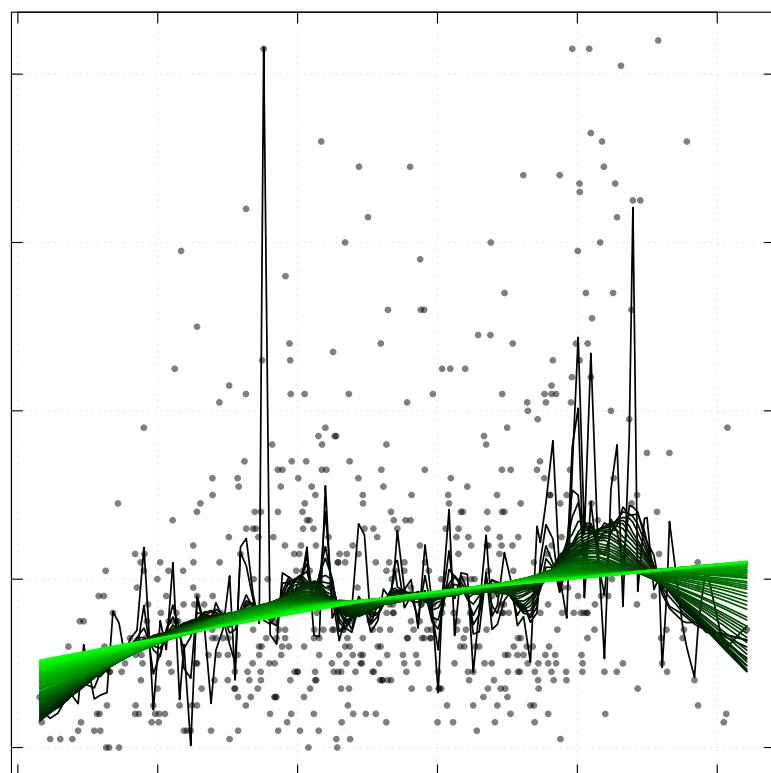


现代统计图形

谢益辉

2010 年 7 月 2 日



版权声明

本书电子版采用Creative Commons（简称CC）许可证“署名—非商业性使用—相同方式共享2.5中国大陆”，该许可证的全文可以从<http://creativecommons.org/licenses/by-nc-sa/2.5/cn/>获得；一份普通人可以理解的法律文本概要可以从<http://creativecommons.org/licenses/by-nc-sa/2.5/cn/legalcode>获得。



责任权利

本CC许可证赋予读者复制、发行、展览、表演、放映、广播或通过信息网络传播本作品以及创作演绎作品的自由，而无需向原作者征求许可或支付任何费用；本许可证与出版社版权独立，因此复制、传播或演绎本作品也无须征求出版社许可。您需要遵循的条件是：

- 声明原作者的署名（Attribution）：不得将本作品归为自己的劳动
- 不得将本作品用于商业目的（Noncommercial）
- 基于本作品的演绎作品须遵守同样许可证发布（Share Alike）

作者采用CC许可证的考虑主要有三点：

- 让读者能免费、自由获得本书，节省经济支出；在网络和电子文档的时代，我们应该充分利用这些工具的优势，如传播快捷、读者交流反馈方便（以便提高书籍质量）等
- 版权的本来意义不在于控制所有权，它只不过是为了对原创者的一种署名激励；如果版权的存在妨碍了知识的传播，那么本人认为版权就没有太大的意义；CC许可证中的“非商业”和“同样许可证”限制条款在书籍出版14年后会自动取消，即读者可以用于商业目的或更改至其它许可证；CC许可证规定的14年似乎是很长的时间，但读者须知：通常的版权只有在原作者去世后50年才会被取消！换句话说，版权告诉我们一个很深刻的哲理：长寿是很重要的

- 自由软件用户往往有某种痴狂的特征，而这种痴狂往往来源于自由软件的分享精神；R语言让本人受益颇多，这本书可视作是对它的一种回馈；既然R语言是自由的，那么本书也将尽量“自由”

特别声明

尽管CC许可证没有限制作品的传播方式，但本作者不愿看到本书被任何人以论坛附件的方式发布在任何论坛，原因是本书稿尚未成熟，或许有诸多不完善之处甚至严重错误，作者在不断更新中，若要传播本书稿给他人，请仅仅给出本书的原始链接<http://yihui.name/cn/publication/>，否则作者对传播过程中的错误概不负责。

捐赠说明

如果本书对您有任何帮助，您不妨考虑为“统计之都”网站（自愿）捐赠：<http://cos.name/donate/>；捐赠对象非作者本人，但本作者将从一定程度上根据捐赠情况判断本书工作的价值。捐赠所得将用于推广统计学和自由统计软件。捐赠之后请及时告知网站管理人员：admin@cos.name。

致谢

本书写作过程中收到了不少读者反馈，在此一并感谢。感谢魏太云对本书文字的校对和建议；感谢赵彦云老师对本书书名和写作风格的建议；感谢李皞对写lattice系统和rgl包的提议；感谢李丰的“彩蛋”建议。欢迎更多意见和建议，包括：

- 内容是否全面，如：有哪些您经常用而且觉得有用的图形这里没有收录
- 主题是否清楚，如：数学、统计理论与图形的对应关系
- 指引是否充分，如：读完某一节是否知道该怎么做
- 文字是否自然，如：有没有“爱上层楼”现象或者过于古板或过于口语化

- 案例是否实用，如：满篇模拟的数据而没有实际数据，或者数据离您的领域太远

目录

序言	i
代序一	i
代序二	i
作者导读	i
第一章 历史	1
1.1 饼图和线图的起源	1
1.2 霍乱传染之谜	2
1.3 提灯女士的玫瑰图	4
1.4 拿破仑的俄罗斯远征	4
1.5 小结与开始	7
第二章 工具	11
2.1 选择作图工具	11
2.2 R语言简介	13
2.3 安装R语言	17
第三章 细节	21
3.1 <i>par()</i> 函数的参数详解	22
3.2 <i>plot()</i> 及相关函数的参数说明	30
第四章 元素	33
4.1 颜色	34
4.1.1 固定颜色选择函数	34
4.1.2 颜色生成和转换函数	36
4.1.3 特定颜色主题调色板	38

4.1.4 演变色的简单原理及应用	40
4.2 点	43
4.3 曲线、直线、线段、箭头、X-样条	44
4.4 矩形、多边形	48
4.5 网格线	50
4.6 标题、任意文本、周边文本	52
4.7 图例	53
4.8 坐标轴	55
第五章 图库	59
5.1 直方图	59
5.2 茎叶图	63
5.3 箱线图	67
5.4 条形图	71
5.5 散点图	73
5.6 关联图	74
5.7 条件密度图	77
5.8 等高图	80
5.9 条件分割图	84
5.10 一元函数曲线图	87
5.11 Cleveland点图	88
5.12 颜色等高图	89
5.13 四瓣图	91
5.14 颜色图	95
5.15 矩阵图	98
5.16 马赛克图	100
5.17 散点图矩阵	103
5.18 三维透视图	105
5.19 因素效应图	108
5.20 坐标轴须	110
5.21 平滑散点图	111
5.22 棘状图	113
5.23 星状图	116
5.24 带状图	118

5.25 向日葵散点图	120
5.26 符号图	122
5.27 饼图	125
5.28 热图	128
5.29 交互效应图	128
5.30 QQ图	128
5.31 生存函数图	131
5.32 分类与回归树图	133
5.33 小提琴图	135
5.34 地图	136
5.35 脸谱图	138
5.36 平行坐标图	139
5.37 调和曲线图	139
5.38 二元箱线图	139
5.39 习题	139
第六章 系统	141
6.1 网格图形	141
6.2 lattice图形	142
6.3 ggplot2图形	142
第七章 模型	143
7.1 线性回归模型	143
7.2 方差分析	144
7.3 非参数回归模型	144
7.3.1 局部加权回归散点平滑法	144
7.4 稳健回归模型	144
7.5 广义线性模型	144
7.6 分类数据模型和列联表	144
7.7 混合效应模型	144
7.8 主成分分析和因子分析	145
7.9 聚类分析	145
7.10 判别分析	145
7.11 对应分析	145

7.12 多维标度分析	145
7.13 时间序列模型	145
7.14 生存分析	146
7.15 空间统计学	146
7.16 数据挖掘和机器学习	146
7.16.1 分类与回归树	146
7.16.2 Bootstrap	146
7.16.3 支持向量机	146
第八章 数据	147
8.1 离散数据	147
8.1.1 一维数据	147
8.1.2 多维数据	147
8.2 连续数据	148
8.2.1 一维数据	148
8.2.2 二维数据	148
8.2.3 高维数据	148
8.3 混合数据	148
8.3.1 一维数据	148
8.3.2 二维数据	148
8.3.3 高维数据	148
附录 A 程序初步	149
A.1 对象类型	149
A.1.1 向量	149
A.1.2 因子	153
A.1.3 数组和矩阵	154
A.1.4 数据框和列表	157
A.1.5 函数	158
A.2 操作方法	160
A.2.1 选择与循环	160
A.2.2 输入与输出	161
A.3 习题	161

附录 B 作图技巧	163
B.1 添加数学公式	163
B.2 一页多图	165
B.2.1 设置图形参数	165
B.2.2 设置图形版面	165
B.2.3 拆分设备屏幕	166
B.3 交互操作	170
B.3.1 获取鼠标位置的坐标	170
B.3.2 识别鼠标附近的数据	170
B.3.3 响应鼠标键盘的动作	171
B.4 分类变量散点图示	171
B.4.1 向日葵散点图	174
B.4.2 随机打散方法	174
B.5 图形设备	174
附录 C 统计动画	177
附录 D 本书R包	179
D.1 函数说明	179
D.2 数据说明	180
参考文献	181
索引	189

插图

1.1	William Playfair的时序线图	2
1.2	William Playfair的饼图（史上第一幅饼图）	3
1.3	John Snow的霍乱传染原因探索图	5
1.4	Florence Nightingale的极坐标面积图	6
1.5	Charles Joseph Minard的拿破仑远征图	8
2.1	正态分布密度曲线及其5%到95%分位数区间表示	19
3.1	参数adj、mgp、tcl和ljoin设置演示	23
3.2	其它主要参数的效果演示：bty, font, las, family等	24
3.3	图形的各种区域说明	28
3.4	plot()作图的九种样式	31
4.1	RColorBrewer包中所有调色板颜色的演示	41
4.2	New Haven地区的年均气温（1912~1971年）	42
4.3	点的类型：pch参数取值从0到25及其它符号	44
4.4	鸢尾花的花瓣长宽散点图	45
4.5	曲线、直线、直线段、箭头的展示说明	47
4.6	X-样条各种形状的展示	49
4.7	多边形和矩形结合使用的一个巧妙图示	51
4.8	添加标题、任意文本和周边文本的一个演示	54
4.9	中美出口额双坐标轴图示	56
5.1	喷泉间隔时间直方图	60
5.2	直方图与密度曲线的结合	61
5.3	世界各地大陆块面积茎叶图	64

5.4	泊松分布随机数茎叶图	65
5.5	各种杀虫剂下昆虫数目的箱线图	69
5.6	箱线图的凹槽与统计推断	70
5.7	弗吉尼亚死亡率数据条形图	72
5.8	半透明散点图中的规律	75
5.9	眼睛颜色与头发颜色的关联图	76
5.10	航天飞机O型环在不同温度下失效的条件密度图	78
5.11	网格数据的示意图	80
5.12	中国31地区国民预期寿命和高学历人数密度等高图	81
5.13	与等高图对应的三维透视图	82
5.14	给定震源深图的地震经纬度条件分割图	85
5.15	函数 $f(x) = \sin(\cos(x) * \exp(-x/2))$ 的曲线图	87
5.16	弗吉尼亚死亡率数据的Cleveland点图	89
5.17	新西兰Maunga Whau火山高度数据颜色等高图	90
5.18	加州伯克利分校录取数据四瓣图	92
5.19	颜色图中色块与数值的对应关系	96
5.20	新西兰Maunga Whau火山高度数据颜色图	97
5.21	用矩阵图画出的一系列正弦曲线	99
5.22	泰坦尼克号乘客生存数据马赛克图	101
5.23	鸢尾花数据的散点图矩阵	104
5.24	新西兰Maunga Whau火山的三维透视图	106
5.25	向三维透视图中添加图形元素的展示	107
5.26	经纱断裂数据的因素效应图	109
5.27	带坐标轴须的喷泉间隔时间密度曲线图	111
5.28	BinormCircle数据的平滑散点图	112
5.29	航天飞机O型环在不同温度下失效的棘状图	114
5.30	Motor Trend杂志1974年汽车数据的星状图	117
5.31	各种杀虫剂下昆虫数目的带状图	119
5.32	鸢尾花花瓣长和宽的向日葵散点图	121
5.33	符号图提供的六种基本符号	123
5.34	中国31地区五大国民素质特征分布温度计图	126
5.35	各类馅饼销售数据的饼图	127
5.36	喷泉间隔时间的正态分布QQ图	129

5.37 急性髓细胞白血病病人生存函数图	131
5.38 脊椎矫正手术结果的分类树图	134
5.39 三组双峰数据的小提琴图比较	135
5.40 2005年世界各国农业进出口竞争力地图	137
B.1 正态分布密度函数公式的表示	164
B.2 函数 <i>layout()</i> 的版面设置示意图	166
B.3 回归模型中边际分布的展示	167
B.4 拆分作图设备屏幕区域的示例	169
B.5 鼠标在图形窗口中移动的效果图	172
B.6 分类变量的散点图示方法示例	173

X

表格

5.1 二维列联表的经典形式 93

序言

代序一

代序二

作者导读

我们常说“一图胜千言”，然而现实情况是我们了解的图形种类太少、使用的作图工具缺乏灵活性，这在很大程度上制约了统计图形的发展，使得统计图形在数据分析中应有的潜力没有被充分挖掘出来，正是这样的背景催生了本书的写作。

本书根据统计图形制作的需要，将所有内容分为七章：第一章先选择性回顾历史上的四幅著名统计图形，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；第二章介绍图形工具，本书主要以R软件为制图工具，因此本章也会介绍关于R语言的一些基础知识；第三章详细介绍图形参数，用以对图形进行细节调整，若读者对图形细节要求不高则可直接跳过这一章；第四章讲解基础图形元素的使用，包括点、线、多边形、颜色和文本等，本章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；第五章是本书的一大核心，集中介绍讲解现有的统计图形种类如直方图、条形图、茎叶图、饼图、箱线图等，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；第六章介绍基础图形系统(base graphics)之外的其它图形系统如grid、lattice和ggplot2，第七章和第八章对各种统计图形分别从模型方法和数据类型的角度给出一些实例并作出归纳总结，以便让读者清楚区分统计图形运用的条件和场合；附录中

给出了一些作图方面的技巧。

本书的所有图形文件和程序代码可以从作者的个人主页下载 (<http://yihui.name/cn/publication/>)，另外本书也配有相应的R包**MSG**(Xie, 2010b)，使用说明参见附录D。阅读本书过程中若有任何疑问请Email联系作者：xie@yihui.name。

关于本书中R程序代码，记号说明如下：

- > 表示一段R程序的开始；在R中可以用`options(prompt = '> ')`设置程序行的起始符号，默认为“>”；后文中只要遇到这个标记，则说明该标记之后的程序语句可以直接在R中运行（读者在运行书中的示例时不要把这个符号也敲进命令行中）
- + 续行符；当一段程序在某一行中没有完整显示出来时，就会折到下一行，此时R会以“+”表示程序语句上不完整，正在继续
- [n] 其中“n”表示一个整数，中括号括上一个整数表示R程序输出的行号，比如[1]表示这是第1行输出
- # 表示R程序注释，即不会被执行的语句（只是为了增强程序的可读性而做的“标记”）

另外，本书R代码以等宽斜体排版，代码中带有程序起始符以及续行符，左侧以数字标出行号，例：

```
1 > plot(x, y)
2 > plot(cumsum(rnorm(80)), type = "l", col = "blue",
3 +       xlab = "Step", ylab = "Brownian Motion")
```

代码的输出以等宽正体排版，左侧不带行号提示，例：

```
1 > library(MSG)
2 > data(PlantCounts)
3 > summary(lm(counts ~ altitude, PlantCounts))

Call:
lm(formula = counts ~ altitude, data = PlantCounts)

Residuals:
    Min      1Q  Median      3Q     Max
-27.63 -10.27  -3.78   6.49  65.79
```

```

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -26.6722     6.4529   -4.13  4.1e-05 ***
altitude      0.0816     0.0111    7.37  5.6e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.3 on 598 degrees of freedom
Multiple R-squared: 0.0833,          Adjusted R-squared: 0.0818
F-statistic: 54.4 on 1 and 598 DF,  p-value: 5.61e-13

```

正文中的代码以等宽正体表示，如“`inline R code`”，函数名称以斜体表示，如“`function()`”，对象类名称和参数名称用无衬线字体表示，如“`class` using sans serif”，R程序包用粗体表示，如“`package`”。

本书以`LATEX`结合R (Sweave, 参见Leisch (2002)) 写成，所有图形均为R代码动态生成，因此整本书稿具有可重复性 (reproducible)，读者可从网上下载书稿源代码并配合适当的工具编译生成本书稿。习惯了阅读Word书稿的读者可能对本书的图表排版有所不适，因为图表看似与正文脱离，即：图/表不一定紧接着出现在正文中提到该图/表的位置，这是因为`LATEX`排版哲学是让读者专注于阅读正文，而图表为陪衬，因此读者在阅读本书时，请以正文为主，如果正文提到图表时，再按序号去找相应的内容。

顾炎武在《日知录》中曾有一句话：“形而上者谓之道，形而下者谓之器。”对本书来讲，统计作图的（计算机）技术本身即为“器”，而数据处理以及统计图形的灵活应用则为“道”。本书的写作目的正是希望能够基于“器”的练习和启发，让读者在统计数据处理和分析中真正得“道”，使统计图形在数据的探索分析中发挥福尔摩斯探案般的功效。

谢益辉
于爱荷华州立大学统计系
2010年7月

第一章 历史

“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，统计图形的重要性自然不言而喻。

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四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。

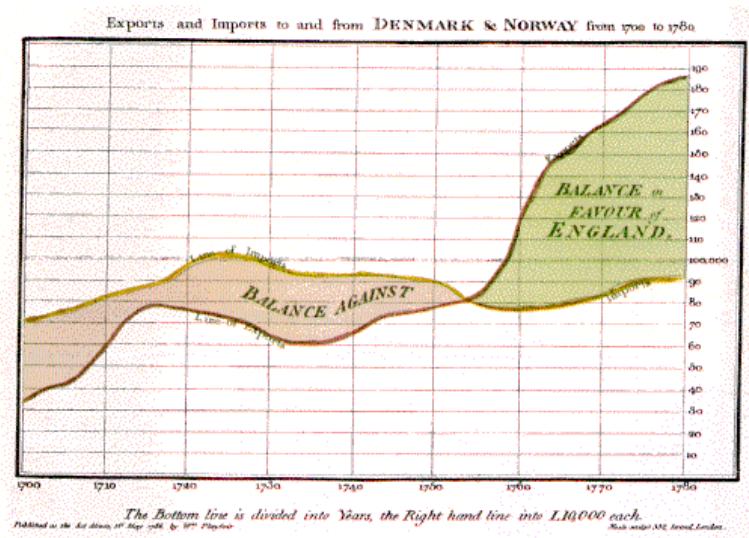


图 1.1: Playfair (1786) 绘制的线图。这幅图主要展示了 1700 年至 1780 年间英格兰的进出口时序数据，左边表明了对外贸易对英格兰不利，而随着时间发展，大约 1752 年后，对外贸易逐渐变得有利。图片来源: http://en.wikipedia.org/wiki/William_Playfair

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。从左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。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。事实上，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，伦敦成为无药可医的流行病滋生的最佳地点。公众一致认为霍乱是由空气传播的，如果呼

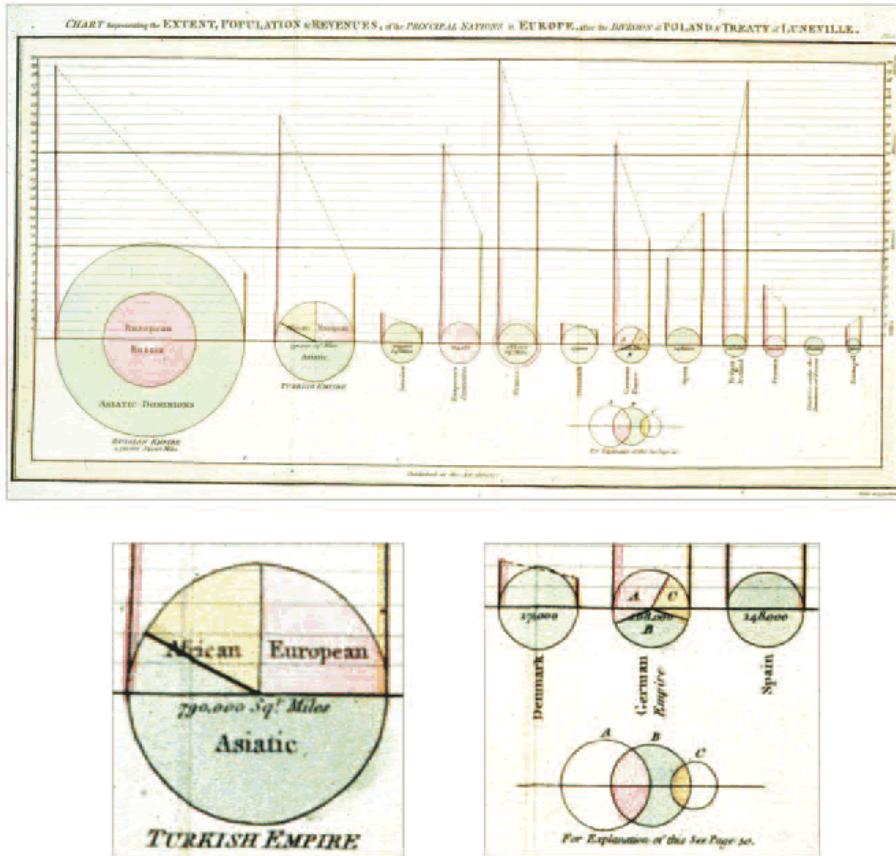


图 1.2: Playfair (1801) 绘制的饼图。这是历史上第一幅饼图，描述了法国大革命前后一些欧洲国家的统计数据。上方的大图展示了各个国家的领土面积（和圆圈成比例）以及人口（左垂线）、税收（右垂线）、国土在各大洲分布比例等数据，两条垂线连线的斜率可表示税负的轻重（这一点颇有争议，因为斜率与圆的半径有关）。左下方的饼图展示了土耳其帝国在三大洲的国土面积分布。图片来源：<http://www.psych.utoronto.ca/users/spence/Spence%202005.pdf>

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，他最终确定了霍乱的源头。后来证实离这口井仅三英尺远的地方有一处污水坑，坑内释放出来的细菌正是霍乱发生的罪魁祸首。

1.3 提灯女士的玫瑰图

南丁格尔（Florence Nightingale）是我们耳熟能详的“提灯女士”，她不仅是现代护理的鼻祖及现代护理专业的创始人，而且是历史上使用极坐标面积图的先驱。这种图形外形如玫瑰，因此后来也称之为玫瑰图，其主要构思是用“花瓣”的面积表示统计数值的大小。图1.4反映了克里米亚战争（英国等与俄国争夺巴尔干半岛的战争）中英国军队自1854年4月至1856年3月的逐月死亡人数(Nightingale, 1858)；其中，右图为1854年4月至1855年3月的死亡人数，左图为1855年4月至1856年3月的死亡人数。玫瑰图不仅清楚展示了这两年军队死亡人数的变化，而且更重要的是，她将每个月中三种死亡情况也分别用不同颜色标记出来：蓝色表示死于可预防的疾病、红色表示死于战争伤害、黑色表示死于其它原因。这样我们可以清楚知道军队伤亡原因的结构，尤其是“绝大多数士兵死于可预防的疾病”（图中最高的花瓣）。凭借这一条重要信息，她让英国政府意识到，真正影响战争伤亡的并非战争本身，而是由于军队缺乏有效的医疗护理！

1.4 拿破仑的俄罗斯远征

1812年6月24日，拿破仑率领的691,501人的大兵团—同时也是欧洲历史上集结的最大规模的部队—开赴莫斯科。但等他们到达那里，看到的只是一座空城。城里的人都被遣散，所有的供给也被中断。由于没有正式的投降，拿破仑觉得俄国人从他那儿剥夺了一场传统意义上的胜利。

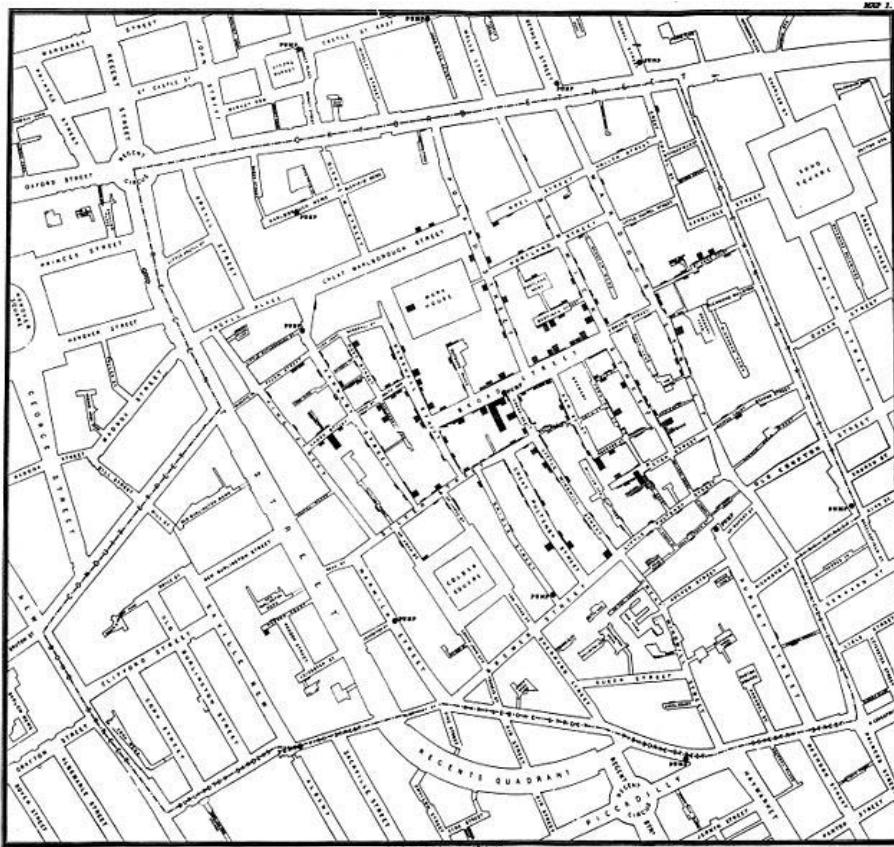


图 1.3: 1854年英国Broad大街大规模爆发霍乱，当时了解微生物理论的人很少，人们不清楚霍乱传播途径，而“瘴气传播理论”是当时的主导理论；John Snow对这种理论表示了怀疑，于1849年发表了关于霍乱传播理论的论文，本图即其主要依据。图中心东西方向的街道即为Broad大街，黑点表示死亡的地点。这幅图形揭示了一个重要现象，就是死亡发生地都在街道中部一处水源（水井）周围，市内其它水源周围极少发现死者。进一步调查他发现这些死者都饮用过这里的井水。图片来源：<http://upload.wikimedia.org/wikipedia/commons/2/27/Snow-cholera-map-1.jpg>

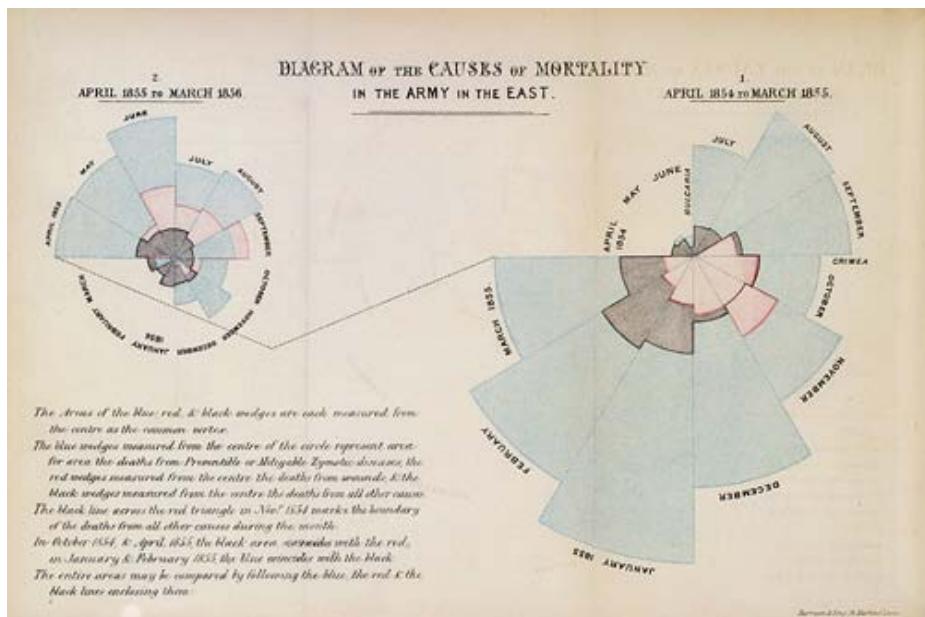


图 1.4: 南丁格尔的极坐标面积图: 两幅图分别是1854年和1855年的军队伤亡人数, 一年12个月恰好可以将极坐标分为12等分, 每一瓣代表一个月。图中用颜色标记出了三种死亡原因。南丁格尔的重大贡献在于使得英国政府意识到真正影响战争伤亡的并非战争本身, 而是由于军队缺乏有效的医疗护理, 导致大量的士兵死于可预防的疾病。1857年, 在她的努力下, 英国皇家陆军卫生委员会成立。同年, 军医学校成立。图片来源: http://en.wikipedia.org/wiki/Florence_Nightingale

军队不得不撤退。在归途中，给军队提供补给几乎是不可能的，主要是因为天气过于恶劣。马匹因为缺少粮草而变得虚弱，所有的马要么饿死，要么被饥饿的士兵拿去果腹。没有了坐骑，法国骑兵们成了步兵，大炮和马车被迫丢弃，部队没了装甲。饥饿与疾病带来惨重的伤亡，而逃兵增速也直线上升。大军团的小分队在Vyazma, Krasnoi和Polotsk也被俄国人击溃。法国军队在渡贝尔齐纳河时遭到俄军两面夹击，伤亡惨重，这也是法军在俄国遭遇的最后一场灾难。1812年12月14日，大军团被驱逐出俄国领土。在这场远征俄罗斯的战役中，拿破仑的士兵只有大约22,000人得以幸存。

这一历史事件被Charles Joseph Minard用一张二维平面图形记录了下来，Minard是一位法国工程师，他以在工程和统计中应用图形而闻名。图1.5就是他的著名作品：在一张二维图形中，他成功地展示了如下信息：

- 军队的位置和前进方向，以及一路上军队的分支和汇合情况
- 士兵数目的减少（图形顶端最粗的线条表示最初渡河的422,000人，他们一路深入到俄国领土，在莫斯科停下来的时候还有100,000人左右。从右到左，他们朝西走回头路，渡过Niemen河的时候，仅仅剩下10,000。随着大部队和余部会师（比如在渡贝尔齐纳河之前），图中显示的数字降中也有升）
- 撤退时的气温变化（参见图的下半部分，可知当时气候条件极其恶劣）

这幅图形在统计图形界内享有至高无上的地位，被Edward Tufte¹称为“有史以来最好的统计图形”。

1.5 小结与开始

在前面四节中，我们看到了具有历史意义的四幅统计图形，它们融入了前人的智慧与艺术，有些甚至具有重大社会价值。当然我们不能苛求每一幅统计图形都能达到那样的效果，但至少我们了解到了统计图形在揭示特殊现象或规律上的功能，这种功能是数据本身不能替代的。试想，若只是将每一个霍乱死者的数据列在纸上，那么要观察出霍乱发生的规律是何其艰难。

¹Tufte是统计图形和信息可视化领域的领军人物，人称“数据达芬奇”。

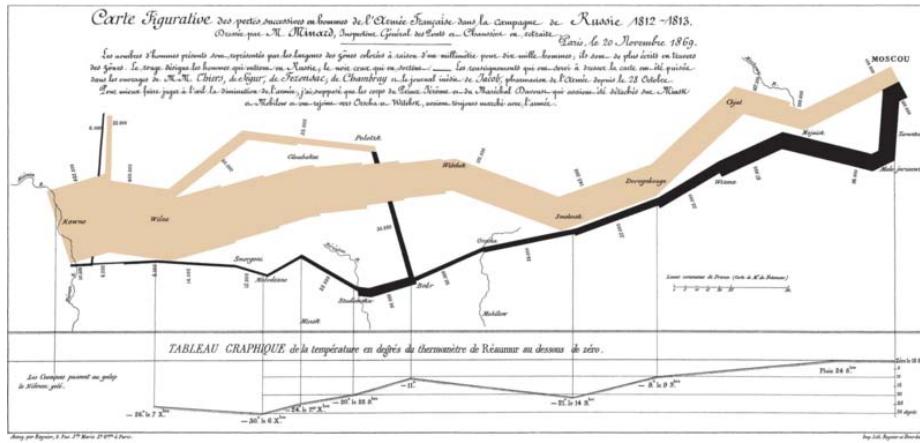


图 1.5: Minard绘制的地图，展现了1812年拿破仑的大军团进军俄国的路线（上半部分）和撤退时的气温变化（下半部分）。这一历史事件中，法军数量的急剧减少以及恶劣的气候条件一览无遗，法国科学家Étienne-Jules Marey称“该图所展现出的雄辩对历史学家的笔是一种极大的挑战”。图片来源：<http://upload.wikimedia.org/wikipedia/commons/2/29/Minard.png>

统计图形领域还有大批卓有成就的研究者，为统计图形的发展做出了不少贡献。早在上个世纪八九十年代，国外已经有比较全面的图示书籍文献资料，如前文提到的“数据达芬奇”(Tufte, 1992, 2001), Wainer and Thissen (1981), Wilkinson (2005)，以及贝尔实验室的Cleveland (1985, 1993)等，其中尤其是Cleveland在数据可视化和统计图示方面撰写了大量的论文，还提出了不少原创图形类型，感兴趣的读者可以访问他在贝尔实验室的个人主页<http://cm.bell-labs.com/cm/ms/departments/sia/wsc/>。

关于统计图形的历史总结，Friendly and Denis (2001)是一份非常详尽的资料，该文档整理、记载了自17世纪以前至今数百年历史中较有影响力的统计图形。

如今统计图形的使用已经比较普遍，饼图、条形图都已经不是什么新鲜内容，但是一方面统计图形的价值并没有被很好地体现出来，另一方面人们对统计图形的了解和使用也被统计软件所限，而不能随心创造图形。我们来看这样一组事实(谢益辉, 2008):

以期刊《统计研究》在2006年12月~2007年11月期间共12个月的所有论文作为统计对象，剔除部分非学术研究型论文之后，挑选论文总数为168篇，其中使用表格的论文篇数为136篇（81.43%），表格总数为528个，而使用图形的论文仅有63篇（37.72%），若将仅仅使用示意图（非统计图形）、条形图和折线图的论文排除在外，使用其它图形的论文仅剩下9篇。

这可算国内统计图形应用现状的一个缩影。为了改变这种局面、发掘出统计图形在数据分析中应有的潜力，我们特别撰写这本小书，供广大统计研究者参考。我们的目的并非仅限于如何作出漂亮的统计图形，而是在作图的同时，强调图背后更重要的工作，那就是“数据分析与统计图形的有机结合”。传统的统计分析大约可以分为三类：

- 描述性统计分析：Descriptive Statistical Analysis
- 推断性统计分析：Inferential Statistical Analysis
- 探索性统计分析：Exploratory Statistical Analysis

前两类统计分析往往都是从既定的统计模型、方法的角度入手，而探索统计分析则主要借助图形对数据进行探索性分析，这对于数据分析的手段是一种重要的拓展（姑且称之为“图形统计分析”）；然而要使用这种手段，则必须清楚了解如何制图以及现有图形有哪些种类，这样才能真正开发出统计图形的价值。

其实，“图形统计分析”也不是一个新概念，平常的统计图示已经或多或少用到了这样的思想，只是我们往往更倾向于数理意义上的统计模型分析，而不会把图形统计分析作为主要分析手段，当然，由于图形的表达限制以及统计图形的普及程度，也使得它不可能替代模型分析，但无论如何，我们对统计图形在统计分析中的地位应该加深认识，不仅是因为这是一个信息爆炸的时代、大量的信息让我们无法在短时间内获取核心信息，更重要的是，目前在国内仍有大量的统计图形未被开发介绍出来，图形种类过于单一，表达信息的效果大打折扣。

本书介绍统计图形的方式主要是从两方面入手，第一，阐明各种统计图形所用到的统计量；第二，与实例结合，解释图形中表现的统计量的实际含义。在本书的附录B中，我们也会介绍一些有用的作图技巧，用以辅助完善统计图形。

总的说来，要把图形提到“统计分析”的高度，就一定要搞清楚统计图形的来龙去脉，包括原始数据的来源和类型、统计量的计算、图形的构造与组合机制等，这与统计模型实际上没有本质区别：若不清楚模型的假设前提、计算原理以及相应的结果解释，同样也不能随便使用模型分析。除了图形本身之外，用好图形分析还需要一定的观察力，最简单的莫过于观察数据的分布状况、离群点、线性/非线性关系等表面观察，而更重要也是最本质的莫过于洞察到种种规律或异常现象背后的深刻原因，至此，我们才达到了分析的目的。

第二章 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》

当今统计软件有许多种，如SPSS、SAS、S-Plus、Statistica、Stata、Systat甚至Excel等等，它们都有作图功能，那么我们面对这么多工具应该如何选择呢？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，三是图形类型多种多样。

2.1 选择作图工具

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，然而这里的“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。

我们通常见到的所谓“统计图形”也许只是Microsoft Office 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淡薄。而统计学的核心研究对象是什么？答案当然是分布（Distribution）¹。无论Excel的图形如何搭配色彩、怎样变得立体化，都跳不出上面三种类型的限制，而且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”。本书所要介绍的R语言所能表达的统计量种类极其丰富，甚至可以毫不夸张地说，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一—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 and Bassett, 1978)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。

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）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。

要想真正精通统计图形，则应首先要练好基本功（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）。

¹分布不仅包含一元变量的分布，而且（更重要的是）包括多元变量的分布，诸如“相关”、“概率”等概念都可以归为“分布”的范畴

2.2 R语言简介

“R” (R Development Core Team, 2009) 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）已经是数以千计，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，那就是它是免费、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，加上目前国内学术和应用界内人士对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）；一般而言，图形往往比较直观而且相对简易，因此本书从R图形入手，以期能给初学者提供一份好的R作图入门学习资料。

在启动R之后²，屏幕上会显示类似如下信息，告诉我们R是由很多贡献者一起协作编写的免费软件，用户可以在一定许可和条件（GPL）下重新发布它：

```

1 > cat(head(system("R -e R.version.string", TRUE)[-1],
2 +      15), sep = "\n")
R version 2.11.1 (2010-05-31)
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
```

²Windows用户一般从程序快捷方式直接启动，Linux用户一般可以从终端敲入命令R启动

```
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
```

```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
```

从技术上来讲，R是一套用于统计计算和图示的综合系统，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³(Becker *et al.*, 1988)以及Sussman的Scheme⁴，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，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, 2009)。

本书不会过多涉及到附加包，所介绍图形主要基于R自身的**graphics**包，当然也不可避免会使用**base**和**grDevices**等基础包中的函数⁵；在第五章中会使用一些附加包介绍特殊的统计数据和统计方法、模型涉及到的图形，例如分类数据（Categorical Data）会用到**vcd**包，生存分析会用到**survival**包。当我们需要调用附加包时，可以使用*library()*函数，例如加载**MSG**包：

```
1 > library(MSG)
```

R的官方网站<http://www.R-project.org>中对R有详细介绍，我们也可以从它在世界各地的镜像（CRAN: <http://CRAN.R-project.org>，全称Comprehensive R Archive Network）下载R的安装程序和附加包，通常

³<http://cm.bell-labs.com/cm/ms/departments/sia/S/history.html>

⁴<http://www.cs.indiana.edu/scheme-repository/home.html>

⁵这些基础包一般不用特别加载，R在启动的时候会自动加载进来，我们随时可以用*search()*函数来查看目前的工作环境中有哪些包已经被加载

我们可以用函数*install.packages()*安装附加包，Windows用户也可以从菜单中点击安装： Packages ⇒ Install Package(s)，注意，附加包都是从镜像上下载的，因此安装时要保证网络连接正常；当然我们也可以先将程序包下载到本地计算机上然后安装。

可能令读者感到不便的一点是，R不像别的统计软件那样有图形用户界面（GUI，即**Graphical User Interface**）⁶，因此它不会显得很“傻瓜”，它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。

著名计算机科学家Niklaus 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，附录A.1中给出了一些简单数据操作例子，请读者通过阅读该章熟悉R的数据对象；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2000个，这还只是CRAN上的仓库，不算另外两个著名站点“Bioconductor”和“Omegahat”。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通常在几十K到几

⁶事实上也不是完全没有，比如John Fox的Rcmdr(Fox et al., 2009)包就是一个比较成熟和著名的R用户界面，我们可以在其中点菜单操作，但我们并不推荐用户过多依赖于GUI，因为GUI会让我们难以驾驭很多统计分析的细节处理，况且R语法命令也不是那样难学

百K的一个附加包。值得注意的是，R的主安装程序大小约为30M，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，如官方的六本手册（均可从R的安装目录或者网站<http://cran.r-project.org/manuals.html>上找到）：

- An Introduction to R (R-intro)
- R Data Import/Export (R-data)
- R Installation and Administration (R-admin)
- Writing R Extensions (R-exts)
- R Internals (R-ints)
- The R Language Definition (R-lang)

其中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，不适合对计算机程序没有深入了解的初学者阅读；另外还有Emmanuel Paradis编写的“R for Beginners”也是较好的入门教材⁷。鉴于目前R的中文资料并不多，我们也推荐R的初学者和爱好者通过访问“统计之都”中文网站（<http://cos.name>；R版块<http://cos.name/cn/forum/15>）共同学习、讨论和研究。

最后需要特别指出的是，R拥有一套完善而便利的帮助系统，这对于初学者也是很好的资源。若已知函数名称，我们可以简单用问号“?”来获取该函数的帮助，比如查询计算均值的函数帮助，只需要在命令行中敲入“?mean”，则会弹出一个窗口显示该函数的详细说明。大多数情况下“?”等价于help()函数；但有少数特殊的函数在查询其帮助时需要在函数名上加引号，比如“?+”就查不到加法的帮助信息，而“?’+’”或者“help('+')”则可顺利查到加法帮助信息，类似的还有if、for等。

一般来说，帮助窗口会显示一个函数所属的包、用途、用法、参数说明、返回值、参考文献、相关函数以及示例，这些信息是相当丰富的。当

⁷丁国徽已将几本官方手册翻译为中文，R-intro的中文翻译文档可以在官方网站上下载，其它中文手册可从<http://www.biosino.org/R/R-doc/>得到；R for Beginners也已由本书作者及其他几位合作者翻译为中文，可从“统计之都”获得：<http://cos.name>

然，更多情况下是我们并不知道函数名称是什么，此时也可以使用搜索功能，即函数`help.search()`（几乎等价于双问号“??”），例如我们想知道方差分析的函数名称，则可输入命令`help.search('analysis of variance')`，弹出的信息窗口会显示与搜索关键词相关的所有函数，如`aov()`等。

知道名称以后，接下来我们就可以通过前面讲到的“?”来查询具体函数的帮助信息。函数`help.start()`可以打开一个网页浏览器用来浏览帮助。如果这些帮助功能还不够用，例如有时候需要的函数在已经安装的包中找不到，那么可以到官方网站上搜索：<http://www.r-project.org/search.html>，网站上的邮件列表导航（Mailing List Archives）也是很有用的资源，其中有大批统计相关学科的著名教授以及世界各地的统计研究者和R爱好者在那里用邮件的方式公开回答各种关于R的问题。

我们在此如此强调帮助系统，目的在于告诉读者，要想学好R语言，除了阅读相关书籍资料，也应该自己多多动手利用信息资源解决自己的问题。

2.3 安装R语言

读者在进入下一章学习之前，可以先将R安装在自己的计算机上，以便阅读时可以随时打开R进行实际操作。R软件可以在多种操作系统上运行，包括Windows、Linux以及Mac OS X等，进入官方网站主页会发现左栏有“Download”一项（CRAN），点击进入便可以看到世界各地的R镜像，任意选择一个进入（比如选择美国加州大学伯克利分校<http://cran.cnr.Berkeley.edu>），我们就会看到R安装程序和源代码的下载页面，此时只需要根据自己的操作系统选择相应的链接进入下载即可，例如Windows用户应该选择“Windows (95 and later)”进入，然后下载基础安装包（base），其中R-*.*-win32.exe字样⁸的链接便是Windows安装程序；Linux用户则可根据具体的系统如RedHat、Ubuntu等选择对应的链接。

由于R是完全开放源代码的，所以我们可以自由修改代码以构建符合自己需要的程序，但是一方面这需要一定的其它程序语言技能（典型的如C语言），因为R的很多基础函数都是用C写的，另一方面，对于绝大多数用户，

⁸*.*表示版本号，例如2.3.1；本书的程序版本是2010年5月发布Windows 2.11.1版，从2000年2月至今，R已经更新了30多个版本，更新速度非常快

其实没有必要对基础包进行修改—既然是开源软件，其代码必然受到很多用户“监视”，这样就会最大程度减少程序错误、优化程序代码，而且我们也可以在R里面自定义函数，这些函数可以保存起来，以后同样还能继续使用。对比起来，现今的商业统计软件都将源代码和计算过程封闭在用户完全不知道的“黑匣子”中，而在用户界面上花大量的功夫，这对统计来说，毫无疑问并非长久之计。我们相信，随着读者对R的深入了解，一定能体会到这个软件的真正强大之处。

下面我们以一个简单的例子开始R图形之旅，见图2.1。

```
1 > # 生成x (从-3到3长度为200的数列) 和y (正态密度)
2 > x = seq(-3, 3, length = 200)
3 > y = dnorm(x)
4 > # 建立图形框架
5 > plot(x, y, type = "n")
6 > # 设定5%和95%分位数之间的X点和相应密度值
7 > xx = seq(-1.65, 1.65, length = 100)
8 > yy = c(0, dnorm(xx), 0)
9 > xx = c(-1.65, xx, 1.65)
10 > # 灰色多边形
11 > polygon(xx, yy, col = "gray", border = NA)
12 > # 添加密度曲线
13 > lines(x, y)
14 > # 文本标注
15 > text(0, 0.05, "$P(-1.65 < X < 1.65) = 90\\%$")
```

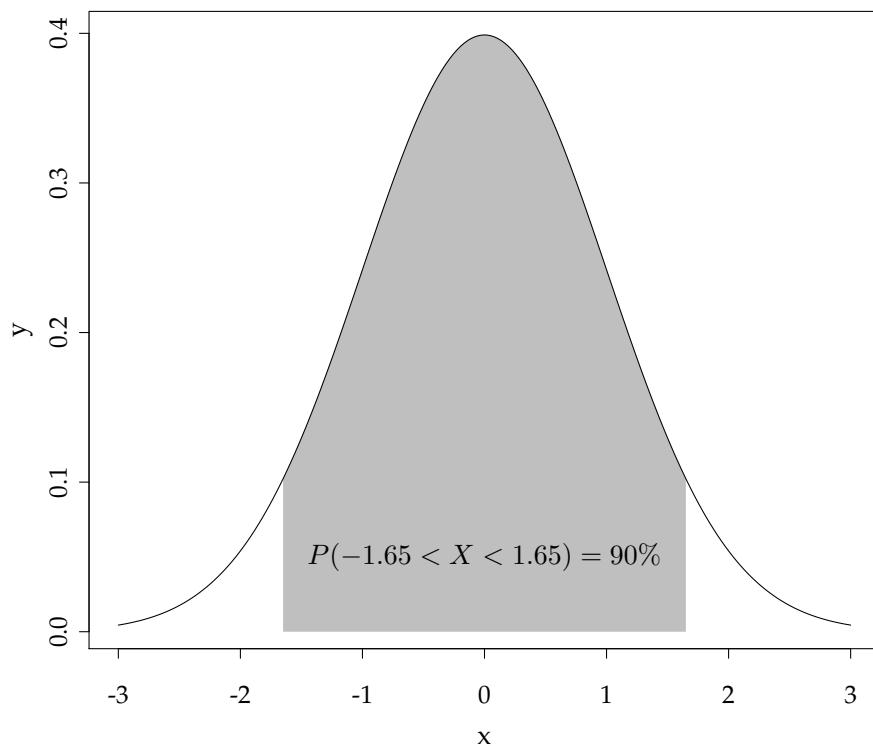


图 2.1: 正态分布密度曲线及其5%到95%分位数区间表示

第三章 细节

“那么，关于绳子的话题就谈到这里吧。”福尔摩斯微笑着说，“现在，我们来看看包裹纸吧。牛皮纸，带有一股明显的咖啡味。怎么，你还没有观察到？我想，肯定没有检查过。地址的字写得很零乱：‘克罗伊登十字大街S·库欣小姐收’，是用笔尖很粗的钢笔写的，或许是一支J字牌的，墨水很差。‘克罗伊登’一词原来是拼写的字母‘i’，后来才被改成字母‘y’的。而且，这个包裹是一个男人直接寄的—字体很明显是男人的字体—这个男人文化程度不高，对克罗伊登镇也不怎么熟悉。到目前为止，一切都顺利！这个纸盒子是一个半磅装甘露烟草盒。除了盒子右下角有两个指印外，再没有明显的痕迹。里面装的粗盐是用来保存兽皮和其他粗制商品的那种。埋在盐里的就是这奇怪的东西。”

—柯南·道尔《硬纸盒子》

统计图形都是通过相应的图形函数生成的，R当中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。

R的图形参数可以通过函数`par()`预先全局设置，也可以在具体作图函数（如`plot()`、`lines()`等）中设置临时参数值；二者的区别在于前者的设置会一直起作用，除非将图形设备（参见附录B.5）关闭，而后的设置只是临时性的，不会影响后面其它作图函数的图形效果。函数`par()`中涵盖了大部分图形参数，因此专用一节讲述。

3.1 *par()*函数的参数详解

函数*par()*可以用来设置或者获取图形参数，*par()*本身（括号中不写任何参数）返回当前的图形参数设置（一个list）；若要设置图形参数，则可用*par(tag = value)*的形式，其中tag的详细说明参见下面的列表，value就是参数值，例如：

```
1 > # 设置边距参数和背景色
2 > par(mar = c(4, 4, 1, 0.5), bg = "yellow")
```

目前*par()*函数涉及到的图形参数大约有70个，这里只是选取其中40多个常用且较易理解的参数进行解释说明如下列表，其它参数请参阅R帮助?*par*。

adj 调整图中字符的相对位置；取值：长度为2的数值向量，分别表示字符串矩形框的左下角相对坐标点(x, y)位置的调整，向量的两个数值一般都在[0, 1]范围内（有些图形设备中也可以超出此范围），表示字符串以左下角为基准、根据自身的宽度和高度分别向左和向下移动的比例，默認為c(0.5, 0.5)。例如c(0, 0)表示整个字符串（串）的左下角对准设定的坐标点，而c(1, 0)则表示字符串横向移动了自身宽度的距离，而纵向不受影响。具体示例参见图3.1左上图

ask 切换到下一个新的作图设备（通常是作一幅新图）时是否需要用户输入（敲回车键或点鼠标）；TRUE表示是；FALSE表示否。当有多幅图将逐一出现而需要按顺序一步步在图形设备上展示时很有用，这种情况下若设置ask为TRUE，那么作图时每一副新图的出现都要先等待用户输入，否则所有的图将会一闪而过

bg 设置图形背景色；关于颜色值的设置请参见4.1节

bty 设置图形边框样式；取值为字符o, l, 7, c, u,]之一；这些字符本身的形状对应着边框样式，比如（默認為）o表示四条边都显示，而c表示不显示右侧边，参见图3.2四幅图的边框样式

cex 图上元素（文本和符号等）的缩放倍数；取值为一个相对于1的数值（默認為1）。具体的细节缩放可以通过如下参数设置（默認為1）：

cex.axis 坐标轴刻度标记的缩放倍数

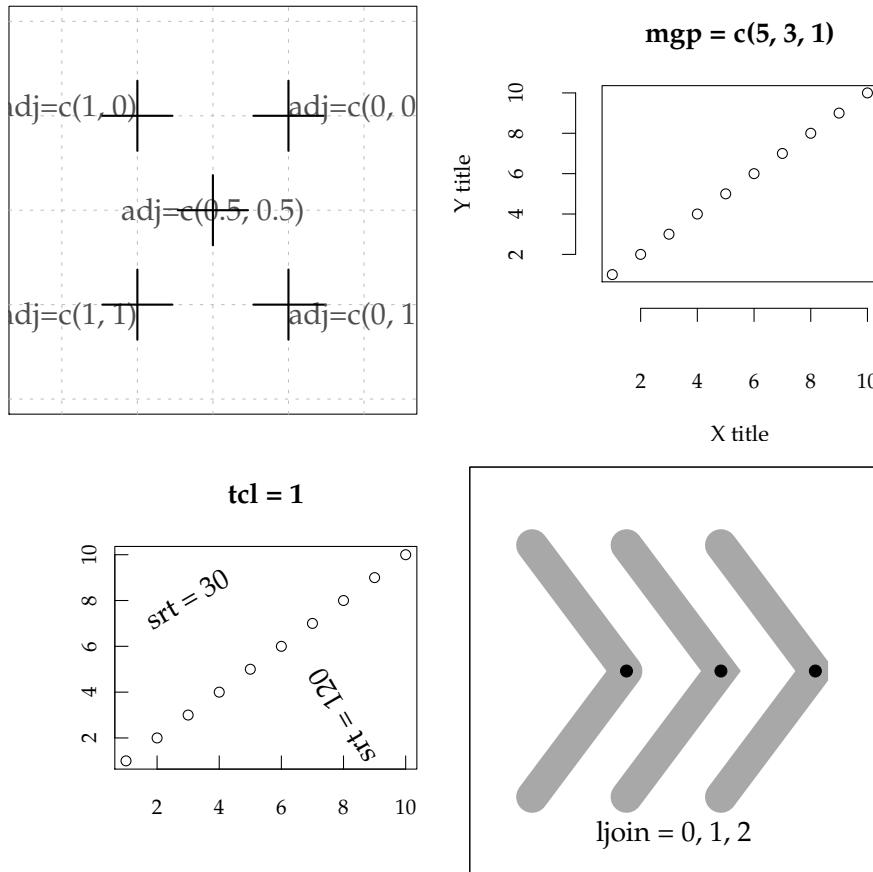


图 3.1: 左上: `adj`用于字符相对位置的调整, “+” 表示真实坐标点的位置, 通过一个长度为2的向量 $c(x, y)$ 可以分别调整字符在横纵坐标上相对平移的位置; 右上: 坐标轴元素的边界距离, 参数 `mgp` 的三个数值分别控制了坐标轴标题、坐标轴刻度数字以及坐标轴线到图形的距离; 左下: `tcl` 控制了坐标轴刻度线的方向和长度, `srt` 控制了字符串的旋转角度; 右下: 线条相交处的样式。

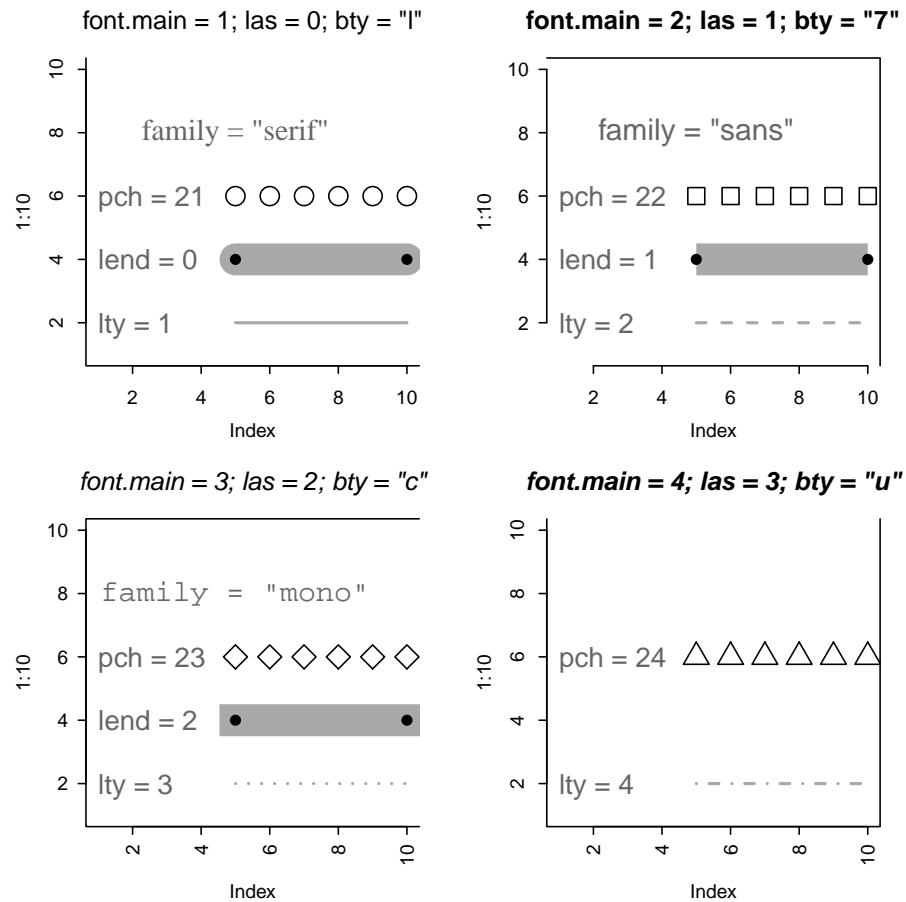


图 3.2: 四幅图形演示了不同的图形边框（上右开、上右闭、右开和上开）、字体样式（正常、粗体、斜体和粗斜体）、字体族（衬线、无衬线、等宽、符号）、坐标轴标签样式、点样式（圆圈、方框、菱形、三角）、线末端样式和线样式（实线、虚线、点线、点划线）。

cex.lab 坐标轴标题的缩放倍数

cex.main 图主标题的缩放倍数

cex.sub 图副标题的缩放倍数

col 图中符号（点、线等）的颜色；取值参见4.1节。与cex参数类似，具体的细节颜色也可以通过如下参数设置：

col.axis 坐标轴刻度标记的颜色

col.lab 坐标轴标题的颜色

col.main 图主标题的颜色

col.sub 图副标题的颜色

family 设置文本的字体族（衬线、无衬线、等宽、符号字体等）；标准取值有：`serif`, `sans`, `mono`, `symbol`，参见图3.2坐标(2, 8)处的文本；`family = 'symbol'`的情况没有显示出来

fg 设置前景色（若后面没有指定别的颜色设置，本参数会影响几乎所有的后续图形元素颜色，若后续图形元素有指定的颜色设置，那么只是影响图形边框和坐标轴刻度线的颜色）；颜色值参见4.1节。

font 设置文本字体样式；取值为一个整数；通常1、2、3、4分别表示正常、粗体、斜体和粗斜体¹，4.6节有进一步的介绍，参见图3.2的图主标题字体

font.axis 坐标轴刻度标签的字体样式

font.lab 坐标轴标题的字体样式

font.main 图主标题的字体样式

font.sub 图副标题的字体样式

lab 设置坐标轴刻度数目（R会尽量自动“取整”²）；取值形式`c(x, y, len)`：x和y分别设置两轴的刻度数目，len目前在R中尚未生效，因此设置任意值都不会有影响（但用到lab参数时必须写上这个参数）

¹对于添加文本，`text()`函数及其`vfont`参数可以设置更为详细的字体族和字体样式；参见这两个演示：`demo(Hershey)`和`demo(Japanese)`，前者演示Hershey向量字体，后者演示日语的表示。

²尽量向0.5、1或10的幂次靠近

las 坐标轴标签样式；取0、1、2、3四个整数之一，分别表示“总是平行于坐标轴”、“总是水平”、“总是垂直于坐标轴”和“总是竖直”。仔细观察图3.2中四幅图的不同坐标轴标签方向

lend 线条末端的样式（圆或方形）；取值为整数0、1、2之一（或相应的字符串'round'，'mitre'，'bevel'），注意后两者的细微区别³

lheight 图中文本行高；取值为一个倍数，默认为1

ljoin 线条相交处的样式；取值为整数0、1、2之一（或相应的字符串'round'，'mitre'，'bevel'），分别表示画圆角、画方角和切掉顶角，观察图3.1的三个直角的顶点

lty 线条虚实样式：0⇒不画线，1⇒实线，2⇒虚线，3⇒点线，4⇒点划线，5⇒长划线，6⇒点长划线；或者相应设置如下字符串（分别对应前面的数字）：'blank'，'solid'，'dashed'，'dotted'，'dotdash'，'longdash'，'twodash'；还可以用由十六进制的数字组成的字符串表示线上实线和空白的相应长度，如'F624'，详细解释请参见4.3一节。

lwd 线条宽度；默认为1

mar 设置图形边界空白宽度；按照“下、左、上、右”的顺序，默认为c(5, 4, 4, 2) + 0.1

mex 设置坐标轴的边界宽度缩放倍数；默认为1，本参数会影响到mgp参数

mfrow, mfcoll 设置一页多图；取值形式c(nrow, ncol)长度为2的向量，分别设置行数和列数，参见附录B.2

mgp 设置坐标轴的边界宽度；取值长度为3的数值向量，分别表示坐标轴标题、坐标轴刻度线标签和坐标轴线的边界宽度（受mex的影响），默认为c(3, 1, 0)，意思是坐标轴标题、坐标轴刻度线标签和坐标轴线离作图区域的距离分别为3、1、0；参见图3.1右上方小图

oma 设置外边界（Outer Margin）宽度；类似mar，默认为c(0, 0, 0, 0)，当一页上只放一张图时，该参数与mar不好区分，但在一页多图的情况下就容易可以看出与mar的区别

³仔细观察图3.2中宽线条中黑点的位置，在画线时，这些线条的起点和终点（分别用图中的两个黑点表示）都是选择同样的坐标位置！

pch 点的符号; `pch = 19`⇒实圆点、`pch = 20`⇒小实圆点、`pch = 21`⇒圆圈、`pch = 22`⇒正方形、`pch = 23`⇒菱形、`pch = 24`⇒正三角尖、`pch = 25`⇒倒三角尖, 其中, 21-25可以填充颜色(用`bg`参数), 参见图4.3

pty 设置作图区域的形状; 默认为'`m`': 尽可能最大化作图区域; 另外一种取值'`s`', 表示设置作图区域为正方形

srt 字符串的旋转角度; 取一个角度数值, 参见图3.1左下方小图中分别旋转 30° 和 120° 的字符串

tck 坐标轴刻度线的高度; 取值为与图形宽高的比例值(0到1之间); 正值表示向内画刻度线, 负值表示向外; 默认为不使用它(设为NA), 而使用`tcl`参数

tcl 坐标轴刻度线的高度; 取一个与文本行高的比例值; 正负值意义类似`tck`, 默认值为-0.5, 即向外画线, 高度为平行文本高; 观察图3.1左下角小图的坐标轴刻度线

usr 作图区域的范围限制, 取值长度为4的数值向量`c(x1, x2, y1, y2)`, 分别表示作图区域内x轴的左右极限和y轴的上下极限; 注意, 若坐标取了对数(参见`xlog, ylog`两个参数), 那么实际上设置的极限都是10的相应幂次

xaxs, yaxs 坐标轴范围的计算方式; 默认'`r`': 先把原始数据的范围向外扩大4%, 然后用这个范围画坐标轴; 另外一种取值'`i`', 表示直接使用原始数据范围; 实际上还有其它的坐标轴范围计算方式, 但是鉴于它们目前在R中都尚未生效, 所以暂不加介绍

xaxt, yaxt 坐标轴样式; 默认'`s`'为标准样式; 另外一种取值'`n`', 意思是不画坐标轴

xlog, ylog 坐标是否取对数; 默认`FALSE`

xpd 对超出边界的图形的处理方式; 取值`FALSE`: 把图形限制在作图区域内, 出界的图形截去; 取值`TRUE`: 把图形限制在图形区域内, 出界的图形截去; 取值`NA`: 把图形限制在设备区域内。这些区域的说明参见下文和图3.3

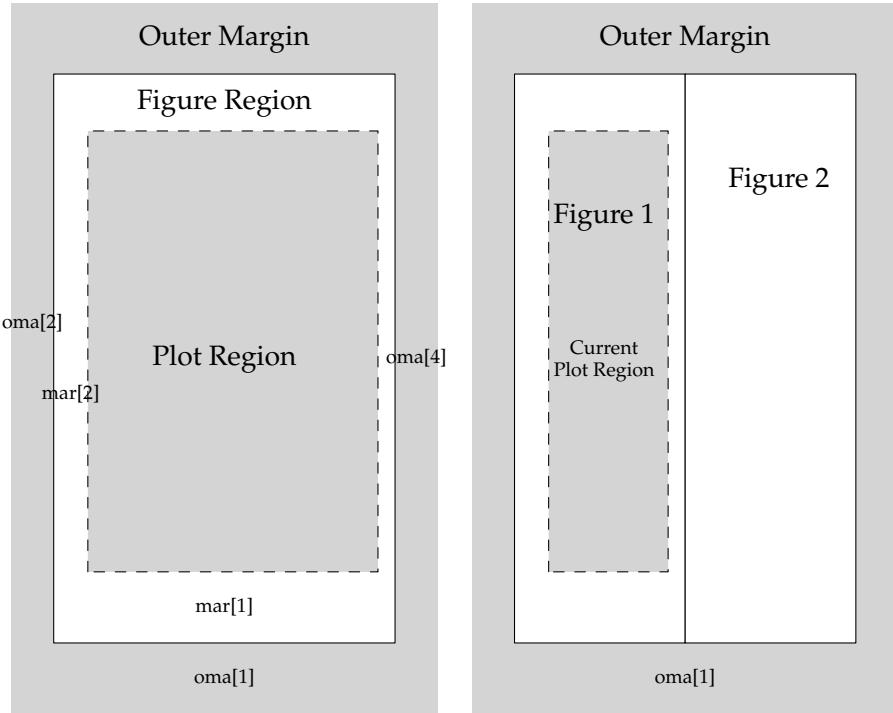


图 3.3: 图形的各种区域和边界说明: 作图区域 (当前作图区域)、图形区域和设备区域; 图形边界和外边界。

整个作图设备实际上可以分为三个区域, 分别是: “作图区域 (Plot Region)”、“图形区域 (Figure Margin)” 和 “设备区域 (Device Region)”, 这三个区域也对应着两种边界: “图形边界 (Figure Margin)” 和 “外边界 (Outer Margin)”, 这些概念对于初学者来说可能会感到迷惑, 然而图3.3是一个很好的说明。R中的图形都是作在一个图形设备中, 最常见的图形设备就是一个图形窗口, 也可以在其它设备中 (参见附录B.5); 整个设备内的区域就称为设备区域, 也就是图3.3中最大的灰色区域, 图形区域是设备区域内的白色实框方形区域, 最里面的灰色虚框区域就是作图区域, 我们的图形实体部分就作在这个区域; 从设备区域的边界向内, 到图形区域之间这一段称为外边界 (用`oma`参数设定), 图形区域边界再向内到作图区域的边界称为图形边界 (用`mar`参数设定)。图3.3的左图是一页一图的展示, 右

图则是一页多图的展示，该展示更清楚地说明了`oma`与`mar`参数的区别，因为一页一图的情况下，外边界和图形边界完全融合在一起，很难分辨。

下面列表中的九组参数只能通过`par()`函数调用⁴，而在其它作图函数中不可设置（否则会导致错误或者被忽略）：

- `ask`
- `fig, fin`
- `lheight`
- `mai, mar, mex, mfcoll, mfrow, mfg`
- `new`
- `oma, omd, omi`
- `pin, plt, ps, pty`
- `usr`
- `xlog, ylog`

介绍完上面的参数之后，我们顺便提一下关于`par()`的常用技巧。本节开头提到过，这个函数会“永久性”改变作图设置，我们有时并不想要这种功能，特别是在一幅图作完之后到准备下一幅图时，我们可能希望之前的参数可以被“还原”回来，此时，我们就需要在一幅图开始之前先把作图参数保存到一个对象中，比如`op = par()`，然后我们可以在作这幅图的过程中用`par()`函数任意更改设置以适合需要，作完这一幅图之后，我们再用`par(op)`语句把之前保存的参数设置“释放”出来，这样，中间过程对图形参数的更改就不再会影响到下一幅图。当然，也可以每作完一幅图都把图形设备关掉，然后再作下一幅图，这样也能达到目的，只是稍显麻烦而已，尤其是有时候对一幅图形反复重作、调整、比较，那时不断关闭、打开图形设备就显得更繁琐了。

⁴与此对应的是，有些参数同样可以在别的作图函数中调用，如`las`参数：`plot(..., las = 1)`；但要提醒读者注意，在其它函数中即使调用与`par()`相同的参数，也可能会有不同效果，典型的如`col`、`pch`等参数，请注意查看相应函数帮助

3.2 *plot()*及相关函数的参数说明

R中最普通的作图函数就是*plot()*函数，它是一个泛型函数（R中的泛型函数与Java、C++等语言可能有所不同，附录A.1中会对它进行介绍），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。

先介绍*plot()*的通用参数：

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⁵；'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3.4的九幅图清楚说明了这九种类型

main 主标题；也可以在作图之后用数*title()*添加上，参见4.6节

sub 副标题；同上

xlab x轴标题；同上

ylab y轴标题；同上

asp 图形纵横比 y/x ；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

⁵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！

```

1 > par(mfrow = c(3, 3), mar = c(2, 2.5, 3, 2))
2 > for (i in c("p", "l", "b", "c", "o", "h", "s", "S",
3 +   "n")) {
4 +   plot(c(1:5, 5:1), type = i, main = paste("Plot type: \\"", 
5 +     i, "\"", sep = ""), xlab = "")
6 + }

```

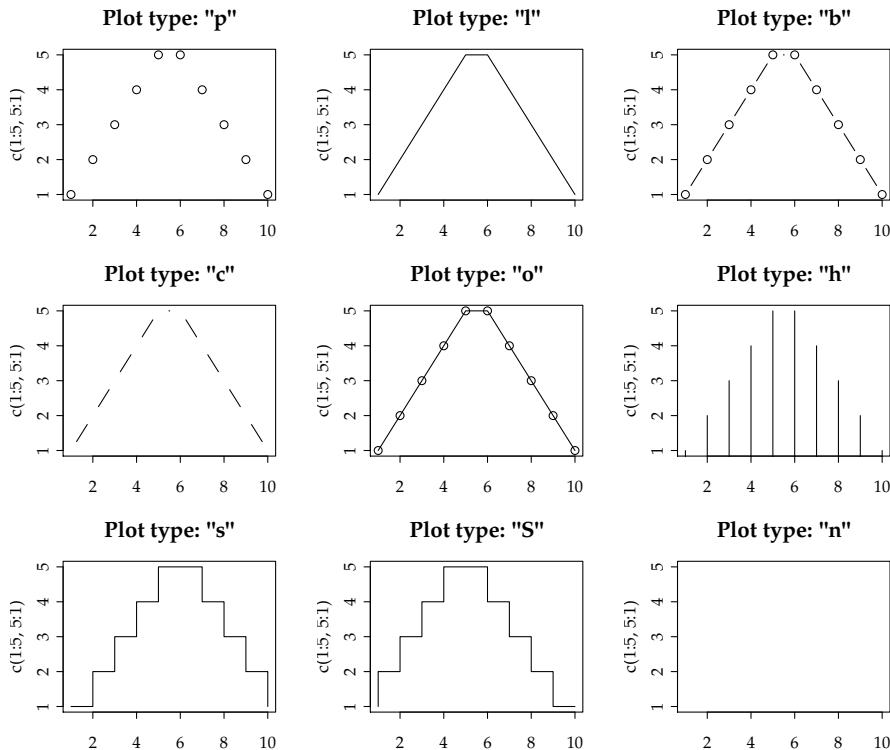


图 3.4: plot()作图的九种样式类型: 点、线、点线(不相接)、擦掉点的线、点线(相接)、垂线、阶梯(水平起步)、阶梯(垂直起步)、无。

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⁶

log 坐标是否取对数，**TRUE**或者**FALSE**

ann 一些默认的标记是否显示，如坐标轴标题和图标题

axes 是否画坐标轴；注意只会影响到是否画出坐标轴线和刻度，不会影响坐标轴标题

frame.plot 是否给图形加框；可以查阅*box()*函数，作用类似但功能更详细

panel.first 在作图前要完成的工作；这个参数常常被用来在作图之前添加背景网格（参见4.5节）或者添加散点的平滑曲线，比如**panel.first = grid()**

panel.last 作图之后要完成的工作；与上一个参数类似

... 其它常用参数如下：

col, pch, cex, lty, lwd 这些参数的意思与*par()*中的参数基本相同，有所区别的是，*par()*中这些参数只能设置一个值，而这里可以对它们设置一个向量，这个向量的值将依次运用到各个元素上，若向量长度短于元素个数，那么向量会被循环使用，直到所有的元素都被画出来，事实上，向量的循环使用也是R图形参数的一大特点

bg 背景色；注意与*par()*不同的是，这里设置的只是可以画背景色的点⁷的背景色，而不是设置整幅图形的背景色！

至此，我们基本上已经介绍完所有常用的图形参数，但这些参数的作用没有必要全都烂熟于心；本章可以仅作为参考资料，需要时查阅即可。后面的章节中，我们就会看到一些参数在图形元素和统计图形中的微调作用（但也许不是微小作用）。

⁶但我们可以*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，因为一般来说作图函数不会返回任何值（或者说返回值为空：**NULL**）

⁷在4.2节中说明了什么类型的点可以画背景色

第四章 元素

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》

任何一幅统计图形都是由最基础的图形元素构成的，这些元素我们并不陌生，无非就是颜色、点、线（直线、曲线、线段甚至箭头）、矩形、任意多边形、文本以及图例。R提供了相应的一系列函数，用以向已有的图形中添加图形元素。事实上，R的所有作图函数分为两类，一类是高层函数（high-level），用以生成新的图形，另一类就是低层函数（low-level），这一类指的正是绘制图形元素的这些基础函数。

4.1 颜色

默认情况下，R中颜色的设置主要需要依靠**grDevices**包的支持，其中提供了大量的颜色选择函数以及生成函数，以及几种预先设置好的调色板(Palette)，用以表现不同的主题。我们把**grDevices**包中所有关于颜色的函数大致分为三类：固定颜色选择函数、颜色生成和转换函数和特定颜色主题调色板，这些函数将在下面三小节中详细介绍。

4.1.1 固定颜色选择函数

固定颜色选择函数也就是R提供的它自带固定种类的颜色，主要是函数`colors()`以及`palette()`：

colors(), colours() 这两个函数完全一样，只是英文的两种不同拼写而已，它们不需要任何参数，会生成657种颜色名称，如：`'beige'`（米色）、`'bisque'`（桔黄色）、`'chocolate'`（巧克力色）、`'cyan'`（青色）、`'gold'`（金黄色）、`'ivory'`（象牙色）、`'lavender'`（浅紫色）等。
下面的代码表示从`colors()`中随机抽取20种颜色¹：

```
1 > sample(colors(), 20)

[1] "gray86"           "springgreen"      "mediumturquoise"
[4] "lightpink2"        "yellow4"          "ghostwhite"
[7] "oldlace"           "grey99"           "burlywood1"
[10] "mediumorchid1"    "goldenrod3"       "orangered4"
[13] "brown2"            "khaki1"          "gray47"
[16] "darkorchid4"       "gold3"            "gray27"
[19] "gray60"            "grey85"
```

有兴趣观看所有657种颜色的读者可以试着运行下面的语句，其中`pdf()`函数的作用是打开一个作图设备，详情参见附录B.5节，可以把参数`'C:/colors-bar.pdf'`替换为任意一个可以读写的路径²，条形图`barplot()`的说明参见5.4小节。最后结果是在设定文件路径上获得一个PDF文件，展示所有颜色名称及其对应的颜色。

¹提醒读者注意，`sample()`是随机抽样函数，因此重复这个语句每次得到的结果可能会不一样，这是正常的

²Windows用户注意路径的写法，不能用通常的单反斜杠“\”，而应该把路径中的单反斜杠换成双反斜杠“\\”或者用单斜杠“/”；如`'D:/colors.pdf'`或`'D:\\colors.pdf'`

```

1 > pdf("C:/colors-bar.pdf", height = 120)
2 > par(mar = c(0, 10, 3, 0) + 0.1, yaxs = "i")
3 > barplot(rep(1, length(colors())), col = rev(colors()),
4 +   names.arg = rev(colors()), horiz = TRUE, las = 1,
5 +   xaxt = "n", main = expression("Bars of colors in" ~
6 +     italic(colors())))
7 > dev.off()

```

palette() 调色板函数；用法`palette(value)`，这个函数用来设置调色板或者获得调色板颜色值；注意，实际上这个函数的结果可能并非“固定”颜色，但是只要设定好了调色板，它的取值就不会再改变（直到下一次重新设定调色板）。如果不写任何参数，那么该函数返回当前的调色板设置，即一个包含当前调色板中所有颜色的向量；若参数长度为1则将当前调色板重新设置为以该参数为名称的调色板，目前这种参数只有'default'这一种，即设置为默认调色板：`palette('default')`；若参数为一个颜色向量，那么将当前调色板中的颜色更改为该参数表示的颜色。如下例：

```

1 > # 默认的调色板颜色
2 > palette()

[1] "black"      "red"        "green3"      "blue"        "cyan"        "magenta"
[7] "yellow"     "gray"

1 > # 重新设置调色板为colors()的前10种颜色
2 > palette(colors()[1:10])
3 > # 更改后的调色板颜色
4 > palette()

[1] "white"       "aliceblue"    "antiquewhite"
[4] "antiquewhite1" "antiquewhite2" "antiquewhite3"
[7] "antiquewhite4" "aquamarine"   "aquamarine"
[10] "aquamarine2"

1 > # 恢复默认调色板
2 > palette("default")

```

调色板的好处在于，我们可以在R中使用一个整数来表示颜色，而这个整数对应的颜色就是调色板中相应位置的颜色，比如在某作图函数中调用参数`col = 2`表示取调色板中第2种颜色。若整数值超过了调色板颜色向量的长度，那么R会自动取该整数除以调色板颜色向量长度的余数。

4.1.2 颜色生成和转换函数

R提供了一系列利用颜色生成原理如RGB模型（红绿蓝三原色混合）、HSV色彩模型（色调、饱和度和纯度）、HCL色彩模型（色调、色度和亮度）和灰色生成模型等构造的颜色。颜色的构造原理比较复杂，超出了本书讨论范围，因此这里仅对相关函数的用法作介绍。

rgb() 红绿蓝三原色混合，用法`rgb(red, green, blue, alpha, names = NULL, maxColorValue = 1)`，其中前四个参数都取值于区间 $[0, \text{maxColorValue}]$ ，`names`参数用来指定生成颜色向量的名称。这里前三个参数不用过多解释，值越大就说明那种颜色的成分越高；可能`alpha`我们不太熟悉，它指的是颜色的透明度，取0表示完全透明，取最大值表示完全不透明（默认），透明度在统计图形中有着重要地位，因为它具有一个非常有用的性质—透明度可以叠加，即：两个或多个带有透明色的图形元素重叠在一起时，重叠部分的透明度会变小；这在某些统计图形中可以找到很好的应用，例如当散点图中点的数目过多而导致大量的点相互重叠时，我们可以使用透明色来看清图中的深层规律，其中一个直接的规律就是二维密度，点重叠越密集，则颜色越深（由于透明度的叠加），该处的密度值也越大，图5.8给出了一个半透明色应用的示例。

hsv() 用色调（Hue）、饱和度（Saturation）和纯度（Value）来构造颜色，用法`hsv(h = 1, s = 1, v = 1, gamma = 1, alpha)`，前三个参数分别对应色调、饱和度和纯度，取值于区间 $[0, 1]$ ，参数`gamma`表示伽玛校正³（Gamma Correction）；`alpha`意思同上，但取值在区间 $[0, 1]$ 上

hcl() 用色调（Hue）、色度（Chroma）和亮度（Luminance）构造颜色，用法为`hcl(h = 0, c = 35, l = 85, alpha, fixup = TRUE)`；参数`h`取值在区间 $[0, 360]$ 上，可以将它想象为一个角度： 0° 表示红色， 120° 表示绿色， 240° 表示蓝色，中间的都是过渡色；参数`c`取值受`h`和`l`限制；参数`l`取值在区间 $[0, 100]$ 上，取值越大生成的颜色越亮；`alpha`意思同`hsv()`；`fixup`表示是否修正生成的颜色值，之所以要修

³对于在RGB颜色空间中的任意一种颜色 (r, g, b) ，伽玛校正的意思就是求`gamma`次幂，将颜色转换为 $(r^{\text{gamma}}, g^{\text{gamma}}, b^{\text{gamma}})$

正，是因为有些搭配生成的RGB颜色(r, g, b)可能出现某一个元素超过1的情形

gray0, grey0 生成灰色系列；只有一个参数level，表示灰度水平，取值在0到1之间，其中0表示纯黑色，而1表示纯白色；level取一个向量则可以生成一系列灰色值，如下例：

```
1 > gray(seq(0, 1, length = 5))
[1] "#000000" "#404040" "#808080" "#BFBFBF" "#FFFFFF"
```

熟悉十六进制的人应该能看出这些颜色都是由六个十六进制数字组成的，每两位数字（合起来取值从0到255）分别表示红绿蓝（RGB颜色）的比例，我们知道，当三原色完全混合时，生成的颜色是白色，上面结果的最后一个，#FFFFFF’正是纯白色。

除了颜色生成函数之外，**grDevices**包还提供了两种颜色转换函数，作用就是把一种颜色从一种颜色系统空间转移到另一种颜色系统空间中表示。这两个函数分别是：

rgb2hsv() 将RGB颜色转换为HSV颜色，用法`rgb2hsv(r, g = NULL, b = NULL, gamma = 1, maxValue = 255)`；所有参数意思已经在上面的列表中解释过，只是要注意，当r是一个矩阵时，另外两个参数g和b就应省略不写。下例中我们将一个 3×4 的RGB颜色矩阵传入函数`rgb2hsv()`，该函数会把每一列RGB颜色都转化为相应的HSV颜色。颜色矩阵的前三列分别是红、绿和蓝色，请观察和对比两种颜色系统的表示方法。

```
1 > # 赋值给变量 rgb.mat
2 > (rgb.mat = matrix(c(255, 0, 0, 0, 255, 0, 0, 0, 255,
3 +      10, 100, 200), nrow = 3))

 [,1] [,2] [,3] [,4]
[1,] 255     0     0    10
[2,]     0 255     0   100
[3,]     0     0 255   200

1 > rgb2hsv(rgb.mat)
```

```
[,1] [,2] [,3] [,4]
h     0 0.3333 0.6667 0.5877
s     1 1.0000 1.0000 0.9500
v     1 1.0000 1.0000 0.7843
```

col2rgb() 将任意一种R颜色值转换为RGB表示，用法`col2rgb(col, alpha = FALSE)`；参数`col`的取值可以有三种形式，第一种是`colors()`函数中的任意一种颜色名称（字符串），第二种是如`#rrggbb`十六进制形式的RGB颜色表示，第三种是一个整数，即调色板中相应位置的颜色。

```
1 > # 调色板中第 4 种颜色默认是蓝色
2 > col2rgb(4)
```

```
[,1]
red      0
green    0
blue    255
```

```
1 > # 黄色是由红绿混合得到的
2 > col2rgb("yellow")
```

```
[,1]
red    255
green  255
blue    0
```

```
1 > # 红蓝混合为紫色
2 > col2rgb("#FF00FF")
```

```
[,1]
red    255
green   0
blue   255
```

4.1.3 特定颜色主题调色板

前面介绍的颜色生成过程对于一般人来说也许显得太复杂，除了美术和绘图专业人士，大部分人可能并不懂颜色的透明度、饱和度等概念，那么在配制大批量颜色的时候可能会比较迷惑。此时，R提供了第三种选择，那就是特定颜色主题的调色板。这些调色板都用一系列渐变的颜色表现了特定的主题，例如彩虹颜色系列、白热化颜色系列、地形颜色系列等等。

rainbow() 顾名思义，就是用彩虹的颜色（“红橙黄绿青蓝紫”）来产生一系列颜色，用法`rainbow(n, s = 1, v = 1, start = 0, end = max(1, n - 1)/n, gamma = 1)`；参数n设定产生颜色的数目，参数s、v和gamma前文已经解释过；参数start和end设定彩虹颜色的一个子集，生成的颜色将从这个子集中选取，这个子集选取的大致分界线为：红色（red）为0，黄色（yellow）为1/6，绿色（green）为2/6，青色（cyan）为3/6，蓝色（blue）为4/6，红紫色（magenta）为5/6

heat.colors() 从红色渐变到黄色再变到白色（以体现“高温”、“白热化”）⁴

terrain.colors() 从绿色渐变到黄色再到棕色最后到白色（这些颜色适合表示地理地形⁵）

topo.colors() 从蓝色渐变到青色再到黄色最后到棕色

cm.colors() 从青色渐变到白色再到粉红色⁶

若想要获得更复杂更精细的颜色或调色板，不妨使用**grDevices**包中的颜色“插值”函数，如`colorRamp()`和`colorRampPalette()`函数，读者可以根据需要产生符合特定要求的调色板，以适应展示主题的需要，这里不详述这两个函数的使用方法。

对于缺乏耐心和兴趣去研究颜色的用户来说，附加包**RColorBrewer** (Neuwirth, 2007)也不失为一个好的选择。这个包提供了三类调色板，用户只需要指定调色板名称，就可以用包中的`brewer.pal()`函数生成颜色。这三类调色板包括：

连续型调色板 Sequential palettes，生成一系列连续渐变的颜色，通常用来标记连续型数值的大小

极端化调色板 Diverging palettes，生成用深色强调两端、浅色标示中部的系列颜色，可用来标记数据中的离群点

离散型调色板 Qualitative palettes，生成一系列彼此差异比较明显的颜色，通常用来标记分类数据

⁴请读者在R中运行`demo(image)`并观察第二幅等高线图

⁵运行`demo(persp)`并观察最后两幅关于火山的三维图形，以及运行`demo(image)`并观察第一幅等高线图

⁶运行`demo(image)`并观察最后几幅颜色图

每一类调色板下有若干种具体的实现，例如Blues是连续型调色板下的一种，用以蓝色主题的渐变颜色：

```

1 > library(RColorBrewer)
2 > brewer.pal(9, "Blues")

[1] "#F7FBFF" "#DEEBF7" "#C6DBEF" "#9ECAE1" "#6BAED6" "#4292C6"
[7] "#2171B5" "#08519C" "#08306B"
```

所有调色板名称及其展示参见图4.1。这些调色板并非一些简单的颜色组合，而是有一定科学根据的。例如离散型调色板下的颜色对大多数人来说都有较好的区分度，甚至色盲也可以辨认其中不同类的颜色。如果用户对颜色选取拿捏不准，不妨用这个包来生成颜色。实际上这个R包是一款叫ColorBrewer的产品的重新实现，更多信息可以访问网站<http://www.colorbrewer.org/>。

4.1.4 渐变色的简单原理及应用

在此我们特别用一节来讲述“渐变色”，原因在于在图形中应用渐变色往往能让图形看起来更美观、避免单调的颜色在图形中显得突兀。不难想象，所谓“渐变”，也就是逐渐变化的意思，这种变化必然对应着某种单调或非单调的（可导）函数，这些函数用来控制颜色值逐步变化。最简单的例子莫过于线性函数：从一种颜色值到另一种颜色值线性变化。比如我们在`rgb()`函数中用一元线性函数控制绿色在[0, 1]上的取值，同时将红色和蓝色分别控制为1和0，那么我们将得到从纯红色到黄色的一个颜色渐变。如：

```

1 > (x = rgb(1, seq(0, 1, length = 30), 0))

[1] "#FF0000" "#FF0900" "#FF1200" "#FF1A00" "#FF2300" "#FF2C00"
[7] "#FF3500" "#FF3E00" "#FF4600" "#FF4F00" "#FF5800" "#FF6100"
[13] "#FF6A00" "#FF7200" "#FF7B00" "#FF8400" "#FF8D00" "#FF9500"
[19] "#FF9E00" "#FFA700" "#FFB000" "#FFB900" "#FFC100" "#FFCA00"
[25] "#FFD300" "#FFDC00" "#FFE500" "#FFED00" "#FFF600" "#FFFF00"

1 > # 读者不妨用barplot(rep(1, 30), col = x)看看效果
```

显然本小节的内容与前一小节所讲到的调色板是一致的，只不过调色板是预先配置好了的渐变色系列；我们在这里“重复”讲述渐变色的简单

```

1 > layout(matrix(1:3, 3), heights = c(2, 1, 1))
2 > par(mar = c(0, 4, 0, 0))
3 > # 连续型: 18种
4 > display.brewer.all(type = "seq")
5 > # 极端化: 9种
6 > display.brewer.all(type = "div")
7 > # 离散型: 8种
8 > display.brewer.all(type = "qual")

```

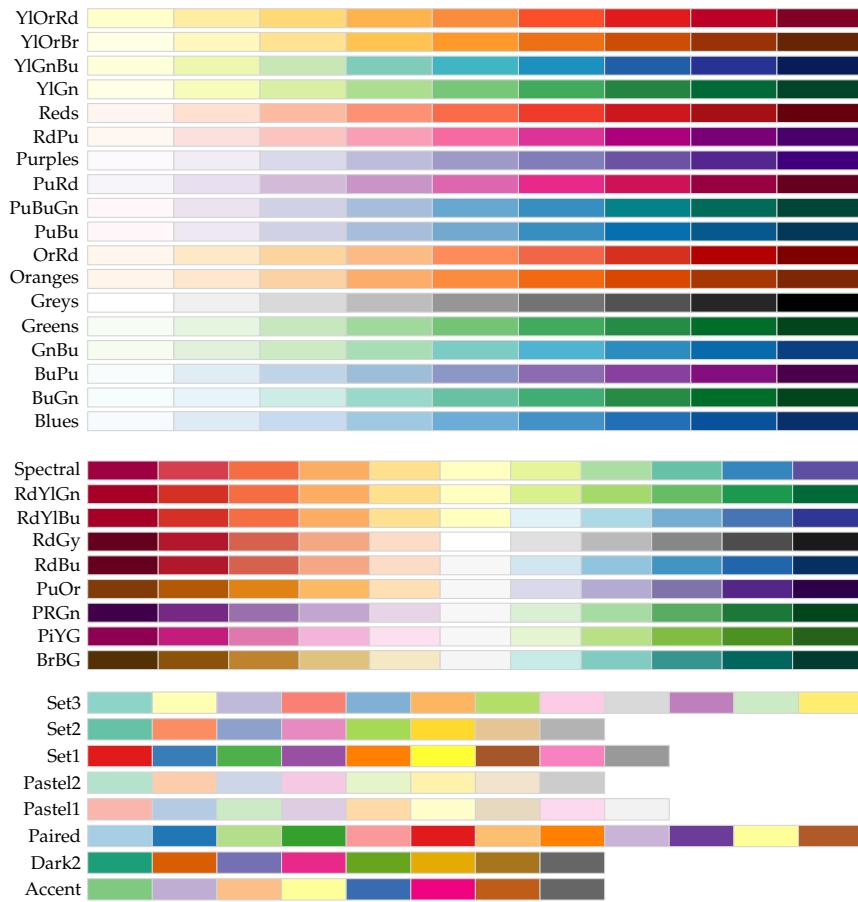


图 4.1: *RColorBrewer*包中所有调色板颜色的演示: 从上至下依次是连续型、极端化和离散型调色板。

```

1 > xx = c(1912, 1912:1971, 1971)
2 > yy = c(min(nhtemp), nhtemp, min(nhtemp))
3 > plot(xx, yy, type = "n", xlab = "Year", ylab = "Temperatures")
4 > for (i in seq(255, 0, -3)) {
5 +   yy = c(45, nhtemp - (nhtemp - min(nhtemp)) *
6 +         (1 - i/255), 45)
7 +   # rgb()中的绿色成分逐渐变小
8 +   polygon(xx, yy, col = rgb(1, i/255, 0), border = NA)
9 +   # Sys.sleep(0.05)
10 +  # 为了看清作图过程, 可以让每步循环延迟0.05秒
11 + }
12 > box()

```

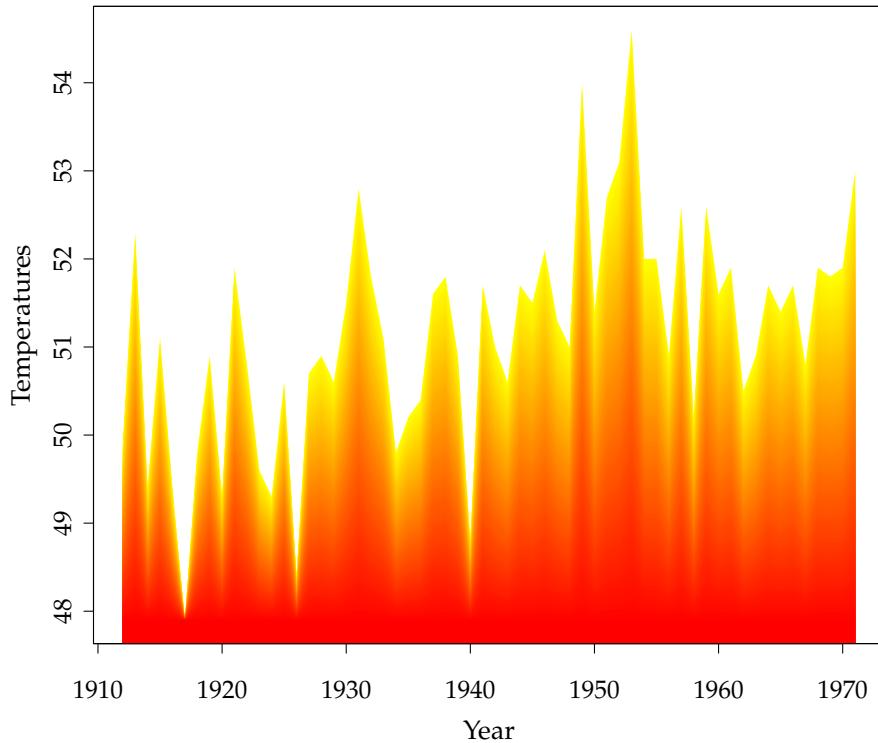


图 4.2: *New Haven*地区的年均气温 (1912~1971年): 统计图形的应用应该根据数据和事实的环境灵活选取图形的样式。本图的数据为1912年至1971年*New Haven*地区的年均气温。数据来源: *datasets*包中的*nhtemp*。

原理，目的是尽量让读者了解各种看起来比较神秘和高深的颜色使用招数的本质。

在本节的最后，我们要再次强调颜色的艺术性：根据不同的环境选用不同的颜色。虽然统计专业人士对绘画、美学可能不必深入了解，但是漂亮、适宜的统计图形总是受人欢迎的。作者曾经在Ross Ihaka的个人主页上看到一个很好的例子，网址在<http://www.stat.auckland.ac.nz/ihaka/Graphics/warming.html>，例子的内容是展示全球气候变暖的温度随年份变化的折线图，这幅图使用了从黄色到红色的渐变色，而这样的颜色恰好能体现温度的灼热感，对于警示温室效应来说，这种渐变色是极为恰当的（思考一下我们平时作折线图是否会考虑这样的颜色搭配）；Ross Ihaka并没有给出具体的代码，但通过前面的介绍和后面4.4小节的阅读，相信读者一定可以自己动手作出类似的折线图。这里我们也给出一段示例代码，核心部分在于控制多边形的col参数；事实上，这幅“折线图”是由多个颜色渐变的多边形重叠而成的，效果见图4.2（图中数据为真实气温数据）。

4.2 点

关于点的设置，我们既可以在很多作图函数中用等参数实现，也可以在用低层函数*points()*向已有图形中添加点时实现。后一种方法往往更灵活自由。*points()*用法：

```
1 > usage(points, "default")
points(x, y = NULL, type = "p", ...)
```

函数*points()*的参数如和等在3.1小节中我们其实已经都见过了，但这两个函数中相同名称的参数最主要的一点区别就是，前者可以使用向量，而后者只能接受一个单值作为参数，此外还有一点细小差异在于参数，它在函数*points()*中表示的是点的背景色而非图形的背景色。为了向图中添加一系列不同样式的点，我们可以使用向量作为*points()*的参数。

本节不再赘述参数的意思（读者可以自行查阅帮助），但有三点仍需特别说明一下。首先是*lwd*参数，我们知道这是设定线条宽度的，对于点来说，这个参数同样可以设置点的边缘“线条”宽度；其次，*pch*参数同样可以接受字符作为参数值，而不仅仅是数字；最后，参数*pch*取值从21~25的点可以填充背景颜色。注意观察，图4.3中21~25的点边缘线颜色和背景色

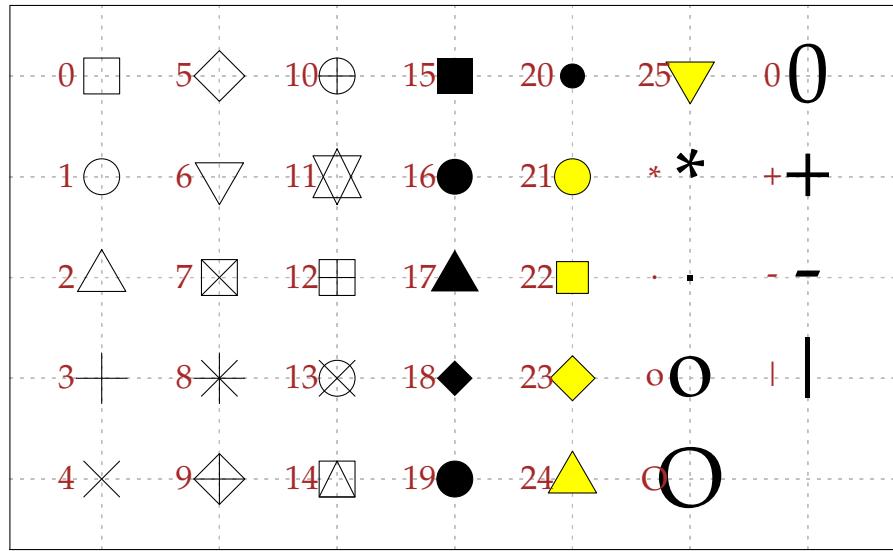


图 4.3: 点的类型: `pch`参数取值从1到25及其它符号。注意: 21~25的点可以填充背景颜色。本图的代码参考 `example(points)`, 或者简单用 `plot(0:25, pch = 0:25)` 观察。

是不同的, 而在25之后的点则采用了文本符号作为参数`pch`的值。图4.3中也有几对形状一样的点, 但是它们实质上是不一样的, 例如0、15和22对应的点都是正方形, 但前二者分别是空心正方形和实心正方形, 且都不可填充背景, 而22对应的点是空心正方形, 可以填充背景。

一次性作出不同样式的点对于统计分析来说是很有利的, 比如有时候我们可以把样本根据某一特征分为几组, 对每一组子样本都采用不同的样式作点图, 那么我们或许可以从图中点的散布情况发现组间的差异特征。图4.4给出了一个鸢尾花散点图的示例 (注意向量的使用)。

4.3 曲线、直线、线段、箭头、X-样条

在介绍函数`par()`时我们曾经提到过关于线条的一些参数设置, 例如`lwd`和`lty`等。类似地, 我们可以用函数`lines()`来向图中添加曲线⁷, 本书的

⁷这里所说的曲线本质上是一些线段的连接, 并非光滑的曲线

```

1 > # 先将鸢尾花的类型转化为整数1, 2, 3, 以便使用向量
2 > idx = as.integer(iris[["Species"]])
3 > plot(iris[, 3:4], pch = c(24, 21, 25)[idx], col = c("black",
4 +     "red", "blue")[idx], panel.first = grid())
5 > legend("topleft", legend = levels(iris[["Species"]]),
6 +     col = c("black", "red", "blue"), pch = c(24,
7 +         21, 25), bty = "n")

```

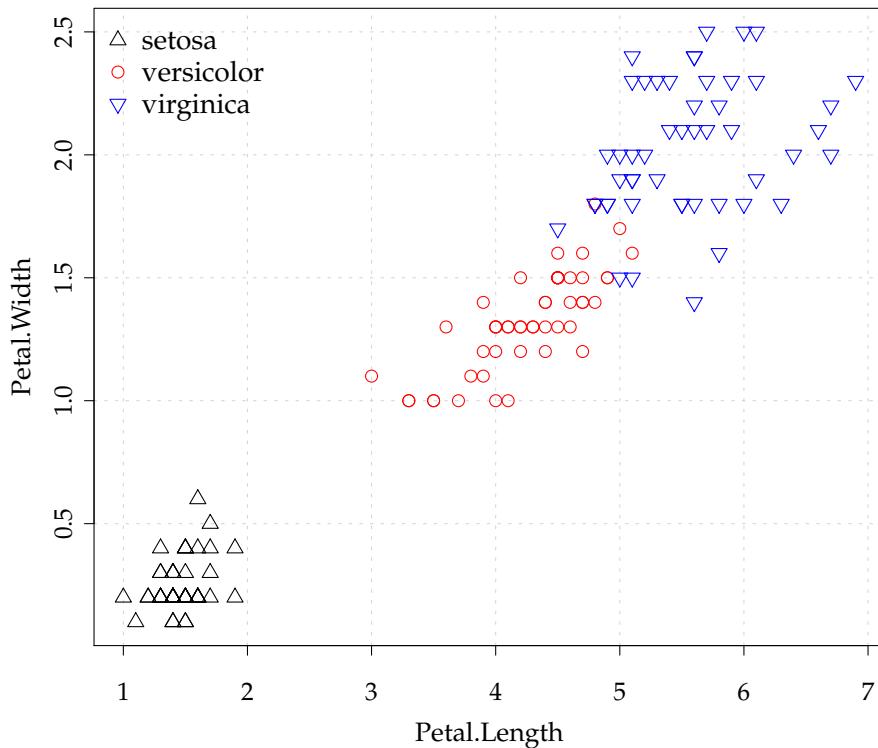


图 4.4: 鸢尾花的花瓣长宽散点图：我们可以给每一种鸢尾花标记不同类型点和颜色，这样可以帮助我们更清楚地区分鸢尾花类型。

图2.1已经使用该函数添加过正态密度曲线；下面主要补充说明一下关于线条样式lty的设定。

R中可以实现几乎无数种线条样式，因为它的lty参数相当灵活，除了取值0~6之外，可以根据一个十六进制的数字串（位数必须是偶数位，且非零）来设定线条的虚实，具体原理是这样：数字串的奇数位上的数字表示画相应长度的实线，然后偶数位上的数字则表示空缺相应的长度，这样就构成了一条虚线。例如，'A5'表示先画11单位长的实线，再接着画5单位长的空白，紧接着又画11单位长的实线，……，就这样重复下去，完成一条虚线；同理，'711911'表示：7单位长实线、1单位长空白、1单位长实线、9单位长空白、1单位长实线、1单位长空白。这个十六进制的数字串的最长长度限制为8位。

当设定type = 'h'时，col参数可以使用向量，此时各条竖线都将使用不同的颜色；除此情况之外，若其它参数使用了向量，那么只有向量的第一个元素会被使用，其它元素都将被忽略掉。

关于直线，我们在平面坐标系中只需要确定两个因素就可以确定它的位置：即斜率和截距。函数abline()就是用来添加直线的，参数同样可以使向量（这一点在低层函数中几乎普遍适用，所以后面不再重复说明）。函数用法如下：

```
1 > usage(abline)
abline(a = NULL, b = NULL, h = NULL, v = NULL,
       reg = NULL, coef = NULL, untf = FALSE, ...)
```

其中，a是截距，b是斜率，h是画水平线时的纵轴值，v是画垂直线时的横轴值，reg是一个能用函数coef()提取系数（包含斜率和截距）的R对象，典型的就是用线性模型（回归）生成的对象，系数是一个长度为2的向量，分别为截距和斜率；后面的...表示还可以传入其它参数（比如lty、col等）。

线段可以用函数segments()生成，用法如下：

```
1 > usage(segments)
segments(x0, y0, x1 = x0, y1 = y0, col = par("fg"),
          lty = par("lty"), lwd = par("lwd"), ...)
```

前四个参数表示线段的起点和终点坐标，后面的参数相信读者也都已经熟悉。

箭头可以用函数arrows()生成，用法如下：

```

1 > # 不作图, 只画出框架, 且指定坐标轴范围
2 > plot(1:10, type = "n", xlim = c(0, 10), ylim = c(0,
3 +      10))
4 > # 10个正态随机数绝对值的波动线
5 > lines(1:10, abs(rnorm(10)))
6 > # 不同的直线
7 > abline(a = 0, b = 1, col = "gray")
8 > abline(v = 2, lty = 2)
9 > abline(h = 2, lty = 2)
10 > #添加文本
11 > text(8, 3, "abline(a = 0, b = 1)")
12 > # 添加箭头
13 > arrows(8, 3.5, 6, 5.7, angle = 40)
14 > # 参数用了向量: 不同灰度的线段
15 > segments(rep(3, 4), 6:9, rep(5, 4), 6:9, col = gray(seq(0.2,
16 +      0.8, length = 4)))
17 > text(4, 9.8, "segments")

```

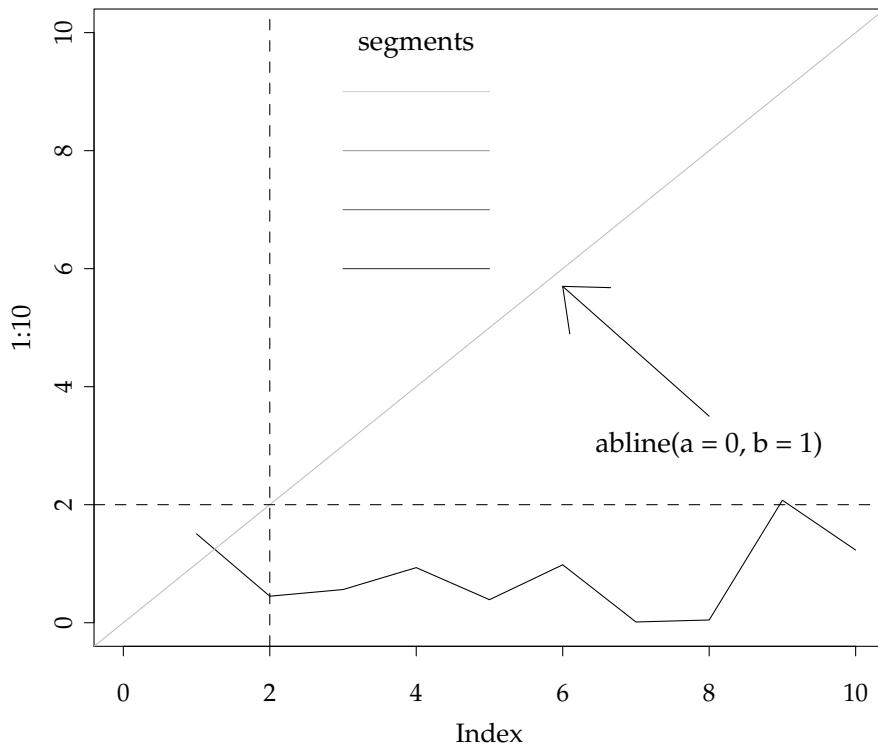


图 4.5: 曲线、直线、线段、箭头的展示说明

```

1 > usage(arrows)

arrows(x0, y0, x1 = x0, y1 = y0, length = 0.25,
       angle = 30, code = 2, col = par("fg"), lty = par("lty"),
       lwd = par("lwd"), ...)

```

类似于线段，前四个参数表示箭头的起点和终点坐标，`length`表示箭头尖上短线的长度（单位：英寸），`angle`表示箭头尖短线的角度（默认为 30° ⁸），`code`表示箭头的样式（整数1~3分别表示尾部箭头、首部箭头和两端都带箭头），注意若`length`设置为0，那么将不会画出箭头（只有箭头线的主体即一条线段）。

关于曲线、直线、线段和箭头函数的示例代码和效果参见图4.5。

下面我们再介绍一下一种特殊的曲线—X-样条（X-spline），样条是用光滑曲线连接若干数据点的曲线，注意它与前面提到的曲线`lines()`的区别在于数据点之间的连接线。样条函数用法如下：

```

1 > usage(xspline)

xspline(x, y = NULL, shape = 0, open = TRUE,
        repEnds = TRUE, draw = TRUE, border = par("fg"),
        col = NA, ...)

```

前两个参数给定点的位置，`shape`为样条的形状，取值在[-1, 1]之间，当取值为负数时，曲线穿过给定的点，负值绝对值越小则曲线的角度越尖锐，反之角度越圆滑，`shape`取值为正数时，曲线脱离给定的点，正值越小越靠近给定点；`open`决定是否样条曲线封闭；`repEnds`为逻辑值，当样条曲线不封闭时，该参数决定是否重复使用端点上的点；`draw`决定是否画线，若为`FALSE`，则仅仅计算曲线的坐标位置而不画线；`border`为曲线的颜色；`col`为封闭曲线的填充颜色。图4.6为各种形状的X-样条，注意观察`shape`参数与曲线形状的对应关系。

4.4 矩形、多边形

R中绘制多边形也是很方便的，主要使用`polygon()`函数，矩形是多边形的特例，不过R也提供了专门的函数`rect()`来绘制它。多边形的主要特征在

⁸请读者想象一下，当箭头角度为 90° 时箭头的形状是怎样的，以及这种形状有什么用途，后面第七章会用到

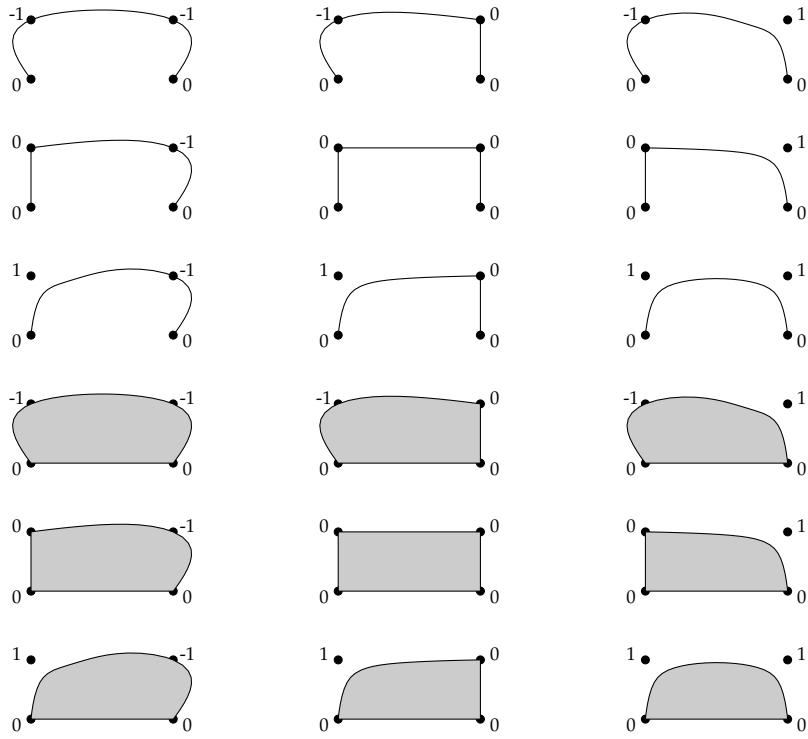


图 4.6: X-样条各种形状的展示: 观察`shape`参数的取值 (图中标注的数字) 与样条形状的对应关系。本图代码参考 `example(xspline)`。

于增加了一些填充选项, 比如颜色填充, 或者用阴影线填充。关于颜色、线条样式的设置就不必再重复说明。矩形和多边形的用法如下:

```

1 > usage(rect)
rect(xleft, ybottom, xright, ytop, density = NULL,
      angle = 45, col = NA, border = NULL, lty = par("lty"),
      lwd = par("lwd"), ...)

1 > usage(polygon)
polygon(x, y = NULL, density = NULL, angle = 45,
       border = NULL, col = NA, lty = par("lty"), ...,
       fillOddEven = FALSE)

```

矩形函数的前四个参数分别制定左下角和右上角的坐标, 用以确定矩形的位置, 同样, 多边形函数的前两个参数给出一系列坐标点, 用以围成

一个多边形; `density`参数设置阴影线的填充密度 (每英寸填充多少条线), 如果设置了一个正值, 那么颜色填充参数`col`将被忽略, 只有当它被设置为负数或NA或NULL时才可以填充颜色; `angle`参数设置填充线条的角度; `col`设置填充颜色; `border`设置边框颜色, 若设置为FALSE或NA, 那么边框线将被省略。

绘制多边形时要清楚它的过程: 线条会随着横纵坐标点延伸, 当走到最后一点时, 就会重新延伸回第一点, 这就是多边形的绘制基本原理。一般来说, 大部分作图函数对于缺失数据都会默认省略不画, 不会对图形造成什么影响, 而对于多边形函数, 数据中的缺失将构成“分界点”, 用以分隔缺失点两端的点群, 因此, 若数据含缺失值, 那么会有多个多边形被作出。这一点性质对与多边形的灵活运用也是很重要的, 我们有时可以故意设置缺失值, 用以将图形分割为不同的区域。

最后要说明, 其实还有一个特殊的“矩形”, 那就是整幅图形的边框, 我们可以用`box()`函数来完成方框的绘制, 我们可以不使用任何参数以添加默认的边框, 也可以调整一些参数画出不同样式的方框 (如虚线框等), 具体参见`?box`。

关于多边形和矩形, 我们用一个比较巧妙的例子来说明具体用法, 代码和效果参见图4.7。

这幅图的目的在于只填充y值在0以上的部分, 0以下的部分留空。采取的手段是: 先用多边形整体填充颜色, 然后根据当前图形的坐标范围 (用`par('usr')`获得 (或者等价使用`par()$usr`), 参见3.1小节) 画一个白色的矩形覆盖0以下的图形部分, 此时下面部分的线条也被覆盖了, 所以接着我们再次画线, 将所有线条补充完整, 最后, 添加一条高度为0的水平线, 即完成本图。

4.5 网格线

有时为了方便图形阅读者知道图中元素的更精确的位置, 我们可以用添加背景网格线的办法来辅助读者的视线对齐坐标轴。函数`grid()`所实现的就是这一个功能, 它的用法较简单:

```
1 > usage(grid)

grid(nx = NULL, ny = nx, col = "lightgray",
      lty = "dotted", lwd = par("lwd"), equilogs = TRUE)
```

```
1 > # 产生40个正态随机数
2 > x = rnorm(40)
3 > # 画线图
4 > plot(x, xlab = "", type = "l")
5 > # 请思考为什么坐标值要这样设置: 多边形的连线路径
6 > polygon(c(1, 1:40, 40), c(0, x, 0), col = "gray")
7 > # 获取当前图形区域坐标范围, 以便下用
8 > xy = par("usr")
9 > # 用白色矩形挡住了0以下的部分
10 > rect(xy[1], xy[3], xy[2], 0, col = "white")
11 > # 重画一遍x的线条
12 > lines(x)
13 > # 添加水平线
14 > abline(h = 0, col = "lightgray")
```

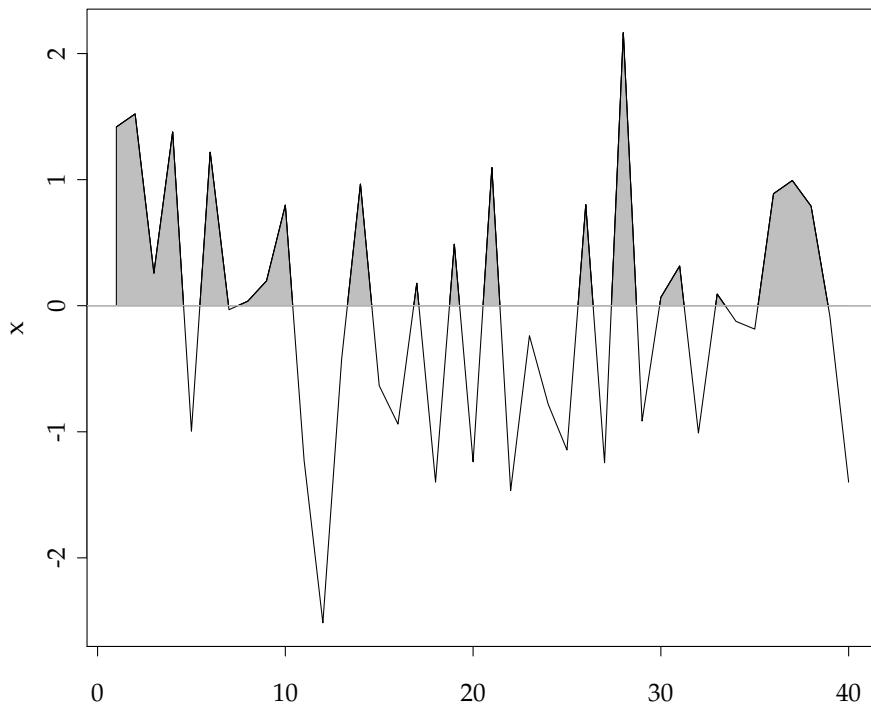


图 4.7: 多边形和矩形结合使用的一个巧妙图示: 将0上下的数值分别用不同颜色填充。本图也可以利用clip()函数更方便地完成, 请读者自行查阅帮助。

可以看到，这个函数已经使用了一些默认的参数设置，例如将网格线的颜色设置为浅灰色，线条样式设置为点线，这是一种比较美观的设置，让网格线既不显得太突兀，又能达到辅助的效果；一般情况下，我们可以直接使用不带参数的`grid()`函数添加网格。参数`nx`和`ny`分别表示横纵轴上网格线的条数，颜色、线条样式和线条宽度参数不必再说明，最后`equilogs`参数意思是，当坐标取了对数之后，是依然使用等距的网格线（`TRUE`）还是根据对数函数使用不等距的网格线（`FALSE`）。

细心的读者也许能发现，其实前面讲到的内容中已经有两处可以实现“网格线”的功能，第一处是`par()`函数中的`tcl`（或`tck`），将坐标轴的刻度线长度设置为图形的宽高就可以构成一种粗略的网格；第二处是`abline()`函数，使用参数`h`和`v`即可实现更细致的网格线。

4.6 标题、任意文本、周边文本

本节中的文本以及下一节中的图例都是用来辅助解释图形的信息，图形中的所有文本可以分为三类：标题（主副标题和坐标轴标题）、任意文本和图形周边文本。`title()`函数用来添加标题，`text()`函数用来向图形中任意位置添加文本，`mtext()`函数用来向图的四条边上添加文本。用法如下：

```

1 > usage(title)

title(main = NULL, sub = NULL, xlab = NULL,
      ylab = NULL, line = NA, outer = FALSE, ...)

1 > usage(text, "default")

text(x, y = NULL, labels = seq_along(x),
      adj = NULL, pos = NULL, offset = 0.5, vfont = NULL,
      cex = 1, col = NULL, font = NULL, ...)

1 > usage(mtext)

mtext(text, side = 3, line = 0, outer = FALSE,
      at = NA, adj = NA, padj = NA, cex = NA, col = NA,
      font = NA, ...)

```

若无特别设定，这些文本的样式都将根据当前的函数`par()`结果来设置，比如颜色、字体等。函数`title()`的前四个参数就是主、副、x轴、y轴标题的字符串，`line`设置一个距离图形边缘的行数（即：文本与图形边缘的距离

为 $\text{line} \times \text{行高}$; outer 表示是否将文本放在外边界中 (参见3.1小节的说明); 函数 $\text{text}()$ 的参数 labels 就是欲添加的文本 (对应横纵坐标的位置, 可以是字符串向量), 若不指定本参数, 那么默认将以数字 $1:\text{length}(x)$ 作为文本标记添加到图中; adj 与 $\text{par}()$ 中说明相同; pos 参数取值整数1~4分别表示文本的位置在坐标点的下、左、上、右方, 注意, 它会覆盖参数 adj 的设置; offset 参数会根据 pos 参数的取值将文本向相应的方向移动一定比例的距离; vfont 参数用Hershey矢量字体来设置文本的字体式样, 取值长度为2的向量, 第一个元素指定字体 (Typeface), 第二个元素指定式样 (Style), 关于字体和式样的搭配种类, 请查看帮助? ?Hershey , 使用Hershey矢量字体的优势在于:

- Hershey字体会产生更好的输出, 特别在计算机屏幕上, 或者用于旋转以及小字体时
- Hershey字体提供一些标准字体库没有的字体。如提供星座记号, 地图符号和天文学符号
- Hershey字体提供西里尔字符 (cyrillic) 和日语字符 (假名和日本汉字)

此外, 等高线图中通常使用Hershey矢量字体以使等高线上的文本更清晰好看, 由于字体设置搭配内容体系比较庞大, 因此感兴趣的读者请仔细阅读帮助⁹; side 参数取值为整数1~4分别把周边文本作在表示图形的下、左、上、右边; 其它参数基本已经都已经接触过, 有些不常用的参数在此处省略不讲。

4.7 图例

函数 $\text{legend}()$ 的作用是添加图例, 总所周知, 图例也是很重要的辅助解释信息, 告诉图形使用者图中各组不同样式的元素分别代表何种对象。它的参数比较多, 但实际应用中通常仅仅用到其中少数几个:

```
1 > usage(legend, w = 0.73)

legend(x, y = NULL, legend, fill = NULL,
       col = par("col"), border = "black", lty, lwd,
```

⁹该字体的一个缺陷是不能用在数学公式中

```

1 > par(mar = c(4, 4, 4, 3))
2 > plot(0:10, type = "n", xlab = "", ylab = "", xlim = c(0,
3 + 12))
4 > grid(col = "gray")
5 > title(main = "Demonstration of text in R Graphics",
6 + xlab = "X-axis title", ylab = "Y-axis title")
7 > mtext("Here is \"side = 4\"", side = 4, line = 1)
8 > x = c(6, 4, 6, 8)
9 > y = c(8, 5, 2, 5)
10 > s = c(0, 90, 180, 270)
11 > for (i in 1:4) text(x[i], y[i], sprintf("srt = %d",
12 + s[i]), srt = s[i])
13 > segments(c(6, 0, 6, 12), c(10, 5, 0, 5), c(0, 6,
14 + 12, 6), c(5, 0, 5, 10), lty = c(2, 1, 1, 2))
15 > legend(-0.2, 9.8, c("Upper", "Lower"), lty = 2:1,
16 + cex = 0.8, bty = "n")

```

Demonstration of text in R Graphics

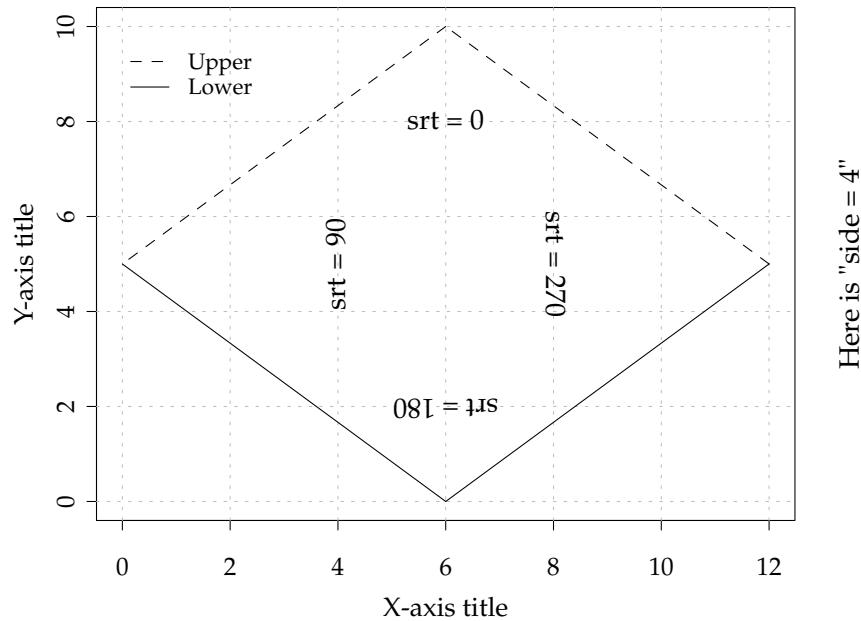


图 4.8: 添加标题、任意文本和周边文本的一个演示

```
pch, angle = 45, density = NULL, bty = "o",
bg = par("bg"), box.lwd = par("lwd"), box.lty = par("lty"),
box.col = par("fg"), pt.bg = NA, cex = 1, pt.cex = cex,
pt.lwd = lwd, xjust = 0, yjust = 1, x.intersp = 1,
y.intersp = 1, adj = c(0, 0.5), text.width = NULL,
text.col = par("col"), merge = do.lines && has.pch,
trace = FALSE, plot = TRUE, ncol = 1, horiz = FALSE,
title = NULL, inset = 0, xpd, title.col = text.col)
```

前两个参数x和y表示图例的坐标位置（左上角顶点的坐标）；legend参数通常为一个字符向量，表示图例中的文字；fill参数指定一个与图例字符向量对应的颜色向量用以在文本左边绘制一个颜色填充方块；col参数设置图例中点和线的颜色；lty、lwd和pch参数指定图例中点线的样式；angle和density参数效果类似于fill参数，只是换成指定角度和密度的阴影线填充方块；bty参数设置图例框的样式，很类似par()中的同名参数；title参数设定图例的标题；剩余参数用来设置更细微的地方，不太常用。

从以上几节对图形元素的介绍可以看出，R可以设置极其细微的元素特征，当然，读者不必过于细究所有的参数，对于大多数函数来说，都只有两三个核心参数，比如，坐标位置向量是几乎所有函数的共同核心参数，而legend()函数的主要参数除此之外还有字符向量legend。对于其它参数，请在使用时即时查阅即可。

关于网格线、文本和图例，我们也用一个简单的例子给出它们的图示效果，参考图4.8。

4.8 坐标轴

坐标轴是图中元素所代表数值大小的参照物，因此它在图形中的作用也很重要，特别是有时候我们需要对坐标轴做一定的特殊设置，比如作一幅双坐标轴的图形，或者在坐标轴标记中使用特殊的文本，那么就必须使用axis()函数来辅助完成对坐标轴的设置和调整。该函数的用法如下：

```
1 > usage(axis)

axis(side, at = NULL, labels = TRUE, tick = TRUE,
line = NA, pos = NA, outer = FALSE, font = NA,
lty = "solid", lwd = 1, lwd.ticks = lwd, col = NULL,
col.ticks = NULL, hadj = NA, padj = NA, ...)
```

```

1 > data(Export.USCN)
2 > par(mar = c(4, 4.5, 0.1, 4.5))
3 > # 看似条形图, 实为粗线条, 宽度lwd = 10
4 > plot(1:13, Export.USCN$Export, xlab = "Year / Country",
5 +       ylab = "US Dollars (\$\$10^{16})", axes = FALSE,
6 +       type = "h", lwd = 10, col = c(rep(2, 6), NA,
7 +                                         rep(4, 6)), lend = 1, panel.first = grid())
8 > # 设置x轴的刻度标记: \n的意思是换行符
9 > xlabel = paste(Export.USCN$Year, "\n", Export.USCN$Country)
10 > xlabel[7] = ""
11 > # 添加一条分隔线
12 > abline(v = 7, lty = 2)
13 > # 使用带有换行符的刻度标记
14 > axis(1, 1:13, labels = xlabel, tick = FALSE, cex.axis = 0.75)
15 > axis(2)
16 > # 换算为人民币再计算另一个坐标轴刻度 (汇率8.27)
17 > ylabel = pretty(Export.USCN$Export * 8.27)
18 > axis(4, at = ylabel/8.27, labels = ylabel)
19 > mtext("Chinese RMB", side = 4, line = 2)
20 > box()

```

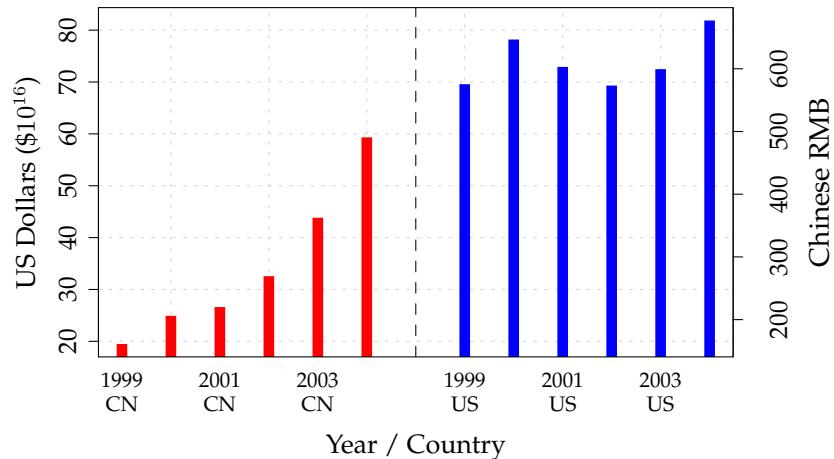


图 4.9: 双坐标轴图示: 中美两国1999~2004年出口额, 分别以美元和人民币表示。注意图中x轴标签文本是如何换行的。

其中，`side`参数与`mtext()`函数中的参数意思相类似，表示将坐标轴画在哪条边上，事实上通过前面一些图形元素参数的讲解，读者应该能意识到，R中上下左右方向的顺序一般都是“下、左、上、右”，分别用1、2、3、4表示；`at`参数表示在什么位置画坐标轴标记线；`labels`参数指定坐标轴刻度标记的字符。

Murrell (2005)中第3.4.5小节举了一个双坐标轴图形的例子，图中用左右两边的纵轴分别表示摄氏和华氏的温度，即：对于图中同一点，既可以对照左边看摄氏度，也可以对照右边看华氏度；这就是双坐标轴的用途。此处我们不妨也举一个类似的例子来说明`axis()`函数的功能。图4.9展示的是从1999~2004年中美两国的出口贸易总额，其中`axis()`函数主要作用在于两个地方：第一点是横轴的刻度标记，注意这些刻度标记与我们平时看到的标记有所不同，主要是这一系列标记都是两行文本，原因就在于`labels`参数中使用了换行符，对C语言比较熟悉的读者对此不会感到陌生，这里第一行是年份，第二行说明了国家：CN表示中国，US表示美国；第二点是图的右边多了一根坐标轴，左边的纵轴表示出口额按美元计价（单位： 10^{16} 美元，出口额数值太大，因此采用较大的单位），右边表示同样的出口额用人民币计价（例如右边轴上400人民币对应左边 $400/8.27 \approx 48.4$ 美元），同一个数值在图中既可以从左边观察美元金额，又可以从右边观察人民币金额。本例中使用的数据来源为：汇率数据来自联合国统计署网站<http://unstats.un.org>，从1999~2004年人民币和美元的平均汇率稳定在8.27，出口额年度数据来自WTO网站：<http://stat.wto.org>。整理后的数据如下，程序代码以参见图4.9。

```

1 > data(Export.USCN)
2 > Export.USCN

  Export Year Country
1   19.49 1999      CN
2   24.92 2000      CN
3   26.61 2001      CN
4   32.56 2002      CN
5   43.82 2003      CN
6   59.33 2004      CN
7     NA    NA    <NA>
8   69.58 1999      US
9   78.19 2000      US
10  72.91 2001      US
11  69.31 2002      US

```

12	72.48	2003	US
13	81.85	2004	US

本章对图形元素的介绍就到此为止，后面章节的统计图形构造仍然会用到这些基础元素。具备这些低层函数的知识之后，只要我们不嫌麻烦，其实所有的统计图形都可以用它们的组合使用加上一定的统计计算而一步步构建出来。当然，一般也不会有人真用这样的方法去作图（练习除外），我们指出这一点，也是为了说明，统计图形并没有任何神秘可言，随着对图形元素的绘制以及统计量的逐渐深入了解，当一幅统计图形摆在我们面前，我们也应该能如“庖丁解牛”一般洞穿其架构¹⁰。例如，从纯粹作图的意义上来讲，直方图、条形图、棘状图和马赛克图不过是矩形，条件密度图和饼图是多边形，散点图和带状图是点，折线图和向日葵散点图是线段，等等。本书附录C中所谈到的统计学动画则是图形元素的另类应用，动画包**animation**(Xie, 2010a)几乎完全是依靠统计计算与R基础图形系统而写成。

前两章中，我们对R的作图基础元素和大部分参数都作了比较详细的介绍，大致掌握这些知识之后，下一章我们就可以正式进入本书的核心部分—统计图形，在那些图形中，我们还会多次遇到这些基础知识的应用，到时候我们会指明章节，以便读者进行回顾复习。

¹⁰事实上，用R语言的作图功能作音乐五线谱、画地图、做动画等等，都不足为奇

第五章 图库

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》

本章中，我们结合R语言中的相关函数以及数据实例对各种统计图形依次作出介绍。从第5.1节到5.27节的所有图形都是基于**graphics**包所作，其后的图形均来自于其它函数包。图形的介绍顺序大致按函数的字母序，但直方图、箱线图和散点图等常见图形放在前面，而饼图被有意安排在最后。

5.1 直方图

直方图（Histogram）是展示连续数据分布最常用的工具，它本质上是对密度函数的一种估计。在介绍作图方法之前我们有必要先了解一下它的基本数学思想，本节仅作简要介绍，详细的数学理论参见Scott (1992)。

我们知道，对于连续随机变量来说，其密度函数即为分布函数的导数：

```

1 > par(mfrow = c(2, 2), mar = c(2, 3, 2, 0.5), mgp = c(2,
2 + 0.5, 0))
3 > data(geyser, package = "MASS")
4 > hist(geyser$waiting, main = "(1) freq = TRUE", xlab = "waiting")
5 > hist(geyser$waiting, freq = FALSE, xlab = "waiting",
6 + main = "(2) freq = FALSE")
7 > hist(geyser$waiting, breaks = 5, density = 10, xlab = "waiting",
8 + main = "(3) breaks = 5")
9 > hist(geyser$waiting, breaks = 40, col = "red", xlab = "waiting",
10 + main = "(4) breaks = 40")

```

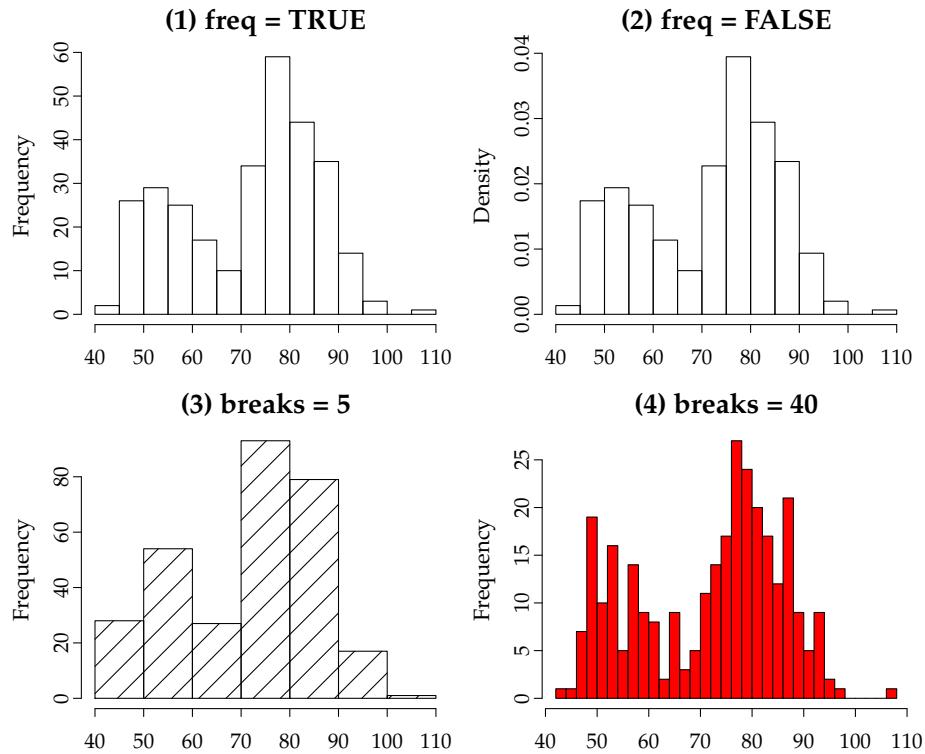


图 5.1: 喷泉间隔时间直方图: (1) 使用默认参数值 (作频数图); (2) 概率密度直方图; (3) 减小区间段数, 直方图看起来更平滑 (偏差大, 方差小); (4) 增大区间段数, 直方图更突兀 (偏差小, 方差大)。

```

1 > # 注意: 以下两行代码足够添加密度曲线, 本代码只是为了美观效果
2 > # hist(geyser$waiting, probability = TRUE, main = '')
3 > # lines(density(geyser[['waiting']])))
4 > par(mar = c(1.8, 3, 0.5, 0.1), mgp = c(2, 0.5, 0))
5 > data(geyser, package = "MASS")
6 > hst = hist(geyser$waiting, probability = TRUE, main = "", 
7 + xlab = "waiting")
8 > d = density(geyser$waiting)
9 > polygon(c(min(d$x), d$x, max(d$x)), c(0, d$y, 0),
10 + col = "lightgray", border = NA)
11 > lines(d)
12 > ht = NULL
13 > brk = seq(40, 110, 5)
14 > for (i in brk) ht = c(ht, d$y[which.min(abs(d$x -
15 + i))])
16 > segments(brk, 0, brk, ht, lty = 3)

```

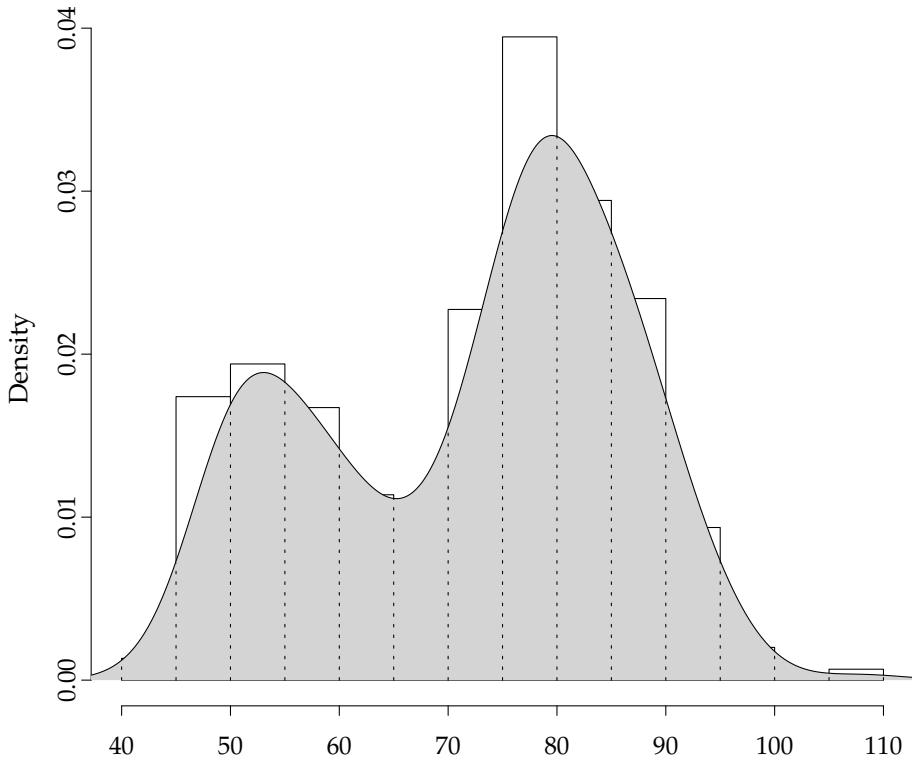


图 5.2: 直方图与密度曲线的结合: 借助函数density()可以计算出数据的核密度估计, 然后利用低层作图函数lines()将核密度估计曲线添加到直方图中。注意本图中其它低层作图函数的使用。

$$f(x) = F'(x) = \lim_{h \rightarrow 0} \frac{F(x+h) - F(x)}{h} \quad (5.1)$$

因此我们不妨自然而然地从分布函数的估计出发得到密度函数的估计。当我们拿到一批数据 X_1, X_2, \dots, X_n 时，我们最容易想到的分布函数估计就是经验分布函数：

$$\hat{F}_n(x) = \frac{1}{n} \sum_{i=1}^n \mathbf{I}(X_i \leq x) \quad (5.2)$$

其中 $\mathbf{I}(\cdot)$ 为示性函数；结合公式 5.1 和 5.2 以及示性函数的性质，我们可以直接得到以下密度函数估计：

$$\hat{f}_n(x) = \lim_{h \rightarrow 0} \frac{1}{n} \sum_{i=1}^n \frac{\mathbf{I}(x < X_i \leq x+h)}{h} \quad (5.3)$$

公式 5.3 实际上已经给出了直方图作为密度函数估计工具的基本思想：划分区间并计数有多少数据点落入该区间。实际数据不可能无限稠密，因此 $h \rightarrow 0$ 的条件往往是不可能实现的，于是我们退而求其次，只是在某一些区间段里面估计区间上的密度。首先我们将实数轴划分为若干宽度为 h 的区间（我们称 h 为“窗宽”）：

$$b_1 < b_2 < \dots < b_j < b_{j+1} < \dots ; b_{j+1} - b_j = h, j = 1, 2, \dots \quad (5.4)$$

然后根据以下直方图密度估计表达式计算区间 $(b_j, b_{j+1}]$ 上的密度估计值：

$$\hat{f}_n(x) = \frac{1}{nh} \sum_{i=1}^n \mathbf{I}(b_j < X_i \leq b_{j+1}); x \in (b_j, b_{j+1}] \quad (5.5)$$

最后我们将密度估计值以矩形的形式表示出来，就完成了直方图的基本制作。当然我们没有必要使用这样原始的方式制作直方图，R 中提供了 `hist()` 函数，其默认用法如下：

```
1 > usage(hist, "default")

hist(x, breaks = "Sturges", freq = NULL,
      probability = !freq, include.lowest = TRUE, right = TRUE,
      density = NULL, angle = 45, col = NULL, border = NULL,
      main = paste("Histogram of", xname), xlim = range(breaks),
      ylim = NULL, xlab = xname, ylab, axes = TRUE, plot = TRUE,
      labels = FALSE, nclass = NULL, ...)
```

其中，`x`为欲估计分布的数值向量；`breaks`决定了计算分段区间的方法，它可以是一个向量（依次给出区间端点），或者一个数字（决定拆分为多少段），或者一个字符串（给出计算划分区间的算法名称），或者一个函数（给出划分区间个数的方法），区间的划分直接决定了直方图的形状，因此这个参数是非常关键的；`freq`和`probability`参数均取逻辑值（二者互斥），前者决定是否以频数作图，后者决定是否以概率密度作图（这种情况下矩形面积为1）；`labels`为逻辑值，决定是否将频数的数值添加到矩形条的上方；其它参数诸如`density`、`angle`、`border`均可参见低层作图函数“矩形”(`rect()`, 4.4节)。

我们以黄石国家公园喷泉数据`geyser`(Venables and Ripley, 2002)为例。图5.1展示了喷泉喷发间隔时间的分布情况。(1) 和 (2) 中的直方图看起来形状完全一样，区别仅仅是前者为频数图，后者为密度图，二者在统计量上仅相差一个常数倍，但密度直方图的一个便利之处在于它可以方便地添加密度曲线，用以辅助展示数据的统计分布(图5.2即为一个示例)；(3) 和 (4) 的区别在于区间划分段数，我们可以很清楚看出区间划分的多少对直方图的直接影响。关于区间划分的一些讨论可以参考Venables and Ripley (2002)，这里我们需要特别指出的是，直方图的理论并非想象中或看起来的那么简单，窗宽也并非可以任意选择，不同的窗宽或区间划分方法会导致不同的估计误差，关于这一点，Excel的直方图可以说是非常不可靠的，因为它把区间的划分方法完全交给了用户去选择，这样随意制作出来的直方图很可能会导致大的估计误差、掩盖数据的真实分布情况。另外一点需要提醒的是关于直方图中的密度曲线，SPSS软件在绘制直方图时会有选项提示是否添加正态分布密度曲线，这也是完全的误导，因为数据不一定来自正态分布，添加正态分布的密度曲线显然是不合理的，相比之下，图5.2的做法才是真正从数据本来的分布出发得到的密度曲线。

直方图函数在作图完毕之后会有一些计算返回值，这些值对于进一步的作图或者分析很有用，例如区间划分端点、频数（或密度）、区间中点等等，这些信息可以被灵活应用在图形定制上（例如图B.3）。

5.2 茎叶图

茎叶图（Stem-and-Leaf Plot）与直方图的功能类似，也是展示数据密度的一种工具，但相比之下茎叶图对密度的刻画显得非常粗略，而且

图 5.3: 本图展示了世界上主要的48块大陆面积的分布状况, 可以明显看出, 这些面积数据是严重右偏的, 意即: 少数陆地块的面积非常大, 而大多数陆地块的面积相对来说都很小。事实上, 主要是七大洲的大陆块面积非常大, 而其它岛屿诸如海南岛、帝汶岛、九洲岛等面积都相对较小。

```
1 > # lambda为10的泊松分布随机数
2 > (x = rpois(50, 10))

[1] 10 9 6 9 9 10 3 18 12 6 10 8 12 11 10 7 8 9 6 13
[21] 9 11 6 12 6 6 11 11 13 16 14 8 7 7 6 8 10 14 6 11
[41] 6 7 12 7 12 10 9 6 5 14

1 > stem(x, scale = 2)

The decimal point is at the |

 3 | 0
 4 |
 5 | 0
 6 | 0000000000
 7 | 00000
 8 | 0000
 9 | 000000
10 | 000000
11 | 00000
12 | 00000
13 | 00
14 | 000
15 |
16 | 0
17 |
18 | 0
```

图 5.4: $\lambda = 10$ 的泊松分布随机数茎叶图：可以看出数据密度在 10 附近最高，这与理论相符。注意这幅图可以完全还原到原始数据。

对原始数据通常会作舍入处理，它只是在早期计算机尚不发达时对于手工整理数据来说比较方便。茎叶图的整体形状如同植物的茎和叶，对于一个数据，通常取其 10^n 部分为茎（ n 视所有数据的数量级而定），剩下的尾数为叶，放置于茎旁，这样每隔 $m10^n$ 就对数据作一次归类汇总，将落入区间 $[km10^n, (k+1)m10^n]$ 的数据汇集为叶子（ k, m 为整数， m 通常取1， $k = 1, 2, 3, \dots$ ），我们不妨称这种区间为一个“节”，节的长度与直方图的“窗宽”本质上是同样的概念。显然，叶子越长则表明该节上数据频数越高。

R中茎叶图的函数为`stem()`，其用法为：

```
1 > usage(stem)
stem(x, scale = 1, width = 80, atom = 1e-08)
```

参数`scale`控制着 m ，即节与节之间的长度（`scale`越大则 m 越小）；`width`控制了茎叶图的宽度，若叶子的长度超出了这个设置，则叶子会被截取到长度`width`，然后以一个整数表示后面尚有多少片叶子没有被画出来。

下面我们以**datasets**包中`islands`数据为例说明茎叶图的作法。该数据记录了世界上各大陆地块的面积大小，原始数据前10条如下（单位：千平方英里）：

```
1 > head(islands, 10)
```

Africa	Antarctica	Asia	Australia	Axel	Heiberg
11506	5500	16988	2968	16	
Baffin	Banks	Borneo	Britain	Celebes	
184	23	280	84	73	

可以看出，以上数据中最大的数量级为 10^4 ，而大部分数据的数量级集中在 10^1 ，因此茎上的数量级取作 10^3 相对比较合适—更大的数量级会导致茎的节数非常少、对分布的刻画过于粗略，而更小的数量级会导致节数过多，使得茎叶图几乎退化为数据的原始表示，这样也难以看出数据的集中趋势。图5.3展示了48块大陆块面积的分布，该茎叶图窗宽为 2×10^3 ，图中注明了原始数据小数点位置在“|”后面三位数处，因此我们从图中“还原”原始数据时，需要用（“茎的区间” + “叶”） $\times 10^3$ 。我们以图5.3为例说明一下茎叶图的制作过程及其相应解释。首先我们将原始数据除以 10^3 ，并四舍五入到小数点后的一位数：

```

1 > # 去掉陆地名称以便显示数据
2 > unname(sort(round(islands/1000, 1)))

[1] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
[13] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
[25] 0.0 0.0 0.0 0.0 0.0 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.2
[37] 0.2 0.2 0.3 0.3 0.8 3.0 3.7 5.5 6.8 9.4 11.5 17.0

```

然后从0到 18×10^3 、以 2×10^3 为窗宽，分段整理数据，每一段（节）中依次放置落入该段的数据的小数位，堆砌起来便形成了茎叶图的叶子。例如11.5落入了[10, 12]的区间，我们就将尾数5放在10的右边；类似地，17.0在[16, 18]之间，我们将0放在16右边；关于茎叶图顶部的一长串0的解释此处不再赘述。

图5.4是利用泊松分布随机数生成的茎叶图，由于窗宽为1，不存在舍入问题，所以图形可以还原到原始数据，请读者自行对应数据观察茎叶图。

经过前面的说明，现在我们不妨将茎叶图简单理解为横放着的直方图，只是茎叶图通常都以某个便利的整数为窗宽，不如直方图那样精细。此外，茎叶图曾经的优势（简单、可手工绘制）在今天这个计算机时代也显得并不突出，因此，除非特殊情况，我们建议主要使用直方图作为密度函数估计工具。

5.3 箱线图

箱线图（Box Plot或Box-and-Whisker Plot）主要是从四分位数的角度出发描述数据的分布，它通过最大值(Q_4)、上四分位数(Q_3)、中位数(Q_2)、下四分位数(Q_1)和最小值(Q_0)五处位置来获取一维数据的分布概况。我们知道，这五处位置之间依次包含了四段数据，每段中数据量均为总数据量的1/4。通过每一段数据占据的长度，我们可以大致推断出数据的集中或离散趋势（长度越短，说明数据在该区间上越密集，反之则稀疏）。

R中相应的函数为`boxplot()`，其用法如下：

```

1 > # 默认用法
2 > usage(boxplot, "default")

boxplot(x, ..., range = 1.5, width = NULL,
        varwidth = FALSE, notch = FALSE, outline = TRUE,

```

```

names, plot = TRUE, border = par("fg"), col = NULL,
log = "", pars = list(boxwex = 0.8, staplewex = 0.5,
outwex = 0.5), horizontal = FALSE, add = FALSE,
at = NULL)

1 > # 公式用法
2 > usage(boxplot, "formula")

boxplot(formula, data = NULL, ..., subset,
na.action = NULL)

```

因为`boxplot()`是一个泛型函数，所以它可以适应不同的参数类型。目前它支持两种参数类型：公式（`formula`）和数据，后者对我们来说可能更容易理解（给一批数据、作相应的箱线图），而前者在某些情况下更为方便，后面我们会举例说明。参数`x`为一个数值向量或者列表，若为列表则对列表中每一个子对象依次作出箱线图；`range`是一个延伸倍数，决定了箱线图的末端（须）延伸到什么位置，这主要是考虑到离群点的原因，在数据中存在离群点的情况下，将箱线图末端直接延伸到最大值和最小值对描述数据分布来说并不合适（图形缺乏稳健性），所以R中的箱线图默认只将图形延伸到离箱子两端 $\text{range} \times (Q_3 - Q_1)$ 处，即上下四分位数分别加/减内四分位距（Interquartile Range，简称IQR $\equiv Q_3 - Q_1$ ）的倍数，超过这个范围的数据点就被视作离群点，在图中直接以点的形式表示出来；`width`给定箱子的宽度；`varwidth`为逻辑值，若为TRUE，那么箱子的宽度与样本量的平方根成比例，这在多批数据同时画多个箱线图时比较有用，能进一步反映出样本量的大小；`notch`也是一个有用的逻辑参数，它决定了是否在箱子上画凹槽，凹槽所表示的实际上是中位数的一个区间估计，其计算式为 $Q_2 + / - 1.58\text{IQR}/\sqrt{n}$ (McGill *et al.*, 1978; Chambers *et al.*, 1983)，区间置信水平为95%，在比较两组数据中位数差异时，我们只需要观察箱线图的凹槽是否有重叠部分，若两个凹槽互不交叠，那么说明这两组数据的中位数有显著差异（P值小于0.05）；`horizontal`为逻辑值，设定箱线图是否水平放置；`add`设置是否将箱线图添加到现有图形上（例：图5.34）；其它参数诸如设置箱子颜色、位置、更详细的宽度等参见?`boxplot`。

绘制单个箱线图时只需要给`boxplot()`传入一个数值向量即可，如：`boxplot(rnorm(100))`；这里我们主要使用公式型的参数，以**datasets**包中的杀虫剂数据`InsectSprays`为例；该数据有两列，第一列为昆虫数目，第二列为杀虫剂种类（ABCDEF），这里是随机抽取的10列数据：

```

1 > InsectSprays[sample(nrow(InsectSprays), 10), ]

```

```

1 > par(mar = c(2, 2.5, 0.2, 0.1))
2 > boxplot(count ~ spray, data = InsectSprays, col = "lightgray",
3 +     horizontal = TRUE, pch = 4)

```

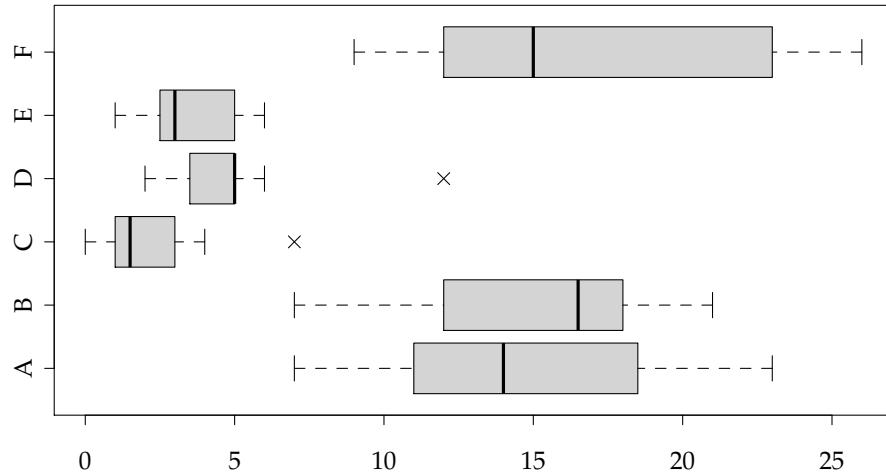


图 5.5: 昆虫数目箱线图: 六种杀虫剂下昆虫的数目分布。

	count	spray
64	22	F
15	21	B
22	21	B
32	1	C
33	3	C
38	5	D
47	2	D
66	16	F
43	5	D
49	3	E

为了了解杀虫剂的效果，我们需要对各种杀虫剂下昆虫的数目作出比较。图5.5是一个简单的箱线图展示，不难看出，除了B和D对应的昆虫数据呈左偏形态外，其它组均有右偏趋势，看起来各组数据的平均水平差异比较明显；另外注意观察图中的两个离群点（以“ \times ”表示）。总体看来，C的效果最好。事实上，我们可以对这个数据作方差分析，检验杀虫剂类型对昆虫数目是否有显著影响：

```

1 > insects.aov = aov(count ~ spray, data = InsectSprays)

```

```

1 > par(mar = c(2, 2.5, 0.2, 0.1))
2 > x = rnorm(150)
3 > y = rnorm(50, 0.5)
4 > boxplot(list(x, y), names = c("x", "y"), col = 2:3,
5 +   horizontal = TRUE, notch = TRUE, varwidth = TRUE)
6 > wilcox.test(x, y)

Wilcoxon rank sum test with continuity correction

data: x and y
W = 2579, p-value = 0.0009585
alternative hypothesis: true location shift is not equal to 0

```

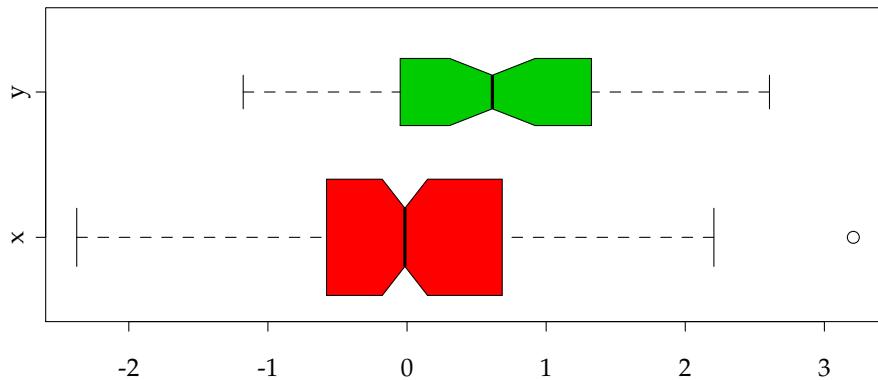


图 5.6: 箱线图的凹槽与统计推断: 从凹槽不交叠的情况来看, 两样本中位数有显著差异。

```

2 > summary(insects.aov)

Df Sum Sq Mean Sq F value Pr(>F)
spray      5    2669     534    34.7 <2e-16 ***
Residuals  66   1015      15
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

上述分析告诉我们杀虫剂类型有显著影响 (P值接近于0), 也印证了我们对图形的观察。

最后我们再以一个模拟数据的例子展示箱线图凹槽的功能。这里我们分别从正态分布 $N(0, 1)$ 和 $N(0.5, 1)$ 中各自产生150和50个随机数, 然后作箱线图比较两组数据中间位置的差异。图5.6为一次模拟的结果, 图中的凹槽

表明了两组数据的中位数有显著差异，Wilcoxon秩和检验也证实了这一结论。此外，该图还使用了varwidth参数以表明两组数据样本量的大小不同。

5.4 条形图

如同前面1.5节中曾经提到的，条形图目前是各种统计图形中应用最广泛的，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展示原始数值，对数据没有任何概括或推断。

R中条形图的函数为`barplot()`，用法如下：

```
> usage(barplot, "default", 0.75)

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, args.legend = NULL, ...)
```

条形图的主要参数是`height`，它指定了长条的长度，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，这些矩形的长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`可以设置条的宽度；`space`用以设置条之间的间距；`names.arg`为条形图的标签，即每一条的名称；`legend.text`参数在`height`为矩阵时比较有用，可以用来添加图例；`horiz`用以设置条形图的方向（水平或垂直）；`density`、`angle`等参数可以参考矩形的章节（4.4节）；`plot`为逻辑值，决定是否将条形图添加到现有图形上。

图5.7展示了参数`beside`和`legend.text`（下图）的效果。该图以1940年弗吉尼亚州分年龄组、分地区和分性别死亡率数据`VADeaths`为基础，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

```

1 > # 用分类调色板
2 > library(RColorBrewer)
3 > par(mfrow = c(2, 1), mar = c(3, 2.5, 0.5, 0.1))
4 > death = t(VADeaths)[, 5:1]
5 > barplot(death, col = brewer.pal(4, "Set1"))
6 > barplot(death, col = brewer.pal(4, "Set1"), beside = TRUE,
7 +     legend = TRUE)

```

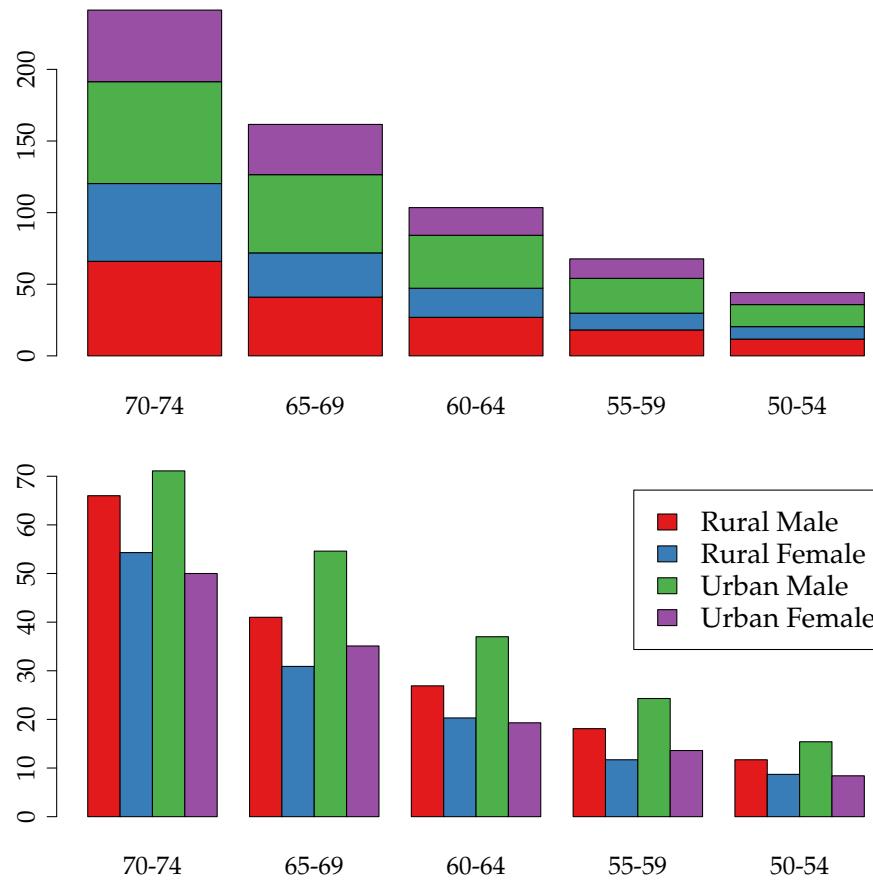


图 5.7: 弗吉尼亚死亡率数据条形图: 堆砌和并列的条形图效果。

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5.7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，即：将最关键的信息用最能激发视觉感知的形式表现出来。

```

1 > # 弗吉尼亚州死亡数据
2 > VADeaths
      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

```

5.5 散点图

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。

R中散点图的函数为`plot.default()`，但由于`plot()`是泛型函数（参见3.2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：

```

1 > usage(plot, "default")
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, ...)

```

其中若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；其它参数均已在3.2小节中介绍。

图5.8展示了一个人造数据的散点图：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。该数据收录在**MSG**包(Xie, 2010b)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，例如线性回归显示两个变量V1和V2的回归系数非常不显著：

```

1 > library(MSG)
2 > data(BinormCircle)
3 > # 数据形式: 前10行
4 > head(BinormCircle, 10)

      V1      V2
1  0.889 -1.764
2  0.072 -0.495
3  0.123 -0.180
4 -0.499  0.030
5  0.252  0.432
6  0.450  0.218
7 -0.317  0.485
8 -0.227 -0.445
9 -0.500 -0.002
10 -0.309  0.393

1 > # 回归系数以及P值(不显著)
2 > coef(summary(lm(V2 ~ V1, BinormCircle)))

            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.006085   0.005258 -1.1573  0.2472
V1          0.003989   0.007015  0.5686  0.5696

```

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）。由于样本量太大，普通的散点图上点与点之间重重重叠，所以也很难看出散点图有何异常，而使用半透明色的散点图则很容易看出，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 <http://yihui.name/en/2008/09/to-see-a-circle-in-a-pile-of-sand> 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5.21小节也将以平滑散点图的方式再回顾这批数据。

5.6 关联图

关联图（Cohen-Friendly Association Plot）是展示二维列联表数据的一种工具(Cohen, 1980; Friendly, 1992)，它主要是基于列联表的独立性检验理论（Pearson χ^2 检验）生成的图形。

```

1 > par(mfrow = c(1, 2), pch = 20, ann = FALSE, mar = c(2,
2 +      2, 0.5, 0.2))
3 > plot(BinormCircle, col = rgb(1, 0, 0))
4 > plot(BinormCircle, col = rgb(1, 0, 0, alpha = 0.01))

```

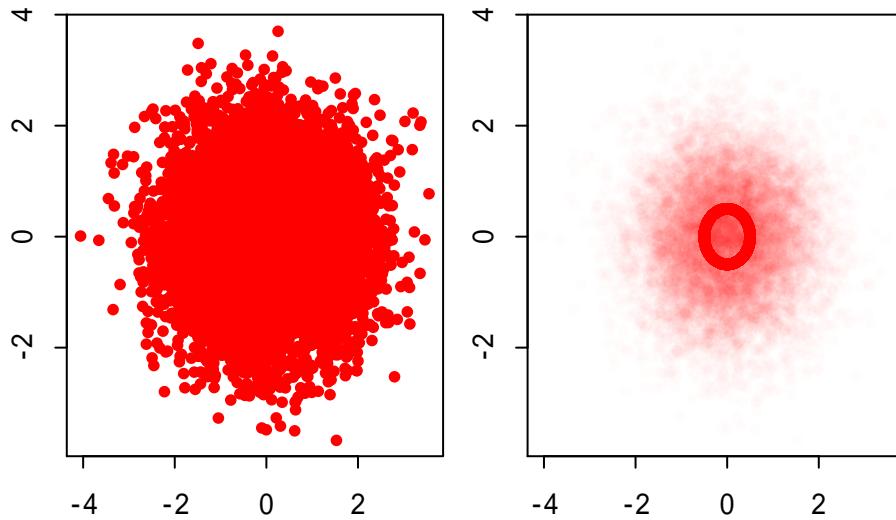


图 5.8: 半透明散点图中的规律: 左图是一幅普通的散点图, 图中几乎看不出数据有任何异常特征; 右图中对点使用了透明度为0.01的红色, 图中立即显示出一个深色的圆圈, 表明该圆圈上集中了大量数据点。

我们知道, 对于一个 $r \times c$ 列联表, χ^2 统计量的定义为如下平方和形式:

$$\chi^2 = \sum_{i=1}^r \sum_{j=1}^c d_{ij}^2; \quad d_{ij} = \frac{f_{ij} - e_{ij}}{\sqrt{e_{ij}}}$$

其中, f_{ij} 为单元格中的观测频数, e_{ij} 为期望频数, 二者相差越大, 则会导致检验统计量的值越大, 说明行变量和列变量越不独立。关联图所展示内容的正是这种差异, 它的设计思路是, 将图形同样以 $r \times c$ 的形式布局, 每一个“单元格”中用一个矩形表示观测频数和期望频数的信息, 具体来说, 矩形的高度与Pearson残差 $f_{ij} - e_{ij}$ 成比例, 宽度与期望频数 $\sqrt{e_{ij}}$ 成比例, 这样一来, 矩形的面积便与 d_{ij} 成比例; 此外, 矩形自身带有方向, 朝上表示残差为正, 朝下则为负, 不同方向的矩形同时也以不同颜色区分开来。

```
1 > (x = margin.table(HairEyeColor, c(1, 2)))
```

	Eye			
Hair	Brown	Blue	Hazel	Green
Black	68	20	15	5
Brown	119	84	54	29
Red	26	17	14	14
Blond	7	94	10	16

```
1 > assocplot(x)
2 > chisq.test(x)
```

Pearson's Chi-squared test

```
data: x
X-squared = 138.3, df = 9, p-value < 2.2e-16
```

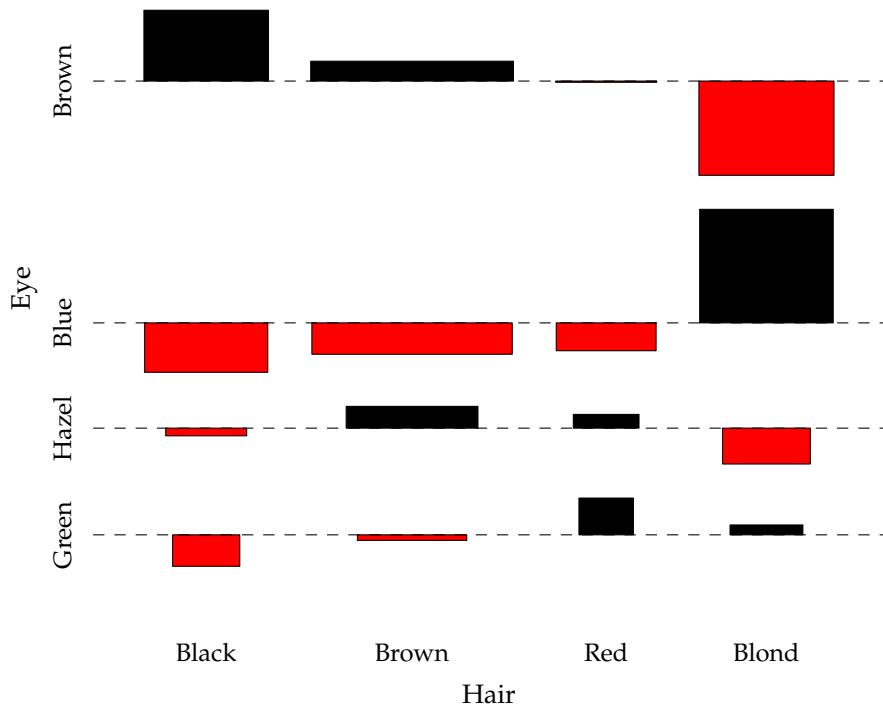


图 5.9: 眼睛颜色与头发颜色的关联图

R中关联图的函数为`assocplot()`, 用法如下:

```
> usage(assocplot)

assocplot(x, col = c("black", "red"), space = 0.3,
          main = NULL, xlab = NULL, ylab = NULL)
```

其中`x`为一个列联表数据 (或者矩阵); `col`为朝上和朝下矩形的颜色; `space`用来设置矩形之间的间距。

图5.9是关于`HairEyeColor`数据的关联图。原始数据为一个三维数组, 首先我们在性别维度上将数据汇总, 得到眼睛颜色 (棕蓝褐绿) 和头发颜色 (黑棕红金) 人数的列联表。我们关心的问题是头发颜色与眼睛颜色之间是否存在关联, 当然我们可以马上用函数`chisq.test()`作检验, 但是检验的结果非常单一, 我们只能知道零假设 (独立) 可否被拒绝, 而图5.9则细致展示了数据的内部信息, 例如从图中我们可以清楚看到, 并非所有单元格都与期望频数有很大差异, 只是少数几个单元格贡献了较大的 χ^2 值, 如金发碧眼、金发棕眼等; 事实上, 这批数据为调查数据, 眼睛颜色和头发颜色都为受访者 (Delaware大学的592名学生) 自己填写, 我们观察到金发碧眼单元格的期望频数和实际频数差异甚大, 据说这背后有一则有趣的故事: 由于“金发碧眼”是大家公认的美的标准, 因此有些学生在填问卷时故意偏向于填写“金发碧眼”, 导致“金发碧眼”的实际频数严重偏高¹。从图5.9的代码输出中我们知道, χ^2 检验可以拒绝零假设, 眼睛的颜色与头发的颜色并不独立, 二者之间存在某种关联关系, 然而这种关联关系是由于生物或遗传原因引起还是受访者有意隐瞒自己的信息, 则需要我们进一步斟酌了。

在`vcd`包(Meyer *et al.*, 2010)中有一个类似的关联图函数`assoc()`, 但功能比本节中介绍的函数要更强大, 详细介绍参见Meyer *et al.* (2006); Zeileis *et al.* (2007)。

5.7 条件密度图

条件密度图 (Conditional Density Plot), 顾名思义, 展示的是一个变量的条件密度, 确切的说是一个分类变量 Y 相对一个连续变量 X 的条件密度 $P(Y|X)$ 。假设 Y 的取值为 $1, 2, \dots, k$, 那么条件密度图将按照 X 的取值从

¹本书作者在爱荷华州立大学统计系读博士期间, 每周统计图形小组有一次讨论, 这则消息来自于作者的一位导师Heike Hofmann教授

```

1 > # 原始数据: 是否失效以及相应温度
2 > fail = factor(c(2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2,
3 +     1, 2, 1, 1, 1, 2, 1, 1, 1, 1), levels = 1:2,
4 +     labels = c("no", "yes"))
5 > temperature = c(53, 57, 58, 63, 66, 67, 67, 67, 68,
6 +     69, 70, 70, 70, 70, 72, 73, 75, 75, 76, 76, 78,
7 +     79, 81)
8 > par(mar = c(4, 4, 0.5, 2))
9 > cdplot(fail ~ temperature, col = c("lightblue", "red"))
10 > # 用黄色的点表示原始数据; 注意y轴位置不是c(1, 2)!
11 > # 请思考为什么 (看源代码 getS3method('cdplot', 'default'))
12 > points(temperature, c(0.25, 0.75)[as.integer(fail)],
13 +     col = "blue", bg = "yellow", pch = 21)

```

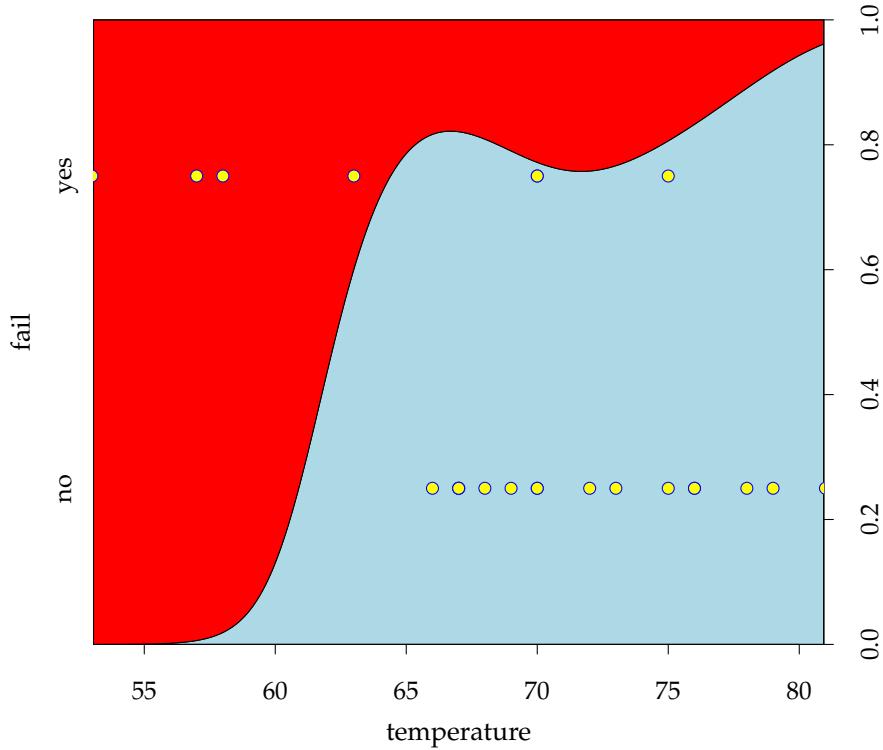


图 5.10: 航天飞机O型环在不同温度下失效的条件密度图: 随着温度升高, O型环越来越不容易失效。

小到大在纵轴方向上依次展示出 $Y = i$ ($i = 1, 2, \dots, k$)的条件概率分布比例 $P_i = P(Y = i|X = x)$, 这些比例大小沿横轴方向上以多边形表示, 在任一个 X 点, 所有比例之和均为1, 这个性质是显而易见的:

$$\sum_{i=1}^k P(Y = i|X = x) = 1; \forall x$$

R中条件密度图的函数为`cdplot()`, 它主要是基于密度函数`density()`完成条件密度的计算(Hofmann and Theus, 2005), 其用法如下:

```

1 > usage(cdplot, "default")
cdplot(x, y, plot = TRUE, tol.ylab = 0.05,
       ylevels = NULL, bw = "nrd0", n = 512, from = NULL,
       to = NULL, col = NULL, border = 1, main = "", xlab = NULL,
       ylab = NULL, yaxlabels = NULL, xlim = NULL, ylim = c(0,
       1), ...)
1 > usage(cdplot, "formula")
cdplot(formula, data = list(), plot = TRUE,
       tol.ylab = 0.05, ylevels = NULL, bw = "nrd0", n = 512,
       from = NULL, to = NULL, col = NULL, border = 1,
       main = "", xlab = NULL, ylab = NULL, yaxlabels = NULL,
       xlim = NULL, ylim = c(0, 1), ..., subset = NULL)

```

函数`cdplot()`是泛型函数, 它可以支持两种参数类型: 直接输入两个数值向量`x`和`y`或者一个公式`y~x`。`x`为条件变量 X , 它是一个数值向量, `y`是一个因子向量, 即离散变量 Y ; `plot`为逻辑值, 决定了是否作出图形 (或者仅仅是计算而不作图); `ylevels`给出因子的取值水平 (或者分类的名称), `bw`、`n`、`from`和`to`都将被传递给`density()`函数以计算密度值, 请参考`density()`帮助文件; `col`给定一个颜色向量用以代表 Y 的各种取值 (默认为不同深浅的灰色); `border`为多边形的边线颜色; 其它参数诸如标题、坐标轴范围等此处略去。

这里我们以美国国家航空和宇宙航行局的一批O型环 (O-ring, 一种由橡胶或塑料制成的平环, 用作垫圈) 失效数据为例, 这批数据有两个变量: 温度变量和是否失效的变量。为了探索温度对O型环失效的影响, 我们可以使用诸如Logistic回归之类的统计模型去计算、分析, 而这里我们用条件密度图来展示温度的影响, 如图5.10。由于因变量是一个二分类变量, 图中相应有两个多边形 (带颜色的区域) 分别表示是否失效, 从图中我们可以清

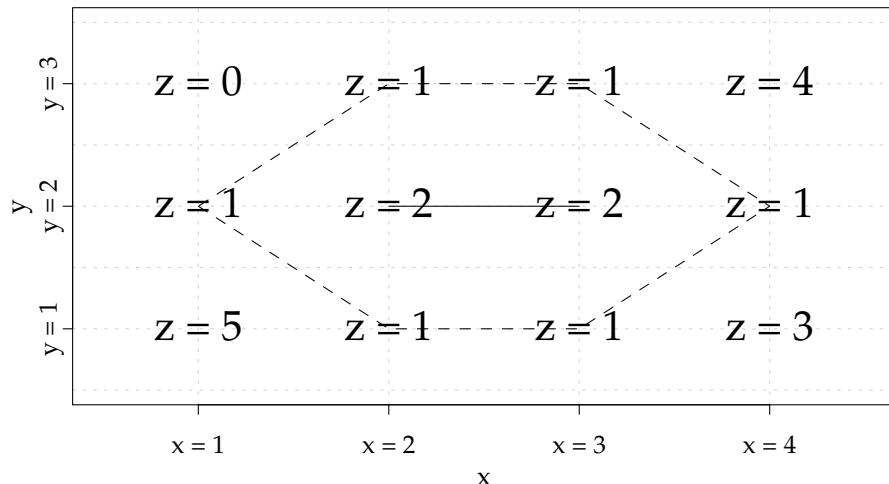


图 5.11: 网格数据的示意图: 体会 x 、 y 与 z 的对应关系

楚观察到, 随着温度的上升, 失效的可能性越来越小 (下面的多边形高度越来越高), 但失效的概率与温度并不是简单的线性关系, 例如55到65之间的温度上升会使得失效概率迅速下降, 而当气温更高的时候, 失效概率下降的速度会减缓。为了更清楚地观察条件密度图的效果, 我们也将原始数据以点的形式添加到图中; 不难发现, O型环失效的情况大多对应着相对较低的温度。

这里需要提醒读者注意的是, 密度值的计算和估计与数据样本量大小有关系, 小的样本量可能会导致密度估计的不精确, 进而导致图形的误导性, 因此使用条件密度图的时候务必注意样本量的问题。

5.8 等高图/等高线

等高图 (Contour Plot) 和等高线 (Contour Line) 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 and Murdoch, 2010), 但是, 在更多的情况下, 我们的图形

```
1 > data(ChinaLifeEdu)
2 > x = ChinaLifeEdu
3 > plot(0, 0, type = "n", xlim = range(x[, 1]), ylim = range(x[, 
4 + 2]), xlab = "预期寿命", ylab = "高学历人数")
5 > u = par("usr")
6 > rect(u[1], u[3], u[2], u[4], col = "antiquewhite",
7 + border = "red")
8 > library(KernSmooth)
9 > est = bkde2D(x, apply(x, 2, dpik))
10 > contour(est$x1, est$x2, est$fhat, nlevels = 15, col = "darkgreen",
11 + add = TRUE, vfont = c("sans serif", "plain"))
12 > points(x)
```

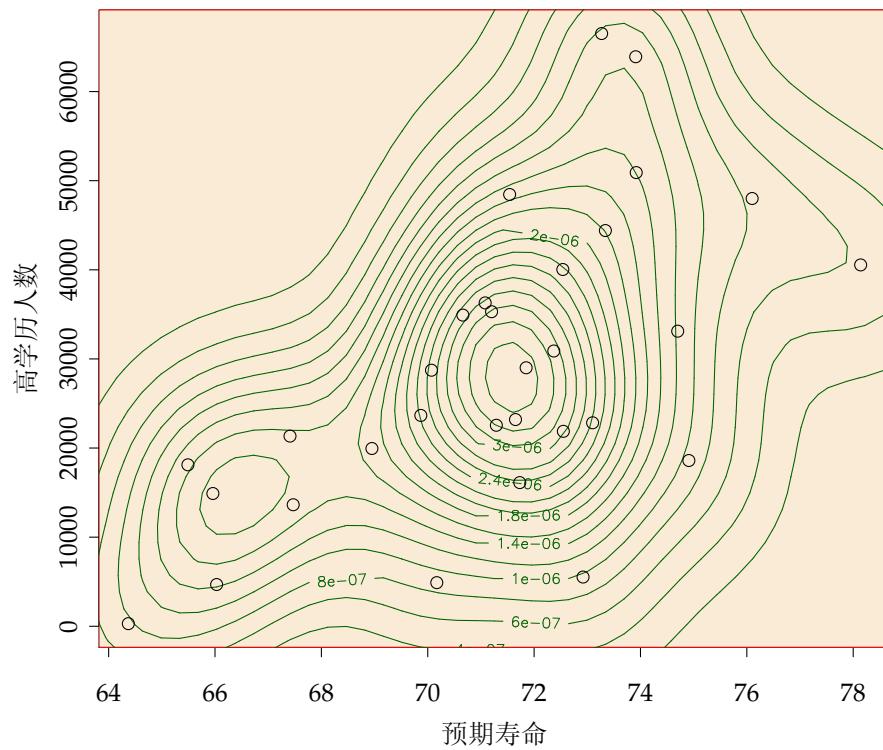


图 5.12: 2005年中国31地区国民预期寿命和高学历人数密度等高图

```
1 > par(mar = rep(0, 4))
2 > # 继续前面的例子
3 > persp(est[["x1"]], est[["x2"]], est[["fhat"]], shade = 0.75,
4 +       border = NA, col = "lightblue", phi = 20, theta = 15,
5 +       box = FALSE)
```

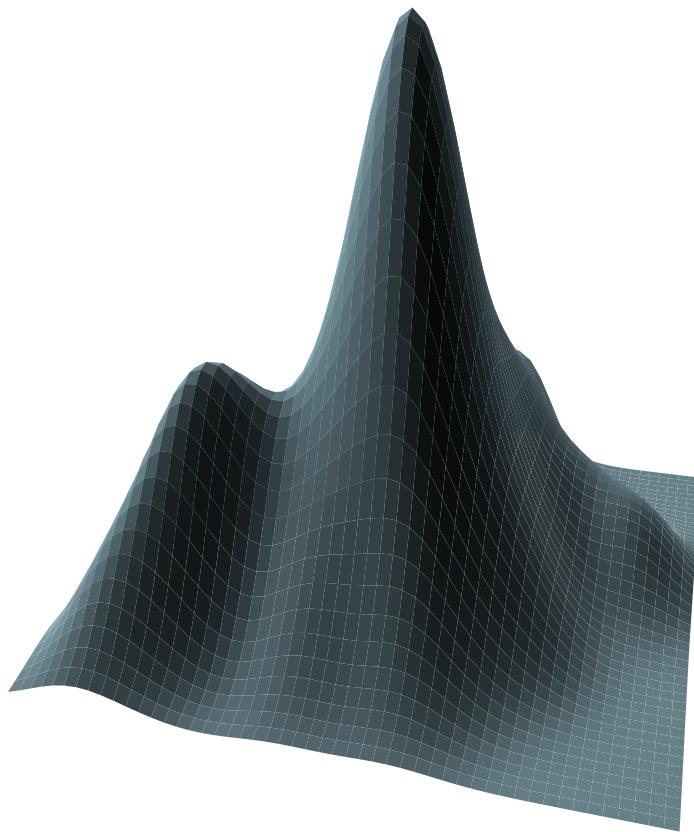


图 5.13: 与等高图对应的三维透视图: 本图与图5.12对应, 从右至左依次有三个山峰, 尤其是中部的山峰最为突出, 对照后面5.26小节中图5.34可知, 这三个山峰分别代表了东中西的省份。

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。

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5.11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5.11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。

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：

```

1 > usage(contour, "default")

contour(x = seq(0, 1, length.out = nrow(z)),
        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 = "flat",
        vfont, axes = TRUE, frame.plot = axes, col = par("fg"),
        lty = par("lty"), lwd = par("lwd"), add = FALSE,
        ...)

1 > usage(contourLines)

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x、y与z此处不再介绍；`nlevels`可以设定等高线的条数、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`为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。

图5.12利用等高图展示了一个聚类现象。数据来源于2005年中国统计年鉴，数值参见**MSG**包中的**ChinaLifeEdu**数据，这里使用了其中两个变量：人口预期寿命（实际数据来自2000年）和高学历人口数量（定义为大专以上学历人数）。首先我们对这二维变量利用**KernSmooth**包(Ripley, 2008)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。由于密度值反映的是某个位置上数据的密集程度，图5.12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5.34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。关于这批数据的分析，我们在5.26小节仍会继续，这里不再深入。

在**graphics**包中还有一个类似的等高图函数`filled.contour()`，它的原理完全类似，只是它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，5.12小节中我们会详细介绍。另外，**lattice**包(Sarkar, 2010)中提供了一个类似的函数`contourplot()`，展示方法更灵活，读者不妨也稍作了解。

5.9 条件分割图

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z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

```

1 > par(mar = rep(0, 4), mgp = c(2, 0.5, 0))
2 > library(maps)
3 > coplot(lat ~ long | depth, data = quakes, number = 4,
4 +     ylim = c(-45, -10.72), panel = function(x, y,
5 +         ...) {
6 +     map("world2", regions = c("New Zealand",
7 +         "Fiji"), add = TRUE, lwd = 0.1, fill = TRUE,
8 +         col = "lightgray")
9 +     text(180, -13, "Fiji", adj = 1)
10 +    text(170, -35, "NZ")
11 +    points(x, y, col = rgb(0.2, 0.2, 0.2, 0.5))
12 + })

```

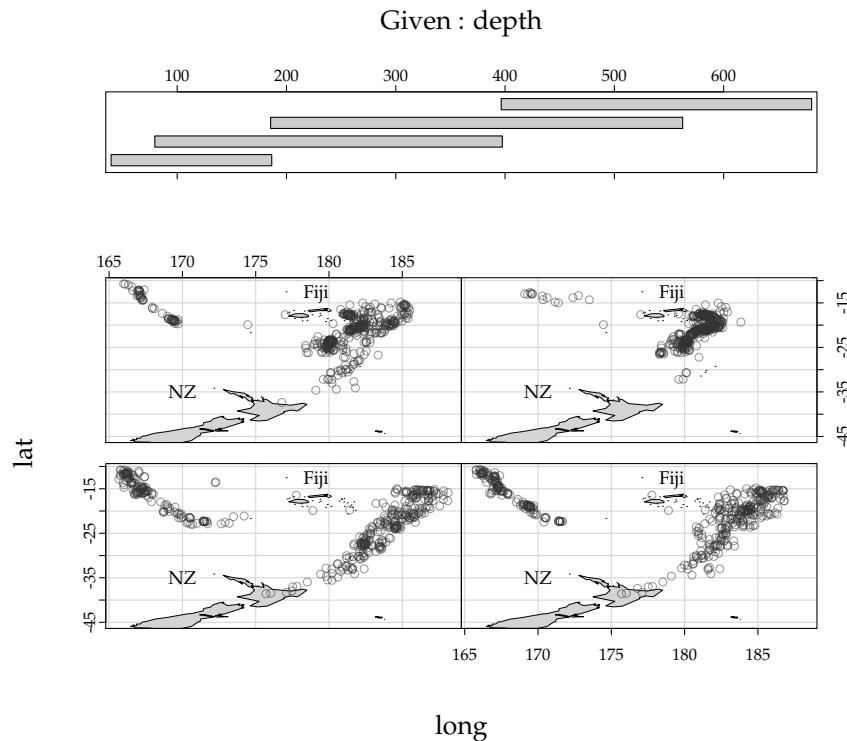


图 5.14: 给定震源深图的地震经纬度条件分割图：四幅散点图有相同的坐标系，震源深度按左下、右下、左上、右上的顺序逐渐增加，可以看到地震发生地点逐渐在向斐济岛靠近。

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。后面我们会结合例子详细说明。

R中条件分割图的函数为`coplot()`，其用法如下：

```

1 > usage(coplot, w = 0.58)

coplot(formula, data, given.values,
       panel = points, rows, columns, show.given = TRUE,
       col = par("fg"), pch = par("pch"),
       bar.bg = c(num = gray(0.8), fac = gray(0.95)),
       xlab = c(x.name, paste("Given :", 
                               a.name)), ylab = c(y.name, paste("Given :", 
                               b.name)), subscripts = FALSE,
       axlabels = function(f) abbreviate(levels(f)),
       number = 6, overlap = 0.5, xlim, ylim,
       ...)

1 > usage(co.intervals)

co.intervals(x, number = 6, overlap = 0.5)

```

参数`formula`为一个公式，形式为`y ~ x | a`（一个条件变量）或`y ~ x | a * b`（两个条件变量），“|”后面即为条件变量；`data`为数据，其中包含了`x`、`y`、`a`和`b`等变量；`given.values`指定条件变量的取值范围；`panel`参数为该函数的关键参数，它决定了每一幅散点图的画法，默认只是画点，我们可以将其任意扩展为我们需要的图示功能，如添加回归直线等等；`rows`和`columns`参数用来设定散点图的摆分行数和列数；`col`和`pch`分别设定散点图中点的颜色和样式；`bar.bg`给定条件变量指示条的填充颜色；`number`和`overlap`传给`co.intervals()`函数用来计算划分连续变量的区间，前者设定划分段数，后者设定区间之间的重叠比例，如：

```

1 > co.intervals(1:10, number = 5, overlap = 0.5)

[,1] [,2]
[1,] 0.5 3.5
[2,] 2.5 5.5
[3,] 3.5 7.5
[4,] 5.5 8.5
[5,] 7.5 10.5

```

```

1 > chippy = function(x) sin(cos(x) * exp(-x/2))
2 > curve(chippy, -8, 7, n = 2008)

```

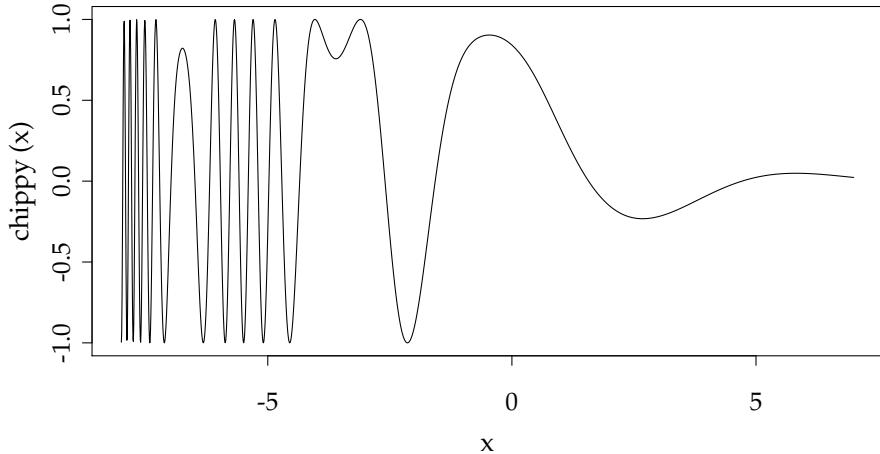


图 5.15: 函数 $f(x) = \sin(\cos(x) * \exp(-x/2))$ 的曲线图

上述代码将数字1:10划分为了5段，每段长度为2，重叠长度为1，因此重叠比例为0.5。条件分割图中散点图的顺序是从左到右、从下到上，分别与条件变量从左到右、从下到上的指示条对应。

图5.14展示了斐济岛（Fiji）附近的地震数据quakes，数据包括地震发生地点的经纬度和震源的深度，我们想知道该地区在地震深度上分布是否均匀，因此我们令深度变量为条件变量，看在不同条件下地震发生地点（经纬度）是否有变化。从图中可以清楚看出，随着深度值的增加，地震发生地点逐渐由西向东、由南向北移动，震源较深的地震都发生在离斐济岛很近的东南侧。另外，图5.14还展示了panel参数的用法，我们借助maps包(Brownrigg, 2010)在散点图上添加了新西兰和斐济岛的地图作为辅助信息，关于R中地图的使用请参考5.34小节。

5.10 一元函数曲线图

函数曲线图没有什么特殊之处，仅仅是一条曲线而已，R专门提供了一个函数，目的是为了节省我们去使用低层作图函数（如`lines()`）的精力和时间。利用这个函数，我们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。

R中函数曲线图的函数为`curve()`, 其用法如下:

```
1 > usage(curve, w = 0.87)

curve(expr, from = NULL, to = NULL, n = 101, add = FALSE,
      type = "l", ylab = NULL, log = NULL, xlim = NULL, ...)
```

参数`expr`为一个一元函数或者该函数的名称; `from`和`to`分别定义了曲线的起点和终点; `n`决定将定义域分成多少个小区间, 以便计算函数值并连接曲线, `n`值越大曲线越光滑; `add`参数决定是否将曲线添加到现有图形上; `type`参数决定了作图类型 (参见3.2小节和图3.4)。注意: 若对一个函数直接应用`plot()`函数, 那么泛型函数`plot()`会自动调用`curve()`完成作图。

图5.15给出了函数 $f(x) = \sin(\cos(x) * \exp(-x/2))$ 的曲线作为示例。由于该函数与数据分析关系不甚密切, 我们在此只是粗略介绍一下。

5.11 Cleveland点图

在前面条形图 (5.4小节) 和后面饼图 (5.27小节) 的章节中我们提到了点图(Cleveland, 1985), 事实上点图和条形图的功能非常类似: 条形图通过条的长度表示数值大小, 点图通过点的位置表示数值大小, 二者几乎可以在任何情况下互换。

R中点图的函数为`dotchart()`, 用法如下:

```
1 > usage(dotchart, w = 0.87)

dotchart(x, labels = NULL, groups = NULL, gdata = NULL,
          cex = par("cex"), pch = 21, gpch = 21, bg = par("bg"),
          color = par("fg"), gcolor = par("fg"), lcolor = "gray",
          xlim = range(x[is.finite(x)]), main = NULL, xlab = NULL,
          ylab = NULL, ...)
```

其中`x`与条形图的`height`参数相同, 为一个数值向量或者矩阵; `labels`为数据的标签; 其它参数主要用来设置图形的样式如颜色、缩放倍数、点的样式等, 此处略去。

图5.16再次以弗吉尼亚死亡率数据为例, 给出了点图的展示。对比图5.7不难发现点图与条形图的相通之处。相比之下, 点图的图形元素更加简洁, 制图时不会显得太拥挤, 我们可以视情况在这二者选其一作为表达工具。

```

1 > dotchart(t(VADeaths)[, 5:1], panel.first = grid(ny = NA),
2 +     cex = 0.8, xlim = c(0, max(VADeaths)), bg = brewer.pal(4,
3 +         "Set1"))

```

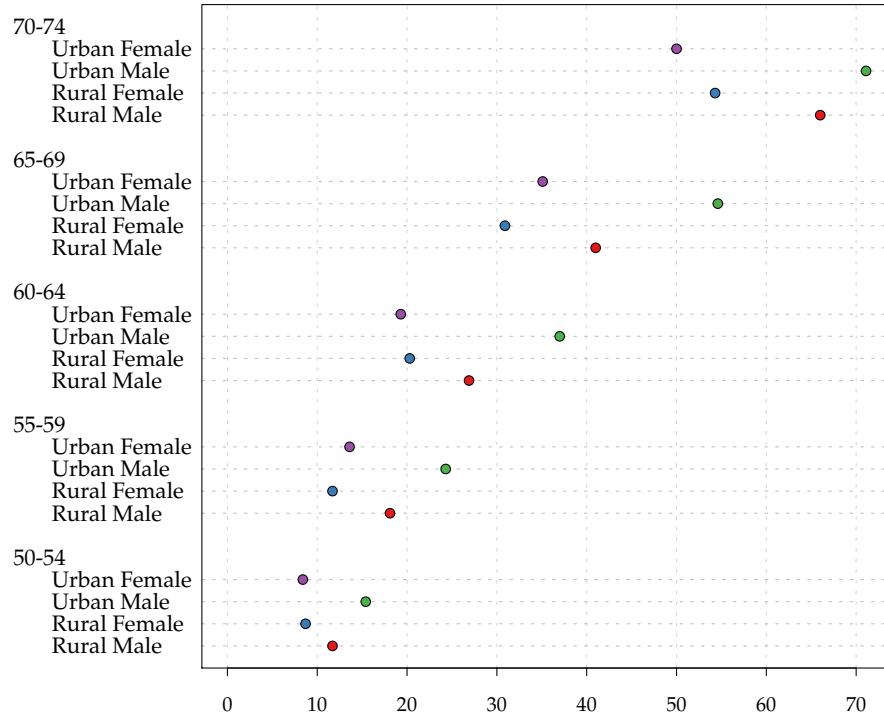


图 5.16: 弗吉尼亚死亡率数据的Cleveland点图

5.12 颜色等高图/层次图

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。读者可以回顾5.8小节关于等高图的介绍。

R中的颜色等高图函数为`filled.contour()`，其用法如下：

```

1 > usage(filled.contour)
filled.contour(x = seq(0, 1, length.out = nrow(z)),
               y = seq(0, 1, length.out = ncol(z)), z, xlim = range(x,

```

```
1 > par(mar = c(4, 4, 2, 2), cex.main = 1)
2 > x = 10 * 1:nrow(volcano)
3 > y = 10 * 1:ncol(volcano)
4 > filled.contour(x, y, volcano, color = terrain.colors,
5 +     plot.title = title(main = "The Topography of Maunga Whau",
6 +         xlab = "Meters North", ylab = "Meters West"),
7 +     plot.axes = {
8 +         axis(1, seq(100, 800, by = 100))
9 +         axis(2, seq(100, 600, by = 100))
10 +     }, key.title = title(main = "Height\n(meters)"),
11 +     key.axes = axis(4, seq(90, 190, by = 10)))
```

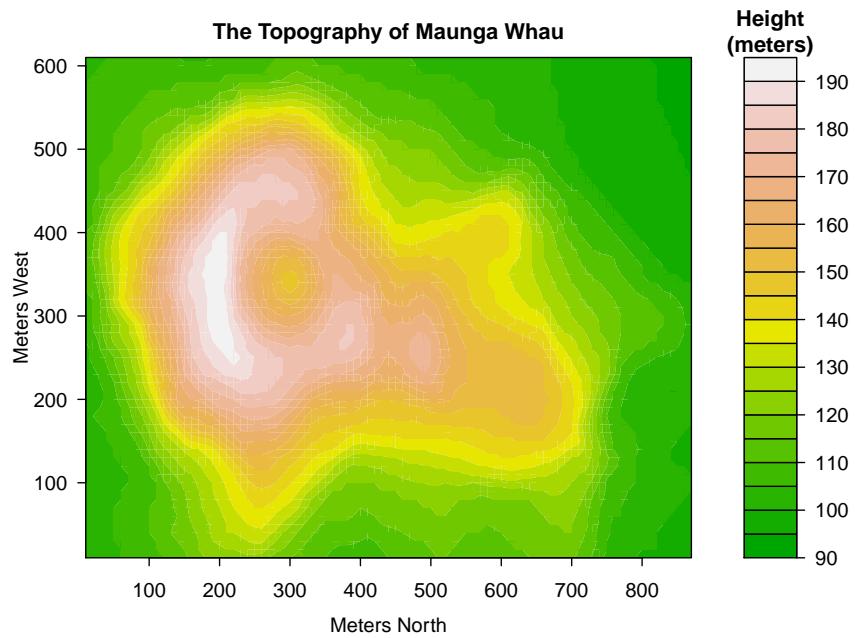


图 5.17: 新西兰 Maunga Whau 火山高度数据颜色等高图

```

finite = TRUE), ylim = range(y, finite = TRUE),
zlim = range(z, finite = TRUE), levels = pretty(zlim,
nlevels), nlevels = 20, color.palette = cm.colors,
col = color.palette(length(levels) - 1), plot.title,
plot.axes, key.title, key.axes, asp = NA, xaxs = "i",
yaxs = "i", las = 1, axes = TRUE, frame.plot = axes,
...)

```

这里面大多数参数与`contour()`函数完全相同，区别在于多了几个定义颜色的参数。`color.palette`给定一个调色板函数，用以生成一系列颜色供等高图填充使用，默认是“青白粉”调色板（回顾4.1.3小节）；如果我们不指定调色板，也可以用`col`参数指定各高度水平对应的颜色；`plot.title`、`plot.axes`、`key.title`和`key.axes`四个参数分别控制着等高图的标题、等高图的坐标轴、图例的标题和图例的坐标轴，它们都能接受若干语句（statement）作为参数值。

图5.17描绘了新西兰Maunga Whau火山的地理数据`volcano`，这份数据包含了在 $10m \times 10m$ 的地理网格上测得的火山高度，是一个 87×61 的矩阵。仔细观察图5.17，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4.1.3小节所介绍的，这种调色板比较适合展示地理数据。图5.24提供了真实的火山立体图形。

颜色等高图中的图形布局（等高图和图例实质上都是独立的图形）是用`layout()`函数（B.2小节）完成的，这给我们带来了扩展上的不便，主要是因为颜色等高图的坐标系统与单幅统计图形的坐标系统并不一样，例如我们无法在作完一幅等高图之后再往图中添加诸如标题、坐标轴等图形元素，这种情况下，我们不妨采用另一种类似的图形—颜色图，参见5.14小节。

5.13 四瓣图

四瓣图（Fourfold Plot）是用来查看 $2 \times 2 \times k$ 列联表中两个二分变量之间关联关系的一种图示工具，它主要是基于二维列联表的检验理论而建立起来的(Friendly, 1994)。

表5.1是一个典型的二维列联表，通常我们想检验的是行变量与列变量

```

1 > # 以紧凑形式展现出来的UCB录取数据
2 > ftable(UCBAmissions)

          Dept   A    B    C    D    E    F
Admit     Gender
Admitted  Male      512 353 120 138  53  22
           Female     89  17 202 131  94  24
Rejected  Male      313 207 205 279 138 351
           Female     19   8 391 244 299 317

1 > # 四瓣图, 2行3列排版
2 > fourfoldplot(UCBAmissions, mfcoll = c(2, 3))
3 > # 对每个系的录取数据进行卡方检验分别得到P值
4 > round(apply(UCBAmissions, 3, function(x) chisq.test(x)$p.value),
5 +       3)

```

	A	B	C	D	E	F
0.000	0.771	0.426	0.638	0.369	0.640	

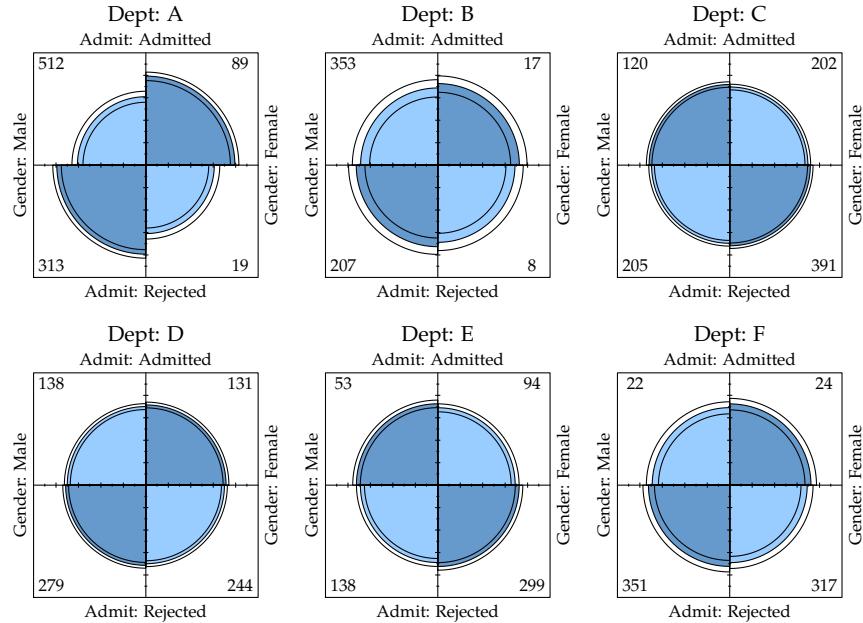


图 5.18: 加州伯克利分校录取数据四瓣图

表 5.1: 二维列联表的经典形式

		事件		
		发生	不发生	
因素	有	P_1	P_3	$P_1 + P_3$
	无	P_2	P_4	$P_2 + P_4$
		$P_1 + P_2$	$P_3 + P_4$	

是否独立；前面5.6小节中曾经利用 χ^2 检验构造了关联图，这里我们从优比(Odds Ratio, OR)的角度出发对列联表进行检验。

首先我们定义以下二式分别为在因素出现和不出现的情况下事件的发生率(医学上常称为风险)：

$$\frac{P_1}{P_1 + P_3}; \quad \frac{P_2}{P_2 + P_4}$$

进而我们用这二式之比得到优比：

$$OR \equiv \frac{P_1}{P_1 + P_3} / \frac{P_2}{P_2 + P_4} = \frac{P_1(P_2 + P_4)}{P_2(P_1 + P_3)} \quad (5.6)$$

由于通常情况下事件发生的几率都比较小(尤其是医学上的疾病)，即 P_1 相对 P_3 来说较小， P_2 相对 P_4 来说较小，因此(5.6)式可以近似用(5.7)式代替：

$$OR \approx \Psi \equiv \frac{P_1 P_4}{P_2 P_3} \quad (5.7)$$

我们记各单元格的样本实现值分别为 a, b, c, d ；如果事件和因素相互独立，那么因素是否发生对事件是否发生(或发生率)没有影响，因此在零假设下优比的样本实现值 ad/bc 应该接近于1，换句话说，如果优比与1显著不同，那么很可能行列变量不独立。

四瓣图正是基于这样一个比例来完成制图的，它将优比体现在两个相邻的四分之一圆的半径之比上，如果两个扇形半径差异显著，那么说明行列变量不独立，即因素对事件有影响，这便是四瓣图最基本的用法，而背后还有关于优比置信区间的计算，并且这个置信区间也在图中用两道弧线表现了出来，四瓣图最终的读法就是观察两瓣相邻扇形的置信区间弧线是否有重叠，有则说明不能拒绝零假设，反之可以拒绝。这是基于假设检验和区间估计之间的转换关系而得以成立的。

计算置信区间需要用到 Ψ 的方差以及正态性假定； Ψ 的方差不容易直接计算，但取对数之后就很容易了：

$$\text{Var}(\log(\Psi)) = \frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d}$$

其置信区间为：

$$\log(\Psi) \pm q_{1-\alpha/2} \sqrt{\text{Var}(\log(\Psi))} \quad (5.8)$$

通过对(5.8)取指数即可还原到 Ψ 本身的置信区间。关于四瓣图的数学理论就介绍到这里，感兴趣的读者可以参阅Friendly (1994)或者直接阅读`fourfoldplot()`的源代码（大约200行）。

R中四瓣图函数`fourfoldplot()`的用法如下：

```
1 > usage(fourfoldplot)

fourfoldplot(x, color = c("#99CCFF", "#6699CC"),
             conf.level = 0.95, std = c("margins", "ind.max",
                                         "all.max"), margin = c(1, 2), space = 0.2,
             main = NULL, mfrow = NULL, mfcoll = NULL)
```

其中 x 是一个 $2 \times 2 \times k$ 的数组，当 $k = 1$ 时，它也可以直接取一个 2×2 的矩阵；`color`设定四分之一圆的填充颜色，处于同一对角线上的扇形颜色相同，颜色填充的顺序也反映出优比与1的大小；`conf.level`为置信水平；`std`为列联表的标准化方法，决定了标准化时分母所除的数。当 $k \geq 1$ 时，该函数会依次生成 k 幅四瓣图。

图5.18是加州伯克利分校 (UCB) 录取数据的四瓣图，数据见于图中的代码输出。这批数据为一个 $2 \times 2 \times 6$ 的数组，我们可以分系别来看学生的录取是否与性别有关。从图中反映的情况来看，只有A系的四瓣图显示出了置信区间弧线不相交的情况，说明A系学生的录取与性别不独立，而其它系都不能拒绝零假设“录取与性别无关”。

现在我们不妨通过一些简单的R语言计算来证实两件事情。首先是扇形颜色的填充与优比的关系，计算优比的代码如下：

```
1 > (x = apply(UCBAdmissions, 3, function(x) (x[1, 1] *
2 + x[2, 2])/(x[1, 2] * x[2, 1])))

          A           B           C           D           E           F
0.3492  0.8025  1.1331  0.9213  1.2216  0.8279
```

C和E系的优比大于1，观察图5.18可知，color参数的第一个颜色值填充第一、三象限的扇形，而优比小于1时，第一个颜色值填充第二、四象限。

其次我们完全可以将优比的置信区间分别算出来，看它们是否包含数值1：

```

1 > y = qnorm(0.975) * sqrt(apply(UCBAdmissions, 3, function(x) {
2 +     sum(1/x)
3 +   }))
4 > conf = exp(cbind(log(x) - y, log(x) + y))
5 > colnames(conf) = c("2.5%", "97.5%")
6 > conf

```

	2.5%	97.5%
A	0.2087	0.5844
B	0.3404	1.8920
C	0.8545	1.5024
D	0.6863	1.2367
E	0.8251	1.8088
F	0.4552	1.5056

显然，这些置信区间中只有A系的不包含1在内，因此对于该系来说可以拒绝零假设。这与图形得到的结论是完全相符的。这里我们提醒读者注意上面的R代码与数学公式的对应关系，很多时候根据数学公式写R代码是非常简单的工作。

5.14 颜色图

颜色图（Color Image）与颜色等高图看起来非常类似，但是等高图需要从网格矩阵中计算等高的数据点，有时还需要一些平滑处理，而颜色图并不涉及任何背后的计算，只是简单将一个网格矩阵映射到指定的颜色序列上、以颜色方块表示数据的大小。在数据规律性较强且数据量较大的时候，这两种图形的区别可以说微乎其微，而当数据没什么规律或者数据量比较小的时候，颜色图的色块就可以很清楚地显露出来了，图5.19为一个简单的示意图。

R中颜色图的函数为`image()`，其用法如下：

```

1 > usage(image, "default", 0.67)
image(x = seq(0, 1, length.out = nrow(z)),
      y = seq(0, 1, length.out = ncol(z)), z,

```

```

1 > par(mar = rep(0, 4))
2 > x = matrix(sample(24), 8)
3 > image(1:8, 1:3, x, col = heat.colors(24), axes = FALSE,
4 + ann = FALSE)
5 > text(rep(1:8, 3), rep(1:3, each = 8), as.vector(x))

```

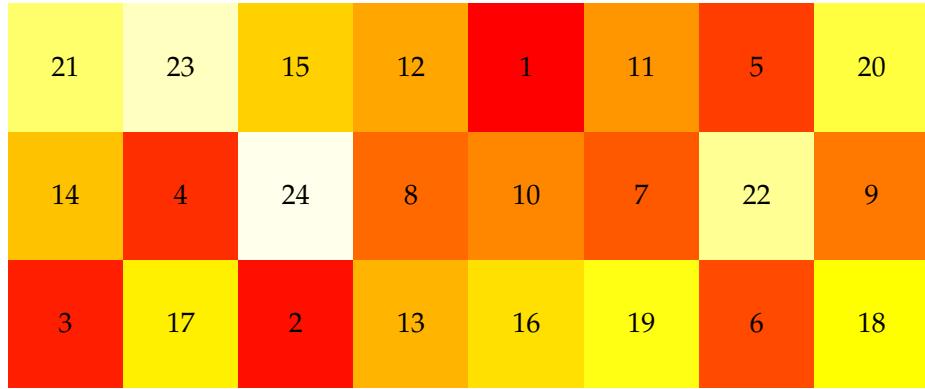


图 5.19: 颜色图中色块与数值的对应关系: 矩阵中数值越大, 色块越趋近于白色, 反之趋近红色。

```

zlim = range(z[is.finite(z)]), xlim = range(x),
ylim = range(y), col = heat.colors(12),
add = FALSE, xaxs = "i", yaxs = "i", xlab,
ylab, breaks, oldstyle = FALSE, ...)

```

参数x、y、z与等高线的参数类似, 不过由于该函数为泛型函数, 因此也可以接受不同类型的参数, 这三个参数除了可以接受两个数值向量和一个矩阵之外, x还可以接受一个列表, 列表中包含三个子对象: x\$x、x\$y和x\$z, 这三个子对象分别为两个数值向量和一个矩阵, 这种情况下就不需要另外单独提供y和z参数了; col设置一个颜色序列以便映射到不同大小的数值; add为逻辑值, 决定是否将颜色图添加到现有图形上; breaks给定z分段的区间端点。

这里我们仍然以新西兰Maunga Whau火山高度数据volcano为例。图5.20是火山数据的颜色图, 从外观上来看, 它与前面的颜色等高图几乎无异(图5.17), 但图5.20中多了一些等高线, 这也说明了颜色图较之颜色等高图的灵活性和可扩展性。在5.12小节的最后我们曾提到颜色等高图作完之后就不容易再往图中添加图形元素, 而这里颜色图只是单幅图形, 作完之

```
1 > par(mar = rep(0, 4), ann = FALSE)
2 > x = 10 * (1:nrow(volcano))
3 > y = 10 * (1:ncol(volcano))
4 > image(x, y, volcano, col = terrain.colors(100), axes = FALSE)
5 > contour(x, y, volcano, levels = seq(90, 200, by = 5),
6 +       add = TRUE, col = "peru")
7 > box()
```

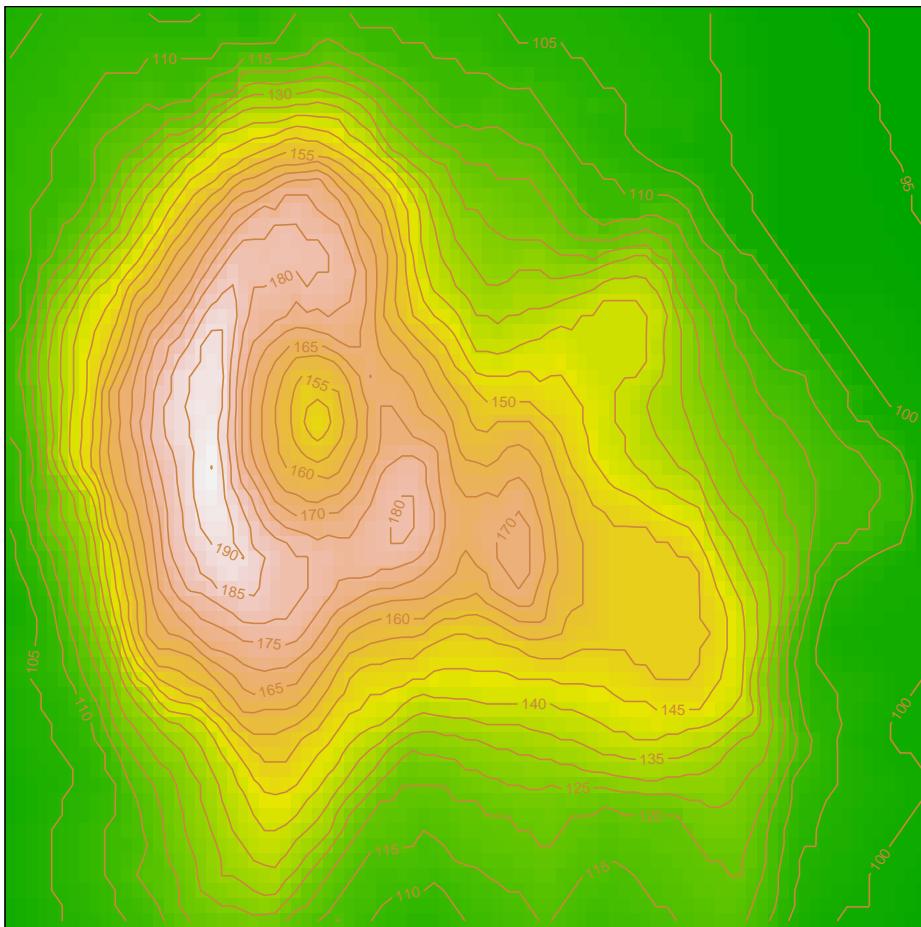


图 5.20: 新西兰 *Maunga Whau* 火山高度数据颜色图: 图5.24提供了真实的火山立体图形; 读者可以到作者个人主页下载原图并放大8倍以上查看颜色图和颜色等高图的区别: 颜色图是用颜色填充方块, 而颜色等高图则是用颜色填充等高线之间的区域。

后仍然可以方便地添加图形元素。

统计数据中有不少是矩阵形式，例如相关系数阵、协方差阵等，我们可以将颜色图应用到这些矩阵形式数据的展示上，尤其是当矩阵行数列数较大时，我们可以借助人眼对颜色的视觉感知从颜色图中迅速找出一定的统计特征来（如很大或者很小的数值），而相比之下，对数据的直接观察并不容易找到规律或特征，因为这种形式下我们必须在脑中对数据两两比较，其速度必然会很慢。

lattice包(Sarkar, 2010)中提供了一个类似的函数*levelplot()*，展示方法更为灵活，感兴趣的读者请参考函数的帮助文件。

5.15 矩阵图、矩阵点、矩阵线

矩阵图的名称来自于其参数类型，它可以针对一个矩阵将所有列以曲线的形式表达出来，同一元函数曲线图（5.10小节）一样，它也没有什么特别之处，仅仅是提供了一个便利的封装，我们可以不必调用*lines()*等函数依次对矩阵的所有列画曲线。

R中矩阵图的函数为*matplot()*，矩阵点的函数为*matpoints()*，矩阵线的函数为*matlines()*，它们的用法如下：

```

1 > usage(matplot, w = 0.7)

matplot(x, y, type = "p", lty = 1:5,
        lwd = 1, lend = par("lend"), pch = NULL, col = 1:6,
        cex = NULL, bg = NA, xlab = NULL, ylab = NULL,
        xlim = NULL, ylim = NULL, ..., add = FALSE,
        verbose =getOption("verbose"))

1 > usage(matpoints, w = 0.99)

matpoints(x, y, type = "p", lty = 1:5, lwd = 1, pch = NULL,
           col = 1:6, ...)

1 > usage(matlines, w = 0.99)

matlines(x, y, type = "l", lty = 1:5, lwd = 1, pch = NULL,
          col = 1:6, ...)

```

函数*matplot()*为高层作图函数（创建新图形），而后两个函数均为低层作图函数（向现有图形上添加元素）。参数x和y为输入的矩阵，做图的方

```

1 > sines = outer(1:20, 1:4, function(x, y) sin(x/20 *
2 + pi * y))
3 > par(mar = c(1, 4, 0.1, 0.1))
4 > matplot(sines, type = "b", pch = 21:24, col = 2:5,
5 + bg = 2:5)
6 > # 数据矩阵的前6行
7 > round(head(sines), 5)

```

	[,1]	[,2]	[,3]	[,4]
[1,]	0.1564	0.3090	0.4540	0.5878
[2,]	0.3090	0.5878	0.8090	0.9511
[3,]	0.4540	0.8090	0.9877	0.9511
[4,]	0.5878	0.9511	0.9511	0.5878
[5,]	0.7071	1.0000	0.7071	0.0000
[6,]	0.8090	0.9511	0.3090	-0.5878

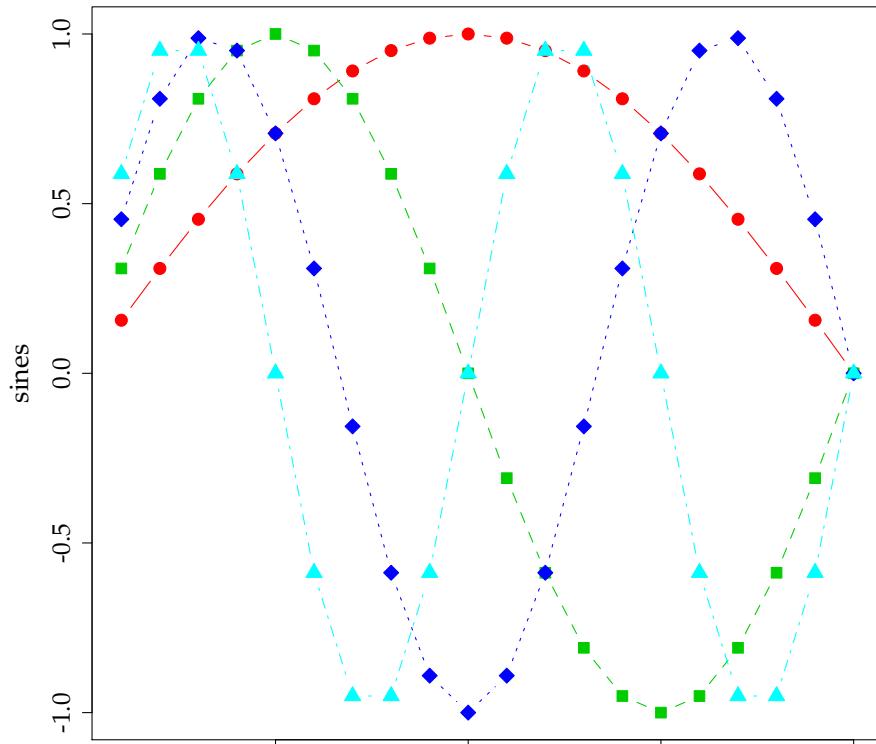


图 5.21: 用矩阵图画出的一系列正弦曲线: 每条曲线都有不同的点线样式和颜色。

式是用x的列为横轴方向的变量，y的列为纵轴方向的变量，然后用这些列依次作散点图（x的第一列对y的第一列，x的第二列对y的第二列，依次类推）；如果这两个参数有一个缺失，那么x将被`1:nrow(y)`代替，y被非缺失的参数矩阵代替；注意两个矩阵要么有一个列数为1，要么列数相等，否则会报错；后面设置颜色、线型等样式的参数`type`、`lty`、`lwd`、`pch`、`col`、`cex`、`bg`等在前面第三章和第四章已经讲述过多次，此处不再赘述。

图5.21展示了一个正弦值矩阵的矩阵图。从图上方的代码中我们可以看到矩阵的前6行数值，该例中只给出了参数x而没有y，所以`matplot()`用矩阵`sines`的每一列依次和`1:20`画曲线。

5.16 马赛克图

马赛克图（Mosaic Plots）是展示多维列联表数据的工具。前面我们已经提到过两种展示列联表数据的工具（5.6和5.13小节），但它们都只能展示低维列联表，而马赛克图对于列联表的维数没有限制。

马赛克图的表现形式为与频数成比例的矩形块，整幅图形看起来就像是若干块马赛克放置在平面上。马赛克图背后的统计理论是对数线性模型（log-linear model），我们先回顾一个最简单的二维列联表的独立模型。

二维列联表的独立性从概率角度来说就是单元格的频率等于边际频率的乘积：

$$\pi_{ij} = \pi_i \cdot \pi_{\cdot j} \quad (5.9)$$

取对数即得：

$$\log(\pi_{ij}) = \log(\pi_i \cdot) + \log(\pi_{\cdot j}) \quad (5.10)$$

根据 $\mu_{ij} = n\pi_{ij}$ 进一步写成频数的形式：

$$\log(\mu_{ij}) = \lambda + \lambda_i^r + \lambda_j^c \quad (5.11)$$

λ_i^r 和 λ_j^c 分别表示行效应和列效应， λ 为常数。（5.11）式就是最普通的对数线性模型，通过计算拟合，我们可以得到行列效应的估计值。对数线性模型在马赛克图中的主要表现是单元格的残差，而单元格的残差可以有三种：似然比残差（离差，deviance） G^2 、Pearson χ^2 残差和Freeman-Tukey残差，前两种定义如下式：

$$G^2 = 2 \sum n_{ij} \log\left(\frac{n_{ij}}{\hat{\mu}_{ij}}\right); \quad \chi^2 = \sum \frac{(n_{ij} - \hat{\mu}_{ij})^2}{\hat{\mu}_{ij}} \quad (5.12)$$

```
1 > ftable(Titanic)
```

			Survived	No	Yes
Class	Sex	Age			
1st	Male	Child	0	5	
		Adult	118	57	
	Female	Child	0	1	
		Adult	4	140	
2nd	Male	Child	0	11	
		Adult	154	14	
	Female	Child	0	13	
		Adult	13	80	
3rd	Male	Child	35	13	
		Adult	387	75	
	Female	Child	17	14	
		Adult	89	76	
Crew	Male	Child	0	0	
		Adult	670	192	
	Female	Child	0	0	
		Adult	3	20	

```
1 > par(mar = c(2, 3.5, 0.1, 0.1))
2 > mosaicplot(Titanic, shade = TRUE, main = "")
```

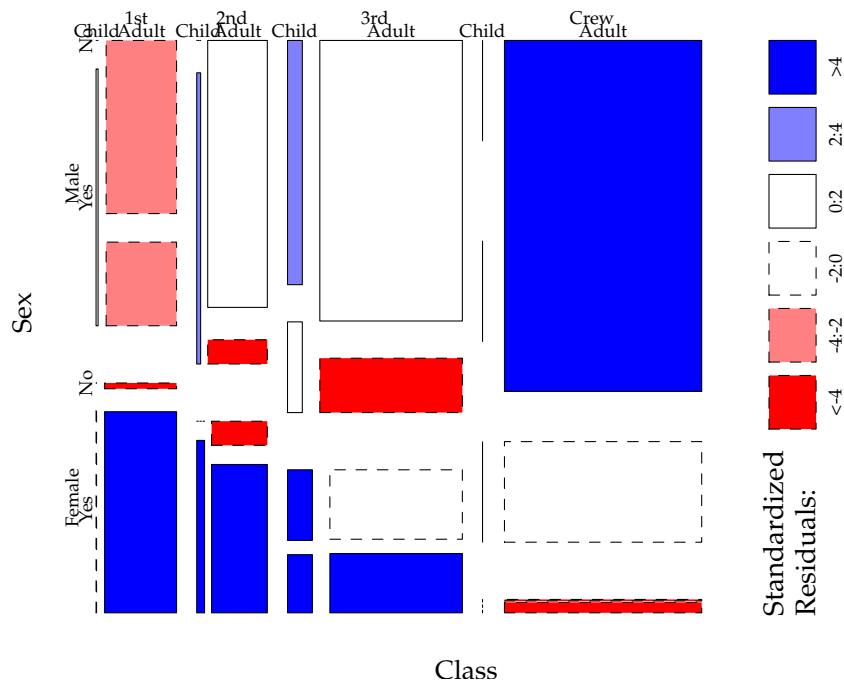


图 5.22: 泰坦尼克号乘客生存数据马赛克图: 按性别、年龄和船舱等级划分

残差反映的是某个单元格拟合的好坏，马赛克图用5级颜色表达了残差的大小，后面我们结合具体例子说明。

R中马赛克图的函数为*mosaicplot()*，其用法如下：

```

1 > usage(mosaicplot, "default")

mosaicplot(x, main = deparse(substitute(x)),
           sub = NULL, xlab = NULL, ylab = NULL, sort = NULL,
           off = NULL, dir = NULL, color = NULL, shade = FALSE,
           margin = NULL, cex.axis = 0.66, las = par("las"),
           type = c("pearson", "deviance", "FT"), ...)

1 > usage(mosaicplot, "formula", 0.7)

mosaicplot(formula, data = NULL, ...,
           main = deparse(substitute(data)), subset,
           na.action = stats::na.omit)

```

马赛克图函数是泛型函数，可以直接接受列联表数据或者公式作为参数，这里我们只介绍前一种情况。*x*为一个列联表数据（可以用函数*table()*生成）；*main*、*sub*、*xlab*和*ylab*分别设定主标题、副标题和坐标轴标题；*sort*指定展示变量的顺序；*dir*指定马赛克图的拆分方向（横向拆分或纵向拆分）；*type*给定残差的类型，即如前所述的三种残差。

下面我们结合泰坦尼克号数据*Titanic*来说明马赛克图的用法。泰坦尼克号乘客生存情况的原始数据参见图5.22的代码输出，该数据给出了分舱位（一二三等舱和船员舱）、分性别（男女）、分年龄（大人小孩）的生存情况。泰坦尼克号的沉没是一件著名的历史事件，至今仍然有很多人在研究它。我们所关心的问题主要是通过一些比例看出当时救援的侧重性，如：是否头等舱的乘客生还比例最高？“女士和孩子优先”的原则在各船舱有没有被很好遵守？……

图5.22以马赛克图的形式将这个 $4 \times 2 \times 2 \times 2$ 的列联表数据展示在了同一张图中，通过矩形块（马赛克）的大小，我们可以清楚看出各舱位、不同性别、年龄的人群的生还状况。例如，对头等舱来说，无论是大人小孩或男女，下方的矩形都比上方的矩形要高（尤其是女性和小孩），这说明头等舱的生还率相对来说都比较高，很可能当时的救援是偏向头等舱的；从年龄来说，头等舱和二等舱中小孩的生存率要远高于大人，但三等舱中小孩的生存率和大人相比差异并不是太显著；但从性别角度来看，各舱位基本上还是将生存机会优先让给女性了，男性的生还率在各舱位来说都相对

较低。类似地，我们还可以从图中挖掘出更多的现象，这里不再深入。另外，图中用不同颜色表示出了个单元格的残差大小，其中虚线框表示残差为负数，我们可以清楚看出哪些单元格的拟合欠佳。感兴趣的读者还可以使用**stats**包中的`loglin()`函数拟合对数线性模型、从统计模型的角度继续分析。

5.17 散点图矩阵

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，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，它通常包含 $p \times p$ 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。

R中散点图矩阵的函数为`pairs()`，其用法如下：

```

1 > usage(pairs, "default", 0.7)

pairs(x, labels, panel = points, ...,
      lower.panel = panel, upper.panel = panel,
      diag.panel = NULL, text.panel = textPanel,
      label.pos = 0.5 + has.diag/3, cex.labels = NULL,
      font.labels = 1, rowtattop = TRUE, gap = 1)

1 > usage(pairs, "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。`x`是一个矩阵或数据框，包含了要作散点图的那些变量；`labels`是变量名称（标签）；`panel`参数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；`label.pos`指定文本标签的位置；`cex.labels`指定标签的缩放

```

1 > panel.hist = function(x, ...) {
2 +   usr = par("usr")
3 +   on.exit(par(usr))
4 +   par(usr = c(usr[1:2], 0, 1.5))
5 +   h = hist(x, plot = FALSE)
6 +   nB = length(breaks <- h$breaks)
7 +   y = h$counts/max(h$counts)
8 +   rect(breaks[-nB], 0, breaks[-1], y, col = "beige")
9 + }
10 > idx = as.integer(iris[["Species"]])
11 > pairs(iris[1:4], upper.panel = function(x, y, ...) points(x,
12 +   y, pch = c(17, 16, 6)[idx], col = idx), pch = 20,
13 +   oma = c(2, 2, 2, 2), lower.panel = panel.smooth,
14 +   diag.panel = panel.hist)

```

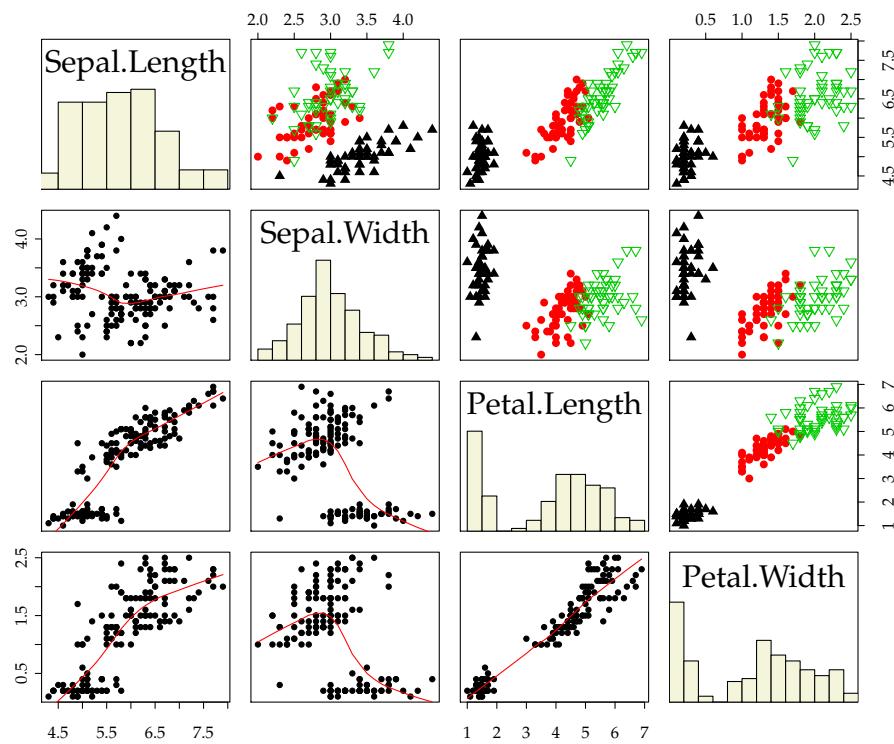


图 5.23: 鸢尾花数据的散点图矩阵: 上三角区域为不同样式的点, 对应着不同种类的鸢尾花, 对角线的直方图展示了花瓣花萼长宽的一维分布, 下三角区域用平滑曲线显示了变量之间的关系。

倍数; `font.labels`指定标签的字体样式; `row1attop`为逻辑值, 指定散点图的第1行出现在顶部还是底部²; `gap`设定窗格之间的间距大小。

图5.23是对鸢尾花数据*iris*所作的散点图矩阵, 注意其中的上三角、下三角和对角线窗格作图函数是如何定义的。

我们可以看到, 主对角线上用了直方图(注意直方图的作法!), 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(回顾图4.4); 下三角窗格中简化了点的样式, 但是利用函数`panel.smooth()`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))。

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。

5.18 三维透视图

相比其二维平面图形来说, 三维透视图(Perspective Plot)可能在视觉上更具有吸引力。三维透视图的数据基础是网格数据(回顾5.8小节和图5.11), 它将一个矩阵中包含的高度数值用曲面连接起来, 便形成了我们所看到的三维透视图。前面等高图一节中我们曾经用到过透视图, 参见图5.13。

R中透视图的函数为`persp()`, 其用法如下:

```

1 > usage(persp)
persp(x, ...)

1 > usage(persp, "default")
persp(x = seq(0, 1, length.out = nrow(z)),
      y = seq(0, 1, length.out = ncol(z)), z, xlim = range(x),
      ylim = range(y), zlim = range(z, na.rm = TRUE),
      xlab = NULL, ylab = NULL, zlab = NULL, main = NULL,
      sub = NULL, theta = 0, phi = 15, r = sqrt(3), d = 1,
```

²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, 所以第1行出现在顶部则是矩阵形式, 在底部则是图的形式。

```
1 > # 放大火山高度, 使立体效果更明显
2 > z = 3 * volcano
3 > x = 10 * (1:nrow(z))
4 > y = 10 * (1:ncol(z))
5 > par(mar = rep(0, 4))
6 > persp(x, y, z, theta = 150, phi = 30, col = "green3",
7 +       scale = FALSE, ltheta = -120, shade = 0.75, border = NA,
8 +       box = FALSE)
```

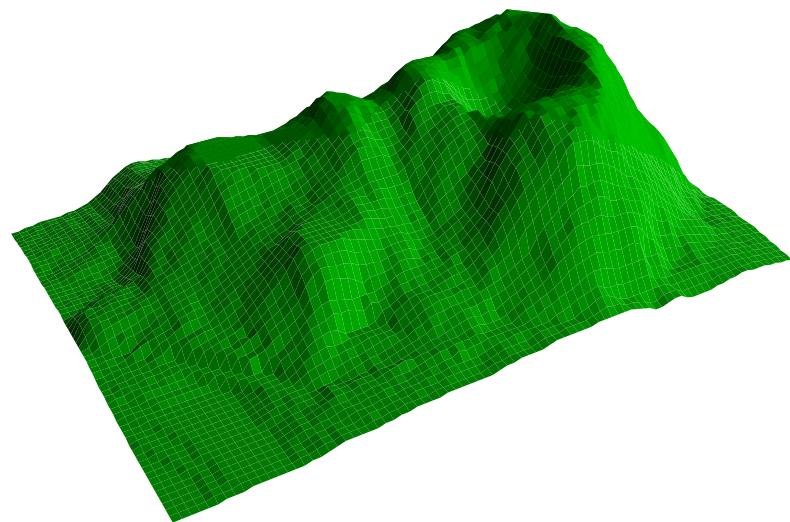


图 5.24: 新西兰 Maunga Whau 火山的三维透视图

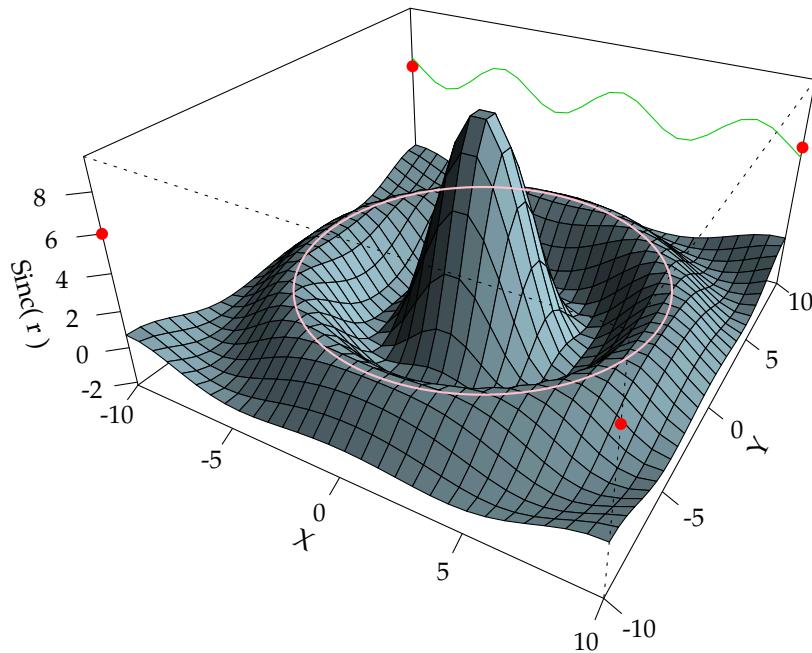


图 5.25: 向三维透視图中添加图形元素的展示: `trans3d()`函数的应用。本图代码参见`example(persp)`。

```
scale = TRUE, expand = 1, col = "white", border = NULL,
ltheta = -135, lphi = 0, shade = NA, box = TRUE,
axes = TRUE, nticks = 5, ticktype = "simple", ...)
```

透視图函数为泛型函数，主要体现在它的第一个参数既可以是单个x向量，也可以是一个包含了向量x和y的列表，这一点与等高图函数是类似的。参数x和y分别是两个数值向量，z是一个与前两个参数对应的矩阵；xlim、ylim和zlim分别设定三个坐标轴的范围；xlab、ylab和zlab分别设定三个坐标轴的标题；theta和phi分别设定立体图形左右方向和上下方向旋转的角度；r设定眼睛离透視图中心的距离，这个距离的远近会给我们一种从远近看物体的感觉；d设定立体效果的程度，大于1的值会减弱立体程度，反之会增强立体程度；scale为逻辑值，决定是否对三个坐标进行缩

放，若为TRUE，则x、y和z都会被缩放到[0, 1]范围内，若为FALSE，那么所有坐标轴都按照数据的原始量纲处理，这样可以得到数据的真实比例；expand为z轴的缩放因子，它决定了z轴的长短；col为组成曲面的所有小方块的颜色；border为组成曲面的小方块的边框样式，设置为NA可以去掉边框；ltheta和lphi设置透视图的光源位置；shade决定的阴影效果；box为逻辑值，设定透视图是否需要画外框；axes决定是否画坐标轴；nticks为坐标轴刻度线的数目；ticktype设定坐标轴刻度类型，取值'simple'则简单画箭头表示坐标轴，'detailed'则将详细的刻度标记在坐标轴上。

图5.24为我们展示了新西兰Maunga Whau火山的“真面目”，读者不妨将这幅立体图形与前面章节中的等高图（图5.17）和颜色图（图5.20）对应起来并分别体会等高图和颜色图是怎样展示三维数据的。

grDevices包提供了一个相关的三维透视图转换函数`trans3d()`，它可以将一个空间的点的三维坐标根据透视图的特征转换为平面坐标，这样我们就可以很方便地使用一般的底层作图函数向立体图中添加图形元素。图5.25就是这样的一个例子，读者可以参考`persp()`帮助文件中的示例。

最后，我们介绍另一个专门的三维图形包：**scatterplot3d** (Ligges and Mächler, 2003)，这个包提供了更方便且美观的作图函数`scatterplot3d()`；在**lattice**包(Sarkar, 2010)中也有三维图形函数`cloud()`和`wireframe()`；此外，**rgl**包(Adler and Murdoch, 2010)也不失为一个非常便利的三维数据探索工具，它基于OpenGL系统写成，最大的优势在于它生成的三维图形可以通过鼠标交互操作，例如拖拽旋转等，立体效果非常逼真，限于**rgl**的系统比较庞大，这里我们就不详细介绍。

5.19 因素效应图

方差分析是很常见的统计模型，它的目的是比较不同组别之间的因变量均值是否有显著差异，因素效应图所展现的就是各种分组条件下因变量的水平，这里的水平可以由任何统计函数定义，例如均值、中位数等。

R中因素效应图的函数为`plot.design()`，其用法如下：

```
1 > usage(plot.design)

plot.design(x, y = NULL, fun = mean, data = NULL,
... , ylim = NULL, xlab = "Factors", ylab = NULL,
main = NULL, ask = NULL, xaxt = par("xaxt"), axes = TRUE,
xtick = FALSE)
```

```
1 > plot.design(warpbreaks, col = "blue")
```

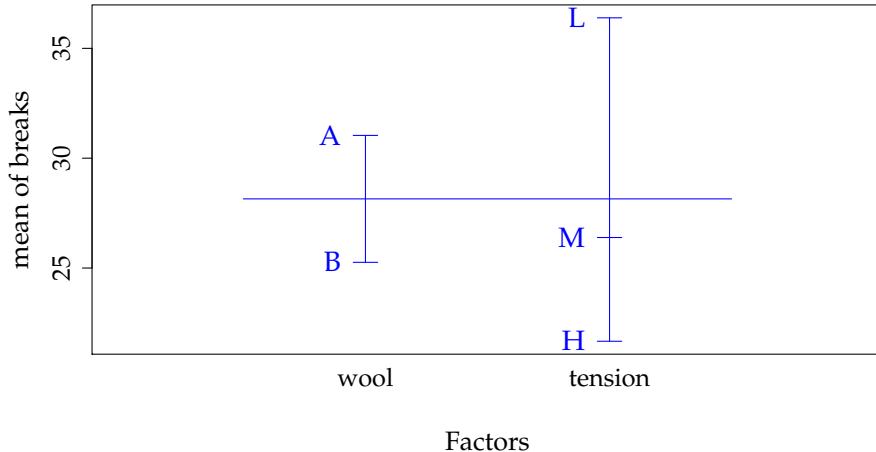


图 5.26: 经纱断裂数据的因素效应图: 每种羊毛 (A、B) 和拉力强度 (L、M、H) 下断裂数目的均值。

`x`为包含自变量（分类变量）的数据框，它也可以包含因变量，这种情况下第二个参数就不必提供了；`y`为因变量；`fun`为计算因变量水平的函数；`data`为所有数据，包含自变量和因变量，当参数`x`为一个公式时，则会用到本参数提取数值，否则不必提供本参数；其它参数用于调整图形外观或设置图形标题、标记等，此处略去。

我们以经纱断裂数据`warpbreaks`为例，这个数据包含三个变量：经纱断裂数目`breaks`、羊毛种类`wool`（A、B两种）和拉力强度`tension`（L、M、H三种强度）。我们可以通过因素效应图看羊毛种类和拉力强度两个变量分别对经纱断裂根数的影响，如图5.26。图中的刻度线展示出了不同羊毛种类和拉力强度下断裂根数的均值。我们也可以通过方差分析模型进一步看这两种因素的影响：

```
1 > summary(aov(breaks ~ wool + tension, data = warpbreaks))

   Df Sum Sq Mean Sq F value Pr(>F)
wool       1    451     451    3.34 0.0736 .
tension     2   2034    1017    7.54 0.0014 **
Residuals  50   6748     135
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

从以上结果中可以看出，拉力强度对经纱断裂根数有显著影响，而羊毛种类则不太显著。

总的来说，因素效应图是一种非常初级的统计图形，在数据的探索性分析中可能会起到一定作用。有时我们完全可以用一些分类汇总函数去计算各因素的效应（各组均值），例如：

```

1 > aggregate(warpbreaks$breaks, list(wool = warpbreaks$wool),
2 +      mean)

      wool      x
1     A 31.04
2     B 25.26

1 > aggregate(warpbreaks$breaks, list(tension = warpbreaks$tension),
2 +      mean)

      tension      x
1       L 36.39
2       M 26.39
3       H 21.67

```

5.20 坐标轴须

坐标轴须（Rug）顾名思义就是往坐标轴上添加短须。短须的作用是标示出相应坐标轴上的变量数值的具体位置，每一根短须都对应着一个数据。这样做好处在于，我们可以从坐标轴须的分布了解到该变量的分布，尤其是当我们使用那些带有汇总性质的图形（如箱线图）时，我们会失去原始数据的位置，得到的只是一幅展示综合统计量的图形。坐标轴须与一些统计图形的结合会让图形表达的信息更丰富，图形的使用者也可以自由选择看图的侧重点。

R中坐标轴须的函数为rug(), 其用法如下:

```

1 > usage(rug, w = 0.9)

rug(x, ticksize = 0.03, side = 1, lwd = 0.5, col = par("fg"),
     quiet = getOption("warn") < 0, ...)

```

其中x为一个向量，给出短须的位置；ticksize为短须的长度；side为欲画短须的坐标轴的位置（与函数axis()的参数相同，参见4.8小节）；lwd和col分别设定短须的宽度和颜色。

```

1 > par(mar = c(3, 4, 0.4, 0.1))
2 > with(faithful, {
3 +   plot(density(eruptions), main = "")
4 +   rug(eruptions)
5 + })

```

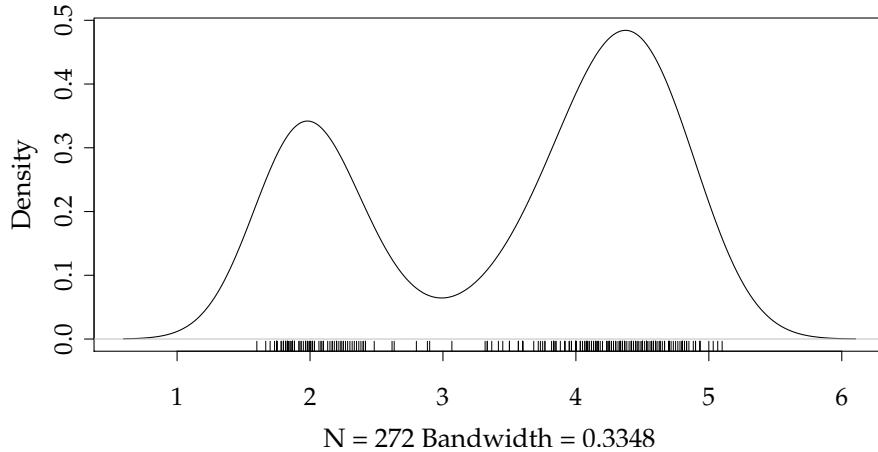


图 5.27: 带坐标轴须的喷泉间隔时间密度曲线图

图5.27为坐标轴须的一个展示；在5.1小节中我们使用喷泉数据制作了直方图和密度图以便查看间隔时间的分布，这里我们在密度曲线的基础上添加上坐标轴须，以使我们能更清楚了解间隔时间数据的具体位置，而不是仅仅看一条曲线。从坐标轴须的疏密我们也可以知道分布密度的大小，这与密度曲线是相辅相成的。

严格来说，坐标轴须不能算作是“图形”，它只是图形的附加物，从R的角度来说它也只是低层作图函数，但由于它在表达数据上的优势，这里我们也将其列入本书的图库中。

5.21 平滑散点图

平滑散点图的基础是散点图，但它并不直接将散点画出来，而是基于二维核密度估计(Wand, 2009)用特定颜色深浅表示某个位置的密度值大小，默认颜色越深说明二维密度值越大，即该处数据点越密集。二维核密度估计的原理和一维情况类似：一维核密度估计是在直线上按照距离远近对每

```

1 > par(mar = c(3.5, 4, 0.3, 0.1))
2 > smoothScatter(BinormCircle)

```

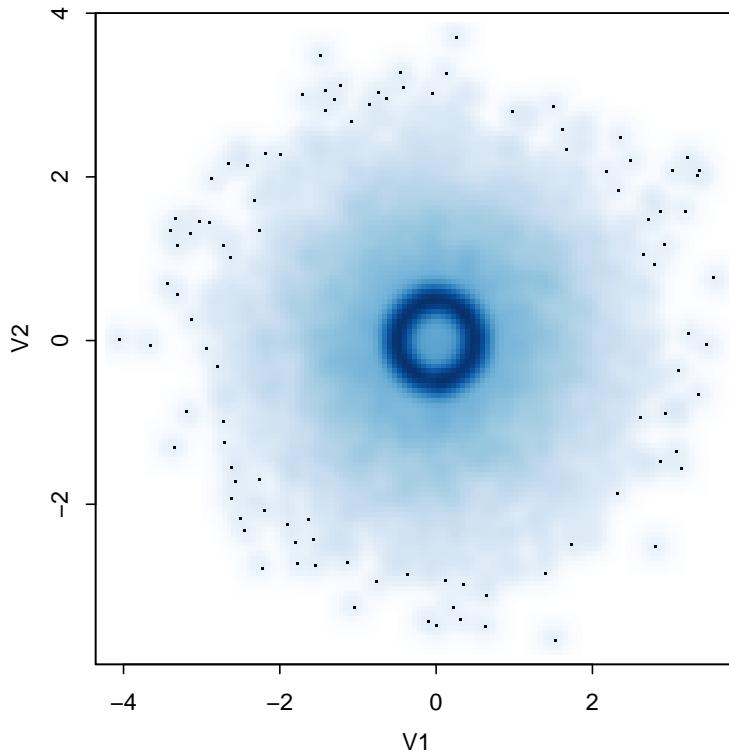


图 5.28: *BinormCircle*数据的平滑散点图: 基于核密度估计找出散点图中暗含的圆圈。

个数据点加权, 距离越近则对密度值贡献越大, 因此数据点密集的地方的核密度值也相应大, 二维情况下只是“距离”的计算放到了平面上, 加权思想相同。

由于平滑散点图大致保留了原始数据点的位置, 因此两个变量之间的关系仍然可以从图中看出来, 这一点和普通的散点图类似。平滑散点图进一步的优势在于它同时还显示了二维变量的密度, 从密度中我们也许可以观察到局部的聚类现象 (大块的深色)。

从计算的角度来说, 我们首先将平面划分为 $n \times n$ 的网格, 计算每个网格点上的二维密度值, 然后用颜色将密度值大小表达出来。关于这一点,

可将图5.28放大8倍，就立刻理解“网格”的意思了。

R中平滑散点图的函数为*smoothScatter()*，其用法如下：

```
1 > usage(smoothScatter)

smoothScatter(x, y = NULL, nbin = 128, bandwidth,
               colramp = colorRampPalette(c("white", "blues9")),
               nrpoints = 100, pch = ".", cex = 1, col = "black",
               transformation = function(x) x^0.25, postPlotHook = box,
               xlab = NULL, ylab = NULL, xlim, ylim, xaxs = par("xaxs"),
               yaxs = par("yaxs"), ...)
```

其中x和y是两个数值向量，或者如果不提供y的话，可以提供一个两列的矩阵/数据框等给x；nbin为横纵坐标方向上划分网格的数目，可以是长度为1或2的整数向量；bandwidth为计算核密度估计时使用的带宽；colramp为生成颜色向量的函数，默认生成从白色到蓝色渐变的颜色向量；nrpoints为需要画出来的点的数目，因为平滑散点图的目的不是画散点，而是画颜色块，但有时候图形中某些地方的密度估计非常低，因此对应颜色也非常浅，导致读者难以察觉那些地方还有数据点的存在，此时不妨直接将这些“离群点”直接画出来，注意画点时按密度值从小到大的顺序画，一直画到第nrpoints个点；其它参数几乎都是画图的一般参数，此处不再介绍。

图5.28是**MSG**包中**BinormCircle**数据的平滑散点图，其原始散点图参见图5.8左图。由于使用了核密度估计，这批数据中藏着的圆圈很容易就能被发现，因为它们的密度值都很大，导致这一圈上颜色也就相应较深。平滑散点图看起来和图5.8右图比较相似，但前者蕴含了更多的数理统计背景。不过，我们也不必一味追求数学理论，“透明色可叠加”这一点性质体现出的原理又何尝不是一种“密度估计”呢？

5.22 棘状图

棘状图（Spine Plot/Spinogram³）可以看作是马赛克图（5.16小节）的特例，也可以看作是堆砌条形图（5.4小节）的推广。棘状图的外观看起来像是高低不齐的荆棘丛，因此而得名。

棘状图的原理和条件密度图非常类似，都展示了在给定某个自变量的情况下因变量的概率分布，但是棘状图首先对连续型的自变量进行了

³对于连续的自变量则称为Spine Plot，离散自变量则为Spinogram，这个单词是仿照Histogram而造。

```

1 > # 原始数据: 是否失效以及相应温度
2 > fail = factor(c(2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2,
3 +     1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1), levels = 1:2,
4 +     labels = c("no", "yes"))
5 > temperature = c(53, 57, 58, 63, 66, 67, 67, 67, 68,
6 +     69, 70, 70, 70, 72, 73, 75, 75, 76, 76, 78,
7 +     79, 81)
8 > par(mar = c(3.5, 4, 0.5, 2))
9 > t(x <- spineplot(fail ~ temperature, col = c("lightblue",
10 +     "red")))

```

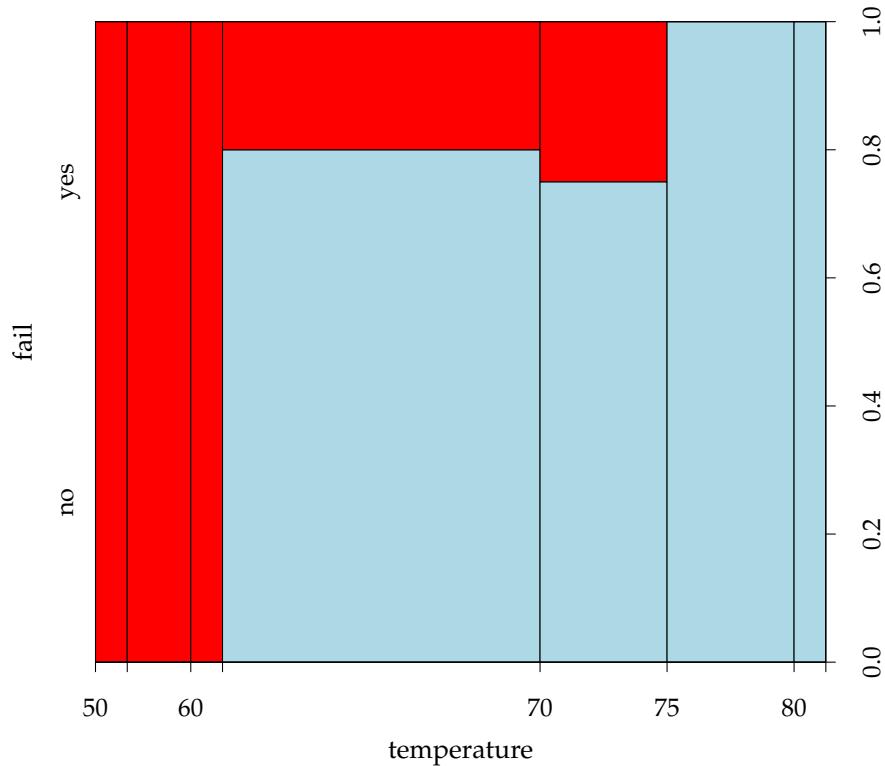


图 5.29: 航天飞机O型环在不同温度下失效的棘状图: 随着温度升高, O型环失效的概率下降; 从横轴方向上看, 观测的温度数据的分布大多集中在 $(60, 75]$ 区间内。

离散化处理，然后在离散的区间内计算因变量的条件分布。除此之外，棘状图还兼顾了自变量的分布，在横轴方向上以不同宽度的矩形表示自变量的分布密度。因此，从纵轴方向上看，棘状图用堆砌的矩形块表示因变量的分布密度，这使得它看起来像堆砌的条形图，而从横轴方向上看，棘状图用不同的矩形宽度表示自变量的密度分布，这使得它又像是马赛克图（马赛克图中矩形的长宽和频数成比例）。尤其是自变量为分类变量时，棘状图与马赛克图几乎无异。从概念和定义上来讲，棘状图实际上是由 $P(Y|X)$ 对 $P(X)$ 所作的图。

R中棘状图的函数为`spineplot()`，其用法如下：

```

1 > usage(spineplot, "default", 0.75)
spineplot(x, y = NULL, breaks = NULL,
          tol.ylab = 0.05, off = NULL, ylevels = NULL,
          col = NULL, main = "", xlab = NULL, ylab = NULL,
          xaxlabels = NULL, yaxlabels = NULL, xlim = NULL,
          ylim = c(0, 1), axes = TRUE, ...)
1 > usage(spineplot, "formula")
spineplot(formula, data = list(), breaks = NULL,
          tol.ylab = 0.05, off = NULL, ylevels = NULL, col = NULL,
          main = "", xlab = NULL, ylab = NULL, xaxlabels = NULL,
          yaxlabels = NULL, xlim = NULL, ylim = c(0, 1),
          axes = TRUE, ..., subset = NULL)

```

棘状图函数是泛型函数，可以直接接受数据或者公式作为参数。`x`可以是一个分类或数值向量（自变量），也可以直接输入一个列联表，后一种情况下则不再需要再输入`y`参数；`y`为因变量，是一个分类变量；`breaks`在自变量是连续变量时给定自变量的分段区间或者分段方法，这个参数最终被传递给直方图函数`hist()`（回顾5.1小节）以便计算自变量的密度；`off`参数指定矩形竖条之间的间距，对于连续自变量来说，`off`默认为0，对于离散自变量，`off`默认为2；`col`用来设定不同类别的`y`的颜色。

图5.29重新使用航天飞机O型环失效数据作了棘状图。我们可以看到，自变量温度被分为了7个长度为5的区间，每个区间内失效的比率都用堆砌的矩形表达。注意棘状图的横坐标轴比较特殊，它对自变量的值来说并不一定是均匀的，因为图中矩形的宽需要表达自变量的密度，所以横坐标的作用仅仅是作为密度值大小的参照物。棘状图会返回一个汇总表，表中给出了自变量的分段情况以及相应的频数。读者可以结合频数表理解棘状图。

5.23 星状图、蛛网图、雷达图

星状图（Star Plot）、蛛网图（Spider Plot）和雷达图（Radar Plot）本质上是一类图形，它们都用线段离中心的长度来表示变量值的大小，这三种图形名称的区别在于星状图用来展示很多个多变量个体，各个个体的图形相互独立，从而整幅图形看起来就像很多星星，而蛛网图和雷达图将多个多变量个体放在同一张图形上，看起来就像是蛛网或雷达的形状，这样重叠的图形就称为蛛网图或者雷达图。简单说来，就是星状图有若干个中心，而蛛网图和雷达图只有一个中心。

R中星状图的函数为`stars()`，其用法如下：

```
1 > usage(stars, w = 0.75)

stars(x, full = TRUE, scale = TRUE, radius = TRUE,
      labels = dimnames(x)[[1L]], locations = NULL,
      nrow = NULL, ncol = NULL, len = 1, key.loc = NULL,
      key.labels = dimnames(x)[[2L]], key.xpd = TRUE,
      xlim = NULL, ylim = NULL, flip.labels = NULL,
      draw.segments = FALSE, col.segments = 1L:n.seg,
      col.stars = NA, axes = FALSE, frame.plot = axes,
      main = NULL, sub = NULL, xlab = "", ylab = "",
      cex = 0.8, lwd = 0.25, lty = par("lty"), xpd = FALSE,
      mar = pmin(par("mar"), 1.1 + c(2 * axes + (xlab !=
      ""), 2 * axes + (ylab != ""), 1, 0)), add = FALSE,
      plot = TRUE, ...)
```

参数`x`为一个多维数据矩阵或数据框，每一行数据将生成一个星形；`full`为逻辑值，决定了是否使用整圆（或半圆）；`scale`决定是否将数据标准化到区间[0, 1]内；`radius`决定是否画出半径；`labels`为每个个体的名称，默认为数据的行名；`locations`以一个两列的矩形给出每个星形的放置位置，默认放在一个规则的矩形网格上，若提供给该参数一个长度为2的向量，那么所有的星形都将被放在该坐标上，从而形成蛛网图或雷达图；`nrow`和`ncol`分别给定网格的行数和列数以便摆放星形，默认`nrow`等于`ncol`；`len`为半径和线段的缩放倍数；`key.loc`提供比例尺的坐标位置；`key.labels`为比例尺的标签，默认为变量名称；`key.xpd`设定比例尺的作图范围，回顾3.1小节（`par('xpd')`）；`flip.labels`设定每个星形底部的名称是否互相上下错位，以免名称太长导致文本之间互相重叠；`draw.segments`设定是否作线段图，即：每个变量以一个扇形表示；`col.segments`设定每个扇形区

```

1 > # 预设调色板, stars()默认用整数来表示颜色
2 > palette(rainbow(12, s = 0.6, v = 0.75))
3 > stars(mtcars[, 1:7], len = 0.8, key.loc = c(14, 1.5),
4 +   ncol = 7, main = "", draw.segments = TRUE)
5 > # 恢复默认调色板
6 > palette("default")

```

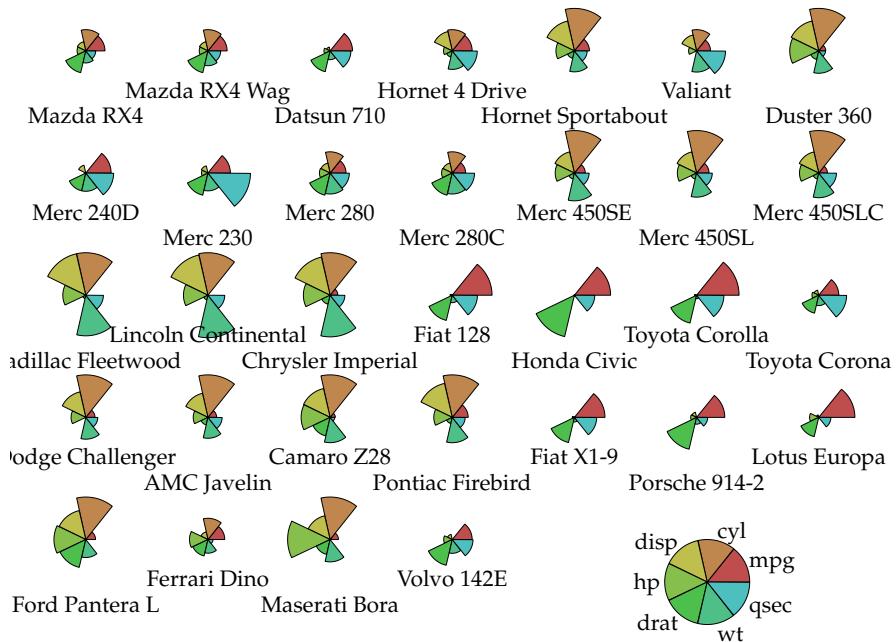


图 5.30: *Motor Trend*杂志1974年汽车数据的星状图

域的颜色（当`draw.segments`为`FALSE`时无效）；`col.stars`设定每个星形的颜色（当`draw.segments`为`TRUE`时无效）；`axes`决定是否画坐标轴；`frame.plot`决定是否画整个图形的边框；`add`决定是否将图形添加到当前图形上。

图5.30为数据`mtcars`的星状图，一共使用了7个变量：`mpg`为每加仑汽油可行驶英里数，`cyl`为汽缸数，`disp`为汽缸排量，`hp`为马力，`drat`为背齿轮比，`wt`为车重，`qsec`为行驶1/4英里的时间。从图中可以看到各种品牌和型号的汽车在这7方面的指标和性能表现。以星状图展示数据可以让我们很快找到一些有突出特征的个体，从而省去了在大批数据中逐个寻找、排序的过程。

```

1 > # mtcars数据前7列的前6行
2 > head(mtcars[, 1:7])

          mpg cyl disp hp drat    wt  qsec
Mazda RX4     21.0   6 160 110 3.90 2.620 16.46
Mazda RX4 Wag 21.0   6 160 110 3.90 2.875 17.02
Datsun 710    22.8   4 108  93 3.85 2.320 18.61
Hornet 4 Drive 21.4   6 258 110 3.08 3.215 19.44
Hornet Sportabout 18.7   8 360 175 3.15 3.440 17.02
Valiant       18.1   6 225 105 2.76 3.460 20.22

```

5.24 带状图

带状图（Strip Chart），又叫一维散点图（1-D Scatter Plot），是针对一维数据的“散点图”，它本质上是数据与固定值（固定x或固定y）之间的散点图，这样形成的图形外观是带状的，因此称之为带状图。

R中带状图的函数为`stripchart()`，其用法如下：

```

1 > usage(stripchart, "default")

stripchart(x, method = "overplot", jitter = 0.1,
           offset = 1/3, vertical = FALSE, group.names, add = FALSE,
           at = NULL, xlim = NULL, ylim = NULL, ylab = NULL,
           xlab = NULL, dlab = "", glab = "", log = "", pch = 0,
           col = par("fg"), cex = par("cex"), axes = TRUE,
           frame.plot = axes, ...)

1 > usage(stripchart, "formula", 0.9)

stripchart(x, data = NULL, dlab = NULL, ..., subset,
           na.action = NULL)

```

```

1 > layout(matrix(1:2, 2), height = c(1.5, 1))
2 > boxplot(count ~ spray, data = InsectSprays, horizontal = TRUE,
3 +     border = "red", col = "lightgreen", at = 1:6 -
4 +     0.3)
5 > stripchart(count ~ spray, data = InsectSprays, method = "stack",
6 +     add = TRUE)
7 > stripchart(count ~ spray, data = InsectSprays, method = "jitter")

```

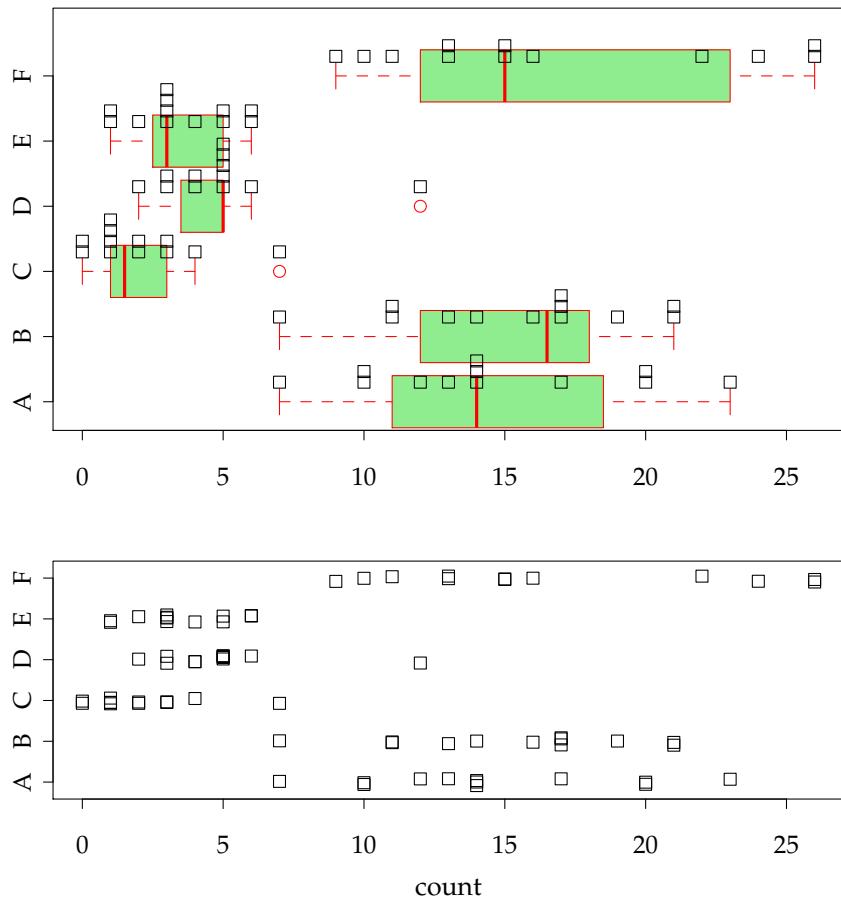


图 5.31: 各种杀虫剂下昆虫数目的带状图：堆砌和随机打乱的带状图。

带状图函数为泛型函数，可以直接接受数据参数或者公式参数。`x`为数据，一般为一个向量；`method`指定作图方法，取值’overplot’意思是将所有的数据点画在一条直线上，不管它们是否有重叠，’jitter’意思是将直线上的数据随机打乱，以免数据重叠导致我们不知道在某个位置究竟有多少个点，’stack’意思是将重叠的数据堆砌起来，某个位置重叠的数据越多，则堆砌越高；`jitter`给定打乱的程度，参见`jitter()`函数（B.4小节有讲解）；`vertical`设定带状图的方向（横向或纵向）；`group.names`为每一组数据的名称标签；`add`决定是否将带状图添加到现有图形上；`at`给定每条带子的位置。

带状图作为描述一维数据分布的工具也有其独特的优势，它的作图方法可以反映出原始数据的疏密，若原始数据有重叠，它也有相应的办法处理，最终使所有的数据点都能够被展示出来。图5.31重新使用了杀虫剂数据`InsectSprays`，上图展示了堆砌的带状图，并且在图中同时放置了箱线图作为对比，如果只是观察箱线图，我们无从得知数据在若干位置有重复，只知道数据四分位点的位置，而带状图则可以告诉我们在哪些位置分别有多少数据点；下图为随机打乱的带状图，作图方法只是将y方向上的固定数值添加了随机数，使原本重叠在一起的数据重新拥有不同的纵坐标，从而将重叠的数据分开来。

回顾前面5.20小节中我们曾经用坐标轴须在坐标轴上标记出原始数据，这与带状图在一条直线上用点表达原始数据的想法有异曲同工之妙，然而坐标轴须没有堆砌和随机打乱数据的功能⁴，所以有时候使用不妨视情况向图中添加带状图作为变量分布的辅助性描述。

5.25 向日葵散点图

向日葵散点图（Sunflower Scatter Plot）是用来克服散点图中数据点重叠问题的特殊散点图工具。它采用的办法是在有重叠的地方用一朵“向日葵花”的花瓣数目来表示重叠数据的个数，这样我们就很容易看出来散点图中哪些地方的数据有重叠，而且能知道重叠的具体数目。向日葵散点图在数据特别密集或者数据类型为分类数据时很有用，因为这两种情况下都容易产生重复的数据点，尤其是后一种情况下，数据几乎必然有重复（除非列联表单元格频数为1）。

⁴后者可以通过函数`jitter()`实现

```
1 > sunflowerplot(iris[, 3:4], col = "gold", seg.col = "gold")
```

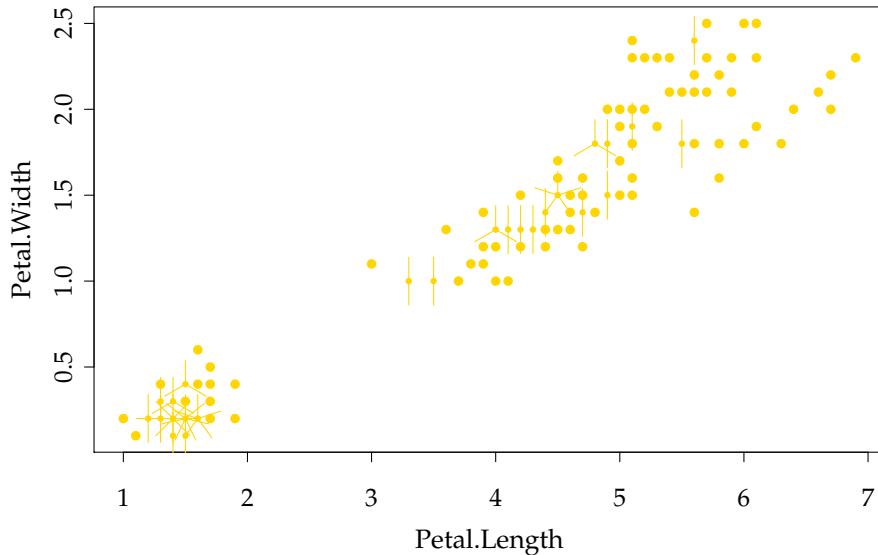


图 5.32: 鸢尾花花瓣长和宽的向日葵散点图

R中向日葵散点图的函数为sunflowerplot(), 其用法如下:

```
1 > usage(sunflowerplot)

sunflowerplot(x, y = NULL, number, log = "",
  digits = 6, xlab = NULL, ylab = NULL, xlim = NULL,
  ylim = NULL, add = FALSE, rotate = FALSE, pch = 16,
  cex = 0.8, cex.fact = 1.5, col = par("col"), bg = NA,
  size = 1/8, seg.col = 2, seg.lwd = 1.5, ...)
```

x和y分别为散点图的两个变量; number为人工给定的数据频数, 即图中的花瓣数目, 若不指定这个参数的话R会自动从x和y计算; digits给定数值的有效数字位数, 在计算重复数据之前原始数据会按照digits四舍五入; add决定是否将向日葵散点图添加到当前图形上; rotate决定是否随机旋转向日葵的角度; pch给定散点图的点的类型; cex给定散点图的点的缩放倍数; cex.fact给定向日葵中心点的缩小倍数, 真正的缩放倍数为cex/cex.fact; col为散点的颜色, bg为点的背景色; size为向日葵花瓣的长度, 单位为英寸; seg.col为花瓣的颜色; seg.lwd为花瓣的宽度。

图5.32为鸢尾花花瓣长和宽的向日葵散点图，注意左下方和中部都有一些重复数据，这幅图的颜色用的是金色'gold'，目的是为了使向日葵散点图看起来与其名称相符，不过似乎使用鸢尾花的颜色更符合数据的背景，读者不妨试试看。

5.26 符号图

符号图是用各种符号展示高维数据的图示工具，它的主要思想是将高维数值体现在图形中符号的特征上。因为受到平面的限制，我们对于高维数据的展示方法总是很有限，仅仅是对于二维数据的展示最为方便，对更高维度的如三维、四维甚至五维的数据相对缺乏好的工具。由于符号的存在，使得我们可以将高于二维的数据“寄托”在符号的各种特征上，如：以矩形为散点图的基本符号，那么我们可以用其长宽分别代表两个变量，这样一幅图形中至少可以放置四个变量；类似地，我们可以以圆圈、正方形、多边形、箱线图、温度计等符号为散点图中的“点”，于是散点图就可以被扩展为高维数据的展示工具。

R中的符号图函数为*symbols()*，它提供了六种基本符号：圆、正方形、长方形、星形、温度计和箱线图，分别由相应的参数指定；*symbols()*的用法如下：

```
1 > usage(symbols)

symbols(x, y = NULL, circles, squares, rectangles,
        stars, thermometers, boxplots, inches = TRUE, add = FALSE,
        fg = par("col"), bg = NA, xlab = NULL, ylab = NULL,
        main = NULL, xlim = NULL, ylim = NULL, ...)
```

如前所述，符号图的基础是散点图，因此首先要给出两个参数x和y以便作散点图，然后在散点的位置上画出符号；接下来的六个参数分别指定符号的形状：

circles 圆：一个数值向量，给定圆的半径（实际上是与圆的半径成比例，下同）

squares 正方形：一个数值向量，给定正方形的边长

rectangles 长方形：一个矩阵，列数为2，这两列分别给定长方形的宽和高

```

1 > par(mar = c(6, 1, 0.1, 0.1), las = 2)
2 > plot(1:6, rep(1, 6), type = "n", axes = FALSE, ann = FALSE,
3 +       xlim = c(0.8, 7.2), ylim = c(0, 5), panel.first = grid())
4 > x = seq(1, 2, length = 4)
5 > symbols(x, 1:4, circles = runif(4, 0.1, 0.6), add = TRUE,
6 +       bg = heat.colors(4), inches = FALSE)
7 > symbols(x + 1, 1:4, squares = runif(4, 0.1, 0.6),
8 +       add = TRUE, bg = terrain.colors(4), inches = FALSE)
9 > symbols(x + 2, 1:4, rect = matrix(runif(8, 0.1, 0.6),
10 +      4), add = TRUE, bg = rainbow(4), inches = FALSE)
11 > symbols(x + 3, 1:4, stars = matrix(runif(20, 0.1,
12 +      0.6), 4), add = TRUE, bg = topo.colors(4), inches = FALSE)
13 > symbols(x + 4, 1:4, therm = matrix(runif(12, 0.1,
14 +      0.7), 4), add = TRUE, inches = FALSE)
15 > symbols(x + 5, 1:4, boxplot = matrix(runif(20, 0.1,
16 +      0.7), 4), add = TRUE, inches = FALSE)
17 > axis(1, 1:6, c("circles", "squares", "rectangles",
18 +      "stars", "thermometers", "boxplots"), cex.axis = 0.85)

```

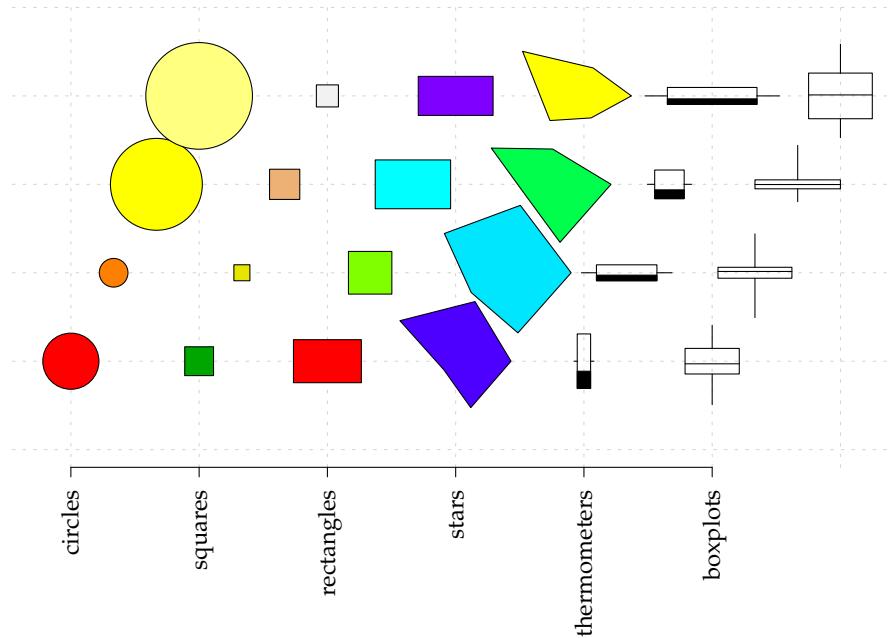


图 5.33: 符号图提供的六种基本符号

stars 星形：一个矩阵，列数 ≥ 3 ，类似雷达图，给定从星星中心向每个方向的射线的长度（严格说是线段），最终这些射线的端点会连接起来形成一个星形，但射线本身不会被画出来；缺失值将被视作0（星形在符号图中并不直观，推荐直接使用星状图，见5.30小节）

thermometers 温度计：一个矩阵，列数为3或4，前两列分别给定温度计的宽和高；若矩阵为三列，那么第三列为温度计内的“温度”高度，注意这一列的值应该小于1，否则温度的填充会超过温度计的范围；若矩阵为四列，那么温度将按照第三列与第四列的比率进行填充，同样，这两列的比率需要小于1

boxplots 箱线图：一个矩阵，列数为5，前两列分别给定箱子的宽和高，第三、四列分别给定两条线（下线和上线）的长度，第五列与温度计类似，给定箱线图内的中位数标记线在箱子内部的高度比例，因此这一列数据也需要在[0, 1]范围内；这里只是借用了箱线图的称谓，符号图中的箱线与真正的箱线图之间没有关系

不难看出，这六种符号图能展示的数据维度分别为3、3、4、 ≥ 5 、5或6、7；参数inches为逻辑值，控制着符号的大小，若为TRUE（默认），那么图中所有符号的最长长度（边长或半径等等）将被设置为1英寸（约2.54厘米），其它长度按比例缩放，若该参数取一个正数，那么类似地，所有符号的最长长度的英寸值为该正数，因此TRUE和1的效果是相同的，如果inches为FALSE那么符号的长度单位取相应坐标轴的单位，例如符号中的1长度即为坐标轴上的1单位；add为逻辑值，设定是否将符号图添加到现有图形上；fg为符号的前景色；bg为符号的背景色或填充色；其它参数用来添加标题、设定坐标轴范围等。

图5.33给出了六种基本符号的形状，注意观察各种符号是如何利用自己的特征将高维数据表达出来的。下面我们通过实际数据来看符号图在展示数据时的效果，以2005年中国31省市自治区的人口特征数据为例，我们选取了人口自然增长率、年底人口总数、城镇人口比重、人口预期寿命和高学历人数五个变量作为国民素质的刻画指标，数据见MSG包中的ChinaPop.R文件（可用load()函数载入，见以下代码），这些数据都可以从《中国统计年鉴》中获得。

```

1 > source(system.file("extdata", "ChinaPop.R", package = "MSG"))
2 > head(ChinaPop)

```

	增长率	总人口	城镇人口比重	预期寿命
北京	1.09	1538	0.8362	76.10
天津	1.43	1043	0.7511	74.91
河北	6.09	6851	0.3769	72.54
山西	6.02	3355	0.4211	71.65
内蒙古	4.62	2386	0.4720	69.87
辽宁	0.97	4221	0.5870	73.34
高学历人数				
北京		48001		
天津		18601		
河北		40036		
山西		23197		
内蒙古		23660		
辽宁		44404		

图5.34(谢益辉, 2008)融合了多种图形和图形元素, 它的基础是一幅等高图(回顾5.8小节和图5.12), 利用人口预期寿命和高学历人数两个变量计算二维密度, 画出等高线, 便完成了底图的制作; 然后我们通过人口预期寿命和高学历人数两个变量的数值往图中添加温度计符号, 温度计宽代表增长率, 高代表总人口数, 温度代表城镇人口比重; 然后我们用text()函数将各省市的文本标签添加到图中。经过这些图形元素的表达, 全国31省市区的五项人口特征便一目了然, 例如通过温度计的高度可以观察出三个人口大省广东、山东、河南(相应的人口总量小的地区如西藏、青海、宁夏等也容易看出), 由宽度可以看出西藏、青海、宁夏、新疆等省市自治区的人口自然增长率非常高(而北京、上海、天津等直辖市的增长率则很低), 从温度指示的情况来看, 北京、上海和天津三大直辖市的城镇人口比例要远高于其它地区; 从整幅散点图来看, 人口平均预期寿命与高学历者人数呈比较明显的正相关关系。箱线图和坐标轴须分别刻画了人口平均预期寿命与高学历者人数各自的分布特征。这样, 我们就完成了在平面上描述五维变量的任务。从这幅图我们可以看出掌握基本图形元素使用的用处和重要性, 试问哪种统计软件能够提供现成的模块或函数来完成类似的任务?

5.27 饼图

饼图是目前应用非常广泛的统计图形, 然而, 根据统计学家(主要是Cleveland和McGill)和一些心理学家的调查结果(Cleveland, 1985), 这

```

1 > x = ChinaPop
2 > x[, 1:2] = apply(x[, 1:2], 2, function(z) 20 * (z -
3 + min(z))/(max(z) - min(z)) + 5)
4 > symbols(x[, 4], x[, 5], thermometers = x[, 1:3],
5 + fg = "gray40", inches = 0.5, xlab = "人均预期寿命",
6 + ylab = "高学历者人数")
7 > est = bkde2D(x[, 4:5], apply(x[, 4:5], 2, dpik))
8 > contour(est$x1, est$x2, est$fhat, add = TRUE, lty = "12")
9 > for (i in 1:nrow(x)) text(x[i, 4], x[i, 5], rownames(x)[i],
10 + cex = 0.75, adj = attr(x, "adj")[i, ])
11 > rug(x[, 4], 0.02, side = 3, col = "gray40")
12 > rug(x[, 5], 0.02, side = 4, col = "gray40")
13 > boxplot(x[, 4], horizontal = TRUE, pars = list(boxwex = 7000,
14 + staplewex = 0.8, outwex = 0.8), at = -6000, add = TRUE,
15 + notch = TRUE, col = "skyblue", xaxt = "n")
16 > boxplot(x[, 5], at = 63, pars = list(boxwex = 1.4,
17 + staplewex = 0.8, outwex = 0.8), add = TRUE, notch = TRUE,
18 + col = "skyblue", yaxt = "n")
19 > text(67, 60000, "2005", cex = 3.5, col = "gray")

```

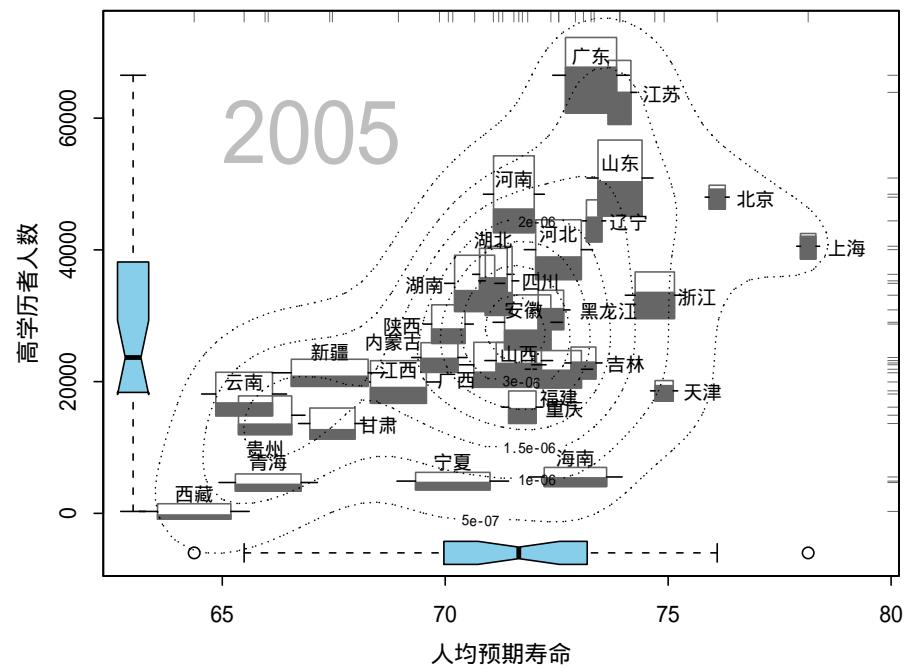


图 5.34: 2005年中国31地区五大国民素质特征分布温度计图

```

1 > # 拆分作图区域
2 > layout(matrix(c(1, 2, 1, 3), 2))
3 > pie.sales = c(0.12, 0.3, 0.26, 0.16, 0.04, 0.12)
4 > names(pie.sales) = c("Blueberry", "Cherry", "Apple",
5 +     "Boston Cream", "Other", "Vanilla Cream")
6 > pie.col = c("purple", "violetred1", "green3", "cornsilk",
7 +     "cyan", "white")
8 > # 饼图、点图和条形图
9 > pie(pie.sales, col = pie.col)
10 > dotchart(pie.sales, xlim = c(0, 0.3))
11 > barplot(pie.sales, col = pie.col, horiz = TRUE, names.arg = "",
12 +     space = 0.5)

```

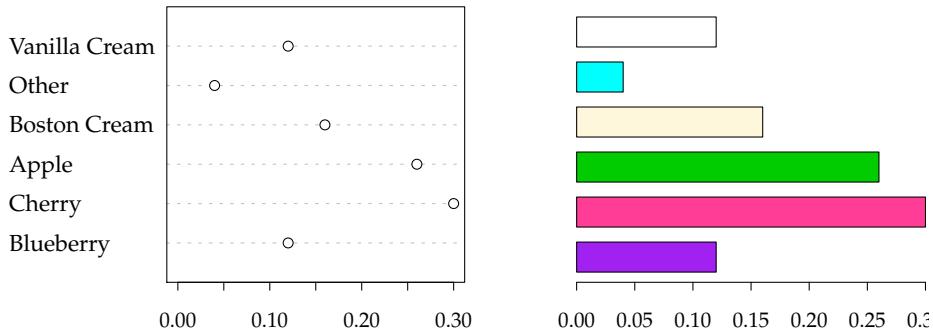


图 5.35: 各类馅饼销售数据的饼图: 对比上面的饼图和下面的点图以及条形图, 人眼对角度或比例的感知能力是否弱于对长度的感知能力?

种以比例展示数据的统计图形实际上是很糟糕的可视化方式，因此，R关于饼图的帮助文件中清楚地说明了并不推荐使用饼图，而是使用条形图（5.4小节）或点图（5.11小节）作为替代。读者若有兴趣可以到COS论坛查看这则关于饼图的幽默：<http://cos.name/cn/topic/101157>。

饼图的原理很简单，每一个扇形的角度与相应数据的数值大小成比例，关于饼图的知识此处不再赘述。R提供了函数`pie()`制作饼图，用法如下：

```
1 > usage(pie)
  pie(x, labels = names(x), edges = 200, radius = 0.8,
       clockwise = FALSE, init.angle = if (clockwise) 90 else 0,
       density = NULL, angle = 45, col = NULL, border = NULL,
       lty = NULL, main = NULL, ...)
```

参数`x`为一个数值向量，`labels`为标签，其它参数基本上都是为多边形准备的，因为扇形实际上是多边形所作，例如`edges`可以设定圆弧的光滑程度（多边形的边越多则越光滑），`density`、`angle`等参数参见多边形的章节（4.4小节）。

图5.35同时给出了饼图、点图和条形图分别对一个不同牌子馅饼的销售数据的展示，请读者对比并思考饼图对于展示数据的弱势。

至此，我们已经全部介绍完`graphics`包中的统计图形函数，读者一般不用安装别的附加包就可以完成以上的图形制作。下面我们开始选择性介绍其它基础包和附加包中的图形函数和图形种类。

5.28 热图

`heatmap()`

5.29 交互效应图

`interaction.plot()`

5.30 QQ图

关于统计分布的检验有很多种，例如KS检验、卡方检验等，从图形的角度来说，我们也可以用QQ图（Quantile-Quantile Plots）来检查数据是

```

1 > par(mfrow = c(1, 2))
2 > x = scale(geyser$waiting)
3 > qqnorm(x, cex = 0.7, asp = 1, main = "")
4 > abline(0, 1)
5 > plot(density(x), main = "", xlim = range(x))
6 > curve(dnorm, from = -3, to = 3, lty = 2, add = TRUE)

```

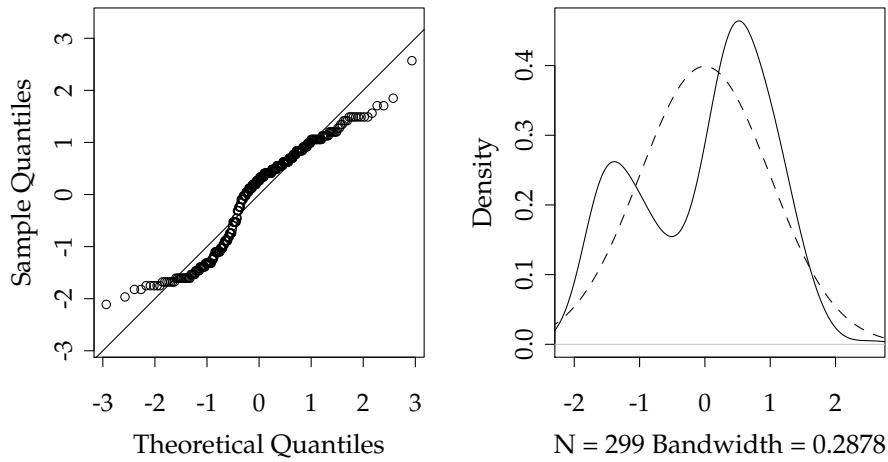


图 5.36: 喷泉间隔时间的正态分布QQ图（左图）及数据密度曲线和实际正态分布密度曲线（右图）。

否服从某种分布。QQ图的原理并不复杂：如果一批数据 x_1, x_2, \dots, x_n 服从某种理论分布，那么将排序后的数据 $x_{(1)}, x_{(2)}, \dots, x_{(n)}$ 和理论分布的分位数 $q_{1/n}, q_{2/n}, \dots, q_{n/n}$ 去画散点图，得到的 n 个点应该大致排列在对角线上，因为这两批数字应该大致相等。从另一个角度来看，检验一批数据是否服从某种理论分布，也就是看其经验分布和理论分布是否一致，而排序后的数据 $x_{(1)}, x_{(2)}, \dots, x_{(n)}$ 可以看作是经验分布的 $1/n, 2/n, \dots, n/n$ 分位数，若这些分位数和理论分位数一致，也就说明了经验分布和理论分布相似。为了说明这一点，我们可以看看数值模拟的结果：

```

1 > # 从 $N(0, 1)$ 中生成1000个随机数的分位数
2 > quantile(rnorm(1000), probs = seq(0.1, 0.9, 0.2))

      10%        30%        50%        70%        90%
-1.267814 -0.495970 -0.006973  0.488255  1.321471

```

```

1 > # 真实的分位数
2 > qnorm(seq(0.1, 0.9, 0.2))
[1] -1.2816 -0.5244  0.0000  0.5244  1.2816

```

以上数据的5个分位数和理论分位数都比较接近，读者可以模拟其它分布，例如从卡方分布中生成随机数，看其分位数是多少，与正态分布分位数差异如何。

R中QQ图的函数为`qqplot()`，由于正态分布是我们经常检验的分布，R也直接提供了一个画正态分布QQ图的函数`qnorm()`，这两个函数都在基础包`stats`包中，它们的用法如下：

```

1 > usage(qqplot)
qqplot(x, y, plot.it = TRUE, xlab = deparse(substitute(x)),
       ylab = deparse(substitute(y)), ...)
1 > usage(qnorm, "default")
qnorm(y, ylim, main = "Normal Q-Q Plot",
      xlab = "Theoretical Quantiles", ylab = "Sample Quantiles",
      plot.it = TRUE, datax = FALSE, ...)
1 > usage(qqline)
qqline(y, datax = FALSE, ...)

```

由于`qqplot()`检验的是两批数据的分布是否相同，所以它需要两个数据参数`x`和`y`，`qnorm()`只需要一个数据参数`x`，其它设置标签和标题等元素的图形参数此处不再赘述。

图5.36左图是喷泉间隔时间数据的正态分布QQ图（5.1小节的直方图用到过），注意其中的数据经过了标准化，使之均值为0，方差为1。可以看出，数据点并不呈直线分布，这说明（标准化后的）数据的分布和标准正态分布有所差异，那么具体是何种差异呢？图的左边有一部分点偏离在直线上方，说明实际分位数大于理论分位数，从密度曲线的角度来说，也就是实际数据的分布曲线更偏右一些，理论分布曲线左边的尾巴向左伸得更远，而图的右边又有一些点在直线下方，说明此处实际分布曲线偏左，即实际分位数偏小。右图画出了数据的核密度估计曲线（实线）和真正的标准正态分布密度曲线（虚线），读者可以将二者的对比结合左图来理解QQ图中数据点偏离直线的方向与分布曲线的偏向关系。

```

1 > library(survival)
2 > leukemia.surv = survfit(Surv(time, status) ~ x, data = aml)
3 > plot(leukemia.surv, lty = 1:2, xlab = "time")
4 > legend("topright", c("Maintenance", "No Maintenance"),
5 +       lty = 1:2, bty = "n")

```

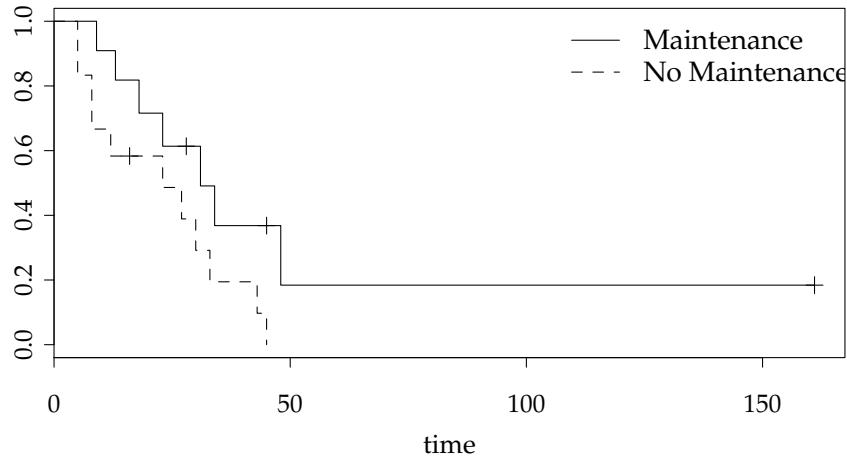


图 5.37: 急性髓细胞白血病病人生存函数图

注意图5.36的QQ图中我们用了纵横比参数`asp = 1`，这样使得图中横纵坐标的单位长度表示的数值大小相同，为我们比较点的横纵坐标值大小提供了更好的视觉辅助。

QQ图的用途不仅在检查数据是否服从某种特定理论分布，它也可以推广到检查数据是否来自某个位置参数分布族。例如，若数据来自正态分布⁵ $N(5, 1)$ ，我们拿它和标准正态分布 $N(0, 1)$ 画QQ图的话，数据点仍然会大致排列在直线上，此时直线的斜率仍然是1，但截距就不是0了，而是5。

5.31 生存函数图

在很多医学研究中，我们主要关心的变量是病人的某种事件发生的时间，例如死亡、疾病复发等。事实上，以“生存时间”为研究对象的领域并不仅限于医学，例如在金融领域，我们可能需要了解信用卡持有者的

⁵正态分布是位置参数分布族中的一种分布：若 $X \sim N(\mu, \sigma^2)$ ，那么 $X + \delta \sim N(\mu + \delta, \sigma^2)$ 仍然是正态分布。均值 μ 是位置参数。

信用风险发生时间。这类数据一般统称为生存数据 (survival data)，而生存数据通常有一个明显特征就是删失。关于生存分析的详细理论请参考Therneau and Grambsch (2000)等。

本节要介绍的图形对象主要是生存函数 (Survival Function)，其定义是个体存活超过时间 t 的概率：

$$S(t) = P(T > t); t \geq 0$$

对于存在删失的生存数据 (t_i, δ_i) , $i = 1, \dots, n$ (其中 t_i 为记录时间, $\delta_i = 0$ 表示存在删失, 1表示个体没有删失), 生存函数的Kaplan-Meier估计(Kaplan and Meier, 1958)为:

$$\hat{S}(t) = \begin{cases} \prod_{i: t_{(i)} \leq t} \left(\frac{n-i}{n-i+1}\right)^{\delta_{(i)}}, & \text{对 } t \leq t_{(n)}; \\ \begin{cases} 0 & \text{如果 } \delta_{(n)} = 1, \\ \text{未定义} & \text{如果 } \delta_{(n)} = 0, \end{cases} & \text{对 } t > t_{(n)}. \end{cases}$$

survival包(Therneau, 2009)提供了生存函数的计算和估计方法。具体函数为`survfit()`, 它返回一个`survfit`类的对象, 而**survival**包扩展了泛型函数`plot()`, 使其拥有子函数`plot.survfit()`, 因此在估计完生存函数之后, 我们可以直接调用`plot()`生成生存函数图。函数`plot.survfit()`的用法如下:

```

1 > library(survival)
2 > usage(plot, "survfit")

plot(x, conf.int, mark.time = TRUE, mark = 3,
      col = 1, lty = 1, lwd = 1, cex = 1, log = FALSE,
      xscale = 1, yscale = 1, firstx = 0, firsty = 1,
      xmax, ymin = 0, fun, xlab = "", ylab = "", xaxs = "S",
      ...)
```

其中`x`为一个`survfit`类的对象, 通常由`survfit()`返回; `conf.int`决定是否在生存曲线上下作置信区间曲线 (当图中只有一条生存曲线时默认会作置信区间); `mark.time`决定是否用短竖线标记出删失的时刻, 或者直接指定一个时间向量以标记删失时刻; `mark`给出删失标记的样式, 即: 标记点的类型 (参见图4.3)。

图5.37展示了急性髓细胞白血病 (Acute Myelogenous Leukemia) 数据`aml`的生存函数图, 该数据有一个分组变量`x`表示病人是否接受了化疗, 从图中可以看出, 接受化疗的病人生存函数的下降速度比没接受化疗的病人要慢, 表明化疗还是有一定作用的。进一步我们可以用对数秩检验知道这两组病人的生存时间在10%的显著水平下有显著差异:

```

1 > survdiff(Surv(time, status) ~ x, data = aml)

Call:
survdiff(formula = Surv(time, status) ~ x, data = aml)

      N Observed Expected (O-E)^2/E (O-E)^2/V
x=Maintained   11        7    10.69     1.27     3.40
x=Nonmaintained 12       11     7.31     1.86     3.40

Chisq= 3.4 on 1 degrees of freedom, p= 0.0653

```

根据3.2小节和图3.4的讲解，读者不难发现，生存函数图实际上就是在生存函数估计值的基础上使用阶梯状参数`type = 's'`制成的线图。

5.32 分类与回归树图

分类与回归树（Classification and Regression Tree, CART）是一种递归分割（Recursive Partition）技术，它的目的是寻找自变量的某种分割，使得样本分割之后因变量各组之间的差异最大。这种分割会一直递归进行下去，直到满足停止条件。详细理论请参见Breiman *et al.* (1984)。

rpart包(Therneau and Atkinson, 2010)提供了分类与回归树的计算拟合函数`rpart()`，该函数包同时也扩充了泛型函数`plot()`，凡是**rpart**类型的对象在作图时都会自动调用`plot.rpart()`生成树图。`plot.rpart()`的用法如下：

```

1 > library(rpart)
2 > usage(plot, "rpart")

plot(x, uniform = FALSE, branch = 1, compress = FALSE,
      nspace, margin = 0, minbranch = 0.3, ...)

```

`x`是一个**rpart**类型的对象，一般由`rpart()`函数拟合产生；`uniform`决定是否在从上至下的枝节点之间使用相等的纵向距离以避免树枝在某些局部区域靠得太近使图形难以辨认，默认情况下每两个枝节点之间的距离与拟合误差成比例；`branch`设定树枝的形状，0为“V”字型，1为垂直的形状，该参数可以取[0, 1]之间的数值以使得数值形状更像“V”或更垂直；`compress`设定是否在横向压缩树枝的间距使得图形更紧凑。

我们利用**rpart**包中的一个脊椎矫正手术数据`kyphosis`来作一棵简单的分类树如图5.38。该数据包含一个因变量`Kyphosis`（术后是否还存在脊

```

1 > library(rpart)
2 > fit = rpart(Kyphosis ~ Age + Number + Start, data = kyphosis)
3 > par(mar = rep(1, 4), xpd = TRUE)
4 > plot(fit, branch = 0.7)
5 > text(fit, use.n = TRUE)

```

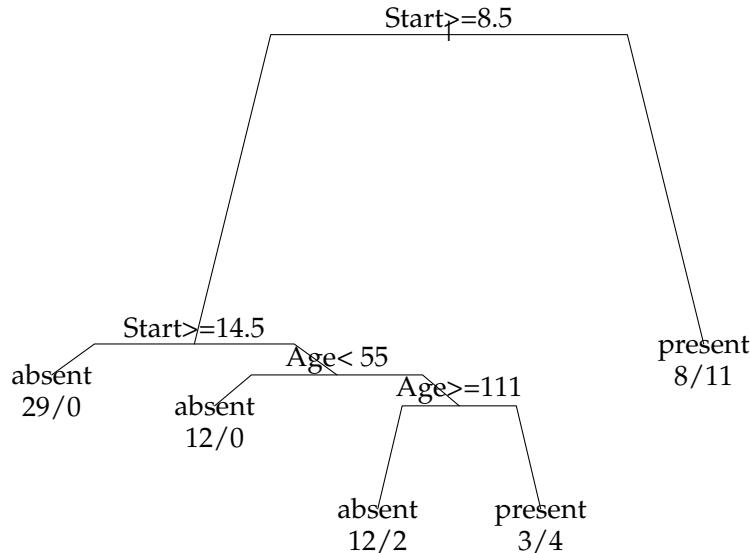


图 5.38: 脊椎矫正手术结果的分类树图

椎畸形) 和三个自变量 **Age** (年龄, 以月计)、**Number** (畸形脊椎的数目) 和 **Start** (从上往下数第一段畸形脊椎的位置)。我们希望知道的是这三个自变量对脊椎矫正手术结果的影响, 例如怎样特征的小孩手术容易失败。分类与回归树的读法为: 每个节点上的条件若满足则树枝向左生长, 否则向右生长, 每片叶子 (最底端, 即不再生长枝节的地方) 上标明了该处的因变量的预测结果⁶, 下方也给出了该叶节点上样本的因变量构成情况。从图5.38中可以看出, **Start**小于8.5的小孩的矫正手术容易失败 (右边叶节点上有11例失败和8例成功), 而对于**Start**大于等于8.5的小孩来说, 手术结果则需要继续按照自变量拆分: **Start**大于等于14.5的29名小孩中, 所有小孩的手术均获成功, 这表明手术成败的重要因素是小孩的第一段畸形脊椎的位置, 这个位置越靠下, 则手术越易成功; 若前面的条件不满足, 则继

⁶若因变量为分类变量, 则预测值按照多数投票表决 (majority vote) 原则计算; 若为数值变量, 则按照叶节点上的样本均值预测。

```

1 > library(vioplot)
2 > f = function(mu1, mu2) c(rnorm(300, mu1, 0.5), rnorm(200,
3 +     mu2, 0.5))
4 > x1 = f(0, 2)
5 > x2 = f(2, 3.5)
6 > x3 = f(0.5, 2)
7 > vioplot(x1, x2, x3, horizontal = TRUE, col = "bisque",
8 +     names = c("A", "B", "C"))

```

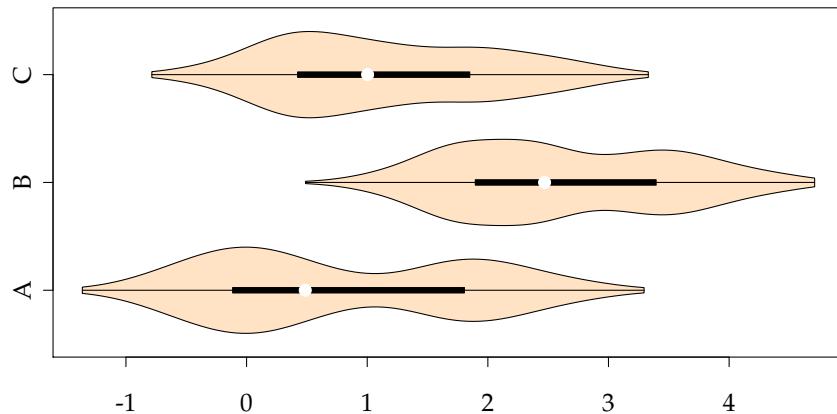


图 5.39: 三组双峰数据的小提琴图比较

续向右拆分，下一个拆分变量为年龄，从下面的几个叶节点来看，年龄越大则手术越不容易成功。

我们还可以将叶节点的信息进一步扩充展示，例如，对于连续型因变量，我们可以将每个叶节点上的样本因变量分布用箱线图或直方图或任何其它展示密度的工具在另一幅图上对应表达出来，而对于离散型因变量，则可以利用条形图等工具将样本因变量的频数表达出来。图形的布局可以利用`layout()`函数（B.2小节）等一页多图的方法来实现。关于这样的例子，读者可以参考Everitt and Hothorn (2006)或Xie (2007)等。

5.33 小提琴图

小提琴图（Violin Plot）是密度曲线图与箱线图的结合，因为它的外观

有时候与小提琴的形状比较相像（尤其是展示双峰数据的密度时），所以我们称之为小提琴图。小提琴图的本质是利用密度值生成的多边形（4.4小节），但该多边形同时还沿着一条直线作了另一半对称的“镜像”，这样两个左右或上下对称的多边形拼起来就形成了小提琴图的主体部分，最后一个箱线图也会被添加在小提琴的中轴线上。

小提琴图来自于**vioplot**包(Adler, 2005)，其函数为*vioplot()*，用法如下：

```

1 > library(vioplot)
2 > usage(vioplot)

vioplot(x, ..., range = 1.5, h = NULL, ylim = NULL,
        names = NULL, horizontal = FALSE, col = "magenta",
        border = "black", lty = 1, lwd = 1, rectCol = "black",
        colMed = "white", pchMed = 19, at, add = FALSE,
        wex = 1, drawRect = TRUE)

```

参数x, ...为一系列数值向量；h传递给**sm**包(Bowman and Azzalini, 2010)中的函数*sm.density()*用来计算密度；至于颜色、方向、边线等样式这里就不再介绍了。

图5.39用三个随机数序列展示了小提琴图的外观及其在表达数据密度和比较统计分布参数（中位数、众数等）上的功效。

lattice包(Sarkar, 2010)中的函数*panel.violin()*也提供了类似的小提琴图展示，鉴于**lattice**包的灵活性，读者不妨也将其作为小提琴图的另一种选择。关于小提琴图的理论请参考Hintze and Nelson (1998)。

5.34 地图

地图毫无疑问是展示地理信息数据时最直观的工具，尤其是当地图和统计量结合时，其功效则会进一步加强。在本书的第一章中曾经提到过John Snow的地图，注意图中不仅标示出了霍乱发生的地点，每个地点的死亡人数也用点的数目标示了出来。历史上还有不少类似的使用地图的例子，而在今天，地理信息系统（GIS）已经成为研究空间和地理数据的热门工具，地图的应用也是屡见不鲜。

地图的本质是多边形（4.4小节），而多边形的边界则由地理经纬度数据确定。R中的附加包**maps**(Brownrigg, 2010)是目前比较完善的地图程序包之一，因此本节主要介绍该程序包。

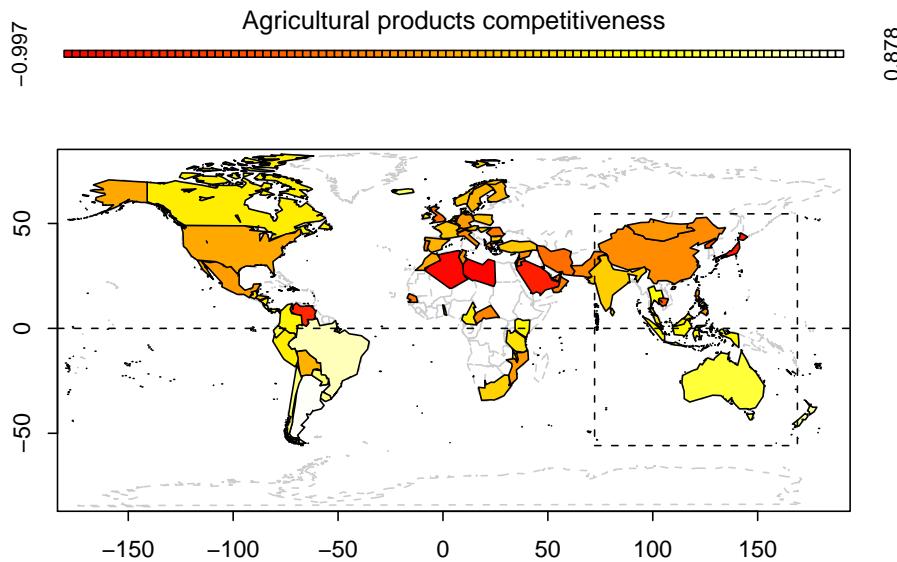


图 5.40: 2005年世界各国农业进出口竞争力地图: 农业出口强国在南美, 弱国在北非。本图代码较长, 此处略去, 感兴趣的读者可从作者个人主页下载。

maps包中核心的函数为 *map()*, 它的用法如下:

```

1 > library(maps)
2 > usage(map, w = 0.75)

map(database = "world", regions = ".",
      exact = FALSE, boundary = TRUE, interior = TRUE,
      projection = "", parameters = NULL, orientation = NULL,
      fill = FALSE, col = 1, plot = TRUE, add = FALSE,
      namesonly = FALSE, xlim = NULL, ylim = NULL,
      wrap = FALSE, resolution = if (plot) 1 else 0,
      type = "l", bg = par("bg"), mar = c(4.1, 4.1,
      par("mar")[3], 0.1), border = 0.01, ...)
```

该函数的两个主要参数为地图数据库 *database* 和地图区域 *region*, 地图数据库中包含了所有区域的经纬度数据以及相应的区域名称, 在指定一个数据库和一系列区域名称之后, 这些区域的地图便可由 *maps()* 生成。其它参

数诸如填充颜色、是否画边界、是否添加到现有图形上等这里就不再介绍，请读者参考帮助文件。

图5.40展示了2005年世界各国地区的农业进出口竞争力指标数据(Xie, 2007)，其中我们将竞争力指标简单定义为(出口-进口)/(出口+进口)。地图上方我们自行添加了颜色图例，从图中可以看出，阿根廷、巴西等南美国家的农业进出口竞争力较强，而利比亚、阿尔及利亚等北非国家的竞争力较弱。该地图的大致制作过程为：首先我们用‘world’数据库作出一幅空白的世界地图，地区边界用灰色线条表示，然后我们根据竞争力数据中的地区名称与地理数据库中地区名称的对应将数据以颜色的形式表示到世界地图中，最后我们在图中添加了赤道线以及东盟国家(ASEAN)的矩形区域⁷。

在地理区域上标记大量的数值信息会遇到一个显而易见的困难，就是由于各个地理区域的面积不同而导致地图的解读失真或某些重要地理单元难以辨认。例如，我们在画中国省级地图时，北京和上海等直辖市相比其它省份显得面积太小，此时若用颜色来标记某个数值指标（如GDP）就会使得各个直辖市的颜色几乎无法辨认。还有另一个更有趣的例子来自美国大选，若用红蓝两种颜色对各个州做标记，以表示该州支持民主党或共和党的总统候选人，那么有些面积大但是权重小的州就会影响整幅美国地图，若我们将各个州的形状进行调整（州的相对位置不变），使其面积与权重成正比，此时红蓝两色的局面就发生了逆转，地图传达信息的偏误才得到了纠正。我们把这种保持地理区域的相对位置不变、调整区域面积与某指标成比例的地图成为“变形地图”(Cartogram)，详细内容可阅读<http://yihui.name/cn/2009/03/cartogram-as-special-maps/>。

5.35 脸谱图

Chernoff faces display multivariate data in the shape of a human face. The individual parts, such as eyes, ears, mouth and nose represent values of the variables by their shape, size, placement and orientation. The idea behind using faces is that humans easily recognize faces and notice small changes without difficulty. Chernoff faces handle each variable differently. Because the features of the faces vary in perceived importance, the way

⁷因为会议论文Xie (2007)的主题是中澳自由贸易区

in which variables are mapped to the features should be carefully chosen.
TeachingDemos包(Snow, 2010)

5.36 平行坐标图

Inselberg (2007)**ggplot2**包(Wickham, 2009)MASS包(Venables and Ripley, 2002)**ipplots**包(Urbaneck and Wichtrey, 2010)

5.37 调和曲线图

(Andrews, 1972)

5.38 二元箱线图

bagplot() **aptpack**包(Wolf and Bielefeld, 2010)

5.39 习题

1. 在第5.30节中我们了解了如何画QQ图，与之对应的还有一种图形叫PP图（Probability-Probability），它也是一种检验数据分布是否和理论分布吻合的图形工具，原理和QQ图类似：对数据的实际概率分布值和理论概率分布值作散点图即可（也可以选择性地添加一条直线）。编写函数画出`geyser$waiting`的PP图（理论分布选择正态分布，均值和标准差用矩估计获得），并评价该数据的正态性。⁸

⁸本题源于COS会员thinkyou的提问：<http://cos.name/cn/topic/18521>

第六章 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》

本章是否要考虑移到最后？……

6.1 网格图形

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）。grid包主要由Paul Murrell开发维护，它的设计初衷是为了克服基础图形系统中图形元素不能动态修改的弱点，例如当一个矩形被画出来之后就无法仅仅更改这个矩形的颜色，而只能重画整幅图形，并更改矩形颜色。例如：

```
1 > # 创建一个名叫rect0的矩形
2 > grid.rect(name = "rect0")
3 > # 修改它的填充颜色为红色
4 > grid.edit("rect0", gp = gpar(fill = "red"))
5 > # 修改为蓝色，不需要重新用grid.rect()画矩形
6 > grid.edit("rect0", gp = gpar(fill = "blue"))
7 > # 若用基础图形系统，则需要这样
```

```

8 > plot(0:1, 0:1)
9 > rect(0, 0, 1, 1, col = "red")
10 > # 为了改变颜色, 重画整幅图形
11 > plot(0:1, 0:1)
12 > rect(0, 0, 1, 1, col = "blue")
13 > # 当然, 这里可以用新的矩形覆盖旧的, 但旧矩形仍然存在

```

6.2 lattice图形

lattice(Sarkar, 2010)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¹, 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”。

6.3 ggplot2图形

上一节中**lattice**虽然灵活, 但它无穷无尽的选项往往让用户感到迷茫。**ggplot2**(Wickham, 2009)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, 构建了一套实用而且美观的图形系统。

¹<http://cm.bell-labs.com/cm/ms/departments/sia/project/trellis/>

第七章 模型

“嗯，人的身高十有八九可以从他的步幅的长度推测出来，这极容易推算，但是让我把枯燥的数字摆出来算给你看实在是毫无用处。我在屋外的粘土路上和屋内的尘土上找到了那人的脚印，此外我还有一个法子验证我的计算：当一个人在墙上写字时，他会本能地将字写在视线以上的地方，而那个血字正好离地面六英尺。这推算实在是简单得像儿戏一般。”

我问：“那他的年龄呢？”

“好的，如果一个人不费吹灰之力就能一步跨出四英尺半，他不可能年老体衰，因为花园小径上的泥坑恰好长四英尺半，他显然是一步跨过去的，漆皮靴则是绕过去的，所以这也没有什么神秘之处。我只是将我在那篇文章中推崇的一些观察和演绎的规则应用在日常生活中罢了。你还有什么不明白的地方吗？”

—柯南·道尔《血字的研究》

在1.5小节中我们曾经提到三种基本的统计分析方式，其中包括推断统计分析和探索统计分析，表面上看来，统计模型在前者中占主导地位，而统计图形在后者中往往起着非常重要的作用，然而本章要探讨和揭示的则是以探索为目标的统计图形与以推断为目标的统计模型之间的联系。

7.1 线性回归模型

- 一元回归：散点图
- 回归诊断：方差齐性假设、线性假设、独立性假设、正态性假设、离群点

7.2 方差分析

- 类似回归
- 箱线图等

7.3 非参数回归模型

7.3.1 局部加权回归散点平滑法

lowess() loess() scatter.smooth()

- <http://cos.name/2008/11/lowess-to-explore-bivariate-correlation-by-yihui/>
- runmed() ?

7.4 稳健回归模型

- 离群点

7.5 广义线性模型

- 回归诊断

7.6 分类数据模型和列联表

- 马塞克图

7.7 混合效应模型

- Gelman讲“爷为啥不关心多重比较的问题”：http://www.stat.columbia.edu/~gelman/presentations/multiple_minitalk2.pdf（随机效应模型究竟做了什么？）

7.8 主成分分析和因子分析

- 成分方差
- Biplot

7.9 聚类分析

- 谱系图
- K-Means的散点图

7.10 判别分析

- 成分的散点图

7.11 对应分析

7.12 多维标度分析

`cmdscale()`, `isoMDS()`

- 高维数据的二维（低维）表示

7.13 时间序列模型

经典ARMA、谱分析等

自相关`acf()` 偏自相关`pacf()` 时滞图`lag.plot()`

累积周期图`cpgram()`

7.14 生存分析

7.15 空间统计学

7.16 数据挖掘和机器学习

7.16.1 分类与回归树

7.16.2 Bootstrap

7.16.3 支持向量机

第八章 数据

“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》

本章中我们主要从数据类型的角度对第五章中所叙述的统计图形进行一个比较全面的梳理与总结，即：什么样的数据适合用什么样的统计图形展示，以及数据中的信息以怎样的方式表达。本章中的数据类型主要分为离散数据和连续数据以及它们的混合，在各种数据类型中，我们分别对低维和高维数据的图示作出总结。

8.1 离散数据

8.1.1 一维数据

- 条形图

8.1.2 多维数据

- 马塞克图

8.2 连续数据

8.2.1 一维数据

- 直方图
- 箱线图

8.2.2 二维数据

- 散点图

8.2.3 高维数据

- 平行坐标图
- 降维

8.3 混合数据

8.3.1 一维数据

8.3.2 二维数据

8.3.3 高维数据

附录 A 程序初步

如第二章所讲，R的编程方式是面向对象（Object-Oriented）的，这里我们把R中的数据类型简要介绍一下，以便读者能熟练操纵数据；此外，我们也简要介绍一下R编程中的选择与循环语句以及输入输出的操作。

A.1 对象类型

在R的系统中，几乎任何东西都是对象。使用对象的好处在于它们都可以重用（Reuse）。例如我们可以建立并拟合一个回归模型（不妨称之为`fit`），这个对象中包含了若干子对象，在后面的计算中我们随时可以调用这个对象中的子对象，如残差向量（`fit$residuals`或`resid(fit)`）、系数估计（`fit$coefficients`或`coef(fit)`）等。面向对象的编程方式尤其在涉及到大量计算的工作中会大显身手，刚才我们提到的只是做一个回归模型，看起来优势并不明显，但如果我们要用某个因变量针对1000个自变量分别作回归，然后看看回归系数的t值或者AIC值的分布情况等等，这时“对象”操作的便利性就充分体现出来了，相比之下，读者不妨考虑用SPSS或其它软件如何完成类似的任务及其难度。掌握了R的对象之后，在R的世界编程基本就可以畅通无阻了。

A.1.1 向量

向量（vector）是最简单的数据结构，它是若干数据点的简单集合，如从1到10的数字：

```
1 > 1:10  
[1] 1 2 3 4 5 6 7 8 9 10
```

通常我们可以用函数`c()`拼接一些数字或字符生成一个向量，如：

```
1 > c(7.11, 9.11, 9.19, 1.23)
[1] 7.11 9.11 9.19 1.23
```

我们可以将一个向量赋值给一个变量：

```
1 > (x <- c(7.11, 9.11, 9.19, 1.23))
[1] 7.11 9.11 9.19 1.23
```

注意R中赋值符号可以是`<-`或`=`，或`->`（从左往右赋值），或者使用`assign()`函数进行赋值。向量的运算一般都是针对每一个元素的运算，如：

```
1 > 1/x
[1] 0.1406 0.1098 0.1088 0.8130
1 > x + 1
[1] 8.11 10.11 10.19 2.23
```

实际上以向量的形式进行元素运算是R语言计算的重要特征。通过中括号和下标值可以提取向量中的元素或者改变相应位置的元素：

```
1 > x[c(1, 4)]
[1] 7.11 1.23
1 > # 将x赋值给tmp
2 > (tmp = x)
[1] 7.11 9.11 9.19 1.23
1 > tmp[1] = 10
2 > tmp
[1] 10.00 9.11 9.19 1.23
```

利用现有的向量可以继续利用`c()`生成新的向量：

```
1 > (y = c(x, 12.19))
[1] 7.11 9.11 9.19 1.23 12.19
```

向量的长度可以用`length()`获得：

```
1 > length(y)
```

```
[1] 5
```

我们还可以用`names()`给向量的每一个元素命名:

```
1 > names(x) = LETTERS[1:length(x)]
```

```
2 > x
```

A	B	C	D
7.11	9.11	9.19	1.23

对于有名称的向量，我们可以用名称提取向量的元素（获取数据子集的方式通常有三种：整数下标、子对象名称以及逻辑值）：

```
1 > x[c("B", "A")]
```

B	A
9.11	7.11

函数`sort()`可以对向量排序（顺序或倒序）：

```
1 > sort(x)
```

D	A	B	C
1.23	7.11	9.11	9.19

因为很多统计量的计算是针对一维数据的，所以用向量操作起来会非常方便，例如计算样本方差 $\sum_{i=1}^n (x_i - \bar{x})^2 / (n - 1)$:

```
1 > sum((x - mean(x))^2)/(length(x) - 1)
```

```
[1] 14.03
```

```
1 > # 自带函数var()作为对比
```

```
2 > var(x)
```

```
[1] 14.03
```

当然R提供了现成的方差函数`var()`，我们不必将代码写得那么复杂，从上面的输出可以看出，直接根据公式写的代码和方差函数计算的结果是一样的。另外，向量操作可以节省显式循环的使用，如果在C语言或VB等其它程序语言中，我们只能使用几段循环来计算方差数值，因为其中涉及到两个求和函数。

使用函数`seq()`和`rep()`可以生成规则的序列，前者提供了等差数列的功能，后者可以将向量或元素重复多次从而生成新的向量，如：

```

1 > # 冒号表示步长为1或-1的序列
2 > 10:1
[1] 10 9 8 7 6 5 4 3 2 1

1 > # 步长为0.2
2 > seq(7, 9, 0.2)
[1] 7.0 7.2 7.4 7.6 7.8 8.0 8.2 8.4 8.6 8.8 9.0

1 > # 生成向量长度为6
2 > seq(7, 9, length.out = 6)
[1] 7.0 7.4 7.8 8.2 8.6 9.0

1 > rep(2, 10)
[1] 2 2 2 2 2 2 2 2 2 2

1 > # 整个向量重复5次
2 > rep(1:3, 5)
[1] 1 2 3 1 2 3 1 2 3 1 2 3

1 > # 每个元素重复5次
2 > rep(1:3, each = 5)
[1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3

1 > # 每个元素分别重复1、2、3次
2 > rep(1:3, 1:3)
[1] 1 2 2 3 3 3

```

向量除了可以是数值型之外，还可以是逻辑值、字符等，如：

```

1 > (z = (x < 5))
      A      B      C      D
      FALSE FALSE FALSE  TRUE

1 > # letters和LETTERS是R中的字符常量
2 > letters
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o"
[16] "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"

```

注意逻辑值的TRUE和FALSE可以简写为T和F，但强烈建议这两个逻辑值不要使用简写¹。我们也可以用逻辑向量提取向量的元素，如：

```
1 > # 满足 “非z” 的元
2 > x[!z]
```

A	B	C
7.11	9.11	9.19

R中有三种特殊的值：缺失值NA，非数值NaN和无穷大/无穷小Inf/-Inf。非数值通常由无意义的数学计算产生，如0/0；注意分子不为0而分母为0时，结果是无穷大。

A.1.2 因子

因子（factor）对应着统计中的分类数据，它的形式和向量很相像，只是因子数据具有水平（level）和标签（label），前者即分类变量的不同取值，后者即各类取值的名称。

因子型数据可以由factor()函数生成，如：

```
1 > (x = factor(c(1, 2, 3, 1, 1, 3, 2, 3, 3), levels = 1:3,
2 +     labels = c("g1", "g2", "g3")))
[1] g1 g2 g3 g1 g1 g3 g2 g3 g3
Levels: g1 g2 g3
```

我们可以对因子型数据求频数、将其转化为整数或字符型向量。注意整数是因子型数据的本质：它本身以整数存储，但表现出来是字符，原理就是把整数对应的标签显示出来，这种存储方式在很多情况下可以大大节省存储空间（存整数往往比存字符串占用空间小）。

```
1 > table(x)

x
g1 g2 g3
3 2 4

1 > as.integer(x)
```

¹ 倾向简写这两个逻辑值的人容易倾向简写任何字符，比如变量名；据作者的经验，很多人的程序出错就在这里，因为使用了T或F作为变量名，而在程序的某些地方又将T或F作为逻辑值对待，如这段条件语句x = 1; T = NULL; ...; if ((x > 0) == T) ...会出错，因为条件(x > 0) == T返回的结果是长度为0的逻辑值，不符合if语句的要求。

```
[1] 1 2 3 1 1 3 2 3 3
1 > as.character(x)
[1] "g1" "g2" "g3" "g1" "g1" "g3" "g2" "g3" "g3"
```

因子型数据在分类汇总时比较有用，例如在`tapply()`中：

```
1 > y = 1:9
2 > # x和y并列放的样子
3 > data.frame(y, x)
```

	y	x
1	1	g1
2	2	g2
3	3	g3
4	4	g1
5	5	g1
6	6	g3
7	7	g2
8	8	g3
9	9	g3

```
1 > # x的每一组的均值
2 > tapply(y, x, mean)
```

	g1	g2	g3
3.333	4.500	6.500	

因子型数据中有一种特殊的类型，就是有序因子，它与普通的因子型数据的区别在于各个分类之间是有顺序的，即统计上所说的定序变量；相关函数为`ordered()`。

A.1.3 数组和矩阵

数组和矩阵是具有维度属性（dimension）的数据结构，注意这里的维度并非统计上所说的“列”，而是一般的计算机语言中所定义的维度（数据下标的个数）。矩阵是数组的特例，它的维度为2。数组和矩阵可以分别由`array()`和`matrix()`函数生成。注意矩阵中所有元素必须为同一种类型，要么全都是数值，要么全都是字符，后面我们会介绍数据框，它会放宽这个要求。这里给出一些示例说明数组和矩阵的用法：

```

1 > # 3维数组示例
2 > (x = array(1:24, c(3, 4, 2)))

, , 1

[,1] [,2] [,3] [,4]
[1,]    1     4     7    10
[2,]    2     5     8    11
[3,]    3     6     9    12

, , 2

[,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24

1 > # 维数
2 > dim(x)

[1] 3 4 2

1 > # 第3维第1个位置上的元
2 > x[, , 1]

[,1] [,2] [,3] [,4]
[1,]    1     4     7    10
[2,]    2     5     8    11
[3,]    3     6     9    12

1 > x[1, , 2]

[1] 13 16 19 22

1 > # 矩阵示例
2 > (x = matrix(1:12, nrow = 2, ncol = 6))

[,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1     3     5     7     9    11
[2,]    2     4     6     8    10    12

1 > x[1, 5]

[1] 9

1 > # 逻辑比较: 是否小于5?
2 > x < 5

```

```

[,1] [,2] [,3] [,4] [,5] [,6]
[1,] TRUE TRUE FALSE FALSE FALSE FALSE
[2,] TRUE TRUE FALSE FALSE FALSE FALSE

1 > # 以逻辑矩阵为下标挑选出小于5的元
2 > x[x < 5]

[1] 1 2 3 4

1 > # 各个元素的平方
2 > x^2

[,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1     9   25   49   81  121
[2,]    4    16   36   64  100  144

1 > # 矩阵乘法 X * X' (t()为转置函数)
2 > x %*% t(x)

[,1] [,2]
[1,] 286 322
[2,] 322 364

1 > # 一个随机方阵
2 > (x = matrix(sample(9), 3))

[,1] [,2] [,3]
[1,]    4     2     9
[2,]    8     7     5
[3,]    1     3     6

1 > # 求逆
2 > solve(x)

[,1]      [,2]      [,3]
[1,] 0.15429  0.08571 -0.30286
[2,] -0.24571  0.08571  0.29714
[3,]  0.09714 -0.05714  0.06857

1 > # 还有eigen(x)求特征根与特征向量, 用qr(x)对矩阵作QR分解, 等等
2 > # 我们还可以用cbind()和rbind()等函数将几个向量拼接成矩阵

```

矩阵相对于数组来说在统计中应用更为广泛, 尤其是大量的统计理论都涉及到矩阵的运算。顺便提一下, R包**Matrix**(Bates and Maechler, 2010)在处理(高维)稀疏或稠密矩阵时效率会比R自身的矩阵运算效率更高。

A.1.4 数据框和列表

数据框（data frame）和列表（list）是R中非常灵活的数据结构。数据框也是二维表的形式，与矩阵非常类似，区别在于数据框只要求各列内的数据类型相同，而各列的类型可以不同，比如第1列为数值，第2列为字符，第3列为因子，等等；列表则是更灵活的数据结构，它可以包含任意类型的子对象。数据框和列表分别可以由`data.frame()`和`list()`生成。

```

1 > # 生成一个数据框，前两列为数值型，后一列为字符型
2 > data.frame(x = rnorm(5), y = runif(5), z = letters[1:5])
      x      y   z
1 0.07055 0.6324 a
2 0.25110 0.2883 b
3 0.33003 0.4333 c
4 -1.03245 0.1963 d
5 0.75708 0.1488 e

1 > # 包含四个子对象的列表
2 > (Lst = list(name = "Fred", wife = "Mary", no.children = 3,
3 +     child.ages = c(4, 7, 9)))
$name
[1] "Fred"

$wife
[1] "Mary"

$no.children
[1] 3

$child.ages
[1] 4 7 9

1 > # 用名称提取子对象
2 > Lst$child.ages
[1] 4 7 9

1 > # 用整数下标提取子对象
2 > Lst[[2]]
[1] "Mary"

1 > # 甚至可以试试list(first = Lst, second = 1:10)之类的语句

```

矩阵的本质是（带有维度属性的）向量，数据框的本质是（整齐的）列表。

A.1.5 函数

R中也可以自定义函数，以便编程中可以以不同的参数值重复使用一段代码。定义函数的方式为：`function(arglist) expr return(value);` 其中`arglist`是参数列表，`expr`是函数的主体，`return()`用来返回函数值。例如我们定义一个求峰度 $\sum_{i=1}^n (x_i - \bar{x})^4 / (n \cdot \text{Var}(x)) - 3$ 的函数`kurtosis()`如下：

```

1 > kurtosis = function(x, na.rm = FALSE) {
2 +   # 去掉缺失值
3 +   if (na.rm)
4 +     x = x[!is.na(x)]
5 +   return(sum((x - mean(x))^4)/(length(x) * var(x)^2) -
6 +         3)
7 + }
```

该函数有两个参数，数据向量`x`和是否删除缺失值`na.rm`，后者有默认值`FALSE`；下面我们用均匀分布的随机数测试一下：

```

1 > # 理论上均匀分布的峰度应该小于1，以下为一个模拟结果
2 > kurtosis(runif(100))

[1] -0.911
```

注意，R中有时候不必用函数`return()`指定返回值，默认情况下，函数主体的最后一行即为函数返回值。

最后，因为本书的很多作图函数都是泛型函数，我们也补充一点泛型函数（generic function）的作用原理。实际上泛型函数是根据第一个参数的类（class）去调用了相应的“子函数”。以`plot()`为例，我们用`methods()`看一下它有哪些作图方法：

```

1 > methods("plot")

[1] plot.Date*           plot.HoltWinters*
[3] plot.POSIXct*        plot.POSIXlt*
[5] plot.TukeyHSD        plot.aareg*
[7] plot.acf*            plot.correspondence*
[9] plot.cox.zph*        plot.data.frame*
[11] plot.decomposed.ts*  plot.default
[13] plot.dendrogram*    plot.density
[15] plot.ecdf            plot.factor*
[17] plot.formula*        plot.hclust*
[19] plot.histogram*      plot.isoreg*
```

```
[21] plot.lda*          plot.lm
[23] plot.mca*         plot.medpolish*
[25] plot.mlm           plot.ppr*
[27] plot.prcomp*       plot.princomp*
[29] plot.profile*      plot.profile.nls*
[31] plot.ridgelm*      plot.rpart*
[33] plot.spec           plot.spec.coherency
[35] plot.spec.phase    plot.spline*
[37] plot.stepfun        plot.stl*
[39] plot.survfit*       plot.table*
[41] plot.ts              plot.tskernel*
[43] plot.xyVector*
```

Non-visible functions are asterisked

可以看出，这些函数都是以`plot.*`的形式定义的，其中*便是类的名称。泛型函数的基本原理就是：传给`plot()`的第一个参数是何种类，则调用何种函数进行作图。例如`iris`的类是`data.frame`，那么`plot(iris)`就会调用`plot.data.frame()`作图（散点图矩阵）。下面的代码有助于进一步说明这种原理：

```
1 > class(iris)
[1] "data.frame"

1 > plot(iris)
2 > x = density(faithful$waiting)
3 > class(x)

[1] "density"

1 > plot(x)
```

我们也可以对现有的泛型函数进行扩充，使它们适应我们自定义的类。例如`print()`也是一个常用的泛型函数，当我们在R控制台中键入一个对象以显示其内容时，实际上是调用了该函数以打印出对象的内容。现在我们定义一个`print.yihui()`，使得类名称为`yihui`的对象在屏幕上打印出来时数值为真实数值的2倍：

```
1 > # 本例告诉我们，眼见未必为实!
2 > x = 1:5
3 > class(x)
```

```

[1] "integer"

1 > class(x) = "yihui"
2 > x

[1] 1 2 3 4 5
attr(,"class")
[1] "yihui"

1 > print.yihui = function(x) print.default(unclass(2 *
2 +      x))
3 > x

[1] 2 4 6 8 10

1 > str(x)

Class 'yihui'  int [1:5] 1 2 3 4 5

```

至此，读者应该能够明白有些作图函数既可以直接接受数据作为参数又可以接受公式作为参数的原因了，如`boxplot()`。

A.2 操作方法

在了解对象的几种基本类型之后，我们需要知道如何对这些对象进行简单的四则运算之外的操作。在计算机程序和算法中，最常见结构的就是选择分支结构和循环结构，通过这样的程序语句，我们可以进一步控制和操纵对象；同时，在执行计算机程序时，我们也常常需要一些输入输出的操作。

A.2.1 选择与循环

一般来说，计算机程序都是按代码先后顺序执行的，而有时候我们希望代码能够按照一定的判断条件执行，或者将一个步骤执行多次，此时我们就需要选择和循环结构的程序。

R提供了如下一些实现选择和循环的方法：

- `if(cond) expr if(cond) cons.expr else alt.expr`
- `for(var in seq) expr`

- `while(cond) expr`

`cond`为条件（其计算结果为逻辑值TRUE或FALSE），`expr`为一段要执行的代码，通常放在大括号{}中，除非代码只有一行。

选择结构的函数还有`ifelse()`和`switch()`，请读者自行查阅帮助文件。关于选择和循环，这里仅给出一个简单的综合示例，不再详加说明：

```

1 > # x初始为空
2 > x = NULL
3 > for (i in 1:10) {
4 +   rnd = rnorm(1)
5 +   # 如果随机数在 [-1.65, 1.65]范围内则放到x中去
6 +   if (rnd < 1.65 & rnd > -1.65)
7 +     x = c(x, rnd)
8 +
9 > x

```

```

[1] -0.5962 -0.7694  0.3263 -0.2106  1.6465  0.3494  0.5809
[8]  0.7325 -0.5273 -0.8510

```

A.2.2 输入与输出

对于统计程序来说，输入的一般是数据，而输出一般是图形和表格，它们在R中都容易实现。前者可以参考`read.table()`系列函数，后者则可以使用图形设备（附录B.5）以及`write.table()`系列函数。具体使用方法请查阅这些函数的帮助文件。

A.3 习题

1. 绝对离差中位数（Median Absolute Deviation, MAD）²: MAD是对数据分散程度的一种刻画，它的定义为 $MAD(x) = C \cdot \text{median}(|x - \text{median}(x)|)$ ，其中 C 是一个给定的常数，通常取值为 $1/qnorm(.75)$ ；若原数据服从正态分布，MAD将是标准差的一个渐近估计。编写函数计算数据`women$height`的绝对离差中位数，并与R自带的`mad()`函数作比较；若有兴趣，请欣赏`mad()`函数的源代码（如何将程序写得安全而简练）。

²本题源于COS会员statsgirl的提问：<http://cos.name/cn/topic/101087>

2. 变量选择问题：某多元线性回归模型包含了20个自变量，根据某些经验，自变量的个数在3个左右是最合适的。要求建立所有可能的回归模型，并根据AIC准则找出最优模型以及相应的自变量。

提示：所有回归模型数目为组合数 $C_{20}^3 = 1140$ ，可使用函数`combn()`生成所有组合，用`lm()`建模，并用`AIC()`提取AIC值。

3. DNA序列的反转互补³：给定一个DNA序列字符串，要求将它的字符序列顺序颠倒过来，然后A与T互换，C与G互换。例如原序列为“TTGGTAGTGC”，首先将它顺序反转为“CGTGATGGTT”，然后互补为“GCACTACCAA”。

提示：可以采用`substr()`暴力提取每一个字符的方式，然后用繁琐的循环和选择语句实现本题的要求；但利用R的向量化操作功能会大大加速计算速度，可采用的函数有：拆分字符串`strsplit()`、反转向量`rev()`、拼接字符串`paste()`。仔细考虑如何利用名字和整数的下标功能实现序列的互补。

4. 接受—拒绝抽样（Acceptance-Rejection Sampling）：当我们不容易从某个分布中生成随机数的时候，若有另一个分布的密度函数 $g(x)$ 能控制前一个分布的密度函数 $f(x)$ 的核 $h(x)$ （此处“控制”的意思是 $\exists M > 0, h(x)/g(x) \leq M, \forall x$ ，核 $h(x)$ 正比于 $f(x)$ ），而且我们很方便从 $g(\cdot)$ 中生成随机数，那么我们可用“接受—拒绝抽样”的方式从 $f(\cdot)$ 中生成随机数。步骤如下：

- (a) 从 $g(\cdot)$ 中生成随机数 x_i ；
- (b) 从均匀分布 $U(0, 1)$ 中生成随机数 u_i ；
- (c) 若 $u_i \leq h(x_i)/(M \cdot g(x_i))$ ，则记下 x_i ；

重复以上步骤若干次，被接受的 x_i 的概率密度函数即为 $f(\cdot)$ 。根据以上抽样步骤用适当的循环和选择语句生成服从以下分布的随机数：

$$f(\theta) = c \frac{\exp(\theta)}{1 + \exp(\theta)} \frac{1}{\sqrt{2\pi}} \exp(-\theta^2/2)$$

其中， c 为使得 $f(\theta)$ 积分为1的常数。

³本题源于COS会员iiiiiiiii的提问：<http://cos.name/cn/topic/18471>

附录 B 作图技巧

本章我们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。本章的内容包括：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。

B.1 添加数学公式

由于统计理论中经常需要用到数学符号，所以向统计图形中添加一些数学说明不仅会使得图形看起来更专业，对图形背后的理论也是一种重要补充。

R的**grDevices**包中提供了一系列数学公式的表达符号，例如运算符（加减乘除乘方开方等）、比较符（等号不等号大于小于号等）、微积分符号、希腊字母（大小写 α 至 ω ）、上标下标等等。这些数学符号的使用与**LATEX**数学公式非常类似，因此如果读者对**LATEX**公式比较熟悉的话，用起R中的数学表达式来也会很顺手。

如果想向图中添加数学表达式的文本标签，只需要将文本设置为表达式（**expression**）的类型即可。图B.1展示了向正态曲线上添加正态分布密度函数表达式的方法，可以看到，表达式中的公式都是**LATEX**与R的混合语法。另外，我们也可以设置符号的外形，如斜体、粗体等。详情参见?**plotmath**或者运行代码**demo(plotmath)**观看R提供的数学公式演示。

注意本书中的所有图形均由**tikzDevice**包(Sharpsteen and Bracken, 2009)生成，其中的数学公式为原始**LATEX**代码，其质量比R自身的数学公式质量高很多，因此图B.1并没有采用**demo(plotmath)**中的写法生成数学公式图上展示的代码为“伪代码”。

```

1 > # 本代码为“伪代码”，下图由pgfSweave生成
2 > plot(seq(-3, 3, 0.1), dnorm(seq(-3, 3, 0.1)), type = "l",
3 +   xlab = "x", ylab = expression(phi(x)))
4 > text(-3, 0.37, adj = c(0, 1), expression(phi(x) ==
5 +   frac(1, sqrt(2 * pi)) ~ e^-frac(x^2, 2)), cex = 1.2)
6 > arrows(-2, 0.27, -1.3, dnorm(-1.3) + 0.02)
7 > abline(v = qnorm(0.95), lty = 2)
8 > text(0, dnorm(qnorm(0.95)), expression(integral(phi(x) *
9 +   dx, -infinity, 1.65) %~~% 0.95))

```

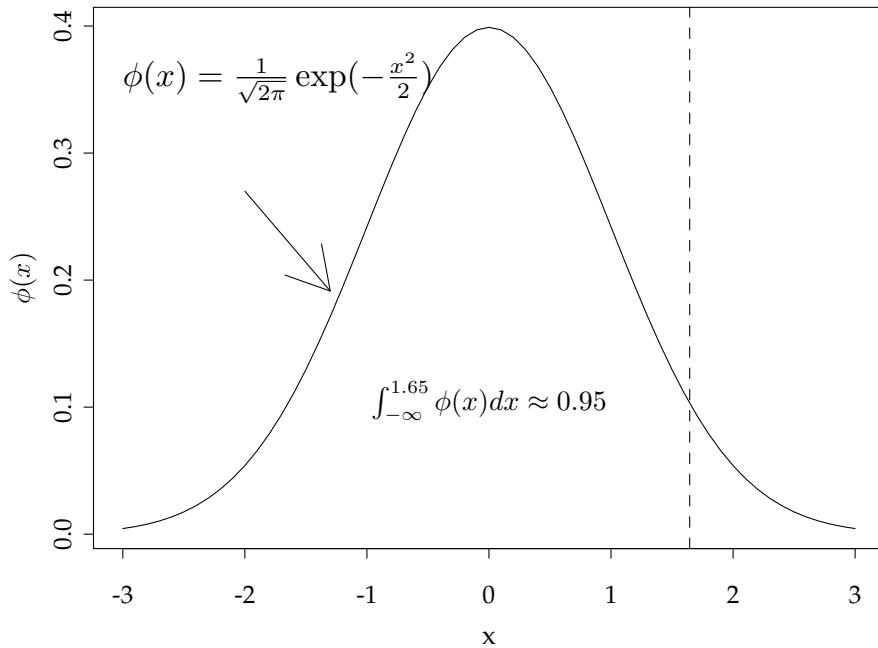


图 B.1: 正态分布密度函数公式的表示

B.2 一页多图

有时候我们需要将多幅图形放在同一页图中，以便对这些图形作出对比，或者使图形的排列更加美观。这种情况下，我们至少可以有三种选择：

B.2.1 设置图形参数

在前面3.1小节中我们曾经讲到过`mfrow`和`mfcol`两个参数，如果我们在`par()`函数中给这两个参数中的一者提供一个长度为2的向量，那么接下来的图形就会按照这两个参数所设定的行数和列数依次生成图形。本书的图形中有很多用到过这两个参数，如图3.4、5.4等，另外有一些统计图形函数也利用了这两个参数设置它们的图形版面，如四瓣图、条件分割图等。

这两个参数的限制在于它们只能将图形区域拆分为网格状，每一格的长和宽都分别必须相等，而且每一格中必须有一幅图形，不能实现一幅图形占据多格的功能。下面的两个函数则灵活许多。

B.2.2 设置图形版面

R提供了`layout()`函数作为设置图形版面拆分的工具，其用法如下：

```
1 > usage(layout)
layout(mat, widths = rep(1, ncol(mat)),
       heights = rep(1, nrow(mat)), respect = FALSE)
1 > usage(layout.show)
layout.show(n = 1)
```

其中`mat`参数为一个矩阵，提供了作图的顺序以及图形版面的安排；`widths`和`heights`提供了各个矩形作图区域的长和宽的比例；`respect`控制着各图形内的横纵轴刻度长度的比例尺是否一样；`n`为欲显示的区域的序号。

`mat`矩阵中的元素为数字1到n，矩阵行列中数字的顺序和图形方格的顺序是一样的。图B.2解释了这种顺序，该图的矩阵为：

```
1 > matrix(c(1, 2, 1, 3), 2)
      [,1] [,2]
[1,]     1     1
[2,]     2     3
```

```

1 > layout(matrix(c(1, 2, 1, 3), 2), c(1, 3), c(1, 2))
2 > layout.show(2)

```

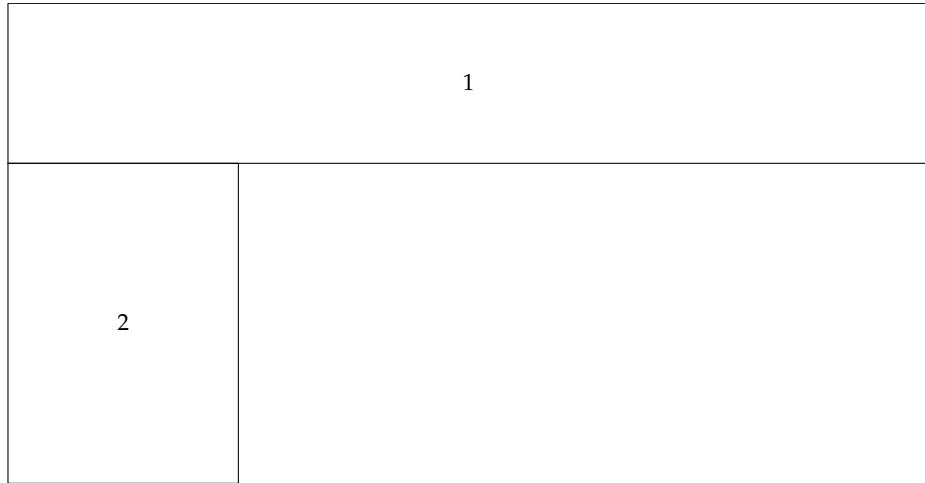


图 B.2: 函数layout()的版面设置示意图

由于这种设置,使得第1幅图占据了(1, 1)和(1, 2)的位置,接下来第2、3幅图分别在(2, 1)和(2, 2)的位置;加上长度和宽度的设置,便产生了图B.2的效果。

前面图5.27和5.24曾经使用该函数设置了图形版面,使得不同方格中的图形长宽不一样。图B.3用layout()安排展示了二元变量的边际分布以及回归直线。

B.2.3 拆分设备屏幕

R中还有另外一种拆分屏幕的方法,即`split.screen()`。这种方法比前两种方法更灵活,它不仅可以像前两种方法一样设定将作图区域拆分为若干行列,也可以随意指定作图区域在屏幕上的位置。该函数及相关函数用法如下:

```

1 > usage(split.screen)

split.screen(figs, screen, erase = TRUE)

1 > usage(screen)

```

```

1 > x = pmin(3, pmax(-3, stats::rnorm(50)))
2 > y = pmin(3, pmax(-3, x + runif(50, -1, 1)))
3 > xhist = hist(x, breaks = seq(-3, 3, 0.5), plot = FALSE)
4 > yhist = hist(y, breaks = seq(-3, 3, 0.5), plot = FALSE)
5 > top = max(c(xhist$counts, yhist$counts))
6 > layout(matrix(c(2, 0, 1, 3), 2, 2, byrow = TRUE),
7 + c(3, 1), c(1, 3))
8 > par(mar = c(2, 2, 1, 1))
9 > plot(x, y, xlim = c(-3, 3), ylim = c(-3, 3), ann = FALSE)
10 > abline(lm(y ~ x))
11 > par(mar = c(0, 2, 1, 1))
12 > barplot(xhist$counts, axes = FALSE, ylim = c(0, top),
13 + space = 0)
14 > par(mar = c(2, 0, 1, 1))
15 > barplot(yhist$counts, axes = FALSE, xlim = c(0, top),
16 + space = 0, horiz = TRUE)

```

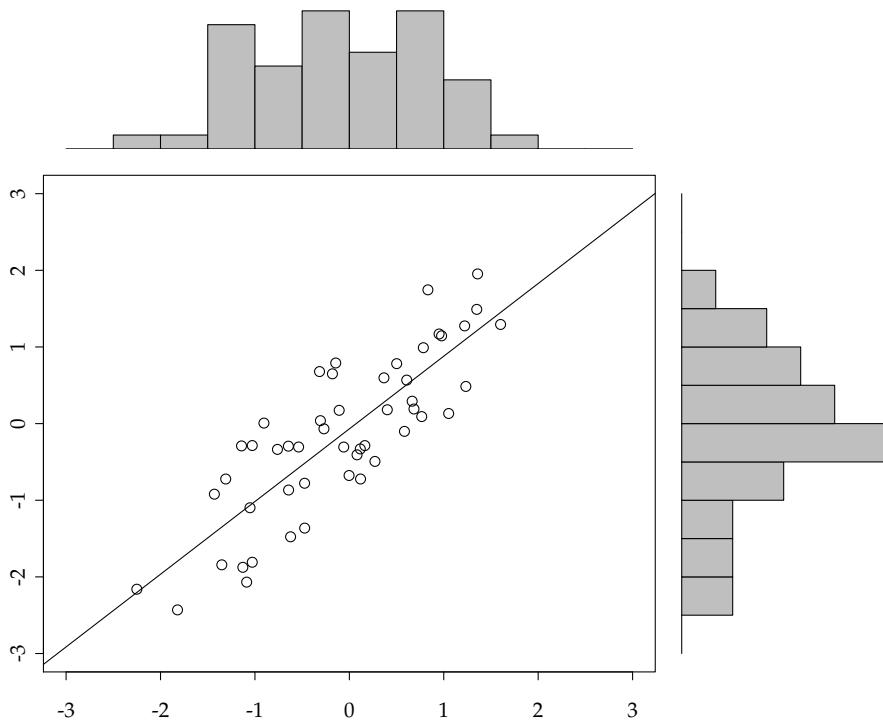


图 B.3: 回归模型中边际分布的展示: 左下方图中展示了散点图和回归直线, 上方和右方的直方图分别展示了自变量和因变量的密度分布。

```

screen(n = cur.screen, new = TRUE)

1 > usage(erase.screen)

erase.screen(n = cur.screen)

1 > usage(close.screen)

close.screen(n, all.screens = FALSE)

```

拆分后的屏幕由若干个区域构成，每个区域有一个编号，即`screen`，我们可以用函数`screen()`指定要作图的区域号，或者用`erase.screen()`擦除该区域的图形，而`split.screen()`的用法主要由`figs`参数控制，该参数既可以取值为一个长度为2的向量（指定行列的数目），也可以是一个4列的数值矩阵，制定图形区域的坐标位置，后一种用法比较灵活，它可以将图形作在屏幕的任意位置上，这里的4列矩阵分别给定区域横坐标的左和右以及纵坐标的下和上的位置，即给定了区域左下角和右上角的坐标，这样就可以划分出一块矩形作图区域来。注意这里的坐标值应该在[0, 1]范围内，整个屏幕左下角坐标为(0, 0)，右上角坐标为(1, 1)。

图B.4给出了用矩阵指定作图区域位置的示例，该矩阵的取值为：

```

1 > matrix(c(0, 0.1, 0.4, 0.3, 0.5, 0.8, 0.9, 1, 0, 0.2,
2 +      0.3, 0.5, 0.4, 0.7, 0.8, 1), 4, 4)

[,1] [,2] [,3] [,4]
[1,] 0.0  0.5  0.0  0.4
[2,] 0.1  0.8  0.2  0.7
[3,] 0.4  0.9  0.3  0.8
[4,] 0.3  1.0  0.5  1.0

```

矩阵一共四行，因此制定了四个屏幕作图区域，四列给定了区域的位置，例如第1个区域的位置在点(0.0, 0.0)与点(0.5, 0.4)之间。该示例中，整个屏幕中划分出了4块有重叠的区域，并分别画出了4幅散点图。

拆分屏幕区域方法的灵活性还在于它可以在拆分的区域中继续拆分（类似于“递归”的做法），而前两节中提到的办法是无法做到这一点的，因此三种方法中这种方法的功能是最强大的，但大多数情况下我们其实用不着如此灵活的定制方法，网格式拆分已经足够使用。

```
1 > split.screen(matrix(c(0, 0.1, 0.4, 0.3, 0.5, 0.8,
2 +     0.9, 1, 0, 0.2, 0.3, 0.5, 0.4, 0.7, 0.8, 1),
3 +     4, 4))
4 > for (i in 1:4) {
5 +   screen(i)
6 +   par(mar = c(0, 0, 0, 0), mgp = c(0, 0, 0), cex.axis = 0.7)
7 +   plot(sort(runif(30)), sort(runif(30)), col = i,
8 +         pch = c(19, 21, 22, 24)[i], ann = FALSE,
9 +         axes = FALSE)
10 +   box(col = "gray")
11 +   axis(1, tcl = 0.3, labels = NA)
12 +   axis(2, tcl = 0.3, labels = NA)
13 + }
```

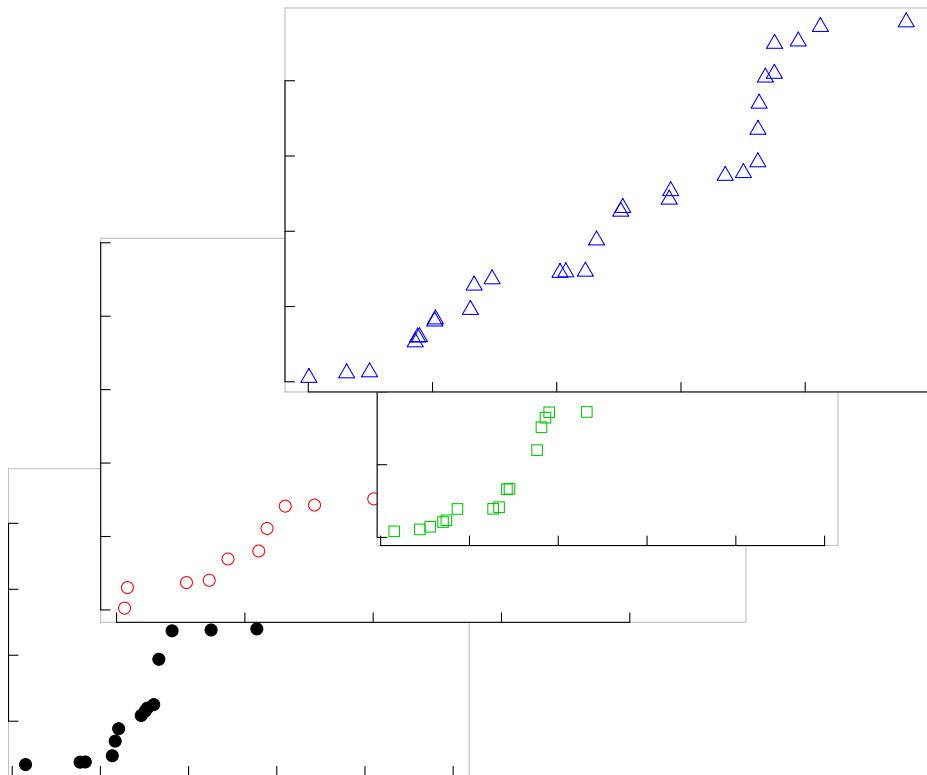


图 B.4: 拆分作图设备屏幕区域的示例: 本图展示了如何用一个矩阵参数控制作图区域在屏幕上的的位置。

B.3 交互操作

R的图形设备可以支持简单的交互式操作，包括支持对鼠标和键盘输入的响应等，这主要由**graphics**和**grDevices**包中的以下几个函数来完成：

B.3.1 获得鼠标位置的坐标

graphics包中的函数*locator()*可以获取当前鼠标在图形坐标系统中的位置坐标，其用法为：

```
1 > usage(locator)
locator(n = 512, type = "n", ...)
```

当我们在图形窗口中创建了一幅图形后，我们可以调用该函数并通过点击鼠标获得坐标。参数n表示鼠标点击的次数，type为点击鼠标之后生成的图形类型，可以边点鼠标边画点或画线，后面的参数为一些图形参数，设定点或线的样式。

该函数在点击鼠标事件结束之后会返回一个包含坐标数据的列表，列表中x和y分别表示横坐标和纵坐标的位置。如下例：

```
1 > plot(1)
2 > # 任意点击三下鼠标
3 > locator(3)
4 > # 返回坐标 (结果取决于用户点击的位置)
5 > # $x
6 > # [1] 0.6121417 0.8046955 1.2561452
7 > # $y
8 > # [1] 0.9562884 0.8710420 1.1648702
```

借助*locator()*返回的坐标数据，我们可以更方便地向图中添加一些图形元素，尤其是图例。因为R的图形设备大多都不支持图形元素的鼠标拖拽，所以事先使用*locator()*在图上“探路”对画图还是很有帮助的。

B.3.2 识别鼠标附近的数据

graphics包中的函数*identify()*可以通过鼠标点击一幅散点图识别鼠标周围的数据点，并且可以给辨识出的数据添加标签，其默认用法如下：

```
1 > usage(identify, "default")
```

```
identify(x, y = NULL, labels = seq_along(x),
  pos = FALSE, n = length(x), plot = TRUE, atpen = FALSE,
  offset = 0.5, tolerance = 0.25, ...)
```

`x`和`y`给出散点图的原始数据，以便鼠标位置坐标与原始数据进行距离匹配，`labels`为数据的标签，默认用数据的序号1、2、3……。

当数据的散点图呈现出异常现象时，如存在离群点等等，我们可以很方便地通过`identify()`函数找出该数据的名称或者序号。

B.3.3 响应鼠标键盘的动作

grDevices包中的函数`getGraphicsEvent()`则提供了更灵活的交互，它可以捕获三种鼠标事件（鼠标按下、鼠标移动和鼠标弹起）和一种键盘事件（键盘输入）。用法如下：

```
1 > usage(getGraphicsEvent)

getGraphicsEvent(prompt = "Waiting for input",
  onMouseDown = NULL, onMouseMove = NULL, onMouseUp = NULL,
  onKeybd = NULL)
```

后面四个参数分别定义了鼠标和键盘事件所对应的行为（通过给定函数实现），具体解释和示例请参见其帮助文件，这里我们只是给出一个例子说明。图B.5演示了鼠标移动的效果：我们在黑色背景的窗口中画了一批数据点，然后通过鼠标的移动在鼠标周围生成一个矩形框，框内的点变成黄色且放大的样式，而框外的点为红色的小点。随着鼠标的移动，矩形框也会在屏幕上移动，从而会框住不同的点。

事实上当今已经有很多类似的交互式图形系统，例如GGobi系统(Cook and Swayne, 2007)、Java的图形系统、OpenGL等，R中也有相应的基于这些系统的函数包如`rggobi`(Temple Lang *et al.*, 2009)、`ipplots`(?)、`rgl`(Adler and Murdoch, 2010)等；感兴趣的读者可以去研究这些图形系统以及函数包。

B.4 分类变量散点图示

我们知道，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

```

1 > xx = runif(100)
2 > yy = runif(100)
3 > mousemove <- function(buttons, x, y) {
4 +   r = 0.2
5 +   idx = (x - r < xx & xx < x + r) & (y - r < yy &
6 +     yy < y + r)
7 +   plot(xx, yy, type = "n")
8 +   rect(-1, -1, 2, 2, col = "black")
9 +   # 鼠标周围画矩形
10 +  rect(x - r, y - r, x + r, y + r, border = "yellow",
11 +    lty = 2)
12 +  points(xx[idx], yy[idx], col = "yellow", cex = 2)
13 +  points(xx[!idx], yy[!idx], col = "red")
14 +  NULL
15 + }
16 > mousedown <- function(buttons, x, y) {
17 +   "Done"
18 + }
19 > par(mar = rep(0, 4), pch = 20)
20 > # plot(xx, yy, type = 'n')
21 > # getGraphicsEvent('Click mouse to exit', onMouseDown = mousedown,
22 > #   onMouseMove = mousemove)
23 > mousemove(, 0.2, 0.3)
24 > arrows(0.22, 0.27, 0.18, 0.33, 0.15, lwd = 4, col = "white")

```

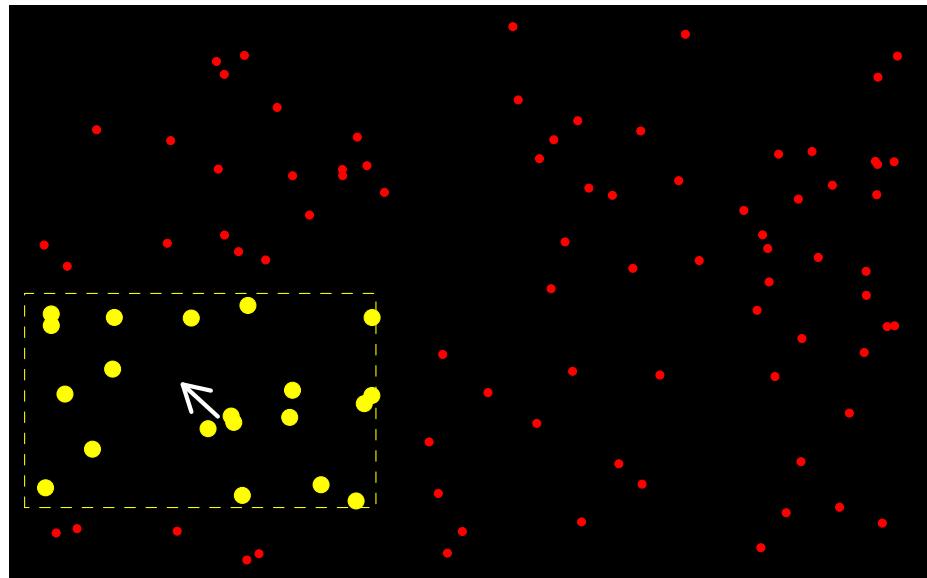


图 B.5: 鼠标在图形窗口中移动的效果图

```

1 > par(mfrow = c(2, 2), mar = c(2.5, 3, 2, 0.1), pch = 20,
2 +     mgp = c(1.5, 0.5, 0), cex.main = 1)
3 > x = sample(rep(1:2, c(12, 18)))
4 > y = rep(1:2, c(18, 12))
5 > plot(x, y, main = "(1) 原始散点图", xlim = c(0.8,
6 +     2.2), ylim = c(0.8, 2.2))
7 > plot(jitter(x), jitter(y), main = "(2) 随机打散后的散点图")
8 > points(x, y, cex = 3)
9 > sunflowerplot(x, y, main = "(3) 向日葵散点图",
10 +      xlim = c(0.8, 2.2), ylim = c(0.8, 2.2))
11 > mosaicplot(table(x, y), main = "(4) 马赛克图")

```

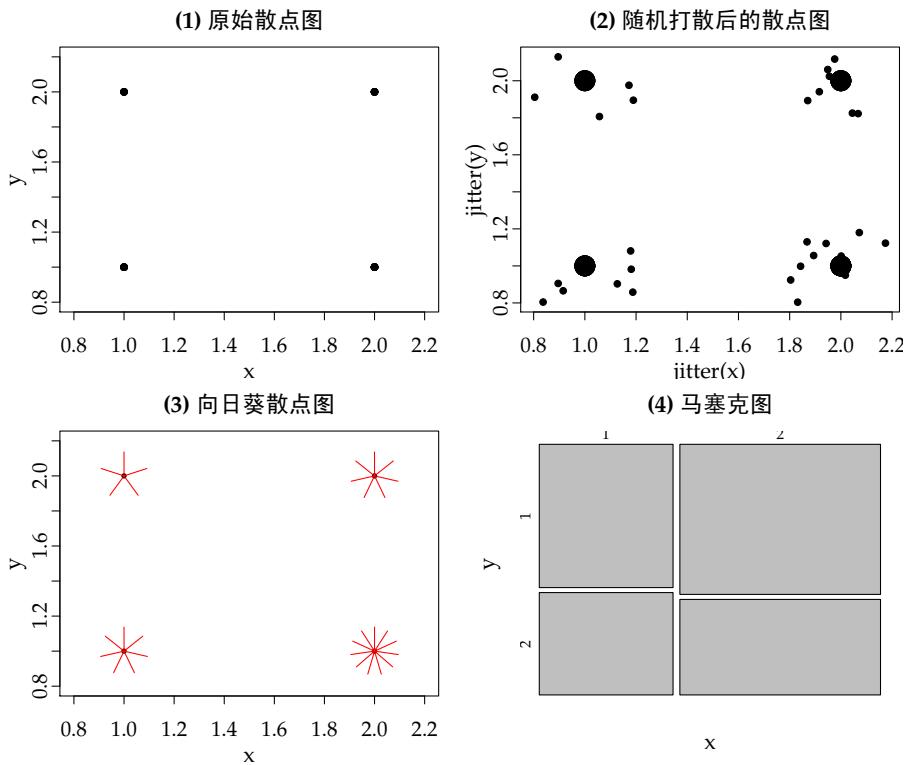


图 B.6: 分类变量的散点图示方法示例: 原始散点图、打散方法、向日葵散点图和马赛克图。

正的频数。我们在第五章中提到过一些分类变量的图示方法，包括关联图（5.6节）、四瓣图（5.13节）和马赛克图（5.16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。这里我们另外介绍两种散点图方法：向日葵散点图和随机打散方法。

B.4.1 向日葵散点图

关于向日葵散点图，在5.25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5.25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B.6(3)给出了一个用向日葵散点图表示分类变量的示例。

B.4.2 随机打散方法

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5.24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B.6(2)给出了一个打散之后的分类变量散点图示例。

B.5 图形设备

利用**grDevices**包中的若干图形设备，我们可以将R的图形输出为各种格式的文件，包括位图文件（BMP、JPEG、PNG、TIFF）和矢量图文件（PDF、EPS）以及TeX或L^AT_EX文件。本书中除了第一章中的历史图形以外，其它大部分图形都是使用**tikzDevice**包(Sharpsteen and Bracken, 2009)中的*tikz()*图形设备生成的（其本质是L^AT_EX）。

基本的图形设备函数有位图设备*bmp()*、*jpeg()*、*png()*和*tiff()*，以及矢量图设备*postscript()*和*pdf()*，打开图形设备之后，所有的R图形都会被生成在该图形设备中，而不会再在窗口中显示，直到图形设备被关闭。详细信息请读者自行查阅相应的帮助文件。

注意位图设备可以支持在图形中使用中文或其它CJK字符，但是在矢量图设备中使用中文字符时则需要设定字体族参数*family*，否则中文不会被

显示出来（例如简体中文应该用`pdf(family = 'GB1')`）。关于非标准字符在图形设备中的使用，请参考Murrell and Ripley (2006)。

最后补充关于图形的一点基础知识：位图文件的图形是由一个个像素点构成的，因此放大之后会变成晶格状从而不太清晰，而矢量图是由内部的数值矢量构成，这些矢量仅仅定义图形元素的始末位置以及其它属性，放大之后清晰度不变。例如一条直线在位图中由若干个点组成，而在矢量图中则是由两个点构成（给定起点和终点），图形放大之后位图的点之间可能会出现空隙，而矢量图随着放大会自动填充两点之间的空隙。为了得到高质量的打印输出，大多数情况下我们建议使用矢量图。

附录 C 统计动画

作者的个人兴趣之一是统计学动画，这里也简要介绍一下动画的基本原理以及它与统计学理论的内在联系。`animation`包(Xie, 2010a)利用R基础图形系统生成了一系列统计学动画；从这个程序包的展示中，读者可以更加深刻认识到对图形元素的控制是一门具有广泛意义的艺术。

附录 D 本书R包

为了配合本书的写作，作者特意编写了一个R包名为**MSG**（Modern Statistical Graphics的缩写），该包目前可从作者主页下载，在本书完成后也会发布到CRAN供读者下载。这里简要介绍一下它包含的函数和数据。

D.1 函数说明

usage() 显示一个函数的用法。该函数可以根据指定的函数名称获取函数的参数列表，并将其以整齐的方式打印出来，如：

```
1 > # plot函数的用法
2 > usage(plot)

plot(x, y, ...)

1 > # plot的默认用法, 即plot.default
2 > usage(plot, "default")

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, ...)

1 > # plot在画经验分布函数的时候的用法, 即plot.ecdf
2 > usage(plot, "ecdf")

plot(x, ..., ylab = "Fn(x)", verticals = FALSE,
      col.01line = "gray70", pch = 19)

1 > # usage自身的用法
2 > usage(usage)

usage(FUN, class = NULL, w = 0.77)
```

D.2 数据说明

PlantCounts 植物数目与海拔高度的数据，共两列，记录了某一海拔高度上植物数目。

参考文献

谢益辉(2008). “统计图形在数据分析中的应用.” In 张波(ed.), *统计学评论*.
中国财政经济出版社.

Adler D (2005). *vioplot: Violin plot*. R package version 0.2, URL <http://wsopuppenkiste.wiso.uni-goettingen.de/~dadler>.

Adler D, Murdoch D (2010). *rgl: 3D visualization device system (OpenGL)*. R package version 0.90, URL <http://CRAN.R-project.org/package=rgl>.

Andrews DF (1972). “Plots of high dimensional data.” *Biometrics*, **28**, 125–136.

Bates D, Maechler M (2010). *Matrix: Sparse and Dense Matrix Classes and Methods*. R package version 0.999375-37, URL <http://CRAN.R-project.org/package=Matrix>.

Becker RA, Chambers JM, Wilks AR (1988). *The New S Language*. Wadsworth & Brooks/Cole.

Bowman AW, Azzalini A (2010). *R package sm: nonparametric smoothing methods (version 2.2-4)*. University of Glasgow, UK and Università di Padova, Italia. URL <http://www.stats.gla.ac.uk/~adrian/sm>.

Breiman, Friedman, Olshen, Stone (1984). *Classification and Regression Trees*. Wadsworth.

Brownrigg R (2010). *maps: Draw Geographical Maps*. R package version 2.1-4. Original S code by Richard A. Becker and Allan R. Wilks. R version by Ray Brownrigg Enhancements by Thomas P

- Minka <surname@stat.cmu.edu>, URL <http://CRAN.R-project.org/package=maps>.
- Chambers JM, Cleveland WS, Kleiner B, Tukey PA (1983). *Graphical Methods for Data Analysis*. Wadsworth & Brooks/Cole.
- Cleveland WS (1979). “Robust locally weighted regression and smoothing scatterplots.” *Journal of American Statistical Association*, **74**, 829–836.
- Cleveland WS (1985). *The Elements of Graphing Data*. Monterey, CA: Wadsworth.
- Cleveland WS (1993). *Visualizing Data*. Hobart Press.
- Cohen A (1980). “On the graphical display of the significant components in a two-way contingency table.” *Communications in Statistics – Theory and Methods*, **A9**, 1025–1041.
- Cook D, Swayne DF (2007). *Interactive and Dynamic Graphics for Data Analysis With R and GGobi*. Springer. ISBN 978-0-387-71761-6.
- Everitt BS, Hothorn T (2006). *A Handbook of Statistical Analyses Using R*. Chapman & Hall/CRC.
- Fox J, with contributions from Liviu Andronic, Ash M, Boye T, Calza S, Chang A, Grosjean P, Heiberger R, Kerns GJ, Lancelot R, Lesnoff M, Ligges U, Messad S, Maechler M, Muenchen R, Murdoch D, Neuwirth E, Putler D, Ripley B, Ristic M, , Wolf P (2009). *Rcmdr: R Commander*. R package version 1.5-4, URL <http://CRAN.R-project.org/package=Rcmdr>.
- Friendly M (1992). “Graphical methods for categorical data.” In *SAS User Group International Conference Proceedings*, volume 17, p. 190–200.
- Friendly M (1994). “A fourfold display for 2 by 2 by k tables.” *Technical Report 217*, York University, Psychology Department.
- Friendly M, Denis DJ (2001). *Milestones in the history of thematic cartography, statistical graphics, and data visualization*. Accessed: March 18, 2010, URL <http://www.math.yorku.ca/SCS/Gallery/milestone/>.

- Hintze JL, Nelson RD (1998). “Violin plots: a box plot-density trace synergism.” *The American Statistician*, **52**(2), 181–4.
- Hofmann H, Theus M (2005). *Interactive graphics for visualizing conditional distributions*. Unpublished Manuscript.
- Hornik K (2009). “The R FAQ.” ISBN 3-900051-08-9, URL <http://CRAN.R-project.org/doc/FAQ/R-FAQ.html>.
- Ihaka R, Gentleman R (1996). “R: A Language for Data Analysis and Graphics.” *Journal of Computational and Graphical Statistics*, **5**(3), 299–314. ISSN 10618600.
- Inselberg A (2007). *Parallel Coordinates: VISUAL Multidimensional Geometry and its Applications*. Springer.
- Kaplan EL, Meier P (1958). “Nonparametric estimation from incomplete observations.” *Journal of the American Statistical Association*, **53**, 457–481.
- Koenker R, Bassett G (1978). “Regression quantiles.” *Econometrica*, **46**, 33–50.
- Leisch F (2002). “Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis.” In W Härdle, B Rönz (eds.), *Compstat 2002 — Proceedings in Computational Statistics*, pp. 575–580. Physica Verlag, Heidelberg. ISBN 3-7908-1517-9, URL <http://www.stat.uni-muenchen.de/~leisch/Sweave>.
- Ligges U, Mächler M (2003). “scatterplot3d – an R Package for Visualizing Multivariate Data.” *Journal of Statistical Software*, **8**, 1–20.
- McGill R, Tukey JW, Larsen WA (1978). “Variations of box plots.” *The American Statistician*, **32**, 12–16.
- Meyer D, Zeileis A, Hornik K (2006). “The Strucplot Framework: Visualizing Multi-Way Contingency Tables with vcd.” *Journal of Statistical Software*, **17**(3), 1–48. URL <http://www.jstatsoft.org/v17/i03/>.

- Meyer D, Zeileis A, Hornik K (2010). *vcd: Visualizing Categorical Data*. R package version 1.2-8, URL <http://CRAN.R-project.org/package=vcd>.
- Murrell P (2005). *R Graphics*. Chapman & Hall/CRC.
- Murrell P, Ripley B (2006). “Non-standard fonts in PostScript and PDF graphics.” *R News*, **6**(2), 41–47. URL http://cran.r-project.org/doc/Rnews/Rnews_2006-2.pdf.
- Neuwirth E (2007). *RColorBrewer: ColorBrewer palettes*. R package version 1.0-2.
- Nightingale F (1858). “Notes on Matters Affecting the Health, Efficiency, and Hospital Administration of the British Army.” *Technical report*.
- Playfair W (1786). *The Commercial and Political Atlas: Representing, by Means of Stained Copper-Plate Charts, the Progress of the Commerce, Revenues, Expenditure and Debts of England during the Whole of the Eighteenth Century*.
- Playfair W (1801). *The statistical breviary*. London: T. Bensley.
- R Development Core Team (2009). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL <http://www.R-project.org>.
- Ripley B (2008). *KernSmooth: Functions for kernel smoothing for Wand & Jones (1995)*. S original by Matt Wand. R port by Brian Ripley. R package version 2.22-22.
- Sarkar D (2010). *lattice: Lattice Graphics*. R package version 0.18-3, URL <http://CRAN.R-project.org/package=lattice>.
- Scott DW (1992). *Multivariate Density Estimation. Theory, Practice and Visualization*. New York: Wiley.
- Sharpsteen C, Bracken C (2009). *tikzDevice: A Device for R Graphics Output in PGF/TikZ Format*. R package version 0.4.8, URL <http://CRAN.R-project.org/package=tikzDevice>.

- Snow G (2010). *TeachingDemos: Demonstrations for teaching and learning.* R package version 2.6, URL <http://CRAN.R-project.org/package=TeachingDemos>.
- Temple Lang D, Swayne D, Wickham H, Lawrence M (2009). *rggobi: Interface between R and GGobi.* R package version 2.1.14, URL <http://CRAN.R-project.org/package=rggobi>.
- Therneau T (2009). *survival: Survival analysis, including penalised likelihood.* R package version 2.35-8. Original R port by Thomas Lumley, URL <http://CRAN.R-project.org/package=survival>.
- Therneau TM, Atkinson B (2010). *rpart: Recursive Partitioning.* R package version 3.1-46. R port by Brian Ripley, URL <http://CRAN.R-project.org/package=rpart>.
- Therneau TM, Grambsch PM (2000). *Modeling Survival Data: Extending the Cox Model.* New York, USA: Springer.
- Tufte ER (1992). *Envisioning Information.* Cheshire, CT, USA: Graphics Press. ISBN 0-961-39211-8.
- Tufte ER (2001). *The Visual Display of Quantitative Information.* 2nd edition. Cheshire, CT, USA: Graphics Press. ISBN 0-9613921-4-2.
- Urbanek S, Wichtrey T (2010). *iplots: iPlots - interactive graphics for R.* R package version 1.1-3, URL <http://www.iPlots.org/>.
- Venables WN, Ripley BD (2002). *Modern Applied Statistics with S.* 4th edition. Springer. ISBN 0-387-95457-0.
- Wainer H, Thissen D (1981). “Graphical Data Analysis.” *Annual Review of Psychology*, **32**(1), 191–241.
- Wand M (2009). *KernSmooth: Functions for kernel smoothing for Wand & Jones (1995).* S original by Matt Wand. R port by Brian Ripley. R package version 2.23-3, URL <http://CRAN.R-project.org/package=KernSmooth>.
- Wickham H (2009). *ggplot2: elegant graphics for data analysis.* Springer New York. ISBN 978-0-387-98140-6. URL <http://had.co.nz/ggplot2/book>.

- Wilkinson L (2005). *The Grammar of Graphics*. 2nd edition. Springer.
- Wolf P, Bielefeld U (2010). *aplpack: Another Plot PACKAGE: stem.leaf, bagplot, faces, spin3R, and some slider functions*. R package version 1.2.3, URL <http://CRAN.R-project.org/package=aplapack>.
- Xie Y (2007). *Visualization of Data and Statistical Models Using R*. Unpublished manuscript, URL http://www.yihui.name/cv/images/Visualization_R_Paper_Yihui.pdf.
- Xie Y (2010a). *animation: Demonstrate Animations in Statistics*. R package version 1.1-0, URL <http://animation.yihui.name>.
- Xie Y (2010b). *MSG: Modern Statistical Graphics*. R package version 0.1-0, URL <http://yihui.name/cn/publication>.
- Zeileis A, Meyer D, Hornik K (2007). “Residual-based Shadings for Visualizing (Conditional) Independence.” *Journal of Computational and Graphical Statistics*, 13(3), 507–525.

索引

- CRAN, 14, 17
- Excel, 11
- graphics包, 59, 170
- grDevices包, 163, 170, 171, 174
- jitter(), 174
- LATEX, 163
- layout(), 165
- par(), 21
- plot(), 30
- R程序包, 13, 14
- R语言, 12, 13
- split.screen(), 166
- S语言, 14
- Wilcoxon秩和检验, 71
- 一页多图, 26, 165
- 中文字符, 174
- 位图, 174
- 低层函数, 33
- 内四分位距, 68
- 函数, 158
- 分类变量, 171
- 列表, 157
- 前景色, 25
- 动画, 177
- 向日葵散点图, 174
- 向量, 149
- 因子, 153
- 图形元素, 12, 33
- 图形参数, 21, 30
- 图形统计分析, 9
- 图形设备, 174
- 坐标轴, 27
- 字体族, 25
- 字体样式, 25
- 密度函数, 62
- 对象, 149
- 帮助系统, 16
- 开源软件, 17
- 循环语句, 160
- 探索性统计分析, 9
- 散点图, 174
- 数学公式, 163
- 数据框, 157
- 数据结构, 15
- 数组, 154
- 条形图, 71
- 泛型函数, 68, 158

点, 27
玫瑰图, 4
直方图, 59
矢量图, 174
矩阵, 154
箱线图, 67
线图, 1
线条宽度, 26
线条样式, 26
经验分布函数, 62
统计之都, 16
缩放倍数, 22

背景色, 22, 32
茎叶图, 63
边框样式, 22
边界宽度, 26

选择语句, 160
面向对象, 12
颜色, 25
饼图, 1
高层函数, 33

后记

2007年2月，我在家读“*A Handbook of Statistical Analysis Using R*”(Everitt and Hothorn, 2006)，书中读完前八章之后，萌发了写一篇关于统计图形综述的念头，于是用`LATEX`匆匆写了几页，但几天后发现这不是一篇综述能讲清楚的事情，于是改为写一本三十来页的小册子，依次介绍R语言中的那些基本的统计图形，然而几天后发现这也不是一本小册子能装得下的内容，因为国内统计界对R语言的了解相对较少，所以有必要把统计图形放在整个R的系统下来讲述，于是在几经折腾之后，一篇论文夭折了，一本书诞生了。

如我在1.5小节中提到的，国内统计图形应用的现状并不是很理想，然而，在这本书中，我结合大量的数据实例给出了各式各样统计图形的展示，其中有相当一部分图形在我们的论文、书籍或报告中并不常见，原因是什么我并不太清楚，也许受软件束缚，也许并不知道有这些图形的存在，无论如何，本书作为一本指南读物，对软件实现以及图形种类都做出了介绍，对统计图形的问题也算是提供了一种解决方案，不过我的主要目的并不在于单纯介绍图形，而是想借现有的图形思想启发读者，一方面使读者能够明白图形的基本构造方法，这样可以让我们容易就能借“它山之石”为自己所用，这种功夫如同《天龙八部》中慕容世家的“以彼之道、还施彼身”，吸收别家的好功夫（好图形），然后施展到自己的领域中去（当然用不着“施彼身”）；另一方面希望读者通过观摩各种图形的思想，在自己的数据分析过程中找到图示的灵感，用更巧妙的方式揭示信息。

国外的统计图形和可视化水平相对国内来说要领先许多，我于2008年6月在德国不来梅的Jacobs大学参加了一个统计图形会议（Data Viz VI），见到了统计图形领域的一些“长老”和新秀，如加拿大York大学的Michael Friendly，这位红脸老头在统计图形上作了大量工作，尤其是分类数据的可视化，以及统计图形历史的总结等等，我正是从他那里了解到了统计图

形的一些历史，包括本书第一章介绍的图形；参会的还有AT&T统计研究部门的Deborah Swayne、Iowa州立大学的Dianne Cook（我的博士导师）、沃顿商学院的Andreas Buja（他们三人发起的GGobi系统作为一种交互式的高维数据展示工具在统计图形届已经是颇有名气）、Systat公司的Leland Wilkinson（“The Grammar of Graphics”的作者），同时我也见到了一些充满潜力的年轻学者如Hadley Wickham、Michael Lawrence等；通过与这些人的接触，我进一步意识到了国内在统计图形方面和国外的差距，回来之后更加坚定了我完成这样一本入门级统计图形书籍的信念。

最后，我要感谢我的父母和亲人们在2008年以来每个暑假给我提供了一个绝佳的写作环境，让我专心完成了本书核心章节的写作；感谢吴喜之老师将R这套工具引入统计学院的课堂，以及王星老师在统计计算和非参数统计课堂上对R的介绍，没有他们的努力，我也许至今还没有踏进R的大门；感谢我的导师赵彦云老师在我的硕博学习期间给我的各种指导，包括论文的写作以及人生的规划；感谢“统计之都”网站的会员们在COS论坛上S-Plus & R版块和我的交流，他们的问题也使我意识到了图形知识的需求；感谢本书写作期间所有给我提供过帮助的人们。