）作散点图。

xlim, ylim 设置坐标系的界限，两个参数都取长度为 2 的向量。它们的作用类似 `par()` 中的 `usr` 参数，但我们可以通过 `par()$usr` 获得一幅图的坐标系界限，而这里的两个参数就没有这个功能了，因

第十章　ggplot2 系统

> “我不想用各种说法和怀疑来影响你，华生，”他说，“我只要求你将各种事实尽可能详尽地报告给我，至于归纳推理就留给我好了。”
>
> “哪些事实呢？”我问道。
>
> “与该案可能有关的任何事实，无论是多么地间接，特别是年轻的巴斯克维尔与邻里的关系或与查尔兹爵士暴卒有关的任何新的问题。”
>
> ——柯南 · 道尔《巴斯克维尔的猎犬》

基础图形系统虽然灵活，但它无穷无尽的选项往往让用户感到迷茫，**lattice** 系统（见 11.2 小节）也有同样的问题（甚至更严重）。**ggplot2** 包 (Wickham et al., 2020) 从易用性出发，结合了基础图形系统的简便以及 **grid** 和 **lattice** 的灵活，并以 *The Grammar of Graphics* 一书 (Wilkinson, 2005) 的理论为支撑，已成为当前最流行的作图系统。

10.1　快速体验

ggplot2 包虽是 R 语言的附加包，但其作图函数的逻辑和语法结构与 R 基础作图函数迥异。R 基础函数用户在初次接触 **ggplot2** 时往往难以适应，而从 **ggplot2** 开始学习 R 语言的用户同样不习惯于基础作图函数。为此，**ggplot2** 包特意提供了一个快速体验函数 `qplot()`，既兼容基础作图函数的语法，又让我们快速画出 **ggplot2** 风格的图形。

`qplot()` 的用法如下：

```
qplot(x, y, ..., data, facets = NULL, margins = FALSE, geom = "auto",
    xlim = c(NA, NA), ylim = c(NA, NA), log = "", main = NULL,
    xlab = NULL, ylab = NULL, asp = NA, stat = NULL, position = NULL)
```

参数含义：

x、y: 分别是要画图的变量。
data: 提供一个数据框（在这里面寻找前面提到的变量）。
geom: 默认值为“自动”，它会根据提供的数据类型自动生成合适的图形。例如，如果我们提供的 x 和 y 是数值型变量，那么画出来的就是散点图；如果只提供数值型的 x，那么就画直方图；如果提供离散型的 x，那么就按照各分类的频数画条形图。
stat: 指定对数据做的统计变换。
facets: 以某些分类变量对数据切分后分别对小块数据作图。

其它参数都是一些细节调整，可以根据具体任务设定。

在下面的代码中，我们使用 qplot() 函数来绘制图 4.4 中的直方图和图 5.1 中的散点图，得到图 10.1：

```
# `qplot()`函数作图示例
library(ggplot2)
library(patchwork)
data(geyser, package = "MASS")
data(BinormCircle, package = "MSG")
print(qplot(x = waiting, data = geyser, xlab = "间隔时间") |
        qplot(x = V1, y = V2, data = BinormCircle, alpha = I(0.01)))
```

虽然 qplot() 使用起来方便快捷，但是我们并不推荐，原因正如前面所述，它并不符合 **ggplot2** 作图的习惯，仅供体验，不宜过度依赖。读者若想深入了解 **ggplot2** 作图的魅力，最好还是从它的基本框架入手。

10.2 基本框架

ggplot2 的核心概念是“层”：所有的图形都是由层的“叠加”构成。“叠加”这个操作在程序实现上很巧妙：它扩展了泛型函数 +。也许用户会对此感到迷茫，后面我们看了具体例子马上就能明白。

ggplot2 作图的基本语法框架是：

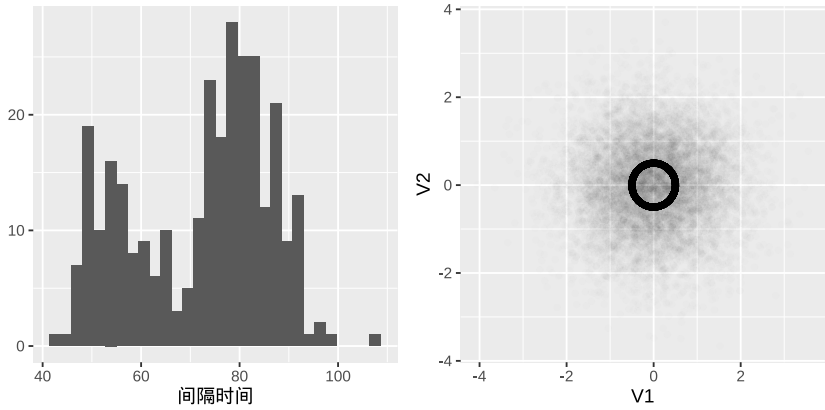


图 10.1: `qplot()` 函数作图示例：与图 4.4 和 5.1 相比，作图所用的函数有哪些区别？

```
# 必选：
ggplot() + geom_xxx() +
# 可选：
  scale_xxx() +
  coordinate_xxx() +
  facet_xxx() +
  theme() +
  ...
```

即：数据映射 + 几何形状 + 可选项（即坐标系 + 切片结构 + 主题 + 其它附加属性）。每个 + 两侧都是图层。

对于这个框架，我们在第二部分给出了很多实例。读者也可以参考第二章的结尾用 **ggplot2** 绘制的拿破仑远征图，这个例子很好地展示了 **ggplot2** 图层叠加的作图特点。

加号 + 本身实际上是一个函数。我们平时看到的加法可以写成函数调用的形式：

```
1 + 2 # 我们看到的加法
`+`(1, 2) # 加号作为一个函数使用
methods("+") # 加号上的 S3 方法
```

在 **ggplot2** 中，画图只需要将若干个图层简单相加即可，语法非常

精炼，如：

```
p = ggplot(aes(x = hp, y = mpg), data = mtcars) +
  geom_point() # 先画一个散点图的图层
p + geom_smooth(method = "loess") # 用散点图加上平滑层
```

使用 **ggplot2** 时，通常我们不必担心细节问题，例如图形的边距会自动调整，不会留出大片空白，元素颜色会自动根据变量取值从调色板中选取，图例会自动添加，等等。这些自动化的设计可以为我们节省大量的调整细节的时间。相比之下，我们使用基础图形常常需要缩小边距（默认值太大）、手工添加图例，显得非常麻烦。

10.3 构成元素

整个 **ggplot2** 系统大致由几何形状（geom）、统计量（statistic）、标度（scale）、坐标系（coordinate system）和切片（facet，或称为“分面”）构成，下面我们分别作介绍。

10.3.1 几何形状

在 **ggplot2** 中，几何形状简称 geom （Geometric objects），这些形状包括：

- 点（`geom_point()`）、
- 条（`geom_bar()`、`geom_col()`）、
- 线（`geom_line()`、`geom_abline()`）、
- 箱线图（`geom_boxplot()`）、
- 文本（`geom_text()`、`geom_label()`）

等等。实际上，它们就是第九章介绍的图形元素，但是 **ggplot2** 在这些元素上做了更多工作。例如，箱线图并非基础图形元素，但它在 **ggplot2** 中的地位也是基础形状；还有平滑曲线和平滑带，背后都涉及大量的统计计算。而 **ggplot2** 对它打包之后，用户用起来就简单多了，否则我们需要手工建立平滑模型（可能是线性回归，可能是 LOWESS，或是分位数回归等），然后取一系列 x 值并拟合 y 值。

几何形状通常和统计量紧密相关。因为要画出几何形状，必须计算一些坐标值（例如箱线图中的分位数、条形图的高度），所以我们必须要知道对数据做什么样的统计变换或汇总。

图 10.2 展示了表 6.3 的汽车数据。从图中我们可以看到，随着汽车马力增大，每加仑汽油能行驶的英里数在降低，但并非直线下降，而是在逐渐变缓。这也符合常识——不可能有哪种汽车的马力大到无法开动的程度。图中的平滑曲线基于 LOWESS 生成，它是散点图的重要辅助工具，在 3.7 小节和 3.8 小节有详细介绍。

```
# 汽车马力与每加仑汽油行驶里程的关系
library(ggplot2)
p = ggplot(aes(x = hp, y = mpg), data = mtcars) +
  geom_point() +
  geom_smooth(method = "loess") +
  labs(x = "马力", y = "每加仑汽油行驶里程")
print(p)
```

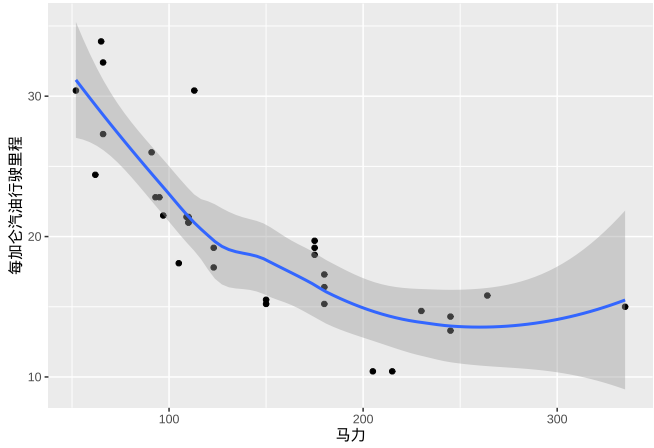


图 *10.2:* 汽车马力与每加仑汽油行驶里程的关系：随着马力增大，汽车油耗也变大，但这个关系并不是线性的

10.3.2 统计量

统计量指定了对原始数据做何种变换，进而用几何形状表达出来。**ggplot2** 中除了划分直方图区间求频数、求分位数、计算密度值这些普通的变换功能之外，还有一些新颖的统计量。例如，根据二维数据用网格划分区间，求每个格子内的数据频数（实际上就是二维直方图）；或者用蜂巢形状将平面划分为一系列的六边形区间，再求数据频数。

图 10.3 是 **ggplot2** 包中 diamonds 数据的一幅蜂巢图，它展示了在每个蜂巢格子里的数据频数大小。这种图形和平滑散点图的思想类似，都是要展示二维数据的密度，只不过实现方法不同而已。蜂巢图的背后是散点图，但具体的点都没有显示出来，我们看到的只有蜂巢及其颜色。从图例中可以看出，浅蓝色表示该单元格内的数据频数为 7000 左右，深蓝色表示 1000。该数据内大多数钻石的价格（纵轴）和重量（横轴为克拉数）都偏小。另外，我们也很容易看出，随着克拉数增大，价格也相应升高，这也是符合常识的，但有些 3 克拉的钻石价格和 0.5 克拉的一样，这可能是因为打磨质量的问题。该图的绘制代码如下：

```
# 钻石重量与价格的蜂巢图
library(ggplot2)
p = ggplot(aes(x = carat, y = price), data = diamonds) +
  geom_hex() + labs(x = "重量", y = "价格", fill = "频数")
print(p)
```

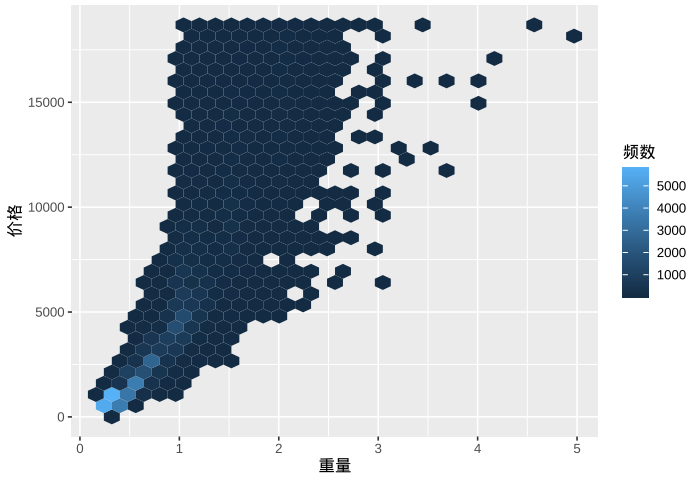


图 *10.3: 钻石重量与价格的蜂巢图：大多数钻石的重量和价格都偏小*

当散点图中的点的数目非常大时，蜂巢图既能保持散点图中两个变量的关系，又能提供对数据密度的概括。蜂巢图的计算基于 **hexbin** 包 (Carr et al., 2020)，这个包自身也可以画蜂巢图，**ggplot2** 包只是调用其中的函数完成蜂巢的计算。

统计量相关函数通常会生成一些新的变量，这些变量可以用来手工构造图形。例如，计算密度的 stat_density() 函数会生成一个 density

第十一章　其它作图系统

他解释说："你要明白，我认为人的大脑原本像一间空空的屋子，必须有选择地用一些家具填满它。只有笨蛋才把他碰到的各种各样的破烂都塞进去。这样的话，那些可能用得上的知识就被挤了出来；或者，充其量也只是把那些破烂同其它东西混杂在一块儿。结果，在需要时却难得找到了。因此，一个善于工作的人，对于将什么东西纳入自己的头脑里是非常仔细的。他只会容纳那些工作时用得着的工具，而且又将这些工具分门别类，安排得井然有序。如果认为这间屋子的墙壁富有弹性，可以随意扩展，那就大错特错了。毫无疑问，总有一天，当你增加点滴知识时，却把从前熟悉的知识给忘记了。因此，不要让无用的信息挤掉那些有用的信息，这一点是至关重要的。"

——柯南·道尔《血字的研究》

除了基础图形系统之外，R 还自带另一套图形系统即 **grid**（网格图形）。网格图形系统是一套基础设施性质的图形系统，它本身不包含统计图形，只是提供了一些画图形元素的工具。在此基础上诞生了 **lattice** 图形系统和 **ggplot2** 图形系统（见第十章）。其中，**lattice** 包已经随 R 本身发布，**ggplot2** 包目前的地位虽然只是一个附加包，但由于它的灵活和美观，用户数量与日俱增，当前已成为最流行的作图系统。因此，我们对 **lattice** 系统仅作粗略介绍。

除了这些静态图形系统之外，R 还有一些附加包支持动态图形和交互式图形，即：用户可以用鼠标或键盘和图形进行交互，比如用鼠标选取图中的点并高亮，或者拖拉旋转图形；代表性的附加包有 **ploty**、**iplots**、**rgl** 和 **playwith** 等。另外我们还可以利用 R 包 **animation** 生成动画。这些附加包极大增强了统计图形的探索功能和趣味性。

11.1 网格图形系统

网格图形（grid graphics）是与基础图形（base graphics）相平行的一套图形系统，在 R 中由 **grid** 包实现。相对于基础图形而言，网格图形具有更高的可控度和灵活性，但其本身只提供了非常底层的绘图命令，如果要绘制较复杂的图形，则可以考虑使用第 11.2 节介绍的 **lattice** 和第十章介绍的 **ggplot2** 图形系统。这两个图形系统都是基于网格图形构建的。

和基础图形系统相比，网格图形有三个非常突出的特性：对绘图区域的灵活控制，支持图形的旋转，以及图形元素的动态编辑。下面本节就从这三个方面对网格图形进行简要的介绍。需要指出的是，由于 **grid** 包主要的作用是提供底层的绘图支持，因此本节不会涉及具体的绘图细节，而是着重介绍其特性和优势所在。这将有助于我们理解 **ggplot2** 包和 **lattice** 包的运行原理。

11.1.1 视图区

在网格图形系统中，最为重要的一个概念即是所谓的“视图区”（Viewport）。一个视图区也就是一个绘图区域。在网格图形中，可以创建任意多个视图区，每个视图区都可以有自己的一套坐标体系和绘图参数，这是与基础图形系统最大的不同。打个比方来说，基础图形系统就是在一张桌子上放置了一张白纸，然后在这唯一的一张白纸上绘图；而在网格图形系统中，你可以拥有任意数量的白纸并在每一张上进行绘图，然后将图纸按一定的层次堆叠好，作为最终的图形展现。

采用这种设计的一个直接好处是可以在已有的一张图形中嵌入子图，如图 11.1 所示。这张完整的图形实际上是由两个视图区堆叠而成的：第一个视图区包含了除虚线框区域之外的坐标轴、散点图和曲线等图形元素；第二个视图区就是图中带虚线框的区域，它将第一个视图区的图形元素进行了复制并放大，然后放置在第一个视图区之上。

在 **grid** 包中，创建视图区的命令是 `viewport()`，函数的参数包括视图区的位置、大小、度量单位、坐标尺度等信息。需要注意的是，`viewport()` 函数只是创建了一个视图区对象，并没有把它应用到图形设备上；如果要完成这一过程，则需要使用 `popViewport()` 函数。此外，**grid** 包还提供了 `upViewport()` 和 `downViewport()` 等函数来对视图区进行进一步的操作，这些函数可以参考 **grid** 包自带的帮助文档来学习使

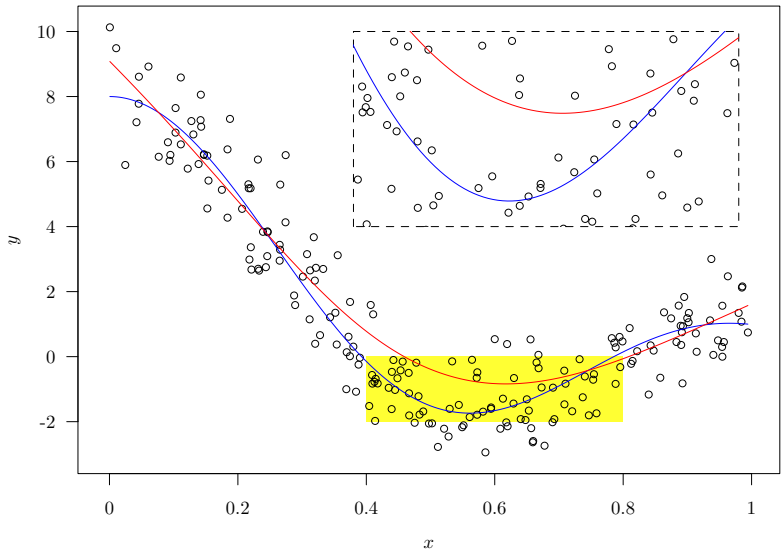


图 11.1: 用 ***grid*** 包实现的图形局部放大效果：将大图的局部区域放大显示在空白处

用。

11.1.2 旋转

图形的旋转是网格图形的一大特色。**grid** 包自带的帮助文档中给出了一个简单的例子，如图 11.2 所示。

该图的绘图代码如下：

```
# grid 系统中旋转的视图区
library(grid)
pushViewport(viewport(h = 0.8, w = 0.8, angle = 15))
grid.multipanel(newpage = FALSE)
```

从程序代码中可以看出，控制图形旋转的实际上就是创建视图区时 `viewport()` 函数的 `angle` 参数，其数值表示图形逆时针旋转的角度。因此，在网格图形系统中旋转图形元素是非常方便的，只需先创建一个新的视图区，并在其中指定 `angle` 参数，然后在这个视图区中绘制图形即可。

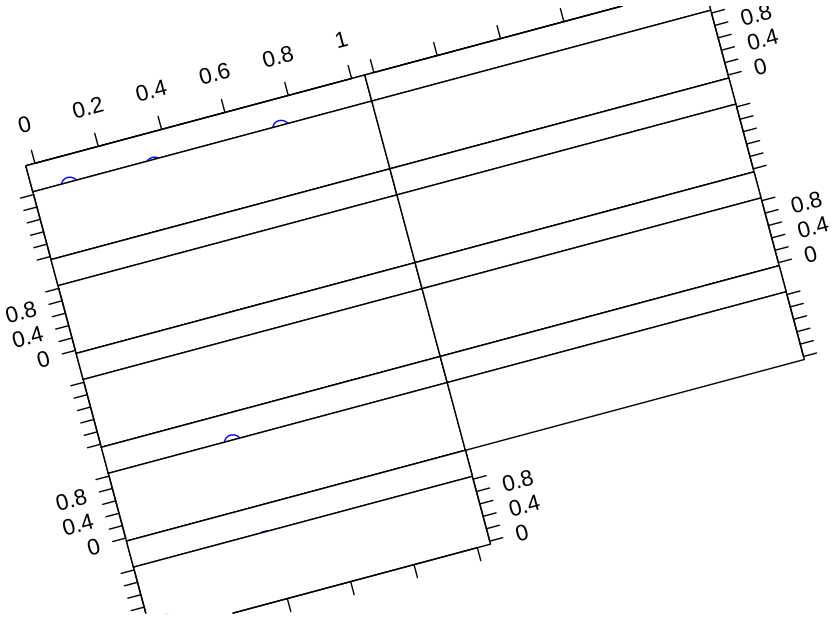


图 *11.2:* ***grid*** 系统中旋转的视图区：倾斜 15° 的图形

图 11.3 是对网格图形旋转功能的一个应用，其中涉及了主成分分析的相关知识。图中首先绘制了两个变量之间的散点图，然后通过主成分分析找到了这两个主成分的方向，在图中是用红色的坐标轴表示的。散点图中的每一个点在红色轴线上的坐标就是它在两个主成分上的得分。

根据主成分分析的相关理论，我们知道主成分实际上就是原始变量的一个线性组合，而且每个主成分彼此之间是相互正交的，这反映在图形中就是一套经过平移和旋转的坐标系，其中坐标系的原点定位在原始变量的重心，而 x 轴则是原始变量方差最大的那个方向。

11.1.3 图形对象

图形对象的动态编辑是网格图形系统的另一大优势。在基础图形系统中，图形元素一旦被绘制到设备上，就无法再对其进行编辑，除非采取覆盖的办法将图形重新绘制一遍。而在网格图形系统中，图形元素会以对象的形式保存在内存中，在需要的时候可以对其进行更改，例如修改颜色、大小等。

下面的例子展示了编辑图形对象的方法：

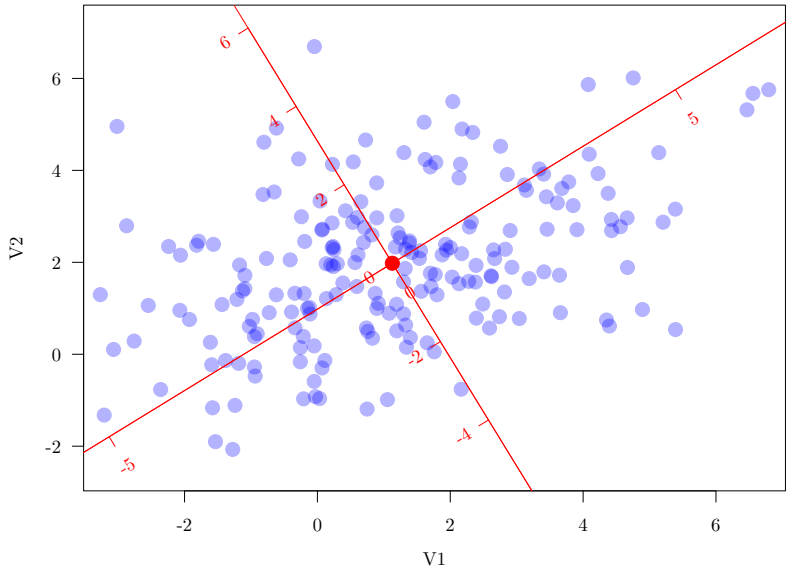


图 11.3: 基于旋转坐标系的主成分分析示意图：红色坐标系为主成分方向

```
library(grid)
grid.rect(x = 0, y = 0, width = 0.1, height = 0.1,
          gp = gpar(col = NA, fill = "red"), name = "rect0")
grid.rect(x = 0.1, y = 0.9, width = 0.1, height = 0.1,
          gp = gpar(col = NA, fill = "green"), name = "rect1")
for (i in 1:100) {
  grid.edit("rect0", x = unit(i/100, "npc"), y = unit(i/100,"npc"),
            gp = gpar(fill = rainbow(100)[i])) # 修改位置和颜色
  Sys.sleep(0.05)
}
```

如果在电脑屏幕上运行这段代码，应该会看到一个静止的正方形（“rect1”）和一个移动的正方形（“rect0”）。rect0 的位置和颜色会在循环语句中不断被修改，而 rect1 则保持不变。

以上介绍的内容只是网格图形系统中非常小的一部分，感兴趣的读者可以自行阅读 **grid** 包附带的帮助文档，来进一步认识和了解网格图形的全貌。当然，在大部分的情况下，读者无需花费太多的时间来研究 **grid** 包的细节，因为以它为基础的 **lattice** 包和 **ggplot2** 包已经可以完

成大部分的绘图任务。

11.2 lattice 网格图形

lattice (Sarkar, 2008) 是基于 **grid** 包的一套统计图形系统，它的图形设计理念来自于 Cleveland (1993) 的 Trellis 图形，其主要特征是根据特定变量（往往是分类变量）将数据分解为若干子集，并对每个子集画图。就像数理统计中的条件期望、条件概率一样，**lattice** 的图形也是一种“条件作图”。

11.2.1 简介

lattice 这个单词是格子、格子状的意思。和它意义相近的还有两个词——grid 和 trellis，这两个词都与 lattice 有着密不可分的关系：**lattice** 中一些底层作图的实现，靠的就是 **grid** 包；而 trellis 则代表了 **lattice** 的另一个特点，即面向对象的思想。

我们知道，在 R 的基础图形中，有一些函数被称为低层作图函数（第九章），例如 `points()`、`lines()`、`arrows()` 等。这些低层作图函数能够帮助我们画出更为复杂的图形。**grid** 包也借鉴了这个方法，内置了 `grid.points()`、`grid.lines()`、`grid.arrows()` 等函数。这些函数在功能上要强于传统作图，并且帮助 **lattice** 实现了自己的低层作图函数。不过在 **lattice** 中，它们的名字相应变为了 `panel.points()`、`panel.lines()`、`panel.arrows()` 等。用好面板（panel）系列函数，是熟练使用 **lattice** 画图的关键。高层作图函数只给我们勾勒出了一幅草图，细节的完善和图形的个性化则需要面板函数来完成。

lattice 的另一显著特点就是面向对象的思想，这是 **lattice** 区别于 R 传统作图的重要特点。在 R 的传统作图中，运行一个高级作图函数，通常会直接生成一幅可见的图形。而在 **lattice** 里，高层作图函数返回结果的是一个 trellis 对象，要想显示图形，必须使用对象的 `print()` 方法才可以。我们通过下面的一个小例子来说明这一点：

```
library(lattice) # 默认 lattice 包不随 R 启动，须手动载入
x = hist(iris$Sepal.Length) # 图形立刻显示出来
y = histogram(~Sepal.Length, data = iris) # 图形不显示
```

这两条语句分别调用了基础图形中的 `hist()` 函数和 **lattice** 中的 `histogram()` 函数，将运行的结果分别赋予了变量 x 和 y。这时，如果我

古统新修记

古统者，古代统计图形也，乃谢公益辉所著现代统计图形之谑称。盖其成书甚早，初行即声名大噪，得者皆读之后快。然久未付梓，仅钞本[1]辗转相传于坊间，其间桃花人面，沧海桑田，忽忽十载，“现”已作“古”，遂有古统之誉。

古统钞本，始于丁亥[2]，发于戊子[3]，兴于己丑[4]，终以庚寅[5]本传世。庚寅者，凡八十回，二百一十页，六十五图，盖八章：历史、工具、细节、元素、图库、系统、模型、数据。另附导读、程序初步、作图技巧、统计动画、本书 R 包。回目有笑傲[6]之韵，兼射雕[7]之风。彼时 R 语言初入中原，小荷乍露，读者莫不折于 R 语言之奇丽，著者之渊博，文笔之雅致，洞见之深远。然系统、模型、数据三章空有其骨，未见其肉，如夜归掷履，邻人闻落地声一，待其二者，终不得焉，念之，憾之，恨之，盼之。

及至戊戌[8]，谢公自云[9]，为生计迫[10]，亦缺良墨[11]，古统残本实难为继，遂传辛卯[12]本。是本增页减章，凡七章八十回：历史、工具、元素、图库、系统、数据、元素，古意犹存。三

[1]即 pdf 版。
[2]即 2007 年。
[3]即 2008 年。
[4]即 2009 年。
[5]即 2010 年。
[6]金庸《笑傲江湖》，每章标题均为两个字。
[7]金庸《射雕英雄传》，每章标题均为四个字。
[8]即 2018 年。
[9]见谢益辉的博客《再清零》一文。
[10]据说是忙于博士资格考试。
[11]据说是所用的文本工具崩溃了。
[12]即 2011 年。

百六十五页，含二表，百卅八图，较庚寅本倍之。尤以图库、数据二章所增为善。天下久饥，今一快朵颐，喜大普奔。时有 *ggplot* 丹青风靡，人尽逐之。古统虽有述及，奈何囿尊古法，修之无肉，弃之有味，谢公绝之。天下闻者，莫不扼腕长叹。盖虽残本，世间莫有比肩者矣。

辛卯毕，钞本一。*bookdown* 兀，己亥[13]出。有黄公湘云者，得谢公雅赐手稿，乃以迅雷之势，着以良墨[14]，迁旧稿于新址，时人异之，不知良墨实出谢公。古统新址，人皆可访之；良墨旧稿，人皆可修之。群雄遂起，竞相争鸣，古统乃初见现统之大观。圣人曰：独乐乐，与民乐乐，孰乐？黄公之功，善莫大焉。

古统八十回之残本，如曹公之石头记，或断臂之维纳斯，美则美矣，终为不全之数，天下憾之久矣。今余受谢黄二公之托，重修现统，以付梓为盼。十年之功，毕此一役，余知兹事体大，如红楼续梦、美神接臂，受命以来，夙夜忧叹，恐托付不效，以伤古统之明。虽肝脑涂地，未敢不尽心竭力，竟披阅十载，增删五次，终得庚子本。凡一十一章，四十表。插图皆以 *ggplot* 新绘，增至百五十图。

嗟乎！余幼年慕周公华健之声，后习丝竹，终于京师学堂遂周公同台之乐。今壮年习 *R*，独行步履维艰，恰行谢公古统庚寅本，旱逢甘露，受益匪浅。未料日后拙作《学 *R*》得谢公赐序。今复为其旧作新修，世事之玄妙莫过于此。正是：

今人不见古时月，今月曾经照古人。

古人今人若流水，共看明月皆如此。

——赵鹏　于　姑苏西浦

[13]即 2019 年。

[14]R bookdown 包。