

④ 黃湘云

# R 语言学习笔记

黄湘云

2022-06-09

## 目录

<b>欢迎</b>	<b>1</b>	<b>3.4 从数据库导入 . . . . .</b>	<b>28</b>
授权说明 . . . . .	1	3.4.1 PostgreSQL . . . . .	29
运行信息 . . . . .	1	3.4.2 MySQL . . . . .	32
3.4.3 Spark . . . . .	33		
<b>第一章 前言</b>	<b>2</b>	<b>3.5 批量导入数据 . . . . .</b>	<b>37</b>
1.1 语言抉择 . . . . .	2	3.6 批量导出数据 . . . . .	38
1.2 数据科学 . . . . .	4	3.7 导出数据 . . . . .	40
1.3 获取帮助 . . . . .	5	3.7.1 导出运行结果 . . . . .	40
1.4 写作环境 . . . . .	5	3.7.2 导出数据对象 . . . . .	41
1.5 记号约定 . . . . .	6	3.8 Spark 与 R 语言 . . . . .	44
1.6 复现环境 . . . . .	7	3.8.1 sparklyr . . . . .	44
1.7 如何发问 . . . . .	7	3.8.2 SparkR . . . . .	48
1.8 作者简介 . . . . .	8	3.9 数据库与 R 语言 . . . . .	49
		3.10 批量读取 csv 文件 . . . . .	50
<b>第一部分 数据整理</b>	<b>9</b>	3.11 批量导出 xlsx 文件 . . . . .	52
<b>介绍</b>	<b>10</b>	3.12 运行环境 . . . . .	52
<b>第二章 数据结构</b>	<b>11</b>	<b>第四章 字符串操作</b>	<b>54</b>
2.1 类型 . . . . .	11	4.1 字符数统计 . . . . .	54
2.2 字符 . . . . .	13	4.2 字符串翻译 . . . . .	55
2.3 向量 . . . . .	13	4.3 字符串连接 . . . . .	56
2.4 矩阵 . . . . .	13	4.4 字符串拆分 . . . . .	56
2.5 数组 . . . . .	13	4.5 字符串匹配 . . . . .	58
2.6 表达式 . . . . .	13	4.6 字符串查询 . . . . .	61
2.7 列表 . . . . .	13	4.7 字符串替换 . . . . .	67
2.8 日期 . . . . .	14	4.8 字符串提取 . . . . .	68
2.9 空值 . . . . .	19	4.9 命名捕捉 . . . . .	69
		4.10 精确匹配 . . . . .	71
<b>第三章 数据搬运</b>	<b>20</b>	4.11 模糊匹配 . . . . .	71
3.1 导入数据 . . . . .	20	4.12 高级的替换 . . . . .	74
3.1.1 scan . . . . .	21	4.13 高级的提取 . . . . .	75
3.1.2 read.table . . . . .	22	4.14 其它操作 . . . . .	76
3.1.3 readLines . . . . .	24	4.14.1 strwrap . . . . .	76
3.1.4 readRDS . . . . .	25	4.14.2 trim . . . . .	82
3.2 其它数据格式 . . . . .	26	4.14.3 strrep . . . . .	82
3.3 导入大数据集 . . . . .	28	4.14.4 trimws . . . . .	83



4.14.5 <code>tolower</code> . . . . .	84	6.23.1 循环合并 . . . . .	150
4.15 字符串加密 . . . . .	84	6.23.2 分组计数 . . . . .	150
4.16 处理性能 . . . . .	85	6.23.3 分组抽样 . . . . .	150
4.17 网络爬虫 . . . . .	85	6.23.4 分组排序 . . . . .	151
4.18 文本挖掘 . . . . .	86	<b>第七章 高级数据操作</b>	155
4.19 运行环境 . . . . .	86	7.1 基础介绍 . . . . .	155
<b>第五章 正则表达式</b>	<b>87</b>	7.1.1 过滤 . . . . .	158
5.1 字符常量 . . . . .	88	7.1.2 变换 . . . . .	159
5.2 软件环境 . . . . .	89	7.1.3 聚合 . . . . .	160
5.3 基本概念 . . . . .	92	7.1.4 命名 . . . . .	161
5.4 字符串匹配 . . . . .	93	7.1.5 排序 . . . . .	162
5.5 级联表达式 . . . . .	93	7.1.6 变形 . . . . .	163
5.6 反向引用 . . . . .	93	7.1.7 分组 . . . . .	165
5.7 命名捕捉 . . . . .	94	7.1.8 合并 . . . . .	166
5.8 表达式注释 . . . . .	95	7.2 高频操作 . . . . .	167
<b>第六章 数据操作</b>	<b>97</b>	7.2.1 选择多列 . . . . .	168
6.1 查看数据 . . . . .	98	7.2.2 过滤多行 . . . . .	169
6.2 提取子集 . . . . .	100	7.2.3 去重多行 . . . . .	170
6.3 数据重塑 . . . . .	103	7.2.4 合并操作 . . . . .	171
6.4 数据转换 . . . . .	106	7.2.5 新添多列 . . . . .	171
6.5 按列排序 . . . . .	106	7.2.6 删除多列 . . . . .	171
6.6 数据拆分 . . . . .	108	7.2.7 筛选多列 . . . . .	172
6.7 数据合并 . . . . .	110	7.2.8 修改多列类型 . . . . .	172
6.8 数据去重 . . . . .	113	7.2.9 取每组第一行 . . . . .	172
6.9 数据缺失 . . . . .	115	7.2.10 计算环比同比 . . . . .	173
6.10 数据聚合 . . . . .	117	7.2.11 合并多个数据框 . . . . .	174
6.11 表格统计 . . . . .	125	7.2.12 分组聚合多个指标 . . . . .	177
6.12 索引访问 . . . . .	128	7.2.13 重命名多个列 . . . . .	179
6.13 多维数组 . . . . .	128	7.2.14 对多个列依次排序 . . . . .	179
6.14 其它操作 . . . . .	130	7.2.15 重排多个列的位置 . . . . .	180
6.14.1 列表属性 . . . . .	130	7.2.16 整理回归结果 . . . . .	180
6.14.2 堆叠向量 . . . . .	131	7.2.17 <code>:=</code> 和 <code>.()</code> . . . . .	181
6.14.3 属性转化 . . . . .	132	7.2.18 去掉含有缺失值的记录 . . . . .	183
6.14.4 绑定环境 . . . . .	133	7.2.19 集合操作 . . . . .	183
6.14.5 数据环境 . . . . .	133	7.2.20 对数值向量按既定分组计数 . . . . .	184
6.15 <code>apply</code> 族 . . . . .	137	7.2.21 分组排序 . . . . .	185
6.16 <code>with</code> 选项 . . . . .	141	7.2.22 分组获取 Top 值 . . . . .	185
6.17 分组聚合 . . . . .	142	7.2.23 分组抽样 . . . . .	186
6.18 合并操作 . . . . .	145	7.2.24 分组计算分位数 . . . . .	186
6.19 长宽转换 . . . . .	146	7.2.25 计算日粒度的 DoD/WoW/MoM/YoY . . . . .	187
6.20 对符合条件的列操作 . . . . .	147	7.3 运行环境 . . . . .	188
6.21 CASE WHEN 和 <code>fcase</code> . . . . .	148	<b>第八章 并行化操作</b>	189
6.22 数据操作实战 . . . . .	149	8.1 <code>apply</code> . . . . .	189
6.23 高频数据操作 . . . . .	149	8.2 <code>MapReduce</code> . . . . .	189



8.3 parallel . . . . .	190	10.2.5 QQ 图 . . . . .	270
8.4 Rmpi . . . . .	190	10.2.6 时序图 . . . . .	272
8.5 gpuR . . . . .	191	10.2.7 饼图 . . . . .	272
8.6 运行环境 . . . . .	192	10.2.8 茎叶图 . . . . .	274
<b>第九章 净土化操作</b>	<b>194</b>	10.2.9 散点图 . . . . .	274
9.1 常用操作 . . . . .	194	10.2.10 抖动图 . . . . .	282
9.1.1 查看 . . . . .	194	10.2.11 箱线图 . . . . .	284
9.1.2 筛选 . . . . .	195	10.2.12 残差图 . . . . .	288
9.1.3 排序 . . . . .	196	10.2.13 提琴图 . . . . .	288
9.1.4 聚合 . . . . .	196	10.2.14 轮廓图 . . . . .	288
9.1.5 合并 . . . . .	197	10.2.15 折线图 . . . . .	288
9.1.6 变换 . . . . .	198	10.2.16 函数图 . . . . .	289
9.1.7 去重 . . . . .	198	10.2.17 马赛克图 . . . . .	291
9.2 高频问题 . . . . .	199	10.2.18 点图 . . . . .	291
9.2.1 初始化数据框 . . . . .	200	10.2.19 矩阵图 . . . . .	291
9.2.2 移除缺失记录 . . . . .	201	10.2.20 雷达图 . . . . .	293
9.2.3 数据类型转化 . . . . .	201	10.2.21 玫瑰图 . . . . .	293
9.2.4 跨列分组求和 . . . . .	201	10.2.22 透视图 . . . . .	295
9.3 管道操作 . . . . .	202	<b>10.3 棚格统计图形</b> . . . . .	296
<b>第二部分 统计图形</b>	<b>204</b>	10.3.1 箱线图 . . . . .	296
<b>介绍</b>	<b>205</b>	10.3.2 折线图 . . . . .	297
<b>第十章 图形基础</b>	<b>206</b>	10.3.3 平滑图 . . . . .	305
10.1 绘图基本要素 . . . . .	206	10.3.4 点图 . . . . .	308
10.1.1 点线 . . . . .	206	10.3.5 阶梯图 . . . . .	309
10.1.2 区域 . . . . .	212	10.3.6 分面图 . . . . .	310
10.1.3 参考线 . . . . .	218	10.3.7 等高线图 . . . . .	311
10.1.4 坐标轴 . . . . .	219	10.3.8 透视图 . . . . .	312
10.1.5 刻度线 . . . . .	229	10.3.9 聚类图 . . . . .	312
10.1.6 标题 . . . . .	231	10.4 运行环境 . . . . .	315
10.1.7 注释 . . . . .	231	<b>第十一章 数据可视化</b>	<b>317</b>
10.1.8 图例 . . . . .	235	11.1 元素 . . . . .	319
10.1.9 边空 . . . . .	240	11.1.1 图层 . . . . .	319
10.1.10 图层 . . . . .	248	11.1.2 标签 . . . . .	320
10.1.11 布局 . . . . .	250	11.1.3 注释 . . . . .	321
10.1.12 组合 . . . . .	251	11.1.4 刻度 . . . . .	324
10.1.13 分屏 . . . . .	254	11.1.5 图例 . . . . .	325
10.1.14 交互 . . . . .	255	11.1.6 坐标系 . . . . .	325
10.2 基础统计图形 . . . . .	255	11.1.7 坐标轴 . . . . .	325
10.2.1 条形图 . . . . .	255	11.1.8 配色 . . . . .	326
10.2.2 直方图 . . . . .	262	11.1.9 主题 . . . . .	327
10.2.3 密度图 . . . . .	266	11.1.10 布局 . . . . .	329
10.2.4 经验图 . . . . .	270	11.2 字体 . . . . .	330
		11.2.1 系统字体 . . . . .	330
		11.2.2 思源字体 . . . . .	335

11.2.3 数学字体 . . . . .	336	11.4.36 龙卷风图 . . . . .	453	
11.2.4 TikZ 设备 . . . . .	339	11.4.37 聚类图 . . . . .	454	
11.2.5 漫画字体 . . . . .	340	11.4.38 主成分图 . . . . .	455	
11.2.6 表情字体 . . . . .	341	11.4.39 组合图 . . . . .	456	
11.3 配色 . . . . .	342	11.4.40 动态图 . . . . .	458	
11.3.1 调色板 . . . . .	345	<b>第十二章 交互图形</b> <span style="float: right;">462</span>		
11.3.2 颜色模式 . . . . .	359	12.1 散点图 . . . . .	464	
11.3.3 LaTeX 配色 . . . . .	365	12.2 条形图 . . . . .	465	
11.3.4 ggplot2 配色 . . . . .	366	12.3 折线图 . . . . .	466	
11.4 图库 . . . . .	366	12.4 双轴图 . . . . .	466	
11.4.1 饼图 . . . . .	366	12.5 直方图 . . . . .	467	
11.4.2 地图 . . . . .	369	12.6 箱线图 . . . . .	467	
11.4.3 热图 . . . . .	371	12.7 提琴图 . . . . .	468	
11.4.4 散点图 . . . . .	374	12.8 气泡图 . . . . .	468	
11.4.5 条形图 . . . . .	385	12.9 曲线图 . . . . .	469	
11.4.6 直方图 . . . . .	399	12.10 堆积图 . . . . .	469	
11.4.7 箱线图 . . . . .	401	12.11 热力图 . . . . .	470	
11.4.8 函数图 . . . . .	406	12.12 地图 I . . . . .	470	
11.4.9 密度图 . . . . .	406	12.13 拟合图 . . . . .	472	
11.4.10 提琴图 . . . . .	414	12.14 轨迹图 . . . . .	473	
11.4.11 抖动图 . . . . .	416	12.15 三维图 (plotly) . . . . .	474	
11.4.12 蜂群图 . . . . .	425	12.16 甘特图 . . . . .	474	
11.4.13 玫瑰图 . . . . .	425	12.17 帕雷托图 . . . . .	476	
11.4.14 瓦片图 . . . . .	428	12.18 时间线 . . . . .	477	
11.4.15 日历图 . . . . .	429	12.19 漏斗图 . . . . .	477	
11.4.16 岭线图 . . . . .	432	12.20 雷达图 . . . . .	478	
11.4.17 椭圆图 . . . . .	433	12.21 瀑布图 . . . . .	478	
11.4.18 Q-Q 图 . . . . .	435	12.22 树状图 . . . . .	480	
11.4.19 包络图 . . . . .	435	12.23 旭日图 . . . . .	480	
11.4.20 拟合图 . . . . .	438	12.24 调色板 . . . . .	480	
11.4.21 地形图 . . . . .	439	12.25 导出静态图形 . . . . .	481	
11.4.22 树状图 . . . . .	440	12.26 静态图形转交互图形 . . . . .	481	
11.4.23 留存图 . . . . .	443	12.27 地图 II . . . . .	482	
11.4.24 瀑布图 . . . . .	444	12.28 动画 . . . . .	485	
11.4.25 桑基图 . . . . .	445	12.29 网络图 . . . . .	487	
11.4.26 词云图 . . . . .	446	12.29.1 networkD3 . . . . .	487	
11.4.27 甘特图 . . . . .	447	12.29.2 visNetwork . . . . .	487	
11.4.28 马赛克图 . . . . .	447	12.29.3 r2d3 . . . . .	488	
11.4.29 凹凸图 . . . . .	447	12.30 运行环境 . . . . .	489	
11.4.30 水流图 . . . . .	449	<b>第十三章 数值优化</b> <span style="float: right;">491</span>		
11.4.31 时间线 . . . . .	450	13.1 线性规划 . . . . .	493	
11.4.32 三元图 . . . . .	452	13.2 整数规划 . . . . .	494	
11.4.33 向量场图 . . . . .	452	13.2.1 一般整数规划 . . . . .	494	
11.4.34 四象限图 . . . . .	452	13.2.2 0-1 整数规划 . . . . .	494	
11.4.35 韦恩图 . . . . .	453			



13.2.3 混合整数规划 . . . . .	495	A.2 创建文件夹 . . . . .	555
13.3 二次规划 . . . . .	497	A.3 移动文件 . . . . .	555
13.3.1 凸二次规划 . . . . .	497	A.4 查看文件大小 . . . . .	556
13.3.2 半正定二次优化 . . . . .	500	A.5 终端模拟器 . . . . .	556
13.4 非线性规划 . . . . .	501	A.6 压缩和解压缩 . . . . .	558
13.4.1 一元非线性优化 . . . . .	501	A.7 从仓库安装 R . . . . .	558
13.4.2 多元非线性无约束优化 . . .	502	A.7.1 Ubuntu . . . . .	558
13.4.3 多元非线性约束优化 . . . . .	521	A.7.2 CentOS . . . . .	559
13.5 非线性方程 . . . . .	537	A.8 源码安装 . . . . .	559
13.5.1 一元非线性方程 . . . . .	537	A.8.1 Ubuntu . . . . .	559
13.5.2 非线性方程组 . . . . .	537	A.8.2 CentOS . . . . .	559
13.6 多目标规划 . . . . .	541	A.9 忍者安装 . . . . .	562
13.7 经典优化问题 . . . . .	542	A.10 配置 . . . . .	563
13.8 回归与优化 . . . . .	542	A.10.1 初始会话 .Rprofile . . . . .	563
13.9 对数似然 . . . . .	544	A.10.2 环境变量 .Renviron . . . . .	563
13.10 微分方程 . . . . .	545	A.10.3 编译选项 Makevars . . . . .	563
13.10.1 常微分方程 . . . . .	546	A.11 命令行参数 . . . . .	563
13.10.2 偏微分方程 . . . . .	546	A.12 从源码安装 R . . . . .	564
13.10.3 延迟微分方程 . . . . .	552	A.13 安装软件 . . . . .	565
13.10.4 随机微分方程 . . . . .	552	A.14 安装 R 包 . . . . .	566
13.11 运行环境 . . . . .	552	A.15 软件包管理器 . . . . .	569
<b>附录 A 命令行操作</b>	<b>554</b>	A.15.1 dnf . . . . .	569
A.1 查看文件 . . . . .	554	A.15.2 apt . . . . .	570

## 插图

## 表格

## 欢迎

警告

Book in early development. Planned release in 202X.

## 授权说明

警告

本书采用 [知识共享署名-非商业性使用-禁止演绎 4.0 国际许可协议](#) 许可, 请君自重, 别没事儿拿去传个什么新浪爱问、百度文库以及 XX 经济论坛, 项目中代码使用 [MIT 协议](#) 开源



## 运行信息

书籍在 R version 4.2.0 (2022-04-22) 下编译, Pandoc 版本 2.16.2, 最新一次编译发生在 2022-06-09 20:04:53。

# 第一章 前言

荃者所以在鱼，得鱼而忘荃；蹄者所以在兔，得兔而忘蹄；言者所以在意，得意而忘言。吾安得夫忘言之人而与之言哉！

— 摘自《庄子·杂篇·物》

庄子谈学习，余深以为然，故引之。

The fish trap exists because of the fish; once you've gotten the fish, you can forget the trap. The rabbit snare exists because of the rabbit; once you've gotten the rabbit, you can forget the snare. Words exist because of meaning; once you've gotten the meaning, you can forget the words. Where can I find a man who has forgotten words so I can have a word with him?

<sup>1</sup>

— Chuang Tzu

## 1.1 语言抉择

行业内可以做统计分析和建模的软件汗牛充栋，比较顶级的收费产品有 SAS 和 SPSS，在科学计算领域的 Matlab 和 Mathematica 也有相当强的统计功能，而用户基数最大的是微软 Excel，抛开微软公司的商业手段不说，Excel 的市场份额却是既成事实。Brian D. Ripley 20 多年前的一句话很有意思，放在当下也是适用的。

Let's not kid ourselves: the most widely used piece of software for statistics is Excel.

— Brian D. Ripley [Ripley, 2002]

有鉴于 Excel 在人文、社会、经济和管理等领域的影响力，熟悉 R 语言的人把它看作超级收费版的 Excel，这实在是一点也不过分。事实上，我司就是一个很好的明证，一个在线教育类的互联网公司，各大业务部门都在使用 Excel 作为主要的数据分析工具。然而，Excel 的不足也十分突出，工作过程无法保存和重复利用，Excel 也不是数据库，数据集稍大，操作起来愈发困难，对于复杂的展示，需要借助内嵌的 VBA，由于缺乏版本控制，随着时间的推移，几乎不可维护。所以，我们还是放弃 Excel 吧，Jenny Bryan 更在 2016 年国际 R 语言大会上的直截了当地喊出了这句话<sup>2</sup>。Nathan Stephens 对 Excel 的缺陷不足做了全面的总结<sup>3</sup>。

Some people familiar with R describe it as a supercharged version of Microsoft's Excel spreadsheet software.

<sup>1</sup>译文摘自 Eric D. Kolaczyk

<sup>2</sup><https://channel9.msdn.com/Events/useR-international-R-User-conference/useR2016/jailbreakr-Get-out-of-Excel-free>

<sup>3</sup><https://resources.rstudio.com/wistia-rstudio-essentials-2/how-to-excel-without-using-excel>

— Ashlee Vance <sup>4</sup>

另一方面，我们谈谈开源领域的佼佼者 — R (<https://cran.r-project.org/>)，Python (<https://www.python.org/>) 和 Octave (<http://www.gnu.org/software/octave/>)。Python 号称万能的胶水语言，从系统运维到深度学习都有它的广泛存在，它被各大主流 Linux 系统内置，语言风格上更接近于基数庞大的开发人员，形成了强大的生态平台。Octave 号称是可以替代 Matlab 的科学计算软件，在兼容 Matlab 的方面确实做的很不错，然而，根据 Julia 官网给出的各大编程语言的测试 <https://julialang.org/benchmarks/>，性能上不能相提并论。

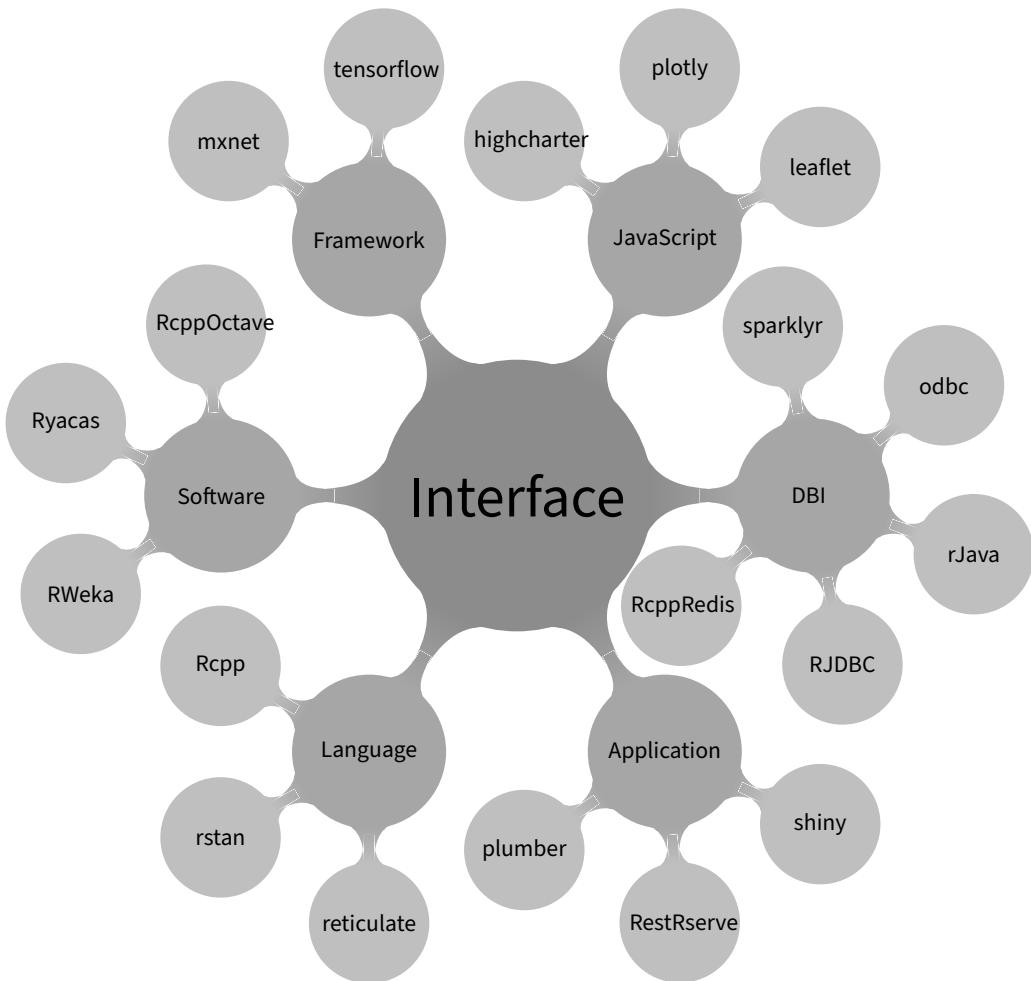


图 1.1: R 语言扩展生态系统

R 提供了丰富的图形接口，包括 Tcl/Tk , Gtk, Shiny 等，以及基于它们的衍生品 rattle ([RGtk2](#))、Rcmdr ([tcl/tk](#))、[radiant](#) ([shiny](#))。更多底层介绍，见 John Chamber 的著作《Extending R》。

[Eviews](#) 时间序列和计量经济模型，相比于 Eviews, [Stata](#) 是综合型的统计软件，提供丰富的统计模型，[SPSS](#) 同 Stata 类似，[Minitab](#)，[JASP](#) 是开源的软件，[Octave](#) 是对标 Matlab 的工程计算软件，有丰富的优化功能，是一门编程语言兼软件，为求解统计模型的参数提供了广泛的基础能力。[Tableau](#) 提供强大的分析和打造数据产品的能力。TikZ 在绘制示意图方面有很大优势，特别是示意图里包含数学公式，这更是 LaTeX 所擅长的方面。

JASP <https://jasp-stats.org> 是一款免费的统计软件，源代码托管在 Github 上 <https://github.com/jasp-stats/jasp-desktop>，主要由阿姆斯特丹大学 E. J. Wagenmakers 教授 <https://www.ejwagenmakers.com/>

<sup>4</sup><https://www.nytimes.com/2009/01/07/technology/business-computing/07program.html>



领导的团队维护开发，实现了很多贝叶斯和频率统计方法，相似的图形用户界面使得 JASP 可以作为 SPSS 的替代，目前实现的功能见 <https://jasp-stats.org/current-functionality/>，统计方法见博客 <https://www.bayesianspectacles.org/>。

国内可视化分析平台，比如 **hiplot** 基于 R 语言实现可视化分析，各类图形的介绍见[文档](#)，极大地降低数据分析人员探索分析的门槛，节省了时间，同时非专业内的人也可借助其完成分析探索的过程，只需明白各类图形的含义即可。美团也建设了自己的可视化分析平台帮助运营人员，详见[文档](#)

**Patrick Burns** 收集整理了 R 语言中奇葩的现象，写成 **The R Inferno** 直译过来就是《R 之炼狱》。这些奇葩的怪现象可以看做是 R 风格的一部分，对于编程人员来说就是一些建议和技巧，参考之可以避开某些坑。Paul E. Johnson 整理了一份真正的 R 语言建议，记录了他自己从 SAS 转换到 R 的过程中遇到的各种问题 <http://pj.freefaculty.org/R/Rtips.html>。Michail Tsagris 和 Manos Papadakis 也收集了 70 多条 R 编程的技巧和建议，力求以更加 R 范地将语言特性发挥到极致 [[Tsagris and Papadakis, 2018](#)], Martin Mächler 提供了一份 [Good Practices in R Programming](#)。Python 社区广泛流传着 Tim Peters 的《Python 之禅》，它已经整合进每一版 Python 软件中，只需在 Python 控制台里执行 `import this` 可以获得。

1. Beautiful is better than ugly.
2. Explicit is better than implicit.
3. Simple is better than complex.
4. Complex is better than complicated.
5. Flat is better than nested.
6. Sparse is better than dense.
7. Readability counts.
8. Special cases aren't special enough to break the rules.
9. Although practicality beats purity.
10. Errors should never pass silently.
11. Unless explicitly silenced.
12. In the face of ambiguity, refuse the temptation to guess.
13. There should be one- and preferably only one -obvious way to do it.
14. Although that way may not be obvious at first unless you're Dutch.
15. Now is better than never.
16. Although never is often better than *right* now.
17. If the implementation is hard to explain, it's a bad idea.
18. If the implementation is easy to explain, it may be a good idea.
19. Namespaces are one honking great idea – let's do more of those!

— The Zen of Python

总之，编程语言到一定境界都是殊途同归的，对美的认识也是趋同的，道理更是相通的，Python 社区的 Pandas <https://github.com/pandas-dev/pandas> 和 Matplotlib <https://github.com/matplotlib/matplotlib> 也有数据框和图形语法的影子。Pandas <https://github.com/pandas-dev/pandas> 明确说了要提供与 `data.frame` 类似的数据结构和对应统计函数等，而 Matplotlib 偷了 ggplot2 绘图样式 [https://matplotlib.org/3.2.2/gallery/style\\_sheets/ggplot.html](https://matplotlib.org/3.2.2/gallery/style_sheets/ggplot.html)。

## 1.2 数据科学

John M. Chambers 谈了数据科学的源起以及和 S、R 语言的渊源 [[Chambers, 2020](#)]。

### 1.3 获取帮助

R 社区提供了丰富的帮助资源，可以在 R 官网搜集的高频问题 <https://cran.r-project.org/faqs.html> 中查找，也可在线搜索 <https://cran.r-project.org/search.html> 或 <https://rseek.org/>，更多获取帮助方式见 <https://www.r-project.org/help.html>。客栈网问题以标签分类，比如 `r-plotly`、`r-markdown`、`data.table` 和 `ggplot2`，还可以关注一些活跃的社区大佬，比如 [谢益辉](#)。

### 1.4 写作环境

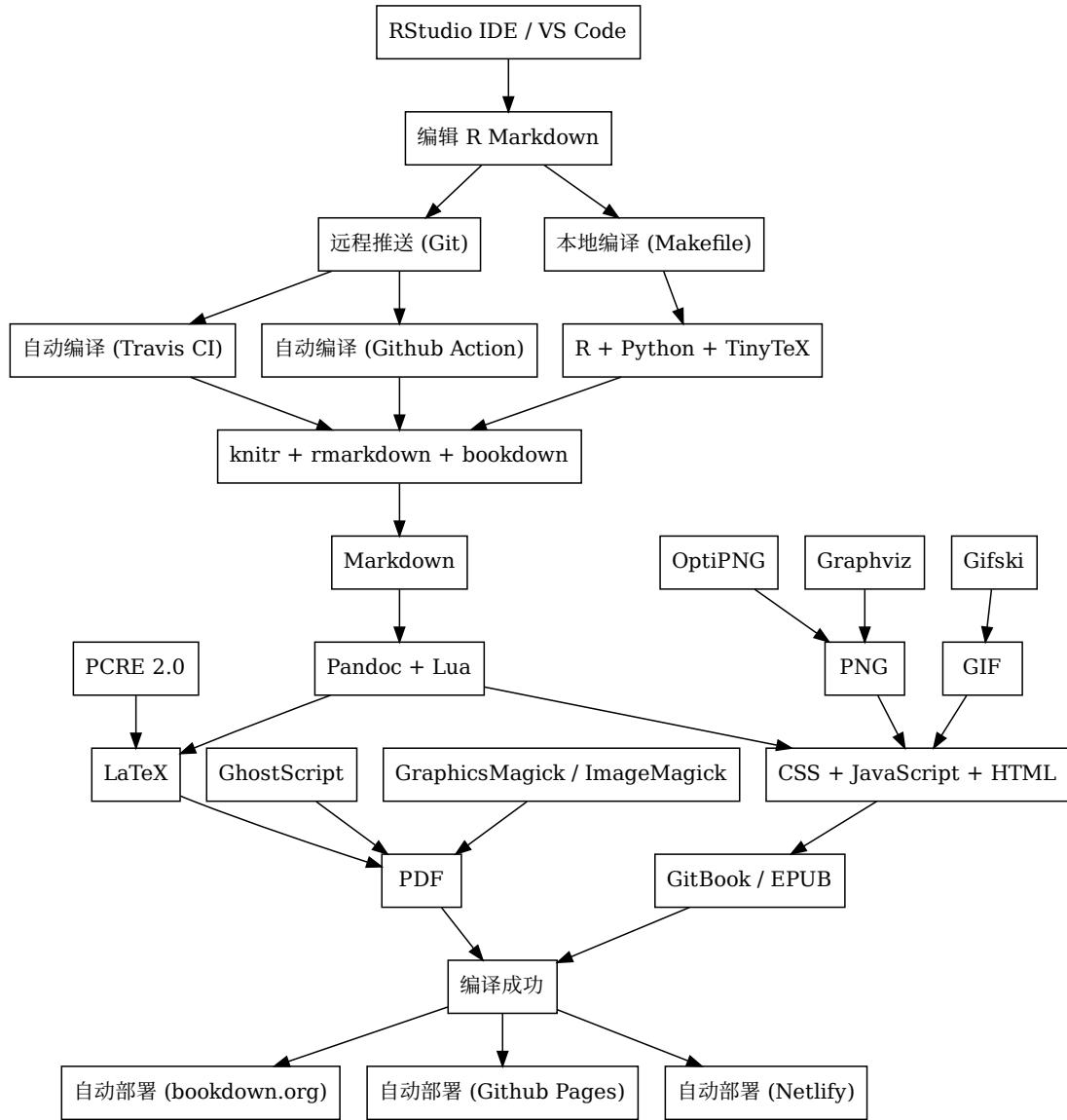


图 1.2: 书籍项目架构图

本书 R Markdown 源文件托管在 Github 仓库里，本地使用 RStudio IDE 编辑，bookdown 组织各个章节的 Rmd 文件和输出格式，使用 Git 进行版本控制。每次提交修改到 Github 上都会触发 Travis 自动编译书

籍，将一系列 Rmd 文件经 knitr 调用 R 解释器执行里面的代码块，并将输出结果返回，Pandoc 将 Rmd 文件转化为 md、html 或者 tex 文件。若想输出 pdf 文件，还需要准备 TeX 排版环境，最后使用 Netlify 托管书籍网站，和 Travis 一起实现连续部署，使得每次修改都会同步到网站。最近一次编译时间 2022 年 06 月 09 日 12 时 04 分 54 秒，本书用 R version 4.2.0 (2022-04-22) 编译，完整运行环境如下：

```
xfun::session_info(packages = c(  
  "knitr", "rmarkdown", "bookdown"  
, dependencies = FALSE)
```

```
## R version 4.2.0 (2022-04-22)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.4 LTS
##
## Locale:
##  LC_CTYPE=en_US.UTF-8          LC_NUMERIC=C
##  LC_TIME=en_US.UTF-8          LC_COLLATE=en_US.UTF-8
##  LC_MONETARY=en_US.UTF-8        LC_MESSAGES=en_US.UTF-8
##  LC_PAPER=en_US.UTF-8          LC_NAME=C
##  LC_ADDRESS=C                  LC_TELEPHONE=C
##  LC_MEASUREMENT=en_US.UTF-8   LC_IDENTIFICATION=C
##
## Package version:
##  bookdown_0.26  knitr_1.39      rmarkdown_2.14
##
## Pandoc version: 2.16.2
```

借助 `bookdown` [Xie, 2016] 可以将 Rmd 文件组织起来, `rmarkdown` [Allaire et al., 2021] 和 `knitr` [Xie, 2015] 将源文件编译成 Markdown 文件, `Pandoc` 将 Markdown 文件转化成 HTML 和 TeX 文件, `TinyTeX` [Xie, 2019] 可以将 TeX 文件进一步编译成 PDF 文档, 书中大量的图形在用 `ggplot2` 包制作 [Wickham, 2016], 而统计理论相关的示意图用 Base R 创作。

## 提示

得益于 Github Action 提供的测试服务, Github Pages、Bookdown 和 Netlify 提供的部署服务, 鉴于国内的网络环境, 本书托管在三个地方, 分别是 <https://xiangyunhuang.github.io/notesdown/>, <https://bookdown.org/xiangyun/notesdown/>, <https://notesdown.netlify.app/>。

## 1.5 记号约定

正文中的代码、函数、参数及参数值以等宽正体表示, 如 `data(list = c('iris', 'BOD'))`, 其中函数名称 `data()`, 参数及参数值 `list = c('iris', 'BOD')`, R 程序包用粗体表示, 如 `graphics`。

`ruler()`

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8  
12345678901234567890123456789012345678901234567890123456789012345678901234567890



## 1.6 复现环境

### 构建容器

本书借助 Github Action 从 Dockerfile 构建容器镜像，然后将镜像文件推送到 Github Package。完成这些操作首先需要从 <https://github.com/settings/tokens> 新建拥有 GitHub Package<sup>5</sup> 读写删的权限的 TOKEN (俗称访问令牌或密钥)，命名为 GITHUB\_PKG，并将此令牌保存到本地 TOKEN.txt 文件中，以备后用。

镜像内默认暴露 8181 端口供外部连接使用，进入容器后，默认工作路径是 /home/docker/。创建好镜像后，要先登陆 GitHub Package 然后才有权限将镜像拉取下来

```
# 登陆 GitHub Package
cat ~/TOKEN.txt | docker login https://docker.pkg.github.com -u XiangyunHuang --password-stdin
# 拉取镜像
docker pull docker.pkg.github.com/xiangyunhuang/notesdown/notesdown-book:devel
```

读者可以先查看下容器内的信息

```
docker run --rm -u root -v "${PWD}://home/docker/" \
  docker.pkg.github.com/xiangyunhuang/notesdown/notesdown-book:devel \
  bash -c "locale; fc-list | sort"
```

### 运行容器

下面从镜像创建一个叫 notesdown-book 的容器，并让它在后台运行，允许以真正的 root 账户权限交互式执行命令，停止容器后，自动销毁容器。此处，不再介绍 Docker 容器的使用，用容器打包本书所有软件环境仅供读者完整复现本书之用，感兴趣的读者可以去参考书籍[Docker 从入门到实践](#)。

```
docker run -itd -p 8282:8787 --rm --name=notesdown-book --privileged=true \
  docker.pkg.github.com/xiangyunhuang/notesdown/notesdown-book:devel /sbin/init
```

接着登陆进入 notesdown-book 容器，

```
docker exec -it notesdown-book bash
```

一番骚操作后，用户退出容器，然后停止容器。

```
docker stop notesdown-book
```

### 使用 RStudio Server

启动容器后，接着获取容器 notesdown-book 的 IP 地址，然后依据端口号 8282 从网页进入 RStudio Sever，比如 <http://192.168.100.23:8282>

```
docker inspect --format='{{ .NetworkSettings.IPAddress }}' notesdown-book
```

## 1.7 如何发问

The phrase “does not work” is not very helpful, it can mean quite a few things including:

- Your computer exploded.

<sup>5</sup><https://docs.github.com/en/packages/using-github-packages-with-your-projects-ecosystem/configuring-docker-for-use-with-github-packages>

- No explosion, but smoke is pouring out the back and microsoft's "NoSmoke" utility is not compatible with your power supply.
- The computer stopped working.
- The computer sits around on the couch all day eating chips and watching talk shows.
- The computer has started picketing your house shouting catchy slogans and demanding better working conditions and an increase in memory.
- Everything went dark and you cannot check the cables on the back of the computer because the lights are off due to the power outage.
- R crashed, but the other programs are still working.
- R gave an error message and stopped processing your code after running for a while.
- R gave an error message without running any of your code (and is waiting for your next command).
- R is still running your code and the time has exceeded your patience so you think it has hung.
- R completed and returned a result, but also gave warnings.
- R completed your command, but gave an incorrect answer.
- R completed your command but the answer is different from what you expect (but is correct according to the documentation).

There are probably others. Running your code I think the answer is the last one.

— Greg Snow<sup>6</sup>

## 1.8 作者简介

热心开源事业,统计之都副主编,经常混迹于统计之都论坛、Github 和客栈网。个人主页 <https://xiangyun.rbind.io/>

---

<sup>6</sup>来自 R 社区论坛收集的智语 `fortunes::fortune(324)`。

## 第一部分

### 数据整理

# ④ 黄湘云

## 介绍

数据整理

## 第二章 数据结构

网站 <https://r-coder.com/> 主要介绍 Base R，特点是全面细致，排版精美

可用于绘图的数据对象，向量 `vector` 等，只涉及基础操作和绘图，关键在入门引导式的介绍，点到即止

数据类型：字符、数值：字符数据操作：按数据类型介绍各类数据操作，重复之处如前所述，数据处理的分类：按数据类型来，一共是 `table` `matrix` `data.frame` 和 `vector`

The trouble with nonstandard evaluation is that it doesn't follow standard evaluation rules...

— Peter Dalgaard (about nonstandard evaluation in the `curve()` function) R-help (June 2011)

向量，列表，

数据框 `data frame` 在 R 里面可以用三种不同类型的数据对象来表达

从历史脉络来看，为什么会出现三种不同的东西，它们之间的区别和联系是什么，能否用一张表来描述

`data.frame` 设计的历史，首次包含在 R 里面是什么时候，R 是否一发布就包含了这个数据类型

The function `data.frame()` creates data frames, tightly coupled collections of variables which share many of the properties of matrices and of lists, used as the fundamental data structure by most of R's modeling software.

`data.table` 2006 年 4 月 15 日首次登陆 CRAN 发布 1.0 版本，差不多恰好 10 年后

`tibble` 在 2016 年 3 月 23 日首次登陆 CRAN 发布 1.0 版本

`data.frame()`, `tibble()` 和 `data.table()` 的区别，去看函数的帮助文档

Provides a 'tbl\_df' class (the 'tibble') that provides stricter checking and better formatting than the traditional data frame.

`vctrs` 和 `rlang` 包 R 内置的 [R Language Definition](#)

### 2.1 类型

```
x <- "abc" # 数据对象
typeof(x) # 数据类型
```

```
## [1] "character"
```



```
mode(x) # 存储模式
## [1] "character"
storage.mode(x) # 存储类型
## [1] "character"
```

表 2.1: 函数 `typeof()` 返回的数据类型<sup>1</sup>

符号	含义
NULL	空值
symbol	可变的名称/变量
pairlist	pairlist 对象 ***
closure	函数闭包
environment	环境
promise	实现惰性求值的对象
language	R 语言构造
special	内部函数, 不计算参数
builtin	内部函数, 计算参数
char	scalar 型字符对象 ***
logical	逻辑向量
integer	整数向量
double	实值向量
complex	复值向量
character	字符向量
...	可变长度的参数 ***
any	匹配任意类型
expression	表达式对象
list	列表
bytecode	位代码 ***
externalptr	外部指针对象
weakref	弱引用对象
raw	位向量
S4	S4 对象

表 2.2: R/Rcpp 提供的基本数据类型

Value	R vector	Rcpp vector	Rcpp matrix	Rcpp scalar	C++ scalar
Logical	logical	LogicalVector	LogicalMatrix	-	bool
Integer	integer	IntegerVector	IntegerMatrix	-	int
Real	numeric	NumericVector	NumericMatrix	-	double

<sup>1</sup>表格中带 \*\*\* 标记的类型, 用户不能轻易获得

Value	R vector	Rcpp vector	Rcpp matrix	Rcpp	C++
				scalar	scalar
Complex	complex	ComplexVector	ComplexMatrix	Rcomplex	complex
String	character	CharacterVector (StringVector)	CharacterMatrix (StringMatrix)	String	string
Date	Date	DateVector	-	Date	-
Datetime	POSIXct	DatetimeVector	-	Datetime	time_t

## 2.2 字符

## 2.3 向量

## 2.4 矩阵

## 2.5 数组

更多数组操作 [rray](#)

## 2.6 表达式

```
# %| |% 中缀符
# x 是空值或者长度为 0 则保留 y 否则保留 x
function(x, y) if (is.null(x) || length(x) == 0) y else x

## function(x, y) if (is.null(x) || length(x) == 0) y else x
```

## 2.7 列表

```
x <- list(a = 1, b = 2, c = list(d = c(1, 2, 3), e = "hello"))
print(x)

## $a
## [1] 1
##
## $b
## [1] 2
##
## $c
## $c$d
```



```
## [1] 1 2 3
##
## $c$e
## [1] "hello"
base::print.simple.list(x)

## -
## a 1
## b 2
## c.d1 1
## c.d2 2
## c.d3 3
## c.e hello
```

## 2.8 日期

注意观察时间转化

```
Sys.Date()
## [1] "2022-06-09"

Sys.time()
## [1] "2022-06-09 12:04:55 UTC"
c(Sys.time(), Sys.Date())
## [1] "2022-06-09 12:04:55 UTC" "2022-06-09 00:00:00 UTC"

data.table::year(Sys.Date())
## [1] 2022

data.table::year(Sys.time())
## [1] 2022

data.table::year(c(Sys.time(), Sys.Date()))
## [1] 2022 2022

x <- Sys.time()
class(x)
## [1] "POSIXct" "POSIXt"

format(x, format = "%Y-%m-%d")
## [1] "2022-06-09"

x <- c("2019-12-21", "2019/12/21")
data.table::year("2019-12-21")
```



```
## [1] 2019  
data.table::year("2019/12/21")
```

```
## [1] 2019
```

但是，下面这样会报错

```
data.table::year(x)
```

```
## Error in as.POSIXlt.character(x): character string is not in a standard unambiguous format
```

正确的姿势是首先将表示日期的字符串格式统一

```
gsub(pattern = "/", replacement = "-", x) %>%  
  data.table::year()
```

```
## [1] 2019 2019
```

date-times 表示 POSIXct 和 POSIXlt 类型的日期对象

```
(x <- Sys.time())
```

```
## [1] "2022-06-09 12:04:55 UTC"
```

```
class(x)
```

```
## [1] "POSIXct" "POSIXt"
```

```
data.table::second(x) # 取秒
```

```
## [1] 55
```

```
format(x, format = "%S")
```

```
## [1] "55"
```

```
data.table::minute(x) # 取分
```

```
## [1] 4
```

```
format(x, format = "%M")
```

```
## [1] "04"
```

```
data.table::hour(x) # 取时
```

```
## [1] 12
```

```
format(x, format = "%H")
```

```
## [1] "12"
```

```
data.table::yday(x) # 此刻在一年的第几天
```

```
## [1] 160
```

```
data.table::wday(x) # 此刻在一周的第几天，星期日是第1天，星期六是第7天
```

```
## [1] 5
```



```
data.table::mday(x) # 此刻在当月第几天
## [1] 9
format(x, format = "%d")

## [1] "09"
weekdays(x)

## [1] "Thursday"
weekdays(x, abbreviate = T)

## [1] "Thu"
data.table::week(x) # 此刻在第几周

## [1] 23
data.table::isoweek(x)

## [1] 23
data.table::month(x) # 此刻在第几月

## [1] 6
format(x, format = "%m")

## [1] "06"
months(x)

## [1] "June"
months(x, abbreviate = T)

## [1] "Jun"
data.table::quarter(x) # 此刻在第几季度

## [1] 2
quarters(x)

## [1] "Q2"
data.table::year(x) # 取年

## [1] 2022
format(x, format = "%Y")

## [1] "2022"
```



## 提示

`format()` 是一个泛型函数，此刻命名空间有 92 方法。`format.Date()`, `format.diffTime()`, `format.POSIXct()` 和 `format.POSIXlt()` 四个函数通过格式化不同类型的日期数据对象抽取指定部分。

```
format(diffTime(Sys.time(), x, units = "secs"))
```

```
## [1] "0.06668901 secs"
```

日期转化详见 [Ripley and Hornik, 2001, Grothendieck and Petzoldt, 2004]

上个季度最后一天

```
# https://d.cosx.org/d/421162/16
as.Date(cut(as.Date(c("2020-02-01", "2020-05-02")), "quarter")) - 1
## [1] "2019-12-31" "2020-03-31"
```

本季度第一天

```
as.Date(cut(as.Date(c("2020-02-01", "2020-05-02")), "quarter"))
## [1] "2020-01-01" "2020-04-01"
```

类似地，本月第一天和上月最后一天

```
# 本月第一天
as.Date(cut(as.Date(c("2020-02-01", "2020-05-02")), "month"))
## [1] "2020-02-01" "2020-05-01"
# 上月最后一天
as.Date(cut(as.Date(c("2020-02-01", "2020-05-02")), "month")) - 1
## [1] "2020-01-31" "2020-04-30"
```

`timeDate` 提供了很多日期计算函数，比如季初、季末、月初、月末等

```
library(timeDate)
# 季初
as.Date(format(timeFirstDayInQuarter(charvec = c("2020-02-01", "2020-05-02"))), format = "%Y-%m-%d")
# 季末
as.Date(format(timeLastDayInQuarter(charvec = c("2020-02-01", "2020-05-02"))), format = "%Y-%m-%d")
# 月初
as.Date(format(timeFirstDayInMonth(charvec = c("2020-02-01", "2020-05-02"))), format = "%Y-%m-%d")
# 月末
as.Date(format(timeLastDayInMonth(charvec = c("2020-02-01", "2020-05-02"))), format = "%Y-%m-%d")
```

`cut.Date()` 是一个泛型函数，查看它的所有 S3 方法

```
methods(cut)
## [1] cut.Date      cut.default    cut.dendrogram* cut.IDate*
## [5] cut.POSIXt
## see '?methods' for accessing help and source code
```



格式化输出日期类型数据

```
formatC(round(runif(1, 1e8, 1e9)), digits = 10, big.mark = ",")  
## [1] "896,491,832"  
# Sys.setlocale(locale = "C") # 如果是 Windows 系统, 必须先设置, 否则转化结果是 NA  
as.Date(paste("1990-January", 1, sep = "-"), format = "%Y-%B-%d")  
## [1] "1990-01-01"
```

获取当日零点

```
format(as.POSIXlt(Sys.Date()), "%Y-%m-%d %H:%M:%S")  
## [1] "2022-06-09 00:00:00"
```

从 POSIXt 数据对象中, 抽取小时和分钟部分, 返回字符串

```
strftime(x = Sys.time(), format = "%H:%M")  
## [1] "12:04"
```

表 2.3: 日期表格

代码	含义	代码	含义
%a	Abbreviated weekday	%A	Full weekday
%b	Abbreviated month	%B	Full month
%c	Locale-specific date and time	%d	Decimal date
%H	Decimal hours (24 hour)	%I	Decimal hours (12 hour)
%j	Decimal day of the year	%m	Decimal month
%M	Decimal minute	%p	Locale-specific AM/PM
%S	Decimal second	%U	Decimal week of the year (starting on Sunday)
%w	Decimal Weekday (0=Sunday)	%W	Decimal week of the year (starting on Monday)
%x	Locale-specific Date	%X	Locale-specific Time
%y	2-digit year	%Y	4-digit year
%z	Offset from GMT	%Z	Time zone (character)

本节介绍了 R 本身提供的基础日期操作, 第??章着重介绍一般的时间序列类型的数据对象及其操作。

Jeffrey A. Ryan 开发的 `xts` 和 `quantmod` 包, Joshua M. Ulrich 开发的 `zoo` 是处理时间序列数据的主要工具

Jeffrey A. Ryan 在开设了一门免费课程教大家如何在 R 语言中使用 `xts` 和 `zoo` 包操作时间序列数据<sup>2</sup>

`xts` (eXtensible Time Series) 扩展的 `zoo` 对象

```
xts(x = NULL,  
    order.by = index(x),  
    frequency = NULL,  
    unique = TRUE,
```

<sup>2</sup><https://www.datacamp.com/courses/manipulating-time-series-data-in-r-with-xts-zoo>

```
tzone = Sys.getenv("TZ"),
...)

library(zoo)
library(xts)
x = matrix(1:4, ncol = 2, nrow = 2)
idx <- as.Date(c("2018-01-01", "2019-12-12"))
# xts = matrix + index
xts(x, order.by = idx)

##          [,1] [,2]
## 2018-01-01    1    3
## 2019-12-12    2    4
```

Date, POSIX times, timeDate, chron 等各种各样处理日期数据的对象

## 2.9 空值

移除 list() 列表里的为 NULL 元素

```
rm_null <- function(l) Filter(Negate(is.null), l)
```

## 第三章 数据搬运

导入数据与导出数据，各种数据格式，数据库

处理 Excel 2003 (XLS) 和 Excel 2007 (XLSX) 文件还可以使用 [WriteXLS](#) 包，不过它依赖于 Perl，另一个 R 包 [xlsx](#) 与之功能类似，依赖 Java 环境。Jennifer Bryan 和 Hadley Wickham 开发的 [readxl](#) 包和 Jeroen Ooms 开发的 [writexl](#) 包专门处理 xlsx 格式并且无任何系统依赖。

### 3.1 导入数据

Base R 针对不同的数据格式文件，提供了大量的数据导入和导出函数，不愧是专注数据分析 20 余年的优秀统计软件。除了函数 `write.ftable` 和 `read.ftable` 来自 `stats` 包，都来自 `base` 和 `utils` 包

```
# 当前环境的搜索路径
searchpaths()

## [1] ".GlobalEnv"
## [2] "/opt/R/4.2.0/lib/R/library/stats"
## [3] "/opt/R/4.2.0/lib/R/library/graphics"
## [4] "/opt/R/4.2.0/lib/R/library/grDevices"
## [5] "/opt/R/4.2.0/lib/R/library/utils"
## [6] "/opt/R/4.2.0/lib/R/library/datasets"
## [7] "/opt/R/4.2.0/lib/R/library/methods"
## [8] "Autoloads"
## [9] "/opt/R/4.2.0/lib/R/library/base"

# 返回匹配结果及其所在路径的编号
apropos("^(read|write)", where = TRUE, mode = "function")
```

##	5	5	9	5
##	"read.csv"	"read.csv2"	"read.dcf"	"read.delim"
##	5	5	5	2
##	"read.delim2"	"read.DIF"	"read.fortran"	"read.ftable"
##	5	5	5	9
##	"read.fwf"	"read.socket"	"read.table"	"readBin"
##	9	5	9	9
##	"readChar"	"readCitationFile"	"readline"	"readLines"
##	9	9	9	5
##	"readRDS"	"readRenvironment"	"write"	"write.csv"

```
##      5         9         2         5
## "write.csv2"     "write.dcf"   "write.ftable"  "write.socket"
##      5         9         9         9
## "write.table"   "writeBin"    "writeChar"    "writeLines"
```

### 3.1.1 scan

```
scan(file = "", what = double(), nmax = -1, n = -1, sep = "",
      quote = if(identical(sep, "\n")) "" else "'\"", dec = ".",
      skip = 0, nlines = 0, na.strings = "NA",
      flush = FALSE, fill = FALSE, strip.white = FALSE,
      quiet = FALSE, blank.lines.skip = TRUE, multi.line = TRUE,
      comment.char = "", allowEscapes = FALSE,
      fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
```

首先让我们用 cat 函数创建一个练习数据集 ex.data

```
cat("TITLE extra line", "2 3 5 7", "11 13 17")
## TITLE extra line 2 3 5 7 11 13 17
cat("TITLE extra line", "2 3 5 7", "11 13 17", file = "data/ex.data", sep = "\n")
```

以此练习数据集，介绍 scan 函数最常用的参数

```
scan("data/ex.data")
## Error in scan("data/ex.data"): scan() expected 'a real', got 'TITLE'
```

从上面的报错信息，我们发现 scan 函数只能读取同一类型的数据，如布尔型 logical，整型 integer，数值型 numeric(double)，复数型 complex，字符型 character，raw 和列表 list。所以我们设置参数 skip = 1 把第一行跳过，就成功读取了数据

```
scan("data/ex.data", skip = 1)
## [1] 2 3 5 7 11 13 17
```

如果设置参数 quiet = TRUE 就不会报告读取的数据量

```
scan("data/ex.data", skip = 1, quiet = TRUE)
## [1] 2 3 5 7
```

参数 nlines = 1 表示只读取一行数据

```
scan("data/ex.data", skip = 1, nlines = 1) # only 1 line after the skipped one
## [1] 2 3 5 7
```

默认参数 flush = TRUE 表示读取最后一个请求的字段后，刷新到行尾，下面对比一下读取的结果

```
scan("data/ex.data", what = list("", "", "")) # flush is F -> read "7"
## Warning in scan("data/ex.data", what = list("", "", "")): number of items read
## is not a multiple of the number of columns
```



```
## [[1]]
## [1] "TITLE" "2"      "7"      "17"
##
## [[2]]
## [1] "extra" "3"      "11"     ""
##
## [[3]]
## [1] "line"  "5"      "13"     ""
## 
## [[1]]
## [1] "TITLE" "2"      "11"
##
## [[2]]
## [1] "extra" "3"      "13"
##
## [[3]]
## [1] "line"  "5"      "17"
```

临时文件 ex.data 用完了，我们调用 unlink 函数将其删除，以免留下垃圾文件

```
unlink("data/ex.data") # tidy up
```

### 3.1.2 `read.table`

```
read.table(file,
  header = FALSE, sep = "", quote = "\"\"",
  dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
  row.names, col.names, as.is = !stringsAsFactors,
  na.strings = "NA", colClasses = NA, nrows = -1,
  skip = 0, check.names = TRUE, fill = !blank.lines.skip,
  strip.white = FALSE, blank.lines.skip = TRUE,
  comment.char = "#",
  allowEscapes = FALSE, flush = FALSE,
  stringsAsFactors = default.stringsAsFactors(),
  fileEncoding = "", encoding = "unknown", text, skipNul = FALSE
)

read.csv(file,
  header = TRUE, sep = ",", quote = "\"\",
  dec = ".", fill = TRUE, comment.char = "", ...
)

read.csv2(file,
  header = TRUE, sep = ";", quote = "\"\",
```



```
  dec = ",", fill = TRUE, comment.char = "", ...
)

read.delim(file,
  header = TRUE, sep = "\t", quote = """",
  dec = ".", fill = TRUE, comment.char = "", ...
)

read.delim2(file,
  header = TRUE, sep = "\t", quote = """",
  dec = ",", fill = TRUE, comment.char = "", ...
)
```

变量名是不允许以下划线开头的，同样在数据框里，列名也不推荐使用下划线开头。默认情况下，`read.table` 都会通过参数 `check.names` 检查列名的有效性，该参数实际调用了函数 `make.names` 去检查。如果想尽量保持数据集原来的样子可以设置参数 `check.names = FALSE, stringsAsFactors = FALSE`。默认情形下，`read.table` 还会将字符串转化为因子变量，这是 R 的历史原因，作为一门统计学家的必备语言，在统计模型中，字符串常用来描述类别，而类别变量在 R 环境中常用因子类型来表示，而且大量内置的统计模型也是将它们视为因子变量，如 `lm`、`glm` 等

```
dat1 = read.table(header = TRUE, check.names = TRUE, text = "
_a _b _c
1 2 a1
3 4 a2
")
dat1
```

```
##   X_a X_b X_c
## 1   1   2   a1
## 2   3   4   a2

dat2 = read.table(header = TRUE, check.names = FALSE, text = "
_a _b _c
1 2 a1
3 4 a2
")
dat2
```

```
##   _a _b _c
## 1  1  2 a1
## 2  3  4 a2

dat3 <- read.table(header = TRUE, check.names = FALSE,
  stringsAsFactors = FALSE, text = "
_a _b _c
1 2 a1
3 4 a2
")
```

```
)  
dat3  
## _a _b _c  
## 1 1 2 a1  
## 2 3 4 a2
```

### 3.1.3 readLines

```
readLines(con = stdin(), n = -1L, ok = TRUE, warn = TRUE,  
encoding = "unknown", skipNull = FALSE)
```

让我们折腾一波，读进来又写出去，只有 R 3.5.3 以上才能保持原样的正确输入输出，因为这里有一个之前版本包含的 BUG

```
writeLines(readLines(system.file("DESCRIPTION", package = "splines")), "data/DESCRIPTION")  
# 比较一下  
identical(  
  readLines(system.file("DESCRIPTION", package = "splines")),  
  readLines("data/DESCRIPTION"))  
)
```

```
## [1] TRUE
```

这次我们创建一个真的临时文件，因为重新启动 R 这个文件和文件夹就没有了，回收掉了

```
fil <- tempfile(fileext = ".data")  
cat("TITLE extra line", "2 3 5 7", "", "11 13 17", file = fil,  
  sep = "\n")  
fil  
## [1] "/tmp/Rtmpk091Q6/file57b76a08d7e2.data"
```

设置参数 `n = -1` 表示将文件 `fil` 的内容从头读到尾

```
readLines(fil, n = -1)  
## [1] "TITLE extra line" "2 3 5 7"           ""           "11 13 17"
```

作为拥有良好习惯的 R 用户，这种垃圾文件最好用后即焚

```
unlink(fil) # tidy up
```

再举个例子，我们创建一个新的临时文件 `fil`，文件内容只有

```
cat("123\nabc")  
  
## 123  
## abc  
  
fil <- tempfile("test")  
cat("123\nabc\n", file = fil, append = TRUE)  
fil
```



```
## [1] "/tmp/Rtmpk091Q6/test57b767035c7f"  
readLines(fil)  
  
## [1] "123" "abc"
```

这次读取文件的过程给出了警告，原因是 `fil` 没有以空行结尾，`warn = TRUE` 表示这种情况要给出警告，如果设置参数 `warn = FALSE` 就没有警告。我们还是建议大家尽量遵循规范。

再举一个例子，从一个连接读取数据，建立连接的方式有很多，参见 `?file`，下面设置参数 `blocking`

```
con <- file(fil, "r", blocking = FALSE)  
readLines(con)
```

```
## [1] "123" "abc"  
cat(" def\n", file = fil, append = TRUE)  
readLines(con)
```

```
## [1] " def"  
  
# 关闭连接  
close(con)  
# 清理垃圾文件  
unlink(fil)
```

### 3.1.4 `readRDS`

序列化数据操作，Mark Klik 开发的 `fst` 和 Travers Ching 开发的 `qs`，Hadley Wickham 开发的 `feather` 包实现跨语言环境快速的读写数据

表 3.1: `fst` 序列化数据框对象性能比较 `BaseR`、`data.table` 和 `feather`<sup>1</sup>

Method	Format	Time (ms)	Size (MB)	Speed (MB/s)	N
<code>readRDS</code>	bin	1577	1000	633	112
<code>saveRDS</code>	bin	2042	1000	489	112
<code>fread</code>	csv	2925	1038	410	232
<code>fwrite</code>	csv	2790	1038	358	241
<code>read_feather</code>	bin	3950	813	253	112
<code>write_feather</code>	bin	1820	813	549	112
<code>read_fst</code>	bin	457	303	2184	282
<code>write_fst</code>	bin	314	303	3180	291

目前比较好的是 `qs` 和 `fst` 包

<sup>1</sup><https://www.fstpackage.org/>

### 3.2 其它数据格式

来自其它格式的数据形式，如 JSON、XML、YAML 需要转化清理成 R 中数据框的形式 data.frame

1. [Data Rectangling with jq](#)
2. [Mongolite User Manual](#) introduction to using MongoDB with the mongolite client in R

`jsonlite` 读取 \*.json 格式的文件, `jsonlite::write_json` 函数将 R 对象保存为 JSON 文件, `jsonlite::fromJSON` 将 json 字符串或文件转化为 R 对象, `jsonlite::toJSON` 函数正好与之相反

```
library(jsonlite)
# 从 json 格式的文件导入
# jsonlite::read_json(path = "path/to/filename.json")
# A JSON array of primitives
json <- '["Mario", "Peach", null, "Bowser"]'

# 简化为原子向量atomic vector
fromJSON(json)

## [1] "Mario"  "Peach"  NA        "Bowser"

# 默认返回一个列表
fromJSON(json, simplifyVector = FALSE)

## [[1]]
## [1] "Mario"
##
## [[2]]
## [1] "Peach"
##
## [[3]]
## NULL
##
## [[4]]
## [1] "Bowser"
```

`yaml` 包读取 \*.yml 格式文件，返回一个列表，`yaml::write_yaml` 函数将 R 对象写入 yaml 格式

```
library(yaml)
yaml::read_yaml(file = '_bookdown.yml')

## $book_filename
## [1] "notesdown"
##
## $delete_merged_file
## [1] TRUE
##
## $language
## $language$label
## $language$label$fig
```

```
## [1] "图 "
##
## $language$label$tab
## [1] "表 "
##
##
## $language$ui
## $language$ui$edit
## [1] "编辑"
##
## $language$ui$chapter_name
## [1] "第 " " 章"
##
## $language$ui$appendix_name
## [1] "附录 "
##
##
## $new_session
## [1] TRUE
##
## $before_chapter_script
## [1] "_common.R"
##
## $rmd_files
## [1] "index.Rmd"           "preface.Rmd"
## [3] "data-wrangling.Rmd"    "data-structure.Rmd"
## [5] "data-transportation.Rmd" "string-operations.Rmd"
## [7] "regular-expressions.Rmd" "data-manipulation.Rmd"
## [9] "advanced-manipulation.Rmd" "parallel-manipulation.Rmd"
## [11] "other-manipulation.Rmd" "statistical-graphics.Rmd"
## [13] "graphics-foundations.Rmd" "data-visualization.Rmd"
## [15] "interactive-web-graphics.Rmd" "numerical-optimization.Rmd"
## [17] "appendix.Rmd"          "references.Rmd"
```

表 3.2: 导入来自其它数据分析软件产生的数据集

统计软件	R 函数	R 包
ERSI ArcGIS	read.shapefile	shapefiles
Matlab	readMat	R.matlab
minitab	read.mtp	foreign
SAS (permanent data)	read.ssd	foreign
SAS (XPORT format)	read.xport	foreign
SPSS	read.spss	foreign
Stata	read.dta	foreign

统计软件	R 函数	R 包
Systat	read.systat	foreign
Octave	read.octave	foreign

表 3.3: 导入来自其它格式的数据集

文件格式	R 函数	R 包
列联表数据	read.ftable	stats
二进制数据	readBin	base
字符串数据	readChar	base
剪贴板数据	readClipboard	utils

read.dcf 函数读取 Debian 控制格式文件，这种类型的文件以人眼可读的形式在存储数据，如 R 包的 DESCRIPTION 文件或者包含所有 CRAN 上 R 包描述的文件 <https://cran.r-project.org/src/contrib/PACKAGES>

```
x <- read.dcf(file = system.file("DESCRIPTION", package = "splines"),
               fields = c("Package", "Version", "Title"))
x
```

```
##      Package Version Title
## [1,] "splines" "4.2.0" "Regression Spline Functions and Classes"
```

最后要提及拥有瑞士军刀之称的 **rio** 包，它集合了当前 R 可以读取的所有统计分析软件导出的数据。

### 3.3 导入大数据集

在不使用数据库的情况下，从命令行导入大数据集，如几百 M 或几个 G 的 csv 文件。利用 data.table 包的 fread 去读取

<https://stackoverflow.com/questions/1727772/>

### 3.4 从数据库导入

[Hands-On Programming with R](#) 数据读写章节<sup>2</sup> 以及 [R, Databases and Docker](#)

将大量的 txt 文本存进 MySQL 数据库中，通过操作数据库来聚合文本，极大降低内存消耗<sup>3</sup>，而 ODBC 与 DBI 包是其它数据库接口的基础，knitr 提供了一个支持 SQL 代码的引擎，它便是基于 DBI，因此可以在 R Markdown 文档中直接使用 SQL 代码块<sup>4</sup>。这里制作一个归纳表格，左边数据库右边对应其 R 接口，两边都包含链接，如表 3.4 所示

<sup>2</sup><https://rstudio-education.github.io/hopr/dataio.html>

<sup>3</sup><https://brucezhaor.github.io/blog/2016/08/04/batch-process-txt-to-mysql>

<sup>4</sup><https://bookdown.org/yihui/rmarkdown/language-engines.html#sql> [rstudio-spark]: <https://spark.rstudio.com/> [rmarkdown-teaching-demo]: <https://stackoverflow.com/questions/35459166>

表 3.4: 数据库接口

数据库	官网	R 接口	开发仓
MySQL	<a href="https://www.mysql.com/">https://www.mysql.com/</a>	RMySQL	<a href="https://github.com/r-dbi/RMySQL">https://github.com/r-dbi/RMySQL</a>
SQLite	<a href="https://www.sqlite.org">https://www.sqlite.org</a>	RSSQLite	<a href="https://github.com/r-dbi/RSSQLite">https://github.com/r-dbi/RSSQLite</a>
PostgreSQL	<a href="https://www.postgresql.org/">https://www.postgresql.org/</a>	RPostgres	<a href="https://github.com/r-dbi/RPostgres">https://github.com/r-dbi/RPostgres</a>
MariaDB	<a href="https://mariadb.org/">https://mariadb.org/</a>	RMariaDB	<a href="https://github.com/r-dbi/RMariaDB">https://github.com/r-dbi/RMariaDB</a>

### 3.4.1 PostgreSQL

odbc 可以支持很多数据库，下面以连接 PostgreSQL 数据库为例介绍其过程

首先在某台机器上，拉取 PostgreSQL 的 Docker 镜像

```
docker pull postgres
```

在 Docker 上运行 PostgreSQL，主机端口号 8181 映射给数据库 PostgreSQL 的默认端口号 5432（或其它你的 DBA 分配给你的端口）

```
docker run --name psql -d -p 8181:5432 -e ROOT=TRUE \
-e USER=xiangyun -e PASSWORD=cloud postgres
```

在主机 Ubuntu 上配置

```
sudo apt-get install unixodbc unixodbc-dev odbc-postgresql
```

端口 5432 是分配给 PostgreSQL 的默认端口，host 可以是云端的地址，如你的亚马逊账户下的 PostgreSQL 数据库地址 <ec2-54-83-201-96.compute-1.amazonaws.com>，也可以是本地局域网 IP 地址，如 <192.168.1.200>。通过参数 dbname 连接到指定的 PostgreSQL 数据库，如 Heroku，这里作为演示就以默认的数据库 postgres 为例

查看配置系统文件路径

```
odbcinst -j
```

```
unixODBC 2.3.6
DRIVERS.....: /etc/odbcinst.ini
SYSTEM DATA SOURCES: /etc/odbc.ini
FILE DATA SOURCES..: /etc/ODBCDataSources
USER DATA SOURCES..: /root/.odbc.ini
SQLLEN Size.....: 8
SQLLEN Size.....: 8
SQLSETPOSIROW Size.: 8
```

不推荐修改全局配置文件，可设置 ODBCSYSINI 环境变量指定配置文件路径，如 ODBCSYSINI=~/ODBC <http://www.unixodbc.org/odbcinst.html>

安装完驱动程序，/etc/odbcinst.ini 文件内容自动更新，我们可以不必修改，如果你想自定义不妨手动修改，我们查看在 R 环境中注册的数据库，可以看到 PostgreSQL 的驱动已经配置好

```
odbc::odbcListDrivers()
```

name	attribute	value
------	-----------	-------



1	PostgreSQL ANSI	Description	PostgreSQL ODBC driver (ANSI version)
2	PostgreSQL ANSI	Driver	psqlodbca.so
3	PostgreSQL ANSI	Setup	libodbcsqlS.so
4	PostgreSQL ANSI	Debug	0
5	PostgreSQL ANSI	CommLog	1
6	PostgreSQL ANSI	UsageCount	1
7	PostgreSQL Unicode	Description	PostgreSQL ODBC driver (Unicode version)
8	PostgreSQL Unicode	Driver	psqlodbcw.so
9	PostgreSQL Unicode	Setup	libodbcsqlS.so
10	PostgreSQL Unicode	Debug	0
11	PostgreSQL Unicode	CommLog	1
12	PostgreSQL Unicode	UsageCount	1

系统配置文件 `/etc/odbcinst.ini` 已经包含有 PostgreSQL 的驱动配置，无需再重复配置

```
[PostgreSQL ANSI]
Description=PostgreSQL ODBC driver (ANSI version)
Driver=psqlodbca.so
Setup=libodbcsqlS.so
Debug=0
CommLog=1
UsageCount=1

[PostgreSQL Unicode]
Description=PostgreSQL ODBC driver (Unicode version)
Driver=psqlodbcw.so
Setup=libodbcsqlS.so
Debug=0
CommLog=1
UsageCount=1
```

只需将如下内容存放在 `~/.odbc.ini` 文件中，

```
[PostgreSQL]
Driver      = PostgreSQL Unicode
Database    = postgres
Servername  = 172.17.0.1
UserName    = postgres
Password    = default
Port        = 8080
```

最后，一行命令 DNS 配置连接 <https://github.com/r-dbi/odbc> 这样就实现了代码中无任何敏感信息，这里为了展示这个配置过程故而把相关信息公开。

注意下面的内容需要在容器中运行，Windows 环境下的配置 PostgreSQL 的驱动有点麻烦就不搞了，意义也不大，现在数据库基本都是跑在 Linux 系统上

`docker-machine.exe ip default` 可以获得本地 Docker 的 IP，比如 192.168.99.101。Travis 上 `ip addr` 可以查看 Docker 的 IP，如 172.17.0.1



```
library(DBI)
con <- dbConnect(RPostgres::Postgres(),
  dbname = "postgres",
  host = ifelse(is_on_travis, Sys.getenv("DOCKER_HOST_IP"), "192.168.99.101"),
  port = 8080,
  user = "postgres",
  password = "default"
)

library(DBI)
con <- dbConnect(odbc::odbc(), "PostgreSQL")
```

列出数据库中的所有表

```
dbListTables(con)
```

第一次启动从 Docker Hub 上下载的镜像，默认的数据库是 `postgres` 里面没有任何表，所以将 R 环境中的 `mtcars` 数据集写入 `postgres` 数据库

将数据集 `mtcars` 写入 PostgreSQL 数据库中，基本操作，写入表的操作也不能缓存，即不能缓存数据库中的表 `mtcars`

```
dbWriteTable(con, "mtcars", mtcars, overwrite = TRUE)
```

现在可以看到数据表 `mtcars` 的各个字段

```
dbListFields(con, "mtcars")
```

最后执行一条 SQL 语句

```
res <- dbSendQuery(con, "SELECT * FROM mtcars WHERE cyl = 4") # 发送 SQL 语句
dbFetch(res) # 获取查询结果
dbClearResult(res) # 清理查询通道
```

或者一条命令搞定

```
dbGetQuery(con, "SELECT * FROM mtcars WHERE cyl = 4")
```

再复杂一点的 SQL 查询操作

```
dbGetQuery(con, "SELECT cyl, AVG(mpg) AS mpg FROM mtcars GROUP BY cyl ORDER BY cyl")
aggregate(mpg ~ cyl, data = mtcars, mean)
```

得益于 knitr [Xie, 2015] 开发的钩子，这里直接写 SQL 语句块，值得注意的是 SQL 代码块不能启用缓存，数据库连接通道也不能缓存，如果数据库中还没有写入表，那么写入表的操作也不能缓存，`tab.cap = "表格标题"` 输出的内容是一个表格

```
SELECT cyl, AVG(mpg) AS mpg FROM mtcars GROUP BY cyl ORDER BY cyl
```

如果将查询结果导出到变量，在 Chunk 设置 `output.var = "agg_cyl"` 可以使用缓存，下面将 `mpg` 按 `cyl` 分组聚合的结果打印出来，`ref.label = "mtcars"` 引用上一个 SQL 代码块的内容

这种基于 `odbc` 的方式的好处就不需要再安装 R 包 `RPostgres` 和相关系统依赖，最后关闭连接通道

```
dbDisconnect(con)
```

### 3.4.2 MySQL

MySQL 是一个很常见，应用也很广泛的数据库，数据分析的常见环境是在一个 R Notebook 里，我们可以在正文之前先设定数据库连接信息

```
```{r setup}
library(DBI)
# 指定数据库连接信息
db <- dbConnect(RMySQL::MySQL(),
  dbname = 'dbtest',
  username = 'user_test',
  password = 'password',
  host = '10.10.101.10',
  port = 3306
)
# 创建默认连接
knitr::opts_chunk$set(connection = 'db')
# 设置字符编码，以免中文查询乱码
DBI::dbSendQuery(db, 'SET NAMES utf8')
# 设置日期变量，以运用在SQL中
idate <- '2019-05-03'
````
```

SQL 代码块中使用 R 环境中的变量，并将查询结果输出为 R 环境中的数据框

```
```{sql, output.var='data_output'}
SELECT * FROM user_table where date_format(created_date, '%Y-%m-%d')>=?idate
````
```

以上代码会将 SQL 的运行结果存在 `data_output` 这是数据库中，`idate` 取之前设置的日期 `2019-05-03`，`user_table` 是 MySQL 数据库中的表名，`created_date` 是创建 `user_table` 时，指定的日期名。

如果 SQL 比较长，为了代码美观，把带有变量的 SQL 保存为 `demo.sql` 脚本，只需要在 SQL 的 chunk 中直接读取 SQL 文件<sup>5</sup>。

```
```{sql, code=readLines('demo.sql'), output.var='data_output'}
````
```

如果我们需要每天或者按照指定的日期重复地运行这个 R Markdown 文件，可以在 YAML 部分引入参数<sup>6</sup>

```
---
params:
  date: "2019-05-03" # 参数化日期
---
```{r setup, include=FALSE}
```

<sup>5</sup><https://d.cosx.org/d/419974>

<sup>6</sup><https://bookdown.org/yihui/rmarkdown/params-knit.html>



```
idate = params$date # 将参数化日期传递给 idate 变量
``
```

我们将这个 Rmd 文件命名为 `MyDocument.Rmd`, 运行这个文件可以从 R 控制台执行或在 RStudio 点击 `knit`。

```
rmarkdown::render("MyDocument.Rmd", params = list(
  date = "2019-05-03"
))
```

如果在文档的 YAML 位置已经指定日期, 这里可以不指定。注意在这里设置日期会覆盖 YAML 处指定的参数值, 这样做的好处是可以批量化操作。

### 3.4.3 Spark

当数据分析报告遇上 Spark 时, 就需要 `SparkR`、`sparklyr`、`arrow` 或 `rsparkling` 接口了, Javier Luraschi 写了一本书 [The R in Spark: Learning Apache Spark with R](#) 详细介绍相关扩展和应用

首先安装 `sparklyr` 包, RStudio 公司 Javier Lurasch 开发了 `sparklyr` 包, 作为 Spark 与 R 语言之间的接口, 安装完 `sparklyr` 包, 还是需要 Spark 和 Hadoop 环境

```
install.packages('sparklyr')
library(sparklyr)
spark_install()
# Installing Spark 2.4.0 for Hadoop 2.7 or later.
# Downloading from:
# - 'https://archive.apache.org/dist/spark/spark-2.4.0/spark-2.4.0-bin-hadoop2.7.tgz'
# Installing to:
# - '/spark/spark-2.4.0-bin-hadoop2.7'
# trying URL 'https://archive.apache.org/dist/spark/spark-2.4.0/spark-2.4.0-bin-hadoop2.7.tgz'
# Content type 'application/x-gzip' length 227893062 bytes (217.3 MB)
# =====
# downloaded 217.3 MB
#
# Installation complete.
```

既然 `sparklyr` 已经安装了 Spark 和 Hadoop 环境, 安装 `SparkR` 后, 只需配置好路径, 就可以加载 `SparkR` 包

```
install.packages('SparkR')
if (nchar(Sys.getenv("SPARK_HOME")) < 1) {
  Sys.setenv(SPARK_HOME = "/spark/spark-2.4.0-bin-hadoop2.7")
}
library(SparkR, lib.loc = c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib")))
sparkR.session(master = "local[*]", sparkConfig = list(spark.driver.memory = "2g"))
```

`rscala` 架起了 R 和 Scala 两门语言之间交流的桥梁, 使得彼此之间可以互相调用

是否存在这样的可能, Spark 提供了大量的 MLib 库的调用接口, R 的功能支持是最少的, Java/Scala 是原生的, 那么要么自己开发新的功能整合到 `SparkR` 中, 要么借助 `rscala` 将 scala 接口代码封装进来

在本地, Windows 主机上, 可以在 `.Rprofile` 中给 Spark 添加环境变量 `SPARK_HOME` 指定其安装路径,

```
# Windows 平台默认安装路径
Sys.setenv(SPARK_HOME = "C:/Users/XiangYun/AppData/Local/spark/spark-2.4.3-bin-hadoop2.7")
library(sparklyr)
sc <- spark_connect(master = "local", version = "2.4")
```



将 R 环境中的数据集 `mtcars` 传递到 Spark 上

```
cars <- copy_to(sc, mtcars)
cars
```

```
# Source: spark<mtcars> [?? x 11]
  mpg cyl  disp  hp drat  wt  qsec  vs  am  gear  carb
  <dbl> <dbl>
1 21       6   160   110  3.9   2.62  16.5     0     1     4     4
2 21       6   160   110  3.9   2.88  17.0     0     1     4     4
3 22.8     4   108   93   3.85  2.32  18.6     1     1     4     1
4 21.4     6   258   110  3.08  3.22  19.4     1     0     3     1
5 18.7     8   360   175  3.15  3.44  17.0     0     0     3     2
6 18.1     6   225   105  2.76  3.46  20.2     1     0     3     1
# ... with more rows
```

监控和分析命令执行的情况, 可以在浏览器中, 见图 3.1

```
spark_web(sc)
```

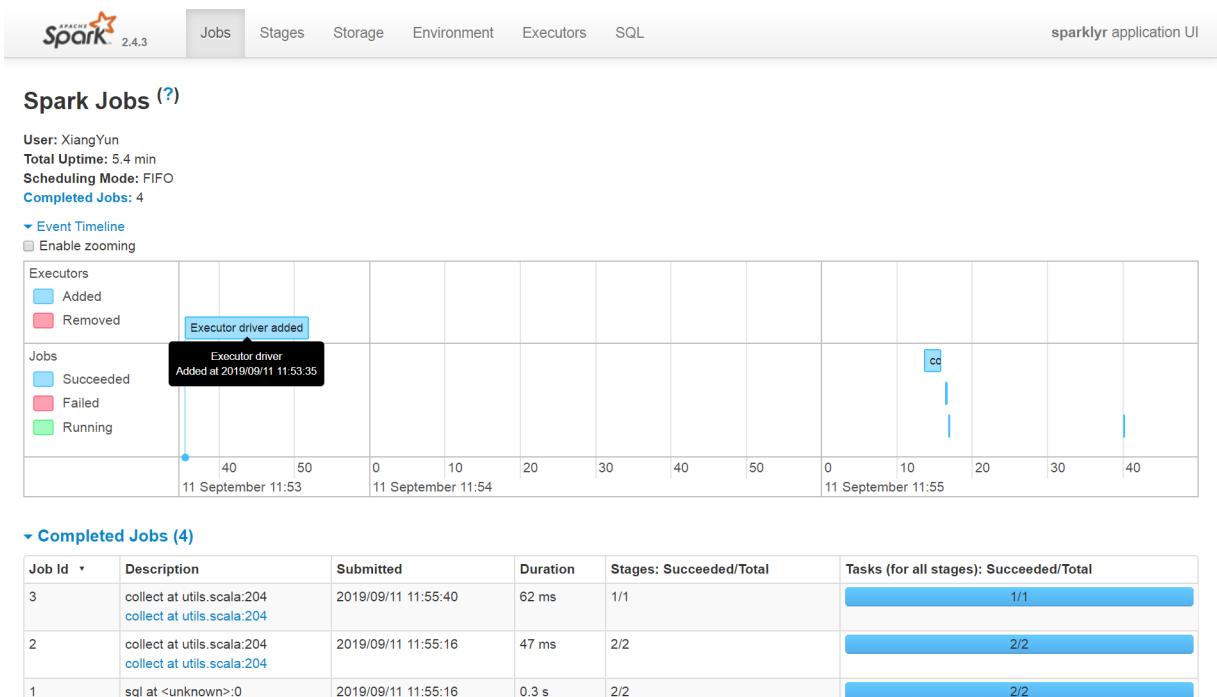


图 3.1: Spark Web 接口

传递 SQL 查询语句, 比如数据集 `mtcars` 有多少行

```
library(DBI)
dbGetQuery(sc, "SELECT count(*) FROM mtcars")

count(1)
1      32
```

进一步地，我们可以调用 `dplyr` 包来写数据操作，避免写复杂逻辑的 SQL 语句，

```
# library(dplyr) # 数据操作
library(tidyverse) # 提供更多功能，包括数据可视化
count(cars)
```

再举一个稍复杂的操作，先从数据集 `cars` 中选择两个字段 `hp` 和 `mpg`

```
select(cars, hp, mpg) %>%
  sample_n(100) %>% # 随机选择 100 行
  collect() %>% # 执行 SQL 查询，将结果返回到本地
  ggplot(aes(hp, mpg)) + # 绘图
  geom_point()
```

数据查询和结果可视化，见图 3.2

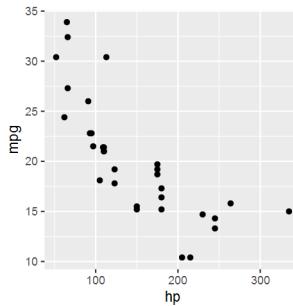


图 3.2: 数据聚合和可视化

用完要记得关闭连接

```
spark_disconnect(sc)
```

不要使用 `SparkR` 接口，要使用 `sparklyr`，后者的功能已经全面覆盖前者，生态方面更是已经远远超越，它有大厂 RStudio 支持，是公司支持的旗舰项目。但是 `sparklyr` 的版本稍微比最新的 Spark 低一两个版本，这是开发周期和出于稳定性的考虑，无伤大雅！

Spark 提供了官方 R 语言接口 `SparkR`。`SparkR` 和 `Spark` 包版本要匹配，比如从 CRAN 上安装了最新版的 `SparkR`，比如版本 2.4.4 就要去 Spark 官网下载最新版的预编译文件 `spark-2.4.4-bin-hadoop2.7`，解压到本地磁盘，比如 `D:/spark-2.4.4-bin-hadoop2.7`

```
Sys.setenv(SPARK_HOME = "D:/spark-2.4.4-bin-hadoop2.7")
# Sys.setenv(R_HOME = "C:/Program Files/R/R-3.6.1/")
library(SparkR, lib.loc = c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib")))
sparkR.session(master = "local[*]",
  sparkConfig = list(spark.driver.memory = "4g"),
  enableHiveSupport = TRUE)
```



从数据集 mtcars (数据类型是 R 的 data.frame) 创建 Spark 的 DataFrame 类型数据

```
cars <- as.DataFrame(mtcars)
# 显示 SparkDataFrame 的前几行
head(cars)
```

```
mpg cyl disp hp drat wt qsec vs am gear carb
1 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4
2 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4
3 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1
4 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1
5 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2
6 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1
```

打印数据集 cars 的 schema 各个字段的

```
printSchema(cars)
```

```
root
|-- mpg: double (nullable = true)
|-- cyl: double (nullable = true)
|-- disp: double (nullable = true)
|-- hp: double (nullable = true)
|-- drat: double (nullable = true)
|-- wt: double (nullable = true)
|-- qsec: double (nullable = true)
|-- vs: double (nullable = true)
|-- am: double (nullable = true)
|-- gear: double (nullable = true)
|-- carb: double (nullable = true)
```

从本地 JSON 文件创建 DataFrame

```
path <- file.path(Sys.getenv("SPARK_HOME"), "examples/src/main/resources/people.json")
peopleDF <- read.json(path)
printSchema(peopleDF)
```

```
root
|-- age: long (nullable = true)
|-- name: string (nullable = true)
```

```
peopleDF
```

```
SparkDataFrame[age:bigint, name:string]
```

```
showDF(peopleDF)
```

```
+----+-----+
| age|    name|
+----+-----+
| null|Michael|
|  30|    Andy|
```



```
| 19| Justin|  
+---+-----+
```

peopleDF 转成 Hive 中的表 people

```
createOrReplaceTempView(peopleDF, "people")
```

调用 sql 传递 SQL 语句查询数据，启动 sparkR.session 时，设置 enableHiveSupport = TRUE，就是执行不出来，报错，不知道哪里配置存在问题

```
teenagers <- SparkR::sql("SELECT name FROM people WHERE age >= 13 AND age <= 19")  
show(people)
```

```
Error in handleErrors(returnStatus, conn) :  
  org.apache.spark.sql.AnalysisException: java.lang.RuntimeException: java.io.IOException: (null) entry  
  at org.apache.spark.sql.hive.HiveExternalCatalog.withClient(HiveExternalCatalog.scala:106)  
  at org.apache.spark.sql.hive.HiveExternalCatalog.databaseExists(HiveExternalCatalog.scala:214)  
  at org.apache.spark.sql.internal.SharedState.externalCatalog$lzycompute(SharedState.scala:114)  
  at org.apache.spark.sql.internal.SharedState.externalCatalog(SharedState.scala:102)  
  at org.apache.spark.sql.internal.SharedState.globalTempViewManager$lzycompute(SharedState.scala:14)  
  at org.apache.spark.sql.internal.SharedState.globalTempViewManager(SharedState.scala:136)  
  at org.apache.spark.sql.hive.HiveSessionStateBuilder$$anonfun$2.apply(HiveSessionStateBuilder.scala:  
  at org.apache.spark.sql.hive.HiveSessionStateBuilder$$anonfun$2.apply(HiveSessionStateBuilder.scala:  
  at org.apache.spark.sql.catalyst.catalog.SessionCatalog.g1
```

调用 collect 函数执行查询，并将结果返回到本地 data.frame 类型

```
teenagersLocalDF <- collect(teenagers)
```

查看数据集 teenagersLocalDF 的属性

```
print(teenagersLocalDF)
```

最后，关闭 SparkSession 会话

```
sparkR.session.stop()
```

## 3.5 批量导入数据

```
library(tidyverse)  
  
read_list <- function(list_of_datasets, read_func) {  
  read_and_assign <- function(dataset, read_func) {  
    dataset_name <- as.name(dataset)  
    dataset_name <- read_func(dataset)  
  }  
  
  # invisible is used to suppress the unneeded output  
  output <- invisible(  
    sapply(list_of_datasets,
```

```
    read_and_assign,
    read_func = read_func, simplify = FALSE, USE.NAMES = TRUE
  )
)

# Remove the extension at the end of the data set names
names_of_datasets <- c(unlist(strsplit(list_of_datasets, "[.]"))[c(T, F)])
names(output) <- names_of_datasets
return(output)
}
```

批量导入文件扩展名为 .csv 的数据文件，即逗号分割的文件

```
data_files <- list.files(path = "path/to/csv/dir",
                         pattern = ".csv", full.names = TRUE)
print(data_files)
```

相比于 Base R 提供的 `read.csv` 函数，使用 `readr` 包的 `read_csv` 函数可以更快地读取 csv 格式文件，特别是在读取 GB 级数据文件时，效果特别明显。

```
list_of_data_sets <- read_list(data_files, readr::read_csv)
```

使用 `tibble` 包的 `glimpse` 函数可以十分方便地对整个数据集有一个大致的了解，展示方式和信息量相当于 `str` 加 `head` 函数

```
tibble::glimpse(list_of_data_sets)
```

## 3.6 批量导出数据

假定我们有一个列表，其每个元素都是一个数据框，现在要把每个数据框分别存入 xlsx 表的工作薄中，以 `mtcars` 数据集为例，将其按分类变量 `cyl` 分组拆分，获得一个列表 `list`

```
dat <- split(mtcars, mtcars$cyl)
dat

## $`4`
##          mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Datsun 710 22.8   4 108.0 93 3.85 2.320 18.61  1  1    4    1
## Merc 240D  24.4   4 146.7 62 3.69 3.190 20.00  1  0    4    2
## Merc 230  22.8   4 140.8 95 3.92 3.150 22.90  1  0    4    2
## Fiat 128   32.4   4  78.7 66 4.08 2.200 19.47  1  1    4    1
## Honda Civic 30.4   4  75.7 52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla 33.9   4  71.1 65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona 21.5   4 120.1 97 3.70 2.465 20.01  1  0    3    1
## Fiat X1-9    27.3   4  79.0 66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2 26.0   4 120.3 91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa 30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Volvo 142E   21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```



```
##  
## $`6`  
##          mpg cyl  disp  hp drat    wt  qsec vs am gear carb  
## Mazda RX4     21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4  
## Mazda RX4 Wag 21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4  
## Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1  
## Valiant      18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1  
## Merc 280      19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4  
## Merc 280C     17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4  
## Ferrari Dino 19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6  
##  
## $`8`  
##          mpg cyl  disp  hp drat    wt  qsec vs am gear carb  
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2  
## Duster 360      14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4  
## Merc 450SE      16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3  
## Merc 450SL      17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3  
## Merc 450SLC     15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3  
## Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4  
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4  
## Chrysler Imperial 14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4  
## Dodge Challenger 15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2  
## AMC Javelin      15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2  
## Camaro Z28        13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4  
## Pontiac Firebird 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2  
## Ford Pantera L     15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4  
## Maserati Bora      15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
```

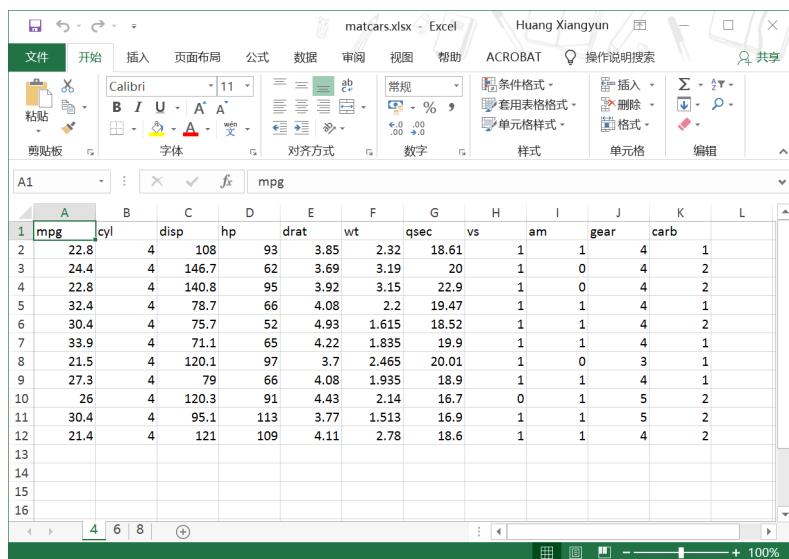
将 `xlsx` 表格初始化，创建空白的工作薄，`openxlsx` 包不依赖 Java 环境，读写效率也高

```
## 加载 openxlsx 包  
library(openxlsx)  
## 创建空白的工作薄  
wb <- createWorkbook()
```

将列表里的每张表分别存入 `xlsx` 表格的每个 `worksheet`，`worksheet` 的名字就是分组变量的名字

```
Map(function(data, name){  
  addWorksheet(wb, name)  
  writeData(wb, name, data)  
  
}, dat, names(dat))
```

最后保存数据到磁盘，见图 3.3



A	B	C	D	E	F	G	H	I	J	K	L
1 mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	
2 22.8	4	108	93	3.85	2.32	18.61	1	1	4	1	
3 24.4	4	146.7	62	3.69	3.19	20	1	0	4	2	
4 22.8	4	140.8	95	3.92	3.15	22.9	1	0	4	2	
5 32.4	4	78.7	66	4.08	2.2	19.47	1	1	4	1	
6 30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2	
7 33.9	4	71.1	65	4.22	1.835	19.9	1	1	4	1	
8 21.5	4	120.1	97	3.7	2.465	20.01	1	0	3	1	
9 27.3	4	79	66	4.08	1.935	18.9	1	1	4	1	
10 26	4	120.3	91	4.43	2.14	16.7	0	1	5	2	
11 30.4	4	95.1	113	3.77	1.513	16.9	1	1	5	2	
12 21.4	4	121	109	4.11	2.78	18.6	1	1	4	2	
13											
14											
15											
16											

图 3.3: 批量导出数据

```
saveWorkbook(wb, file = "data/matcars.xlsx", overwrite = TRUE)
```

## 3.7 导出数据

### 3.7.1 导出运行结果

```
capture.output(..., file = NULL, append = FALSE,
               type = c("output", "message"), split = FALSE)
```

`capture.output` 将一段 R 代码执行结果，保存到文件，参数为表达式。`capture.output` 和 `sink` 的关系相当于 `with` 和 `attach` 的关系。

```
glmout <- capture.output(summary(glm(case ~ spontaneous + induced,
   data = infert, family = binomial()))
                           ), file = "data/capture.txt")
capture.output(1 + 1, 2 + 2)
```

```
## [1] "[1] 2" "[1] 4"
```

```
capture.output({
  1 + 1
  2 + 2
})
```

```
## [1] "[1] 4"
```

`sink` 函数将控制台输出结果保存到文件，只将 `outer` 函数运行的结果保存到 `ex-sink.txt` 文件，`outer` 函数计算的是直积，在这里相当于 `seq(10) %*% t(seq(10))`，而在 R 语言中，更加有效的计算方式是 `tcrossprod(seq(10), seq(10))`



```
sink("data/ex-sink.txt")
i <- 1:10
outer(i, i, "*")

##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    1    2    3    4    5    6    7    8    9   10
## [2,]    2    4    6    8   10   12   14   16   18   20
## [3,]    3    6    9   12   15   18   21   24   27   30
## [4,]    4    8   12   16   20   24   28   32   36   40
## [5,]    5   10   15   20   25   30   35   40   45   50
## [6,]    6   12   18   24   30   36   42   48   54   60
## [7,]    7   14   21   28   35   42   49   56   63   70
## [8,]    8   16   24   32   40   48   56   64   72   80
## [9,]    9   18   27   36   45   54   63   72   81   90
## [10,]   10   20   30   40   50   60   70   80   90  100

sink()
```

### 3.7.2 导出数据对象

```
load(file, envir = parent.frame(), verbose = FALSE)

save(..., list = character(),
      file = stop("'file' must be specified"),
      ascii = FALSE, version = NULL, envir = parent.frame(),
      compress = isTRUE(!ascii), compression_level,
      eval.promises = TRUE, precheck = TRUE)

save.image(file = ".RData", version = NULL, ascii = FALSE,
           compress = !ascii, safe = TRUE)
```

`load` 和 `save` 函数加载或保存包含工作环境信息的数据对象, `save.image` 保存当前工作环境到磁盘, 即保存工作空间中所有数据对象, 数据格式为 `.RData`, 即相当于

```
save(list = ls(all.names = TRUE), file = ".RData", envir = .GlobalEnv)
```

`dump` 保存数据对象 `AirPassengers` 到文件 `AirPassengers.txt`, 文件内容是 R 命令, 可把 `AirPassengers.txt` 看作代码文档执行, `dput` 保存数据对象内容到文件 `AirPassengers.dat`, 文件中不包含变量名 `AirPassengers`。注意到 `dump` 输入是一个字符串, 而 `dput` 要求输入数据对象的名称, `source` 函数与 `dump` 对应, 而 `dget` 与 `dput` 对应。

```
# 加载数据
data(AirPassengers, package = "datasets")
# 将数据以R代码块的形式保存到文件
dump('AirPassengers', file = 'data/AirPassengers.txt')
# source(file = 'data/AirPassengers.txt')
```

接下来，我们读取 `AirPassengers.txt` 的文件内容，可见它是一段完整的 R 代码，可以直接复制到 R 的控制台中运行，并且得到一个与原始 `AirPassengers` 变量一样的结果

```
cat(readLines('data/AirPassengers.txt'), sep = "\n")  
  
## AirPassengers <-  
## structure(c(112, 118, 132, 129, 121, 135, 148, 148, 136, 119,  
## 104, 118, 115, 126, 141, 135, 125, 149, 170, 170, 158, 133, 114,  
## 140, 145, 150, 178, 163, 172, 178, 199, 199, 184, 162, 146, 166,  
## 171, 180, 193, 181, 183, 218, 230, 242, 209, 191, 172, 194, 196,  
## 196, 236, 235, 229, 243, 264, 272, 237, 211, 180, 201, 204, 188,  
## 235, 227, 234, 264, 302, 293, 259, 229, 203, 229, 242, 233, 267,  
## 269, 270, 315, 364, 347, 312, 274, 237, 278, 284, 277, 317, 313,  
## 318, 374, 413, 405, 355, 306, 271, 306, 315, 301, 356, 348, 355,  
## 422, 465, 467, 404, 347, 305, 336, 340, 318, 362, 348, 363, 435,  
## 491, 505, 404, 359, 310, 337, 360, 342, 406, 396, 420, 472, 548,  
## 559, 463, 407, 362, 405, 417, 391, 419, 461, 472, 535, 622, 606,  
## 508, 461, 390, 432), tsp = c(1949, 1960.916666666699, 12), class = "ts")
```

`dput` 函数类似 `dump` 函数，保存数据对象到磁盘文件

```
# 将 R 对象保存/导出到磁盘  
dput(AirPassengers, file = 'data/AirPassengers.dat')  
AirPassengers
```

```
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec  
1949 112 118 132 129 121 135 148 148 136 119 104 118  
1950 115 126 141 135 125 149 170 170 158 133 114 140  
1951 145 150 178 163 172 178 199 199 184 162 146 166  
1952 171 180 193 181 183 218 230 242 209 191 172 194  
1953 196 196 236 235 229 243 264 272 237 211 180 201  
1954 204 188 235 227 234 264 302 293 259 229 203 229  
1955 242 233 267 269 270 315 364 347 312 274 237 278  
1956 284 277 317 313 318 374 413 405 355 306 271 306  
1957 315 301 356 348 355 422 465 467 404 347 305 336  
1958 340 318 362 348 363 435 491 505 404 359 310 337  
1959 360 342 406 396 420 472 548 559 463 407 362 405  
1960 417 391 419 461 472 535 622 606 508 461 390 432
```

```
# dget 作用与 dput 相反  
AirPassengers2 <- dget(file = 'data/AirPassengers.dat')  
AirPassengers2
```

```
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec  
1949 112 118 132 129 121 135 148 148 136 119 104 118  
1950 115 126 141 135 125 149 170 170 158 133 114 140  
1951 145 150 178 163 172 178 199 199 184 162 146 166  
1952 171 180 193 181 183 218 230 242 209 191 172 194  
1953 196 196 236 235 229 243 264 272 237 211 180 201
```



```
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
```

同样地，现在我们观察 `dput` 函数保存的文件 `AirPassengers.dat` 内容，和 `dump` 函数保存的文件 `AirPassengers.txt` 相比，就缺一个赋值变量

```
cat(readLines('data/AirPassengers.dat'), sep = "\n")

structure(c(112, 118, 132, 129, 121, 135, 148, 148, 136, 119,
104, 118, 115, 126, 141, 135, 125, 149, 170, 170, 158, 133, 114,
140, 145, 150, 178, 163, 172, 178, 199, 199, 184, 162, 146, 166,
171, 180, 193, 181, 183, 218, 230, 242, 209, 191, 172, 194, 196,
196, 236, 235, 229, 243, 264, 272, 237, 211, 180, 201, 204, 188,
235, 227, 234, 264, 302, 293, 259, 229, 203, 229, 242, 233, 267,
269, 270, 315, 364, 347, 312, 274, 237, 278, 284, 277, 317, 313,
318, 374, 413, 405, 355, 306, 271, 306, 315, 301, 356, 348, 355,
422, 465, 467, 404, 347, 305, 336, 340, 318, 362, 348, 363, 435,
491, 505, 404, 359, 310, 337, 360, 342, 406, 396, 420, 472, 548,
559, 463, 407, 362, 405, 417, 391, 419, 461, 472, 535, 622, 606,
508, 461, 390, 432), tsp = c(1949, 1960.91666666667, 12), class = "ts")
```

`openxlsx` 可以读写 XLSX 文档

美团使用的大数据工具有很多，最常用的 Hive、Spark、Kylin、Impala、Presto 等，详见 <https://tech.meituan.com/2018/08/02/mt-r-practice.html>。下面主要介绍如何在 R 中连接 MySQL、Presto 和 Spark。

`sparklyr.flint` 支持 Spark 的时间序列库 `flint`，`sparkxgb` 为 Spark 上的 XGBoost 提供 R 接口，`sparkwarc` 支持加载 Web ARCHive 文件到 Spark 里 `sparkavro` 支持从 Apache Avro (<https://avro.apache.org/>) 读取文件到 Spark 里，`sparkbq` 是一个 `sparklyr` 扩展包，集成 Google BigQuery 服务，`geospark` 提供 GeoSpark 库的 R 接口，并且以 `sf` 的数据操作方式，`rsparkling` H2O Sparkling Water 机器学习库的 R 接口。

Spark 性能优化，参考三篇博文

- [Spark 在美团的实践](#)
- [Spark 性能优化指南——基础篇](#)
- [Spark 性能优化指南——高级篇](#)

其他材料

- 朱俊晖收集的 Spark 资源列表 <https://github.com/harryprince/awesome-sparklyr>，推荐使用 `sparklyr` <https://github.com/sparklyr/sparklyr> 连接 Spark
- Spark 与 R 语言 <https://docs.microsoft.com/en-us/azure/databricks/spark/latest/sparkr/>
- Mastering Spark with R <https://therinspark.com/>

## 3.8 Spark 与 R 语言

### 3.8.1 sparklyr

警告

Spark 依赖特定版本的 Java、Hadoop，三者之间的版本应该要相融。

在 MacOS 上配置 Java 环境，注意 Spark 仅支持 Java 8 至 11，所以安装指定版本的 Java 开发环境

```
# 安装 openjdk 11
brew install openjdk@11
# 全局设置 JDK 11
sudo ln -sfn /usr/local/opt/openjdk@11/libexec/openjdk.jdk /Library/Java/JavaVirtualMachines/openjdk-11.jdk
# Java 11 JDK 添加到 .zshrc
export CPPFLAGS="-I/usr/local/opt/openjdk@11/include"
export PATH="/usr/local/opt/openjdk@11/bin:$PATH"
```

配置 R 环境，让 R 能够识别 Java 环境，再安装 rJava 包

```
# 配置
sudo R CMD javareconf
# 系统软件依赖
brew install pcre2
# 安装 rJava
Rscript -e 'install.packages("rJava", type="source")'
```

最后安装 sparklyr 包，以及 Spark 环境，可以借助 spark\_install() 安装 Spark，比如下面 Spark 3.0 连同 hadoop 2.7 一起安装。

```
install.packages('sparklyr')
sparklyr::spark_install(version = '3.0', hadoop_version = '2.7')
```

也可以先手动下载 Spark 软件环境，建议选择就近镜像站点下载，比如在北京选择清华站点 <https://mirrors.tuna.tsinghua.edu.cn/apache/spark/spark-3.0.1/spark-3.0.1-bin-hadoop2.7.tgz>，此环境自带 R 和 Python 接口。为了供 sparklyr 调用，先设置 SPARK\_HOME 环境变量指向 Spark 安装位置，再连接单机版 Spark。

```
# 排错 https://github.com/sparklyr/sparklyr/issues/2827
options(sparklyr.log.console = FALSE)
# 连接 Spark
library(sparklyr)
library(ggplot2)
sc <- spark_connect(
  master = "local",
  # config = list(sparklyr.gateway.address = "127.0.0.1"),
  spark_home = Sys.getenv("SPARK_HOME")
)
# diamonds 数据集导入 Spark
```



```
diamonds_tbl <- copy_to(sc, ggplot2::diamonds, "diamonds")
```

做数据的聚合统计，有两种方式。一种是使用用 R 包 `dplyr` 提供的数据操作语法，下面以按 `cut` 分组统计钻石的数量为例，说明 `dplyr` 提供的数据操作方式。

```
library(dplyr)
# 列出数据源下所有的表 tbls
src_tbls(sc)

diamonds_tbl <- diamonds_tbl %>%
  group_by(cut) %>%
  summarise(cnt = n()) %>%
  collect
```

另一种是使用结构化查询语言 SQL，这自不必说，大多数情况下，使用和一般的 SQL 没什么两样。

```
library(DBI)
diamonds_preview <- dbGetQuery(sc, "SELECT count(*) as cnt, cut FROM diamonds GROUP BY cut")
diamonds_preview
```

```
##      cnt      cut
## 1  21551    Ideal
## 2  13791  Premium
## 3   4906      Good
## 4   1610      Fair
## 5  12082 Very Good
```

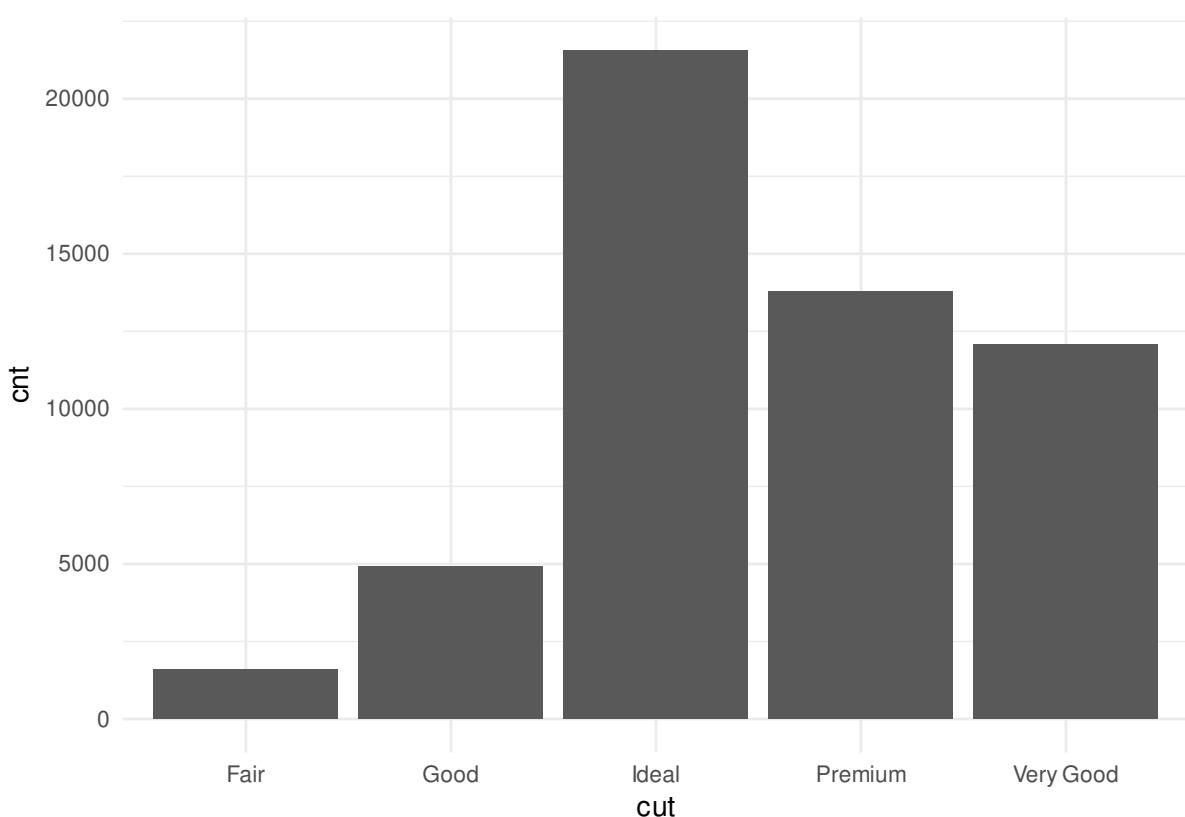
```
# SQL 中的 AVG 和 R 中的 mean 函数是类似的
diamonds_price <- dbGetQuery(sc, "SELECT AVG(price) as mean_price, cut FROM diamonds GROUP BY cut")
diamonds_price
```

```
##      mean_price      cut
## 1    3457.542    Ideal
## 2    4584.258  Premium
## 3    3928.864      Good
## 4    4358.758      Fair
## 5    3981.760 Very Good
```

将结果数据用 `ggplot2` 呈现出来

```
ggplot(diamonds_preview, aes(cut, cnt)) +
  geom_col() +
  theme_minimal()
```

```
library(ggplot2)
library(data.table)
diamonds <- as.data.table(diamonds)
diamonds[,.(mean_price = mean(price)), by = .(cut)]
##      cut mean_price
## 1:    Ideal  3457.542
## 2:  Premium  4584.258
## 3:    Good  3928.864
## 4: Very Good  3981.760
## 5:      Fair  4358.758
```



diamonds 数据集总共 53940 条数据，下面用 BUCKET 分桶抽样，将原数据随机分成 1000 个桶，取其中的一个桶，由于是随机分桶，所以每次的结果都不一样，解释详见<https://spark.apache.org/docs/latest/sql-ref-syntax-qry-select-sampling.html>

```
diamonds_sample <- dbGetQuery(sc, "SELECT * FROM diamonds TABLESAMPLE (BUCKET 1 OUT OF 1000) LIMIT 6")
diamonds_sample
```

```
##   carat      cut color clarity depth table price     x     y     z
## 1  0.80    Premium     D    SI1  62.7     59  3312 5.89 5.85 3.68
## 2  0.36  Very Good     I    VS1  60.3     60   568 4.58 4.64 2.78
## 3  0.90      Good     G    SI1  63.2     62  3574 6.09 6.19 3.88
## 4  0.70  Very Good     D   VVS2  62.3     56  3622 5.66 5.70 3.54
## 5  0.96      Fair     E    SI2  57.7     67  3674 6.49 6.46 3.73
## 6  0.32    Premium     I    VS2  62.8     58   576 4.37 4.33 2.73
```

将抽样的结果用窗口函数 RANK() 排序，详见 <https://spark.apache.org/docs/latest/sql-ref-syntax-qry-select-window.html>

窗口函数 <https://www.cnblogs.com/ZackSun/p/9713435.html>

```
diamonds_rank <- dbGetQuery(sc, "
  SELECT cut, price, RANK() OVER (PARTITION BY cut ORDER BY price) AS rank
  FROM diamonds TABLESAMPLE (BUCKET 1 OUT OF 1000)
  LIMIT 6
")
diamonds_rank
```

```
##      cut price rank
## 1 Fair    1778     1
## 2 Good    669      1
## 3 Good    796      2
## 4 Good    975      3
## 5 Good   1279     4
## 6 Good   3149     5
```

LATERAL VIEW 把一列拆成多行

<https://liam.page/2020/03/09/LATERAL-VIEW-in-Hive-SQL/> <https://spark.apache.org/docs/latest/sql-ref-syntax-qry-select-lateral-view.html>

创建数据集

```
# 先删除存在的表 person
dbGetQuery(sc, "DROP TABLE IF EXISTS person")
# 创建表 person
dbGetQuery(sc, "CREATE TABLE IF NOT EXISTS person (id INT, name STRING, age INT, class INT, address STRING)
# 插入数据到表 person
dbGetQuery(sc, "
INSERT INTO person VALUES
(100, 'John', 30, 1, 'Street 1'),
(200, 'Mary', NULL, 1, 'Street 2'),
(300, 'Mike', 80, 3, 'Street 3'),
(400, 'Dan', 50, 4, 'Street 4')
")
```

查看数据集

```
dbGetQuery(sc, "SELECT * FROM person")
```

```
##      id name age class address
## 1 100 John  30     1 Street 1
## 2 200 Mary  NA     1 Street 2
## 3 300 Mike  80     3 Street 3
## 4 400 Dan   50     4 Street 4
```

行列转换 <https://www.cnblogs.com/kimbo/p/6208973.html>, LATERAL VIEW 展开

```
dbGetQuery(sc, "
SELECT * FROM person
  LATERAL VIEW EXPLODE(ARRAY(30, 60)) tableName AS c_age
  LATERAL VIEW EXPLODE(ARRAY(40, 80)) AS d_age
LIMIT 6
")
```

```
##      id name age class address c_age d_age
## 1 100 John  30     1 Street 1    30    40
## 2 100 John  30     1 Street 1    30    80
## 3 100 John  30     1 Street 1    60    40
```



```
## 4 100 John 30      1 Street 1    60    80
## 5 200 Mary  NA     1 Street 2    30    40
## 6 200 Mary  NA     1 Street 2    30    80
```

日期相关的函数  
<https://spark.apache.org/docs/latest/sql-ref-functions-builtin.html#date-and-timestamp-functions>

# 今天

```
dbGetQuery(sc, "select current_date")
```

```
##   current_date()
## 1      2022-06-09
```

# 昨天

```
dbGetQuery(sc, "select date_sub(current_date, 1)")
```

```
##   date_sub(current_date(), 1)
## 1                  2022-06-08
```

# 本月最后一天 `current_date` 所属月份的最后一天

```
dbGetQuery(sc, "select last_day(current_date)")
```

```
##   last_day(current_date())
## 1      2022-06-30
```

# 星期几

```
dbGetQuery(sc, "select dayofweek(current_date)")
```

```
##   dayofweek(current_date())
## 1                  5
```

最后，使用完记得关闭 Spark 连接

```
spark_disconnect(sc)
```

### 3.8.2 SparkR

注意

考虑到和第3.8.1节的重合性，以及 `sparklyr` 的优势，本节代码都不会执行，仅作为补充信息予以描述。完整的介绍见 [SparkR 包](#)

```
if (nchar(Sys.getenv("SPARK_HOME")) < 1) {
  Sys.setenv(SPARK_HOME = "/opt/spark/spark-3.0.1-bin-hadoop2.7")
}
library(SparkR, lib.loc = c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib")))
sparkR.session(master = "local[*]", sparkConfig = list(spark.driver.memory = "2g"))
```

警告

SparkR 要求 Java 版本满足：大于等于 8，而小于 12，本地 MacOS 安装高版本，比如 oracle-jdk 16.0.1 会报不兼容的错误。

```
Spark package found in SPARK_HOME: /opt/spark/spark-3.1.1-bin-hadoop3.2
```

```
Error in checkJavaVersion() :
```

```
Java version, greater than or equal to 8 and less than 12, is required for this package; found version
```

sparkConfig 有哪些参数可以传递

Property Name	Property group	spark-submit equivalent
spark.master	Application Properties	--master
spark.kerberos.keytab	Application Properties	--keytab
spark.kerberos.principal	Application Properties	--principal
spark.driver.memory	Application Properties	--driver-memory
spark.driver.extraClassPath	Runtime Environment	--driver-class-path
spark.driver.extraJavaOptions	Runtime Environment	--driver-java-options
spark.driver.extraLibraryPath	Runtime Environment	--driver-library-path

将 data.frame 转化为 SparkDataFrame

```
faithful_sdf <- as.DataFrame(faithful)
```

SparkDataFrame

```
head(faithful_sdf)
```

查看结构

```
str(faithful_sdf)
```

## 3.9 数据库与 R 语言

Presto 的 R 接口 <https://github.com/prestodb/RPresto> 和文档 <https://prestodb.io/docs/current/index.html>, Presto 数据库

```
install.packages('RPresto')
```

MySQL 为例介绍 odbc 的连接和使用，详见 [从 R 连接 MySQL](#)

```
-- !preview conn=DBI::dbConnect(odbc::odbc(), driver = "MariaDB", database = "demo",
--                               uid = "root", pwd = "cloud", host = "localhost", port = 3306)
```

```
SELECT * FROM mtcars
LIMIT 10
```

我的系统已经安装了多款数据库的 ODBC 驱动



```
dnf install -y unixODBC unixODBC-devel mariadb mariadb-server mariadb-devel mariadb-connector-odbc
odbc::odbcListDrivers()

# Driver from the mariadb-connector-odbc package
# Setup from the unixODBC package

[MariaDB]
Description      = ODBC for MariaDB
Driver          = /usr/lib/libmaodbc.so
Driver64        = /usr/lib64/libmaodbc.so
FileUsage        = 1
```

## 3.10 批量读取 csv 文件

iris 数据转化为 data.table 类型，按照 Species 分组拆成单独的 csv 文件，各个文件的文件名用鸢尾花的类别名表示

```
# 批量分组导出
library(data.table)
as.data.table(iris)[, fwrite(.SD, paste0("data/user_", unique(Species), ".csv")), by = Species, .SDcols =
```

读取文件夹 data/ 所有 csv 数据文件

```
library(data.table)
merged_df <- do.call('rbind', lapply(list.files(pattern = "*.csv", path = "data/"), fread) )
# 或者
merged_df <- rbindlist(lapply(list.files(pattern = "*.csv", path = "data/"), fread))

xdf$date <- as.Date(xdf$date)
xdf$ts <- as.POSIXct(as.numeric(xdf$ts), origin = "1978-01-01")
split(xdf[order(xdf$ts), ], interaction(xdf$study, xdf$port)) %>%
  lapply(function(.x) {
    .x[nrow(.x), ]
  }) %>%
  unname() %>%
  Filter(function(.x) {
    nrow(.x) > 0
  }, .) %>%
  do.call(rbind.data.frame, .)

library(dplyr)
xdf %>%
  mutate(
    date = as.Date(date),
    ts = anytime::anytime(as.numeric(ts))
  ) %>%
  arrange(ts) %>%
```

```
group_by(study, port) %>%
  slice(n()) %>%
  ungroup()

library(tibble)
library(magrittr)

mtcars <- tibble(mtcars)

mtcars %>%
  print(n = 16, width = 69)

## # A tibble: 32 x 11
##   mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
##   <dbl> <dbl>
## 1 21     6   160   110   3.9   2.62  16.5     0     1     4     4
## 2 21     6   160   110   3.9   2.88  17.0     0     1     4     4
## 3 22.8   4   108   93    3.85  2.32  18.6     1     1     4     1
## 4 21.4   6   258   110   3.08  3.22  19.4     1     0     3     1
## 5 18.7   8   360   175   3.15  3.44  17.0     0     0     3     2
## 6 18.1   6   225   105   2.76  3.46  20.2     1     0     3     1
## 7 14.3   8   360   245   3.21  3.57  15.8     0     0     3     4
## 8 24.4   4   147.   62    3.69  3.19  20       1     0     4     2
## 9 22.8   4   141.   95    3.92  3.15  22.9     1     0     4     2
## 10 19.2   6   168.   123   3.92  3.44  18.3     1     0     4     4
## 11 17.8   6   168.   123   3.92  3.44  18.9     1     0     4     4
## 12 16.4   8   276.   180   3.07  4.07  17.4     0     0     3     3
## 13 17.3   8   276.   180   3.07  3.73  17.6     0     0     3     3
## 14 15.2   8   276.   180   3.07  3.78  18       0     0     3     3
## 15 10.4   8   472    205   2.93  5.25  18.0     0     0     3     4
## 16 10.4   8   460    215   3      5.42  17.8     0     0     3     4
## # ... with 16 more rows

mtcars %>%
  print(., n = nrow(.)/4)

## # A tibble: 32 x 11
##   mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
##   <dbl> <dbl>
## 1 21     6   160   110   3.9   2.62  16.5     0     1     4     4
## 2 21     6   160   110   3.9   2.88  17.0     0     1     4     4
## 3 22.8   4   108   93    3.85  2.32  18.6     1     1     4     1
## 4 21.4   6   258   110   3.08  3.22  19.4     1     0     3     1
## 5 18.7   8   360   175   3.15  3.44  17.0     0     0     3     2
## 6 18.1   6   225   105   2.76  3.46  20.2     1     0     3     1
## 7 14.3   8   360   245   3.21  3.57  15.8     0     0     3     4
## 8 24.4   4   147.   62    3.69  3.19  20       1     0     4     2
```

```
## # ... with 24 more rows
```

### 3.11 批量导出 xlsx 文件

将 R 环境中的数据集导出为 xlsx 表格

```
## 加载 openxlsx 包
library(openxlsx)

## 创建空白的工作簿, 标题为鸢尾花数据集
wb <- createWorkbook(title = "鸢尾花数据集")

## 添加 sheet 页
addWorksheet(wb, sheetName = "iris")

# 将数据写进 sheet 页
writeData(wb, sheet = "iris", x = iris, colNames = TRUE)

# 导出数据到本地
saveWorkbook(wb, file = "iris.xlsx", overwrite = TRUE)

library(openxlsx)
xlsxFile <- system.file("extdata", "readTest.xlsx", package = "openxlsx")
# 导入
dat = read.xlsx(xlsxFile = xlsxFile)
# 导出
write.xlsx(dat, xlsxfile)
```

### 3.12 运行环境

```
xfun::session_info()

## R version 4.2.0 (2022-04-22)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.4 LTS
##
## Locale:
##   LC_CTYPE=en_US.UTF-8      LC_NUMERIC=C
##   LC_TIME=en_US.UTF-8       LC_COLLATE=en_US.UTF-8
##   LC_MONETARY=en_US.UTF-8   LC_MESSAGES=en_US.UTF-8
##   LC_PAPER=en_US.UTF-8      LC_NAME=C
##   LC_ADDRESS=C              LC_TELEPHONE=C
##   LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## Package version:
##   askpass_1.1      assertthat_0.2.1   base64enc_0.1-3   blob_1.2.3
##   bookdown_0.26     bslib_0.3.1       cli_3.3.0        codetools_0.2.18
##   colorspace_2.0-3  compiler_4.2.0    config_0.3.1     cpp11_0.4.2
```



```
## crayon_1.5.1      curl_4.3.2       data.table_1.14.2  DBI_1.1.2
## dbplyr_2.1.1      digest_0.6.29    dplyr_1.0.9       ellipsis_0.3.2
## evaluate_0.15     fansi_1.0.3      farver_2.1.0     fastmap_1.1.0
## forge_0.2.0        fs_1.5.2        generics_0.1.2   ggplot2_3.3.6
## globals_0.15.0    glue_1.6.2       graphics_4.2.0   grDevices_4.2.0
## grid_4.2.0         gtable_0.3.0    highr_0.9        htmltools_0.5.2
## htmlwidgets_1.5.4  httr_1.4.3      isoband_0.2.5    jquerylib_0.1.4
## jsonlite_1.8.0    knitr_1.39     labeling_0.4.2   lattice_0.20.45
## lifecycle_1.0.1   magrittr_2.0.3  MASS_7.3.57     Matrix_1.4.1
## methods_4.2.0     mgcv_1.8.40    mime_0.12       munsell_0.5.0
## nlme_3.1.157      openssl_2.0.1   parallel_4.2.0  pillar_1.7.0
## pkgconfig_2.0.3   png_0.1-7      purrrr_0.3.4    r2d3_0.2.6
## R6_2.5.1          rappdirs_0.3.3  RColorBrewer_1.1.3 rlang_1.0.2
## rmarkdown_2.14     rprojroot_2.0.3  rstudioapi_0.13  sass_0.4.1
## scales_1.2.0       sparklyr_1.7.5  splines_4.2.0   stats_4.2.0
## stringi_1.7.6     stringr_1.4.0   sys_3.4        sysfonts_0.8.8
## tibble_3.1.7       tidyrr_1.2.0    tidyselect_1.1.2 tinytex_0.39
## tools_4.2.0        utf8_1.2.2      utils_4.2.0    uuid_1.1.0
## vctrs_0.4.1        viridisLite_0.4.0 withr_2.5.0    xfun_0.31
## xml2_1.3.3        yaml_2.3.5
```

## 第四章 字符串操作

[Handling Strings with R](#) 和 [R for Data Science](#) 提供字符串入门介绍, Sara Stoudt 整理了 `stringr` 包与 Base R 正则表达式函数的对应表 <https://stringr.tidyverse.org/articles/from-base.html>

`stringr` 基于 `stringi` 包字符串处理包, `re2r` 包基于 Google 开发的 C++ 库 `re2`, Google 编程之夏项目提供了一份 [正则表达式性能综述](#), `stringdist` Approximate String Matching and String Distance Functions 近似字符串匹配和字符串距离计算函数 [[van der Loo, 2014](#)]

- `janitor`
- [Manipulating strings with the stringr package](#)
- `filestrings` 基于 `stringr` 操作字符串
- `strex` 一些没有包含在 `stringr` 或者 `stringi` 中的字符串操作函数
- `tidytext` Text mining using `dplyr`, `ggplot2`, and other tidy tools

`stringdist` `stringfish` `stringb` `stringi` `stringr`

字符和字符串类型的数据值得单独拿出来讲, 不仅因为内容多, 而且比较难, 应用范围最广, 特别是面对文本类型的数据时, 几乎是避不开的! R 的前身是 S, S 的前身是一些 Fortran 和 C 子程序, 最早在贝尔实验室是用于文本分析领域, 因此在 R 基础包中提供了丰富的字符串处理函数, 你可以在 R 控制台中执行如下一行命令查看

```
help.search(keyword = "character", package = "base")
```

本章主要介绍 R 内置的字符串操作函数

### 4.1 字符数统计

`nchar` 函数统计字符串向量中每个元素的字符个数, 注意与函数 `length` 的差别, 它统计向量中元素的个数, 即向量的长度。

```
nchar(c("Hello", "world", "!"))

## [1] 5 5 1

R.version.string

## [1] "R version 4.2.0 (2022-04-22)"

nchar(R.version.string)

## [1] 28
```



```
deparse(base::mean)
## [1] "function (x, ...) "  "UseMethod(\"mean\")"
nchar(deparse(base::mean))

## [1] 18 17

一些特殊的情况

nchar("")

## [1] 0

nchar(NULL)

## integer(0)

nchar(0)

## [1] 1

pi

## [1] 3.141593

nchar(pi)

## [1] 16

exp(1)

## [1] 2.718282

nchar(exp(1))

## [1] 16

nchar(NA)

## [1] NA
```

## 4.2 字符串翻译

`tolower` 将字符串或字符串向量中含有的大写字母全都转化为小写, `toupper` 函数正好与之相反.

```
tolower(c("HELLO", "Hello, R", "hello"))

## [1] "hello"    "hello, r" "hello"

toupper(c("HELLO", "Hello, R", "hello"))

## [1] "HELLO"    "HELLO, R" "HELLO"
```

## 4.3 字符串连接

paste 函数设置参数 sep 作为连接符, 设置参数 collapse 可以将字符串拼接后连成一个字符串

```
paste("A", "B", sep = "")  
## [1] "AB"  
paste(c("A", "B", "C"), 1:3, sep = "-")  
## [1] "A-1" "B-2" "C-3"  
paste(c("A", "B", "C"), 1:3, sep = "-", collapse = ";")  
## [1] "A-1;B-2;C-3"
```

paste0 相当于 sep 设为空, 没有连接符

```
paste0("A", "B")  
## [1] "AB"  
paste0(c("A", "B", "C"), 1:3)  
## [1] "A1" "B2" "C3"  
paste0(c("A", "B", "C"), 1:3, collapse = ";")  
## [1] "A1;B2;C3"
```

## 4.4 字符串拆分

```
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)
```

strsplit 函数用于字符串拆分, 参数 x 是被拆分的字符串向量, 其每个元素都会被拆分, 而参数 split 表示拆分的位置, 可以用正则表达式来描述位置, 拆分的结果是一个列表。

参数 fixed 默认设置 fixed = FALSE 表示正则表达式匹配, 而 fixed = TRUE 表示正则表达式的精确匹配或者按文本字符的字面意思匹配, 即按普通文本匹配。我们知道按普通文本匹配速度快。

当启用 perl = TRUE 时, 由 PCRE\_use\_JIT 控制细节。perl 参数的设置与 Perl 软件版本有关, 如果正则表达式很长, 除了正确设置正则表达式, 使用 perl = TRUE 可以提高运算速度

参数 useBytes 设置是否按照逐个字节地进行匹配, 默认设置为 FALSE, 即按照字符而不是字节进行匹配

```
x <- c(as = "asfef", qu = "qwerty", yuiop[", "b", "stuff.blah.yech")  
# 按字母 e 拆分字符串向量 x  
strsplit(x, "e")  
  
## $as  
## [1] "asf" "f"  
##  
## $qu  
## [1] "qw" "rty"
```



```
##  
## [[3]]  
## [1] "yuiop["  
##  
## [[4]]  
## [1] "b"  
##  
## [[5]]  
## [1] "stuff.blah.y" "ch"
```

参数 `split` 支持通过正则表达式的方式指明拆分位置

```
# 默认将点号 . 看作一个正则表达式, 它是一个元字符, 匹配任意字符  
strsplit("a.b.c", ".")
```

```
## [[1]]  
## [1] "" "" "" "" ""  
  
# 这才是按点号拆分  
strsplit("a.b.c", ".", fixed = TRUE)
```

```
## [[1]]  
## [1] "a" "b" "c"  
  
# 或者  
strsplit("a.b.c", "[.]")
```

```
## [[1]]  
## [1] "a" "b" "c"  
  
# 或者转义点号, 去掉元字符的特殊意义  
strsplit("a.b.c", "\\.")
```

```
## [[1]]  
## [1] "a" "b" "c"
```

这里介绍一个将字符串逆序的函数 `str_rev`

```
str_rev <- function(x)  
  sapply(lapply(strsplit(x, NULL), rev), paste, collapse = "")  
str_rev(c("abc", "Statistics"))
```

```
## [1] "cba"      "scitsitatS"
```

为了加深理解, 再举几个例子

```
# 最后一个空字符没有产生  
strsplit(paste(c("", "a", ""), collapse="#"), split="#")
```

```
## [[1]]  
## [1] "" "a"
```



```
# 空字符串只有有定义的时候才会产生  
strsplit("", " ")
```

```
## [[1]]  
## character(0)  
  
strsplit(" ", " ")  
  
## [[1]]  
## [1] ""
```

## 4.5 字符串匹配

`agrep` 和 `agrepl` 函数做近似（模糊）匹配 (Approximate Matching or Fuzzy Matching) , 对于匹配, 考虑到参数 `pattern` 在参数 `x` 中匹配时, 允许参数值 `x` 存在最小可能的插入、删除和替换, 这种修改叫做 Levenshtein 编辑距离, `max.distance` 控制其细节

```
agrep(pattern, x, max.distance = 0.1, costs = NULL,  
      ignore.case = FALSE, value = FALSE, fixed = TRUE,  
      useBytes = FALSE)  
  
agrepl(pattern, x, max.distance = 0.1, costs = NULL,  
       ignore.case = FALSE, fixed = TRUE, useBytes = FALSE)
```

`agrep` 函数返回 `pattern` 在 `x` 中匹配到的一个位置向量, `agrepl` 返回一个逻辑向量, 这一点类似 `grep` 和 `grepl` 这对函数, 下面举例子说明

```
agrep("lasy", "1 lazy 2")  
  
## [1] 1  
  
# sub = 0 表示匹配时不考虑替换  
agrep("lasy", c("1 lazy 2", "1 lasy 2"), max = list(sub = 0))  
  
## [1] 2  
  
# 默认设置下, 匹配时区分大小写  
agrep("laysy", c("1 lazy", "1", "1 LAZY"), max = 2)  
  
## [1] 1  
  
# 返回匹配到值, 而不是位置下标, 类似 grep(..., value = TRUE) 的返回值  
agrep("laysy", c("1 lazy", "1", "1 LAZY"), max = 2, value = TRUE)  
  
## [1] "1 lazy"  
  
# 不区分大小写  
agrep("laysy", c("1 lazy", "1", "1 LAZY"), max = 2, ignore.case = TRUE)  
  
## [1] 1 3
```



```
startsWith(x, prefix)
endsWith(x, suffix)
```

`startsWith` 和 `endsWith` 函数用来匹配字符串的前缀和后缀, 返回值是一个逻辑向量, 参数 `prefix` 和 `suffix` 不要包含特殊的正则表达式字符, 如点号.,, 举例子

# 字符串向量

```
search()
```

```
## [1] ".GlobalEnv"      "package:stats"    "package:graphics"
## [4] "package:grDevices" "package:utils"     "package:datasets"
## [7] "package:methods"   "Autoloads"       "package:base"
```

# 匹配以 `package:` 开头的字符串

```
startsWith(search(), "package:")
```

```
## [1] FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE
```

# 或者

```
grep("^\w+package:", search())
```

```
## [1] FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE
```

当前目录下, 列出扩展名为 .Rmd 的文件

```
# list.files(path = ".", pattern = "\\.Rmd$")
```

# 而不是 `endsWith(list.files(), "\\.Rmd")`

```
endsWith(list.files(), ".Rmd")
```

```
## [1] FALSE TRUE TRUE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE TRUE FALSE FALSE
## [25] FALSE TRUE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE
## [37] FALSE FALSE TRUE FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE
## [49] TRUE TRUE FALSE TRUE TRUE FALSE FALSE FALSE TRUE TRUE TRUE TRUE
## [61] FALSE TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE TRUE TRUE
## [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE FALSE TRUE FALSE
```

# 或者

```
grep("\\\w+.Rmd$", list.files())
```

```
## [1] FALSE TRUE TRUE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE TRUE FALSE FALSE
## [25] FALSE TRUE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE
## [37] FALSE FALSE TRUE FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE
## [49] TRUE TRUE FALSE TRUE TRUE FALSE FALSE FALSE TRUE TRUE TRUE TRUE
## [61] FALSE TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE TRUE TRUE
## [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE FALSE TRUE FALSE
```

部分匹配 (Partial String Matching)

```
match(x, table, nomatch = NA_integer_, incomparables = NULL)
```

```
x %in% table
```

```
charmatch(x, table, nomatch = NA_integer_)
pmatch(x, table, nomatch = NA_integer_, duplicates.ok = FALSE)
```

这几个 `match` 函数的返回值都是一个向量，每个元素是参数 `x` 在参数 `table` 中第一次匹配到的位置，`charmatch` 与 `pmatch(x, table, nomatch = NA_integer_, duplicates.ok = TRUE)` 类似，所以 `pmatch` 在默认 `duplicates.ok = FALSE` 的情况下，若 `x` 在第二个参数 `table` 中有多个匹配就会返回 `NA`，因此，实际上 `pmatch` 只允许在第二个参数中匹配一次

```
match("xx", c("abc", "xx", "xxx", "xx"))
```

```
## [1] 2
```

```
1:10 %in% c(1,3,5,9)
```

```
## [1] TRUE FALSE TRUE FALSE TRUE FALSE FALSE FALSE TRUE FALSE
```

# `charmatch` 就比较奇怪，规则太多

```
charmatch("", "") # returns 1
```

```
## [1] 1
```

# 多个精确匹配到，或者多个部分匹配到，则返回 0

```
charmatch("m", c("mean", "median", "mode", "quantile")) # returns 0
```

```
## [1] 0
```

# `med` 只在 `table` 参数值的第二个位置部分匹配到，所以返回 2

```
charmatch("med", c("mean", "median", "mode", "quantile")) # returns 2
```

```
## [1] 2
```

```
charmatch("xx", "xx")
```

```
## [1] 1
```

```
charmatch("xx", "xa")
```

```
## [1] 1
```

```
charmatch("xx", "axx")
```

```
## [1] NA
```

# 注意比较与 `charmatch` 的不同

```
pmatch("", "") # returns NA
```

```
## [1] NA
```

```
pmatch("m", c("mean", "median", "mode")) # returns NA
```

```
## [1] NA
```

```
pmatch("med", c("mean", "median", "mode")) # returns 2
```

```
## [1] 2
```



## 4.6 字符串查询

```
grep(pattern, x,
  ignore.case = FALSE, perl = FALSE, value = FALSE,
  fixed = FALSE, useBytes = FALSE, invert = FALSE
)
grepl(pattern, x,
  ignore.case = FALSE, perl = FALSE,
  fixed = FALSE, useBytes = FALSE
)
```

grep 和 grepl 是一对字符串查询函数，查看字符串向量 x 中是否包含正则表达式 pattern 描述的内容

- ignore.case: TRUE 表示忽略大小写，FALSE 表示匹配的时候区分大小写
- fixed = TRUE 表示启用 literal regular expression 字面正则表达式，默认情况下 fixed = FALSE
- grep 函数返回匹配到的字符串向量 x 的元素的下标，如果 value=TRUE 则返回下标对应的值
- grepl 函数返回一个逻辑向量，检查字符串向量 x 中的每个元素是否匹配到，匹配到返回 TRUE，没有匹配到返回 FALSE

```
# 返回下标位置
grep("[a-z]", letters)

## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## [26] 26

# 返回查询到的值
grep("[a-z]", letters, value = TRUE)

## [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
## [20] "t" "u" "v" "w" "x" "y" "z"
```

继续举例子

```
grep(x = c("apple", "banana"), pattern = "a")

## [1] 1 2

grep(x = c("apple", "banana"), pattern = "b")

## [1] 2

grep(x = c("apple", "banana"), pattern = "a", value = TRUE)

## [1] "apple" "banana"

grep(x = c("apple", "banana"), pattern = "b", value = TRUE)

## [1] "banana"
```

关于 grepl 函数的使用例子

```
grepl(x = c("apple", "banana"), pattern = "a")

## [1] TRUE TRUE
```

```
grepl(x = c("apple", "banana"), pattern = "b")
```

```
## [1] FALSE TRUE
```

R 语言是用字符串来表示正则表达式的，但是正则表达式不是字符串，字符串的构造类似算术表达式

在 R 里面分别表示 `a\\b` 和 `a\b`

```
writeLines(c("a\\\\\\b", "a\\b"))
```

```
## a\\b
```

```
## a\b
```

下面在 R 里面分别匹配字符串 `a\\b` 和 `a\b` 中的 `\` 和 `\`

# 匹配字符串中的一个反斜杠

```
grep(x = c("a\\\\\\b", "a\\b"), pattern = "\\\\", value = TRUE, fixed = TRUE)
```

```
## [1] "a\\\\\\b" "a\\b"
```

```
grep(x = c("a\\\\\\b", "a\\b"), pattern = "\\\\\\", value = TRUE, fixed = FALSE)
```

```
## [1] "a\\\\\\b" "a\\b"
```

# 匹配字符串中的两个反斜杠

```
grep(x = c("a\\\\\\b", "a\\b"), pattern = "\\\\\\\\", value = TRUE, fixed = TRUE)
```

```
## [1] "a\\\\\\b"
```

```
grep(x = c("a\\\\\\b", "a\\b"), pattern = "\\\\\\\\\\\\", value = TRUE, fixed = FALSE)
```

```
## [1] "a\\\\\\b"
```

# 匹配字符串中的两个反斜杠 \\

```
grepl(x = "a\\\\\\b", pattern = "\\\\\\\\\\\\", fixed = FALSE)
```

```
## [1] TRUE
```

```
grepl(x = "a\\\\\\b", pattern = "\\\\\\\\\\\\", fixed = TRUE)
```

```
## [1] FALSE
```

```
grepl(x = "a\\\\\\b", pattern = "\\\\", fixed = TRUE)
```

```
## [1] TRUE
```

```
regexp(pattern, text,
```

```
  ignore.case = FALSE, perl = FALSE,
```

```
  fixed = FALSE, useBytes = FALSE
```

```
)
```

```
gregexpr(pattern, text,
```

```
  ignore.case = FALSE, perl = FALSE,
```

```
  fixed = FALSE, useBytes = FALSE
```

```
)
```

```
regexec(pattern, text,
```

```
  ignore.case = FALSE, perl = FALSE,
```

```
    fixed = FALSE, useBytes = FALSE
)
```

当启用 `perl=TRUE` 时, 函数 `regexpr` 和 `gregexpr` 支持 Python 环境下的命名捕获 (named captures), 但是不支持长向量的输入。如果一个分组被命名了, 如 `(?<first>[A-Z][a-z]+)` 那么匹配到的位置按命名返回。函数 `sub` 不支持命名反向引用 (Named backreferences)

函数 `regmatches` 用来提取函数 `regexpr`, `gregexpr` 和 `regexec` 匹配到的子字符串

`useBytes = FALSE` 匹配位置和长度默认是按照字符级别的, 如果 `useBytes = TRUE` 则是按照逐个字节的匹配结果

如果使用到了命名捕获则会返回更多的属性 “`capture.start`”, “`capture.length`” 和 “`capture.names`”, 分别表示捕获的起始位置、捕获的长度和捕获的命名。

- `regexpr` 函数返回一个整型向量, 第一次匹配的初始位置, `-1` 表示没有匹配到, 返回的属性 `match.length` 表示匹配的字符数量, 是一个整型向量, 向量长度是匹配的文本的长度, `-1` 表示没有匹配到

```
text <- c("Hello, Adam!", "Hi, Adam!", "How are you, Adam.")
regexpr("Adam", text)

## [1] 9 5 14
## attr(),"match.length")
## [1] 4 4 4
## attr(),"index.type")
## [1] "chars"
## attr(),"useBytes")
## [1] TRUE

txt <- c(
  "The", "licenses", "for", "most", "software", "are",
  "designed", "to", "take", "away", "your", "freedom",
  "to", "share", "and", "change", "it.",
  "", "By", "contrast", "the", "GNU", "General", "Public", "License",
  "is", "intended", "to", "guarantee", "your", "freedom", "to",
  "share", "and", "change", "free", "software", "--",
  "to", "make", "sure", "the", "software", "is",
  "free", "for", "all", "its", "users"
)
# gregexpr("en", txt)
regexpr("en", txt)

## [1] -1 4 -1 2 -1 4
## [26] -1 4 -1
## attr(),"match.length")
## [1] -1 2 -1
## [26] -1 2 -1
## attr(),"index.type")
## [1] "chars"
```

```
## attr("useBytes")
## [1] TRUE
```

- `gregexpr` 函数返回一个列表，返回列表的长度与字符串向量的长度一样，列表中每个元素的形式与 `regexp` 的返回值一样，except that the starting positions of every (disjoint) match are given.

```
gregexpr("Adam", text)
```

```
## [[1]]
## [1] 9
## attr("match.length")
## [1] 4
## attr("index.type")
## [1] "chars"
## attr("useBytes")
## [1] TRUE
##
## [[2]]
## [1] 5
## attr("match.length")
## [1] 4
## attr("index.type")
## [1] "chars"
## attr("useBytes")
## [1] TRUE
##
## [[3]]
## [1] 14
## attr("match.length")
## [1] 4
## attr("index.type")
## [1] "chars"
## attr("useBytes")
## [1] TRUE
```

- `regexec` 函数返回一个列表，类似函数 `gregexpr` 的返回结果，长度与字符串向量的长度一样，如果没有匹配到就返回 -1，匹配到了就返回一个匹配的初值位置的整型序列，所有子字符串与括号分组的正则表达式的子表达式对应，属性 “`match.length`” 是一个表示匹配的长度的向量，如果是 -1 表示没有匹配到。位置、长度和属性的解释与 `regexp` 一致

```
regexec("Adam", text)
```

```
## [[1]]
## [1] 9
## attr("match.length")
## [1] 4
## attr("index.type")
## [1] "chars"
```

```

## attr(,"useBytes")
## [1] TRUE
##
## [[2]]
## [1] 5
## attr(,"match.length")
## [1] 4
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
##
## [[3]]
## [1] 14
## attr(,"match.length")
## [1] 4
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE

```

由于资源限制（特别是 PCRE）导致的匹配失败，会视为没有匹配，通常伴随一个警告

下面这个将链接分解的例子由 Luke Tierney 提供<sup>1</sup>

```

x <- "http://stat.umn.edu:80/xyz"
m <- regexec("^(([^\:]+)://)?([^\:/]+)(:[([0-9]+))?(./*)", x)
m

## [[1]]
## [1] 1 1 1 8 20 21 23
## attr(,"match.length")
## [1] 26 7 4 12 3 2 4
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE

```

这里 x 是一个字符串，所以函数 `regexec` 返回的列表长度为 1，正则表达式 `^(([^\:]+)://)?([^\:/]+)(:[([0-9]+))?(./*)` 括号分组匹配到了 7 次，第一次匹配整个字符串，所以起始位置是 1，而匹配长度是 26，即整个字符串的长度，读者可以调用函数 `nchar(x)` 算一下，如果你愿意手动数一下也可以哈！余下不一一介绍，可以根据返回结果和图 4.1 一起看，最后还可以调用 `regmatches` 函数抽取匹配到的结果

```

regmatches(x, m)

## [[1]]
## [1] "http://stat.umn.edu:80/xyz" "http://"
## [3] "http"                      "stat.umn.edu"

```

<sup>1</sup><https://homepage.divms.uiowa.edu/~luke/R/regexp.html>

```
## [5] ":"80"           "80"
## [7] "/xyz"
```

我们可以在 <https://regex101.com/> 上测试表达式, 如图4.1所示, 表达式 `^(([^\:]+)://)?([^\:/]+)(:[0-9]+)?(.*?)` 包含 7 个组, 每个组的匹配结果见图的右下角, 这样我们不难理解, 函数 `regmatches` 返回的第列表中, 第 3 个位置是传输协议 `protocol` `http`, 第 4 个位置是主机 `host` `stat.umn.edu`, 第 6 个位置是端口 `port` `80`, 第 7 个位置是路径 `path` `/xyz`, 所以函数 `regmatches` 的作用就是根据函数 `regexec` 匹配的结果抽取子字符串。

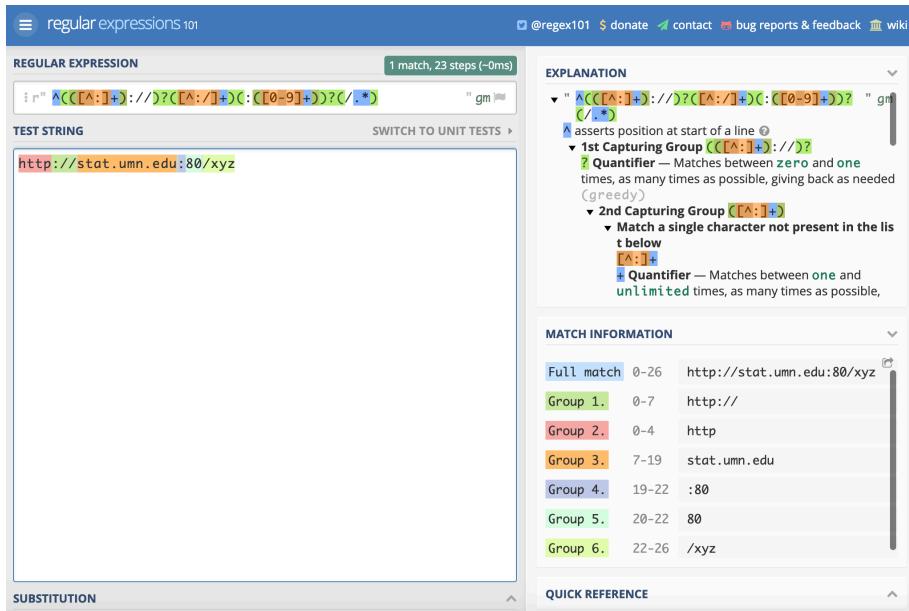


图 4.1: 正则表达式匹配结果

进一步, 我们可以用 `regmatches` 函数抽取 URL 的部分内容, 如前面提到的传输协议, 主机等

```
URL_parts <- function(x) {
  m <- regexec("^(([^\:]+)://)?([^\:/]+)(:[0-9]+)?(.*?)" , x)
  parts <- do.call(
    rbind,
    lapply(regmatches(x, m), `[,` , c(3L, 4L, 6L, 7L)))
  # 3,4,6,7是索引位置
  )
  colnames(parts) <- c("protocol", "host", "port", "path")
  parts
}

URL_parts(x)

##      protocol host      port path
## [1,] "http"   "stat.umn.edu" "80"  "/xyz"
```

目前还没有 `gregexec` 函数, 但是可以模拟一个, 首先用 `gregexpr` 函数返回匹配的位置, `regmatches` 抽取相应的值, 然后用 `regexec` 作用到每一个提取的值, 做再一次匹配和值的抽取, 实现了全部的匹配。另一个例子

```
## There is no gregexec() yet, but one can emulate it by running
## regexec() on the regmatches obtained via gregexpr(). E.g.:
pattern <- "[[:alpha:]]+([[:digit:]]+)"
s <- "Test: A1 BC23 DEF456"
gregexpr(pattern, s)

## [[1]]
## [1] 7 10 15
## attr(),"match.length"
## [1] 2 4 6
## attr(),"index.type"
## [1] "chars"
## attr(),"useBytes"
## [1] TRUE

regmatches(s, gregexpr(pattern, s))

## [[1]]
## [1] "A1"      "BC23"    "DEF456"
lapply(
  regmatches(s, gregexpr(pattern, s)),
  function(e) regmatches(e, regexec(pattern, e)))
)

## [[1]]
## [[1]][[1]]
## [1] "A1" "A"  "1"
##
## [[1]][[2]]
## [1] "BC23" "BC"  "23"
##
## [[1]][[3]]
## [1] "DEF456" "DEF"  "456"
```

## 4.7 字符串替换

`chartr` 支持正则表达式的替换, `chartr` 是对应字符的替换操作

```
x <- "MiXeD cAsE 123"
# 将字符 iXs 替换为 why
chartr("iXs", "why", x)

## [1] "MwheD cAyE 123"

# 将字符串 a-cX 中的字符挨个对应地替换为 D-Fw
chartr("a-cX", "D-Fw", x)
```



```
## [1] "MiweD FAsE 123"
```

两个 `*sub` 函数的区别: `sub` 替换第一次匹配到的结果, `gsub` 替换所有匹配的结果

```
sub(".*", "", extSoftVersion()["PCRE"])
```

## PCRE

## "10.39"

参数 `replacement` 的值是正则表达式, 其包含反向引用的用法, `\1` 即引用表达式 `([ab])`

```
gsub(pattern = "([ab])", replacement = "\1_\1", x = "abc and ABC")
```

```
## [1] "a_a_b_b_c a_a_nd ABC"
```

## 4.8 字符串提取

```
substr(x, start, stop)
substring(text, first, last = 1000000L)
```

`substr` 和 `substring` 函数通过位置进行字符串的拆分和提取, 它们本身不使用正则表达式, 结合其他正则表达式函数 `regexp`, `gregexpr` 和 `regexec`, 可以很方便地从大量文本中提取所需的信息。作用类似之前提到的 `regmatches` 函数

参数设置基本相同

- `x/text` 是要拆分的字符串向量
- `start/first` 截取的起始位置向量
- `stop/last` 截取的终止位置向量

返回值有差别

- `substr` 返回的字串个数等于第一个参数 `x` 的长度
- `substring` 返回字串个数等于三个参数中最长向量长度, 短向量循环使用。

```
x <- "123456789"
```

```
substr(x, c(2, 4), c(4, 5, 8))
```

```
## [1] "234"
```

```
substring(x, c(2, 4), c(4, 5, 8))
```

```
## [1] "234"      "45"      "2345678"
```

```
substr("abcdef", 2, 4)
```

```
## [1] "bcd"
```

```
substring("abcdef", 1:6, 1:6)
```

```
## [1] "a"  "b"  "c"  "d"  "e"  "f"
```

因为 `x` 的向量长度为 1, 所以 `substr` 获得的结果只有 1 个字串, 即第 2 和第 3 个参数向量只用了第一个组合: 起始位置 2, 终止位置 4。而 `substring` 的语句三个参数中最长的向量为 `c(4,5,8)`, 执行时按短向

量循环使用的规则第一个参数事实上就是 `c(x, x, x)`，第二个参数就成了 `c(2, 4, 2)`，最终截取的字符串起始位置组合为：2-4, 4-5 和 2-8。

```
x <- c("123456789", "abcdefghijklmnopqrstuvwxyz")
substr(x, c(2, 4), c(4, 5, 8))

## [1] "234" "de"
substring(x, c(2, 4), c(4, 5, 8))

## [1] "234"      "de"      "2345678"
```

更加高级的字符串抽取

```
# 从字符串中抽取固定模式的文本，替代 stringr::str_extract
# 只抽取一个匹配的
extract_str <- function(text, pattern) regmatches(text, regexpr(pattern, text))
# 符合模式的全部抽取
gextract_str <- function(text, pattern) regmatches(text, gregexpr(pattern, text))
```

举例子，抽取连续的数字

```
# 两个例子
extract_str(text = "abd123da345das", pattern = "(\\d+){3}")

## [1] "123"
gextract_str(text = "abd123da345das", pattern = "(\\d+){3}")

## [[1]]
## [1] "123" "345"
```

例子来自于 <https://recology.info/2018/10/limiting-dependencies/>

## 4.9 命名捕捉

函数 `regexpr(..., perl = TRUE)` 和 `gregexpr(..., perl = TRUE)` 支持命名捕捉

```
## named capture
notables <- c(" Ben Franklin and Jefferson Davis",
            "\tMillard Fillmore")
# name groups 'first' and 'last'
name.rex <- "(?<first>[:upper:]][[:lower:]]+) (?<last>[:upper:]][[:lower:]]+)"

(parsed <- regexpr(name.rex, notables, perl = TRUE))

## [1] 3 2
## attr(),"match.length")
## [1] 12 16
## attr(),"index.type")
## [1] "chars"
## attr(),"useBytes")
```



```
## [1] TRUE
## attr(),"capture.start")
##      first last
## [1,]     3     7
## [2,]     2    10
## attr(),"capture.length")
##      first last
## [1,]     3     8
## [2,]     7     8
## attr(),"capture.names")
## [1] "first" "last"
attr(parsed, 'capture.names')

## [1] "first" "last"
regmatches(notables, parsed)

## [1] "Ben Franklin"      "Millard Fillmore"
```

希望返回一个 data.frame, 列名是指定的 named group 名字

```
# 有多个结果
(idx <- gregexpr(name.rex, notables, perl = TRUE))
```

```
## [[1]]
## [1] 3 20
## attr(),"match.length")
## [1] 12 15
## attr(),"index.type")
## [1] "chars"
## attr(),"useBytes")
## [1] TRUE
## attr(),"capture.start")
##      first last
## [1,]     3     7
## [2,]    20    30
## attr(),"capture.length")
##      first last
## [1,]     3     8
## [2,]     9     5
## attr(),"capture.names")
## [1] "first" "last"
##
## [[2]]
## [1] 2
## attr(),"match.length")
## [1] 16
## attr(),"index.type")
```



```
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
## attr(,"capture.start")
##      first last
## [1,]    2    10
## attr(,"capture.length")
##      first last
## [1,]    7    8
## attr(,"capture.names")
## [1] "first" "last"
regmatches(notables, idx)

## [[1]]
## [1] "Ben Franklin"      "Jefferson Davis"
##
## [[2]]
## [1] "Millard Fillmore"
attr(idx[[1]], 'capture.names')

## [1] "first" "last"
library(magrittr)
data.frame(notable = notables) %>%
tidyrr::extract(
  notable, c("first", "last"), name.rex,
  remove = FALSE
)

##                                     notable   first     last
## 1  Ben Franklin and Jefferson Davis   Ben Franklin
## 2                           \tMillard Fillmore Millard Fillmore
```

## 4.10 精确匹配

```
fixed = TRUE
```

## 4.11 模糊匹配

近似字符串匹配 (Approximate String Matching) 也叫模糊匹配 (Fuzzy Matching)

```
agrep() agrepl() aregexec() adist()
agrep(pattern = "lasy", x = "1 lazy 2")

## [1] 1
```



```
agrep("lasy", c(" 1 lazy 2", "1 lasy 2"), max = list(sub = 0))
## [1] 2

agrep("laysy", c("1 lazy", "1", "1 LAZY"), max = 2)
## [1] 1

agrep("laysy", c("1 lazy", "1", "1 LAZY"), max = 2, value = TRUE)
## [1] "1 lazy"

agrep("laysy", c("1 lazy", "1", "1 LAZY"), max = 2, ignore.case = TRUE)
## [1] 1 3

agrepl(pattern = "lasy", x = "1 lazy 2")
## [1] TRUE

## Cf. the examples for agrep.
x <- c("1 lazy", "1", "1 LAZY")

aregexec("laysy", x, max.distance = 2)
## [[1]]
## [1] 3
## attr(),"match.length")
## [1] 4
##
## [[2]]
## [1] -1
## attr(),"match.length")
## [1] -1
##
## [[3]]
## [1] -1
## attr(),"match.length")
## [1] -1

aregexec("(lay)(sy)", x, max.distance = 2)
## [[1]]
## [1] 3 3 5
## attr(),"match.length")
## [1] 4 2 2
##
## [[2]]
## [1] -1
## attr(),"match.length")
## [1] -1
##
```

```
## [[3]]
## [1] -1
## attr(,"match.length")
## [1] -1
aregexec("(lay)(sy)", x, max.distance = 2, ignore.case = TRUE)

## [[1]]
## [1] 3 3 6
## attr(,"match.length")
## [1] 4 3 1
##
## [[2]]
## [1] -1
## attr(,"match.length")
## [1] -1
##
## [[3]]
## [1] 3 3 6
## attr(,"match.length")
## [1] 4 3 1

m <- aregexec("(lay)(sy)", x, max.distance = 2)
regmatches(x, m)

## [[1]]
## [1] "lazy" "la"   "zy"
##
## [[2]]
## character(0)
##
## [[3]]
## character(0)

## Cf. https://en.wikipedia.org/wiki/Levenshtein\_distance
adist("kitten", "sitting")

##      [,1]
## [1,]    3
## To see the transformation counts for the Levenshtein distance:
drop(attr(adist("kitten", "sitting", counts = TRUE), "counts"))

## ins del sub
##   1   0   2
## To see the transformation sequences:
attr(adist(c("kitten", "sitting"), counts = TRUE), "trajos")

##      [,1]      [,2]
```



```
## [1,] "MMMMMM"  "SMMMSMI"
## [2,] "SMMMSMD" "MMMMMM"
## Cf. the examples for agrep:
adist("lasy", "1 lazy 2")

##      [,1]
## [1,]     5

## For a "partial approximate match" (as used for agrep):
adist("lasy", "1 lazy 2", partial = TRUE)

##      [,1]
## [1,]     1
```

案例

```
help.search()
```

## 4.12 高级的替换

相比于 `sprintf()` 格式化输出字符串的方式替换，它的优势在于提示性，或者说代码的可读性

```
glue_data <- function(param, text) {
  idx <- gregexpr('\\\\{[^}]*\\\\}', text)[[1L]]
  keys <- substring(text, idx, idx + attr(idx, 'match.length') - 1L)
  for (key in keys) {
    text <- gsub(key, param[[gsub('{[]}', '', key)]], text, fixed = TRUE)
  }
  text
}
cat(glue_data(
  param = list(table = 'flights', origin = 'JFK'),
  text = "
  select count(*) as n
  from {table}
  where origin = '{origin}'
  "
))
```

```
## 
##   select count(*) as n
##   from flights
##   where origin = 'JFK'
##
```



## 4.13 高级的提取

从 text 中抽取给定模式 pattern 的字符串

```
str_extract <- function(text, pattern, ...) regmatches(text, regexpr(pattern, text, ...))
```

举个栗子，比如提取数字

```
shopping_list <- c("apples x4", "bag of flour", "bag of sugar", "milk x2")  
stringr::str_extract(shopping_list, "\\d")
```

```
## [1] "4" NA NA "2"
```

# 注意二者的差别

```
str_extract(shopping_list, "\\d")
```

```
## [1] "4" "2"
```

提取所有符合匹配模式的字符串

```
str_extract_all <- function(text, pattern, ...) regmatches(text, gregexpr(pattern, text, ...))
```

举个栗子，提取其中的英文字母

```
str_extract_all(shopping_list, "[a-z]+")
```

```
## [[1]]  
## [1] "apples" "x"  
##  
## [[2]]  
## [1] "bag"    "of"    "flour"  
##  
## [[3]]  
## [1] "bag"    "of"    "sugar"  
##  
## [[4]]  
## [1] "milk"   "x"  
stringr::str_extract_all(shopping_list, "[a-z]+")
```

```
## [[1]]  
## [1] "apples" "x"  
##  
## [[2]]  
## [1] "bag"    "of"    "flour"  
##  
## [[3]]  
## [1] "bag"    "of"    "sugar"  
##  
## [[4]]  
## [1] "milk"   "x"
```

## 4.14 其它操作

### 4.14.1 strwrap

```
strwrap(x, width = 0.9 *getOption("width"), indent = 0,
        exdent = 0, prefix = "", simplify = TRUE, initial = prefix)
```

该函数把一个字符串当成一个段落的文字（不管字符串中是否有换行符），按照段落的格式（缩进和长度）和断字方式进行分行，每一行是结果中的一个字符串。

```
# 读取一段文本
x <- paste(readLines(file.path(R.home("doc"), "THANKS")), collapse = "\n")
## 将文本拆分为段落，且移除前三段
x <- unlist(strsplit(x, "\n[ \t\n]*\n"))[-(1:3)]
# 每一段换两行
x <- paste(x, collapse = "\n\n")
# 每一行的宽度设定为60个字符
writeLines(strwrap(x, width = 60))
```

```
## J. D. Beasley, David J. Best, Richard Brent, Kevin Buhr,
## Michael A. Covington, Bill Cleveland, Robert Cleveland,, G.
## W. Cran, C. G. Ding, Ulrich Drepper, Paul Eggert, J. O.
## Evans, David M. Gay, H. Frick, G. W. Hill, Richard H.
## Jones, Eric Grosse, Shelby Haberman, Bruno Haible, John
## Hartigan, Andrew Harvey, Trevor Hastie, Min Long Lam,
## George Marsaglia, K. J. Martin, Gordon Matzigkeit, C. R.
## Mckenzie, Jean McRae, Cyrus Mehta, Fionn Murtagh, John C.
## Nash, Finbarr O'Sullivan, R. E. Odeh, William Patefield,
## Nitin Patel, Alan Richardson, D. E. Roberts, Patrick
## Royston, Russell Lenth, Ming-Jen Shyu, Richard C.
## Singleton, S. G. Springer, Supoj Sutanthavibul, Irma
## Terpenning, G. E. Thomas, Rob Tibshirani, Wai Wan Tsang,
## Berwin Turlach, Gary V. Vaughan, Michael Wichura, Jingbo
## Wang, M. A. Wong, and the Free Software Foundation (for
## autoconf code and utilities). See also files under
## src/extras.

##
## Many more, too numerous to mention here, have contributed
## by sending bug reports and suggesting various improvements.

##
## Simon Davies whilst at the University of Auckland wrote the
## original version of glm().

##
## Julian Harris and Wing Kwong (Tiki) Wan whilst at the
## University of Auckland assisted Ross Ihaka with the
## original Macintosh port.
```

```
##  
## R was inspired by the S environment which has been  
## principally developed by John Chambers, with substantial  
## input from Douglas Bates, Rick Becker, Bill Cleveland,  
## Trevor Hastie, Daryl Pregibon and Allan Wilks.  
##  
## A special debt is owed to John Chambers who has graciously  
## contributed advice and encouragement in the early days of R  
## and later became a member of the core team.  
##  
## Stefano Iacus (up to 2014, a former member of R Core) and  
## Simon Urbanek developed the macOS port, including the R.app  
## GUI, toolchains and packaging.  
##  
## The Windows port was originally developed by Guido  
## Masarotto (for a while a member of R Core) and Brian  
## Ripley, then further by Duncan Murdoch (a former member of  
## R Core) and then Jeroen Ooms (base) and Uwe Ligges  
## (packages). Tomas Kalibera is the current main developer  
## of the Windows port and provides assistance with package  
## porting.  
##  
## Tomas Kalibera's work has been sponsored by Jan Vitek and  
## funded by his European Research Council grant "Evolving  
## Language Ecosystems (ELE)".  
##  
## Computing support (including hardware, hosting and  
## infrastructure) has been provided/funded by the R  
## Foundation, employers of R-Core members (notably WU Wien,  
## ETH Zurich, U Oxford and U Iowa) and by Northeastern  
## University and the University of Kent.  
##  
## Distributions of R contain the recommended packages, whose  
## authors/contributors are listed in their DESCRIPTION files.
```

# 每一段的段首缩进5个字符

```
writeLines(strwrap(x, width = 60, indent = 5))
```

```
##      J. D. Beasley, David J. Best, Richard Brent, Kevin  
## Buhr, Michael A. Covington, Bill Cleveland, Robert  
## Cleveland,, G. W. Cran, C. G. Ding, Ulrich Drepper, Paul  
## Eggert, J. O. Evans, David M. Gay, H. Frick, G. W. Hill,  
## Richard H. Jones, Eric Grosse, Shelby Haberman, Bruno  
## Haible, John Hartigan, Andrew Harvey, Trevor Hastie, Min  
## Long Lam, George Marsaglia, K. J. Martin, Gordon  
## Matzigkeit, C. R. Mckenzie, Jean McRae, Cyrus Mehta, Fionn
```

```
## Murtagh, John C. Nash, Finbarr O'Sullivan, R. E. Odeh,
## William Patefield, Nitin Patel, Alan Richardson, D. E.
## Roberts, Patrick Royston, Russell Lenth, Ming-Jen Shyu,
## Richard C. Singleton, S. G. Springer, Supoj Sutanthavibul,
## Irma Terpenning, G. E. Thomas, Rob Tibshirani, Wai Wan
## Tsang, Berwin Turlach, Gary V. Vaughan, Michael Wichura,
## Jingbo Wang, M. A. Wong, and the Free Software Foundation
## (for autoconf code and utilities). See also files under
## src/extras.

##
##      Many more, too numerous to mention here, have
## contributed by sending bug reports and suggesting various
## improvements.

##
##      Simon Davies whilst at the University of Auckland
## wrote the original version of glm().

##
##      Julian Harris and Wing Kwong (Tiki) Wan whilst at the
## University of Auckland assisted Ross Ihaka with the
## original Macintosh port.

##
##      R was inspired by the S environment which has been
## principally developed by John Chambers, with substantial
## input from Douglas Bates, Rick Becker, Bill Cleveland,
## Trevor Hastie, Daryl Pregibon and Allan Wilks.

##
##      A special debt is owed to John Chambers who has
## graciously contributed advice and encouragement in the
## early days of R and later became a member of the core team.

##
##      Stefano Iacus (up to 2014, a former member of R Core)
## and Simon Urbanek developed the macOS port, including the
## R.app GUI, toolchains and packaging.

##
##      The Windows port was originally developed by Guido
## Masarotto (for a while a member of R Core) and Brian
## Ripley, then further by Duncan Murdoch (a former member of
## R Core) and then Jeroen Ooms (base) and Uwe Ligges
## (packages). Tomas Kalibera is the current main developer
## of the Windows port and provides assistance with package
## porting.

##
##      Tomas Kalibera's work has been sponsored by Jan Vitek
## and funded by his European Research Council grant "Evolving
## Language Ecosystems (ELE)".
```

```
## Computing support (including hardware, hosting and
## infrastructure) has been provided/funded by the R
## Foundation, employers of R-Core members (notably WU Wien,
## ETH Zurich, U Oxford and U Iowa) and by Northeastern
## University and the University of Kent.
##
## Distributions of R contain the recommended packages,
## whose authors/contributors are listed in their DESCRIPTION
## files.

# 除了段首, 每一段的余下诸行都缩进5个字符
writeLines(strwrap(x, width = 60, exdent = 5))

## J. D. Beasley, David J. Best, Richard Brent, Kevin Buhr,
## Michael A. Covington, Bill Cleveland, Robert
## Cleveland, G. W. Cran, C. G. Ding, Ulrich Drepper,
## Paul Eggert, J. O. Evans, David M. Gay, H. Frick, G.
## W. Hill, Richard H. Jones, Eric Grosse, Shelby
## Haberman, Bruno Haible, John Hartigan, Andrew Harvey,
## Trevor Hastie, Min Long Lam, George Marsaglia, K. J.
## Martin, Gordon Matzigkeit, C. R. Mckenzie, Jean McRae,
## Cyrus Mehta, Fionn Murtagh, John C. Nash, Finbarr
## O'Sullivan, R. E. Odeh, William Patefield, Nitin
## Patel, Alan Richardson, D. E. Roberts, Patrick
## Royston, Russell Lenth, Ming-Jen Shyu, Richard C.
## Singleton, S. G. Springer, Supoj Sutanthavibul, Irma
## Terpenning, G. E. Thomas, Rob Tibshirani, Wai Wan
## Tsang, Berwin Turlach, Gary V. Vaughan, Michael
## Wichura, Jingbo Wang, M. A. Wong, and the Free
## Software Foundation (for autoconf code and utilities).
## See also files under src/extras.
##
## Many more, too numerous to mention here, have contributed
## by sending bug reports and suggesting various
## improvements.
##
## Simon Davies whilst at the University of Auckland wrote the
## original version of glm().
##
## Julian Harris and Wing Kwong (Tiki) Wan whilst at the
## University of Auckland assisted Ross Ihaka with the
## original Macintosh port.
##
## R was inspired by the S environment which has been
## principally developed by John Chambers, with
```



```
## substantial input from Douglas Bates, Rick Becker,  
## Bill Cleveland, Trevor Hastie, Daryl Pregibon and  
## Allan Wilks.  
  
##  
## A special debt is owed to John Chambers who has graciously  
## contributed advice and encouragement in the early days  
## of R and later became a member of the core team.  
  
##  
## Stefano Iacus (up to 2014, a former member of R Core) and  
## Simon Urbanek developed the macOS port, including the  
## R.app GUI, toolchains and packaging.  
  
##  
## The Windows port was originally developed by Guido  
## Masarotto (for a while a member of R Core) and Brian  
## Ripley, then further by Duncan Murdoch (a former  
## member of R Core) and then Jeroen Ooms (base) and Uwe  
## Ligges (packages). Tomas Kalibera is the current main  
## developer of the Windows port and provides assistance  
## with package porting.  
  
##  
## Tomas Kalibera's work has been sponsored by Jan Vitek and  
## funded by his European Research Council grant  
## "Evolving Language Ecosystems (ELE)".  
  
##  
## Computing support (including hardware, hosting and  
## infrastructure) has been provided/funded by the R  
## Foundation, employers of R-Core members (notably WU  
## Wien, ETH Zurich, U Oxford and U Iowa) and by  
## Northeastern University and the University of Kent.  
  
##  
## Distributions of R contain the recommended packages, whose  
## authors/contributors are listed in their DESCRIPTION  
## files.
```

# 在输出的每一行前面添加前缀

```
writeLines(strwrap(x, prefix = "THANKS> "))
```

```
## THANKS> J. D. Beasley, David J. Best, Richard Brent, Kevin Buhr,  
## THANKS> Michael A. Covington, Bill Cleveland, Robert Cleveland, G. W.  
## THANKS> Cran, C. G. Ding, Ulrich Drepper, Paul Eggert, J. O. Evans,  
## THANKS> David M. Gay, H. Frick, G. W. Hill, Richard H. Jones, Eric  
## THANKS> Grosse, Shelby Haberman, Bruno Haible, John Hartigan, Andrew  
## THANKS> Harvey, Trevor Hastie, Min Long Lam, George Marsaglia, K. J.  
## THANKS> Martin, Gordon Matzigkeit, C. R. Mckenzie, Jean McRae, Cyrus  
## THANKS> Mehta, Fionn Murtagh, John C. Nash, Finbarr O'Sullivan, R. E.  
## THANKS> Odeh, William Patefield, Nitin Patel, Alan Richardson, D. E.
```

```
## THANKS> Roberts, Patrick Royston, Russell Lenth, Ming-Jen Shyu, Richard
## THANKS> C. Singleton, S. G. Springer, Supoj Sutanthavibul, Irma
## THANKS> Terpenning, G. E. Thomas, Rob Tibshirani, Wai Wan Tsang, Berwin
## THANKS> Turlach, Gary V. Vaughan, Michael Wichura, Jingbo Wang, M. A.
## THANKS> Wong, and the Free Software Foundation (for autoconf code and
## THANKS> utilities). See also files under src/extras.
## THANKS>
## THANKS> Many more, too numerous to mention here, have contributed by
## THANKS> sending bug reports and suggesting various improvements.
## THANKS>
## THANKS> Simon Davies whilst at the University of Auckland wrote the
## THANKS> original version of glm().
## THANKS>
## THANKS> Julian Harris and Wing Kwong (Tiki) Wan whilst at the
## THANKS> University of Auckland assisted Ross Ihaka with the original
## THANKS> Macintosh port.
## THANKS>
## THANKS> R was inspired by the S environment which has been principally
## THANKS> developed by John Chambers, with substantial input from Douglas
## THANKS> Bates, Rick Becker, Bill Cleveland, Trevor Hastie, Daryl
## THANKS> Pregibon and Allan Wilks.
## THANKS>
## THANKS> A special debt is owed to John Chambers who has graciously
## THANKS> contributed advice and encouragement in the early days of R and
## THANKS> later became a member of the core team.
## THANKS>
## THANKS> Stefano Iacus (up to 2014, a former member of R Core) and Simon
## THANKS> Urbanek developed the macOS port, including the R.app GUI,
## THANKS> toolchains and packaging.
## THANKS>
## THANKS> The Windows port was originally developed by Guido Masarotto
## THANKS> (for a while a member of R Core) and Brian Ripley, then further
## THANKS> by Duncan Murdoch (a former member of R Core) and then Jeroen
## THANKS> Ooms (base) and Uwe Ligges (packages). Tomas Kalibera is the
## THANKS> current main developer of the Windows port and provides
## THANKS> assistance with package porting.
## THANKS>
## THANKS> Tomas Kalibera's work has been sponsored by Jan Vitek and
## THANKS> funded by his European Research Council grant "Evolving
## THANKS> Language Ecosystems (ELE)".
## THANKS>
## THANKS> Computing support (including hardware, hosting and
## THANKS> infrastructure) has been provided/funded by the R Foundation,
## THANKS> employers of R-Core members (notably WU Wien, ETH Zurich, U
## THANKS> Oxford and U Iowa) and by Northeastern University and the
```



```
## THANKS> University of Kent.
## THANKS>
## THANKS> Distributions of R contain the recommended packages, whose
## THANKS> authors/contributors are listed in their DESCRIPTION files.
```

再举一个烧脑的例子

(C)

```
x <- paste(sapply(
  sample(10, 100, replace = TRUE), # 从1-10个数字中有放回的随机抽取100个数
  function(x) substring("aaaaaaaaaa", 1, x)
), collapse = " ")
sapply(
  10:40,
  function(m)
  c(target = m, actual = max(nchar(strwrap(x, m))))
)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## target  10    11    12    13    14    15    16    17    18    19    20    21    22
## actual   10    10    11    12    13    14    15    16    17    18    19    20    21
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## target   23    24    25    26    27    28    29    30    31    32    33    34
## actual   22    23    24    25    26    27    28    29    30    31    32    33
##      [,26] [,27] [,28] [,29] [,30] [,31]
## target   35    36    37    38    39    40
## actual   34    35    36    37    38    39
```

## 4.14.2 strtrim

```
strtrim(x, width)
```

`strtrim` 函数将字符串 `x` 修剪到特定的显示宽度，返回的字符串向量的长度等于字符串向量 `x` 的长度，如果 `width` 的参数值（它是一个整型向量）的长度小于 `x` 的，就循环补齐。

```
strtrim(c("abcdef", "abcdef", "abcdef"), c(1, 5, 10))
```

```
## [1] "a"      "abcde"  "abcdef"
```

## 4.14.3 strrep

```
strrep(x, times)
```

以给定的次数重复字符串向量中每个元素的个数，并连接字符串的各个副本

```
strrep("ABC", 2)
```

```
## [1] "ABCABC"
```



```
strrep(c("A", "B", "C"), 1 : 3)
## [1] "A"    "BB"   "CCC"
# 创建一个字符串向量，指定每个元素中空格的数量
strrep(" ", 1 : 5)
## [1] " "    " "    " "    " "    " "
```

#### 4.14.4 trimws

```
trimws(x, which = c("both", "left", "right"), whitespace = "[ \t\r\n]")
```

`trimws` 函数用于移除字符串中的空格，这种空格可以来自制表符、回车符和换行符，位置可以位于字符串的开头或者结尾，`which` 参数指定空格的大致位置。举例如下

```
x <- " Some text. "
x
## [1] " Some text. "

trimws(x)
## [1] "Some text."

trimws(x, "l")
## [1] "Some text. "

trimws(x, "r")
## [1] " Some text.

shopping_list <- c("apples x4", "bag of flour", "bag of sugar", "milk x2")

stringr::str_replace(string = shopping_list, pattern = "\d", replacement = "aa")

## [1] "apples xaa"   "bag of flour" "bag of sugar" "milk xaa"
# https://github.com/hadley/stringr/issues/5
# x is vector
str_replace <- function(x, pattern, fun, ...) {
  loc <- gregexpr(pattern, text = x, perl = TRUE)
  matches <- regmatches(x, loc)
  out <- lapply(matches, fun, ...)

  regmatches(x, loc) <- out
  x
}

loc <- gregexpr(pattern = "\d", text = shopping_list, perl = TRUE)
```

```

matches = regmatches(x = shopping_list, loc)

matches

out <- lapply(matches, transform, "aa")

regmatches(x = shopping_list, loc) <- out

shopping_list

str_replace(shopping_list, pattern = "\\\\d", replace = "aa")

```

#### 4.14.5 tolower

tolower 和 toupper 是一对，将大写转小写，小写转大写

```

simpleCap <- function(x) {
  x <- tolower(x)
  s <- strsplit(x, " ")[[1]]
  paste(toupper(substr(s, 1, 1)), substr(s, 2),
    sep = "", collapse = " ")
}

# 参考文献条目里需要将每个英文单词的首字母大写
simpleCap(x = "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS")

## [1] "The Use Of Multiple Measurements In Taxonomic Problems"

```

## 4.15 字符串加密

字符串编码加密，**openssl** 包提供了 sha1 函数<sup>2</sup>

```

library(openssl)

encode_mobile <- function(phone_number) paste("*", paste(toupper(sha1(sha1(charToRaw(paste(phone_number,
# 随意模拟两个手机号
mobile_vec <- c("18601013453", "13811674545")
sapply(mobile_vec, encode_mobile)

##                               18601013453
## "*B1D46D1D62C7280137F0E14249EE500865247B7B"
##                               13811674545

```

<sup>2</sup>参考刘思<sup>④</sup>的两篇博文：[利用 R 函数生成差异化密码](#) 和 [在 R 中各种码的转换](#)



```
## "x0554DA6E403491F58F1567DF2EDEB19186B77173"
```

## 4.16 处理性能

当你对一个很长的字符串进行大量的正则表达式匹配的时候，你需要考虑性能问题了，这时候该考虑启用合适的选项，一般来讲，PCRE 比默认的正则表达式引擎快，`fixed=TRUE` 可以继续加快匹配速度，特别是当每个模式只匹配少量次数时。

连接字符串，`paste/c/bfile/bracket` 函数性能比较 [https://wch.github.io/string\\_builder/index.html](https://wch.github.io/string_builder/index.html)

R 内置的默认正则表达式匹配方式是基于 PCRE 的匹配，`options` 控制 PCRE 默认的三个选项 `PCRE_limit_recursion=NA`、`PCRE_study=10` 和 `PCRE_use_JIT=TRUE`，当前系统环境下 PCRE 的支持情况

```
pcre_config()
```

	UTF-8 Unicode properties	JIT	stack
##	TRUE	TRUE	FALSE

查看 R 环境的 PCRE 配置

```
sapply(c("PCRE_limit_recursion", "PCRE_study", "PCRE_use_JIT"),getOption)
```

	PCRE_study	PCRE_use_JIT
## PCRE_limit_recursion	NA	FALSE
##		TRUE

## 4.17 网络爬虫

用 R 语言写爬虫 `curl`、`httr`、`xml2`、`XML` 和 `rvest` 解析网页<sup>3</sup>

```
# 查看 libcurl 库的版本
```

```
libcurlVersion()
```

```
## [1] "7.68.0"
## attr(),"ssl_version")
## [1] "OpenSSL/1.1.1f"
## attr(),"libssh_version")
## [1] "libssh/0.9.3/openssl/zlib"
## attr(),"protocols")
## [1] "dict"   "file"   "ftp"    "ftps"   "gopher" "http"   "https"  "imap"
## [9] "imaps"  "ldap"   "ldaps"   "pop3"   "pop3s"  "rtmp"   "rtsp"   "scp"
## [17] "sftp"   "smb"    "smbs"   "smtp"   "smtps"  "telnet" "tftp"
```

于主编利用 `tidyRSS` 包抓取解析博客站点的订阅信息，并将此设置为定时任务，创建自动更新内容的博客聚合网站 [Daily R](#)

抓取地震台信息

[一个爬网页的练习：看看 R 邮件列表中最热门的讨论是什么](#)

<sup>3</sup>Jeroen Ooms 已经确认 RCurl 早已经不再维护，取代它的是 curl/httr，不要使用不再维护的 R 包 <https://frie.codes/curl-vs-rcurl/>



## 4.18 文本挖掘

How did Axios rectangle Trump's PDF schedule? A try with R 使用 pdftools 和 magick 处理表格，这两个 R 包分别依赖 Poppler C++ 和 ImageMagick++，在 Ubuntu 上安装 pdftools 和 magick 包

```
sudo apt-get install libpoppler-cpp-dev libmagick++-dev
```

```
install.packages(c("pdftools", "magick"))
```

除了 pdftools 包外，PDF 文档中表格抽取工具还有 `tabulizer`。扫描版 PDF 文档需要 OCR 识别技术支持的 `tesseract` 包

## 4.19 运行环境

```
xfun::session_info()

## R version 4.2.0 (2022-04-22)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.4 LTS
##
## Locale:
##   LC_CTYPE=en_US.UTF-8          LC_NUMERIC=C
##   LC_TIME=en_US.UTF-8          LC_COLLATE=en_US.UTF-8
##   LC_MONETARY=en_US.UTF-8       LC_MESSAGES=en_US.UTF-8
##   LC_PAPER=en_US.UTF-8          LC_NAME=C
##   LC_ADDRESS=C                  LC_TELEPHONE=C
##   LC_MEASUREMENT=en_US.UTF-8   LC_IDENTIFICATION=C
##
## Package version:
##   askpass_1.1       assertthat_0.2.1  base64enc_0.1.3  bookdown_0.26
##   bslib_0.3.1       cli_3.3.0        compiler_4.2.0   cpp11_0.4.2
##   crayon_1.5.1      curl_4.3.2       DBI_1.1.2      digest_0.6.29
##   dplyr_1.0.9       ellipsis_0.3.2   evaluate_0.15  fansi_1.0.3
##   fastmap_1.1.0     fs_1.5.2        generics_0.1.2 glue_1.6.2
##   graphics_4.2.0    grDevices_4.2.0  highr_0.9      htmltools_0.5.2
##   jquerylib_0.1.4   jsonlite_1.8.0   knitr_1.39     lifecycle_1.0.1
##   magrittr_2.0.3    methods_4.2.0   openssl_2.0.1  pillar_1.7.0
##   pkgconfig_2.0.3   purrrr_0.3.4    R6_2.5.1       rappdirs_0.3.3
##   rlang_1.0.2       rmarkdown_2.14   sass_0.4.1     stats_4.2.0
##   stringi_1.7.6    stringr_1.4.0   sys_3.4       sysfonts_0.8.8
##   tibble_3.1.7      tidyrr_1.2.0    tidyselect_1.1.2 tinytex_0.39
##   tools_4.2.0       utf8_1.2.2     utils_4.2.0    vctrs_0.4.1
##   xfun_0.31         yaml_2.3.5
```

## 第五章 正则表达式

Douglas Bates: If you really want to be cautious you could use an octal representation like `sep="\\007"` to get a character that is very unlikely to occur in a factor level.

Ed L. Cashin: I definitely want to be cautious. Instead of the bell character I think I'll use the field separator character, "`\\034`", just because this is the first time I've been able to use it for its intended purpose! ;)

Douglas Bates: Yes, but with "`\\034`" you don't get to make obscure James Bond references :-)

— Douglas Bates and Ed L. Cashin R-help (April 2004)

维基百科关于 [正则表达式的描述](#)，学习正则表达式

```
# 毒鸡汤用来做文本分析
# https://github.com/egotong/nows/blob/master/soul.sql
```

R 内置的三种匹配模式

1. `fixed = TRUE`: 字面意思匹配 exact matching.
2. `perl = TRUE`: 使用 Perl 正则表达式.
3. `fixed = FALSE, perl = FALSE`: 使用 POSIX 1003.2 extended 正则表达式 (默认设置).

不要拘泥于一种解决方案，比如清理数据中正则表达式有 Base R 提供的一套，stringr 又一套，提高效率的工具 RStudio 插件 `regeplain` 和辅助创建正则表达式 `RVerbalExpressions` 包。

有几个名词需要单独拎出来解释的

- `literal character strings` 字面字符串
- `metacharacters` 元字符
- `extended regular expressions` 在下文中约定翻译为默认正则表达式
- `character class` 字符集 `[abc]`
- `Perl-like regular expressions` Perl 风格的正则表达式

以下所述，都不考虑函数中参数 `perl=TRUE` 的情况，R 语言中提供了扩展的（默认的）和 Perl 风格的两套正则表达式。作为入门，我们这里只关注前者，启用 Perl 正则表达式只需在函数如 `grep` 中将选项 `perl = TRUE` 即可，并将后者统一命名为 Perl 正则表达式<sup>1</sup>。

正则表达式 (regular expression, 简称 regexp)，函数 `regexpr` 和 `gregexpr` 的名称就好理解了，在控制台输入 `?regex` 查看 R 支持的正则表达式，这个文档看上百八十回也不过分。R 内支持正则表达式的函数有 `grep`、`grepl`、`sub`、`gsub`、`regexpr`、`gregexpr`、`regexec` 和 `strsplit`。函数 `apropos`、`browseEnv`，

<sup>1</sup>推荐的学习正则表达式的路径可以见统计之都论坛 <https://d.cosx.org/d/420410>



help.search, list.files 和 ls 是通过函数 grep 来使用正则表达式的，它们全都使用 extended regular expressions

```
grep(pattern, x, ignore.case = FALSE, perl = FALSE, value = FALSE,
      fixed = FALSE, useBytes = FALSE, invert = FALSE)
```

匹配模式 pattern 的内容可以用函数 cat 打印出来，注意反斜杠进入 R 字符串中时，需要用两个，反斜杠 \ 本身是转义符，否则会报错。

```
cat("\\\\") # \ 反斜杠是转义字符
```

```
## \
```

```
cat("\\\\.")
```

```
## \.
```

```
cat("\\\\n") # 注意 \\n 表示换行
```

```
## \
```

## 5.1 字符常量

单引号 '、双引号 " 和反引号 ` 三种类型的引用 (quotes) 是 R 语法的一部分<sup>2</sup>，此外反斜杠 \ 用来转义下面的字符

表 5.1: 字符常量表

字符常量	含义
\n	换行 newline
\r	回车 carriage return
\t	制表符 tab
\b	退格 backspace
\a	警报 (铃) alert (bell)
\f	换页 form feed
\v	垂直制表符 vertical tab
\\	反斜杠 backslash \
'	单引号 ASCII apostrophe '
"	双引号 ASCII quotation mark "
`	反引号或沉音符 ASCII grave accent (backtick) `
\nnn	八进制 character with given octal code (1, 2 or 3 digits)
\xnn	十六进制 character with given hex code (1 or 2 hex digits)
\unnnnn	Unicode character with given code (1–4 hex digits)
\Unnnnnnnn	Unicode character with given code (1–8 hex digits)

<sup>2</sup><https://stat.ethz.ch/R-manual/R-devel/library/base/html/Quotes.html>



## 5.2 软件环境

R 内置的正则表达式实现是基于 PCRE ICU TRE iconv 等第三方库，搞清楚自己使用的版本信息是重要的，一些字符集的解释与区域环境有关，如 [:alnum:] 和 [:alpha:] 等，所以获取当前的区域设置也很重要

```
# find a suitable coding for the current locale
localeToCharset(locale = Sys.getlocale("LC_CTYPE"))

## [1] "UTF-8"      "ISO8859-1"

# 软件版本信息
extSoftVersion()
```

```
##                               zlib
##                               "1.2.11"
##                               bzlib
##                               "1.0.8, 13-Jul-2019"
##                               xz
##                               "5.2.4"
##                               PCRE
##                               "10.39 2021-10-29"
##                               ICU
##                               "66.1"
##                               TRE
##                               "TRE 0.8.0 R_fixes (BSD)"
##                               iconv
##                               "glibc 2.31"
##                               readline
##                               "8.0"
##                               BLAS
## "/usr/lib/x86_64-linux-gnublas/libblas.so.3.9.0"
```

```
# 区域及其编码信息
l10n_info()
```

```
## $MBCS
## [1] TRUE
##
## $`UTF-8`
## [1] TRUE
##
## $`Latin-1`
## [1] FALSE
##
## $codeset
## [1] "UTF-8"
```

```
# 表示数字、货币的细节
Sys.localeconv()

##      decimal_point      thousands_sep      grouping      int_curr_symbol
##      "."                  ""                  ""                  "USD "
##      currency_symbol  mon_decimal_point  mon_thousands_sep  mon_grouping
##      "$"                  "."                  ","                  "\003\003"
##      positive_sign      negative_sign      int_frac_digits      frac_digits
##      ""                  "--"                  "2"                  "2"
##      p_cs_precedes      p_sep_by_space      n_cs_precedes      n_sep_by_space
##      "1"                  "0"                  "1"                  "0"
##      p_sign_posn      n_sign_posn
##      "1"                  "1"

# PCRE 启用的配置选项
pcre_config()

##      UTF-8 Unicode properties      JIT      stack
##      TRUE      TRUE      TRUE      FALSE

# 比较全的字符信息
stringi::stri_info()

## $Unicode.version
## [1] "13.0"
##
## $ICU.version
## [1] "66.1"
##
## $Locale
## $Locale$Language
## [1] "en"
##
## $Locale$Country
## [1] "US"
##
## $Locale$Variant
## [1] ""
##
## $Locale$Name
## [1] "en_US"
##
## $Charset.internal
## [1] "UTF-8"  "UTF-16"
##
## $Charset.native
```



```
## $Charset.native$name.friendly
## [1] "UTF-8"
##
## $Charset.native$name.ICU
## [1] "UTF-8"
##
## $Charset.native$name.UTR22
## [1] NA
##
## $Charset.native$name.IBM
## [1] "ibm-1208"
##
## $Charset.native$name.WINDOWS
## [1] "windows-65001"
##
## $Charset.native$name.JAVA
## [1] "UTF-8"
##
## $Charset.native$name.IANA
## [1] "UTF-8"
##
## $Charset.native$name.MIME
## [1] "UTF-8"
##
## $Charset.native$ASCII.subset
## [1] TRUE
##
## $Charset.native$Unicode.1to1
## [1] NA
##
## $Charset.native$CharSize.8bit
## [1] FALSE
##
## $Charset.native$CharSize.min
## [1] 1
##
## $Charset.native$CharSize.max
## [1] 3
##
## $ICU.system
## [1] TRUE
##
## $ICU.UTF8
## [1] TRUE
```



需要临时改变区域环境设置，配合特殊的画图和文本输出要求。

```
# 获取当前默认的区域设置
Sys.getlocale()
foo <- Sys.getlocale()
# 恢复默认的区域设置
Sys.setlocale("LC_ALL", locale = foo)
```

### 5.3 基本概念

正则表达式的构造方式类似算术表达式，通过各种操作组合子（更小的）表达式，整个表达式匹配一个或多个字符<sup>3</sup>。大多数字符，包括所有的字母和数字，是匹配自身的正则表达式。元字符 . \ | ( ) [ { ^ \$ \* + ? 需要转义才能表达其自身的含义，转义的方式是在元字符前面添加反斜杠，如要表达点号 . 需要使用 \.。要注意，它们是否有特殊意义取决于所在的内容。

一个字符集 (character class) 是用一对中括号 [] 括起来的字符列表，用来匹配列表中的任意单个字符，除非列表中的第一个字符是 ^，它用来匹配不在这个列表中的字符。[0123456789] 用来匹配任意单个数字，[^abc] 用来匹配除字符 a,b,c 以外的任意字符。字符范围 (character ranges) 可以通过第一个和最后一个字符指定，中间用连字符 (hyphen) 连接，由于这种解释依赖于区域和具体实现，所以指定字符范围的使用方式最好避免。唯一可移植（便携，通用）的方式是作为字符集，在列表中列出所有的 ASCII 字母，[ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz]。

预定义的一些字符类，它们的解释依赖于当前的语言区域，下面是 POSIX locale 环境下的解释

- [:alnum:] 表示 [:alpha:] 和 [:digit:]，含义是 [0-9A-Za-z]，但是前者与区域和字符集无关，后者依赖于当前的区域设置和字符编码。要注意在这些字符集名 class names 中，中括号 [] 是符号名的一部分，是必须要包含的。在字符集中，大多数元字符失去它们特殊的含义。
- [:alpha:] 表示 [:lower:] 和 [:upper:]
- [:blank:] 表示空格 space 制表符 tab
- [:cntrl:] 表示控制符，在 ASCII 字符集里里，这些字符有八进制代码，从 000 到 037，和 177(DEL)。
- [:digit:] 表示数字 0,1,2,3,4,5,6,7,8,9
- [:graph:] 表示 [:alnum:] 和 [:punct:]。
- [:lower:] 表示当前区域下的小写字母
- [:print:] 表示可打印的字符 [:alnum:], [:punct:] 和空格。
- [:punct:] 表示标点字符  
! " # \$ % & ' ( ) \* + , - . / : ; < = > ? @ [ \ ] ^ \_ ` { | } ~`
- [:space:] 表示空格字符：水平制表符 tab，换行符 newline，垂直制表符 vertical tab，换页符 form feed，回车符 carriage return，空格符 space
- [:xdigit:] 表示 16 进制数字 0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f.

<sup>3</sup>useBytes = TRUE 表示把字符看作字节。字符、字节和比特的关系是，一个字节 byte 八个比特 bit，一个英文字符 character 用一个字节表示，而一个中、日、韩文字符需要两个字节表示

要包含字面的 `]` 就把它放在列表的开头，类似地，要包含字面 `^`，除了开头可以放在任意位置。要包含字面 `-` 把它放在开头或者结尾。只有 `^ - \ ]` 在字符集内是有特殊的含义

点号 `.` 匹配任意单个字符，`\w` 匹配一个词 `word` 字符（是 `[:alnum:]_` 的同义词，一个扩展），而 `\W` 是 `\w` 取反，意味着 `^[:alnum:]_`。`\d`, `\s`, `\D` 和 `\S` 表示数字和空格类和它们的取反

脱字符 `caret ^` 和美元符号 `$` 是元字符，分别匹配一行的开头和结尾。符号 `\<` 和 `\>` 分别匹配一个词的开头和结尾的空字符串。`\b` 匹配词边缘的空字符串，`\B` 匹配不在词边缘的空字符串。词 `word` 的解释依赖于区域和实现。

## 5.4 字符串匹配

默认的匹配方式是贪婪的，会使用尽可能多的匹配次数，这个可以变为最小的匹配次数，通过在其之后添加 `?`，一个正则表达式可能跟着重复量词，下面的限定符都是限定在它前面的正则表达式

表 5.2: 贪婪匹配限定符

符号	描述
<code>?</code>	匹配至多 1 次
<code>*</code>	匹配 0 次或多次
<code>+</code>	匹配至少 1 次
<code>{n}</code>	匹配 <code>n</code> 次
<code>{n,}</code>	匹配至少 <code>n</code> 次
<code>{n,m}</code>	匹配至少 <code>n</code> 次，至多 <code>m</code> 次

## 5.5 级联表达式

Regular expressions may be concatenated; the resulting regular expression matches any string formed by concatenating the substrings that match the concatenated subexpressions.

正则表达式可以是级联 `concatenation` 的，是不是在讲一个正则表达式里面嵌套一个正则表达式？

两个正则表达式可以通过中缀符号 `|` 联合，用两个子表达式的任意一个去匹配字符串，例如 `abba | cde` 要么匹配字符串 `abba` 要么匹配字符串 `cde`，要注意在字符集内，即 `abba|cde`，二选一的匹配不凑效，因为中缀符 `|` 有它的字面意思。

重复匹配 `Repetition` 的优先级高于级联，级联高于 `|`。整个子表达式可以括号括起来覆盖这些优先级规则。

## 5.6 反向引用

反向引用 `\N` 这里 `N` 可取 `1,2,...,9` 匹配被之前第 `N` 个括起来的子表达式匹配的子字符串，例子见 COS 论坛 <https://d.cosx.org/d/420570/5>



## 5.7 命名捕捉

模式 `(?:...)` 包住的字符就是括号分组，但是不做反向查找。模式 `(?<=...)` 和 `(?<!...)` 都是反向查找，它们不允许跟限制符，在 `...` 也不允许出现 `\c`。表 5.3 展示四个反向引用

表 5.3: 环顾四周查找

符号	描述
<code>?=</code>	正向肯定查找
<code>?!</code>	正向否定查找
<code>?&lt;=</code>	反向肯定查找
<code>?&lt;!</code>	反向否定查找

函数 `regexp` 和 `gregexpr` 支持命名捕捉 (named capture)。如果一个组被命名了，如 `(?<first>[A-Z][a-z]+)` 那么，匹配的位置是按名字返回。

下面举个例子说明，从字符串向量 `notables` 中获得了三组匹配 `name.rex` 是一段正则表达式，描述的模式是人名

```
## named capture
notables <- c(" Ben Franklin and Jefferson Davis",
             "\tMillard Fillmore")
# name groups 'first' and 'last'
name.rex <- "(?<first>[:upper:][:lower:]+) (?<last>[:upper:][:lower:]+)"
parsed <- regexp(name.rex, notables, perl = TRUE)
parsed

## [1] 3 2
## attr("match.length")
## [1] 12 16
## attr("index.type")
## [1] "chars"
## attr("useBytes")
## [1] TRUE
## attr("capture.start")
##      first last
## [1,]     3     7
## [2,]     2    10
## attr("capture.length")
##      first last
## [1,]     3     8
## [2,]     7     8
## attr("capture.names")
## [1] "first" "last"
```

`notables` 是一个长度为 2 的字符串向量，所以获得两组匹配，捕捉到匹配开始的位置 `capture.start` 和匹配的长度 `capture.length` 都是两组，按列来看，字符 B 出现在字符串 `Ben Franklin and Jefferson`

Davis 的第三个位置，匹配的长度 Ben 是三个字符，长度是 3，如图 5.1 所示，需要注意的是一定要设置 perl = TRUE 才能使用命名捕捉功能，函数 sub 不支持命名反向引用 Named backreferences



图 5.1: 命名捕捉

Atomic grouping 原子分组, possessive qualifiers 占有限定 and conditional 条件 and recursive 递归等模式超出介绍的范围，不在此处详述，感兴趣的读者可参考，此外，插播一条漫画 5.2



图 5.2: 正则表达式漫画

正则表达式的直观解释 <https://github.com/gadenbuie/regexpain>

## 5.8 表达式注释

The sequence (?# marks the start of a comment which continues up to the next closing parenthesis. Nested parentheses are not permitted. The characters that make up a comment play no part at all in the pattern matching.

If the extended option is set, an unescaped # character outside a character class introduces a comment that continues up to the next newline character in the pattern.

批量转换驼峰式命名

```
old_name <- list.files(".", pattern = "^[A-Z].*.Rmd$")  
new_name <- gsub("rmd", "Rmd", tolower(old_name))  
file.rename(from = old_name, to = new_name)  
  
html_lines <- readLines("https://movie.douban.com/top250")  
doc <- paste0(html_lines, collapse = "")  
  
title_lines <- grep('class="title"', html_lines, value = T)  
titles <- gsub(".*>(.*)<.*", "\\\1", title_lines, perl = T)  
  
gsub(".*>(.*)<.*", "\\\1", '<span class="title">肖生克的救赎</span>', perl = T)
```

### 解析术之 XPath

```
library(xml2)  
dom = read_html(doc)  
title_nodes = xml_find_all(dom, './/span[@class="title"]')  
xml_text(title_nodes)
```

### 解析术之 CSS Selector

```
library(rvest)  
read_html(doc) %>%  
html_nodes('.title') %>% # class="title"的标签  
html_text()
```

## 第六章 数据操作

`data.table` 诞生于 2006 年 4 月 15 日（以在 CRAN 上发布的第一个版本时间为准），是基于 `data.frame` 的扩展和 Base R 的数据操作连贯一些，`dplyr` 诞生于 2014 年 1 月 29 日，号称数据操作的语法，其实二者套路一致，都是借用 SQL 语言的设计，实现方式不同罢了，前者主要依靠 C 语言完成底层数据操作，总代码量 1.29M，C 占 65.6%，后者主要依靠 C++ 语言完成底层数据操作，总代码量 1.2M，C++ 占 34.4%，上层的高级操作接口都是 R 语言。像这样的大神在写代码，码力应该差不多，编程语言会对数据操作的性能有比较大的影响，我想这也是为什么在很多场合下 `data.table` 霸榜！

关于 `data.table` 和 `dplyr` 的对比，参看爆栈网的帖子 <https://stackoverflow.com/questions/21435339>

提示

学习 `data.table` 包最快的方式就是在 R 控制台运行 `example(data.table)` 并研究其输出。

`data.table` 大大加强了 Base R 提供的数据操作，`poorman` 提供最常用的数据操作，但是不依赖 `dplyr`，`fst`，`arrow` 和 `feather` 提供更加高效的数据读写性能。

`collapse` 提供一系列高级和快速的数据操作，支持 Base R、`dplyr`、`tibble`、`data.table`、`plm` 和 `sf` 数据框结构类型。关键的特点有：1. 高级的统计编程，提供一系列统计函数支持在向量、矩阵和数据框上做分组和带权计算。`fastverse` 提供丰富的数据操作和统计计算功能，意图打造一个 `tidyverse` 替代品。

更多参考材料见 [A `data.table` and `dplyr` tour](#)，[Big Data in Economics: Data cleaning and wrangling](#) 和 [DataCamp's `data.table` cheatsheet](#)，关于采用 Base R 还是 `tidyverse` 做数据操作的 [讨论](#)，数据操作的动画展示参考 <https://github.com/gadenbuie/tidyexplain>。

什么是 Base R? Base R 指的是 R 语言/软件的核心组件，由 R Core Team 维护

```
Pkgs <- sapply(list.files(R.home("library")), function(x)
  packageDescription(pkg = x, fields = "Priority"))
names(Pkgs[Pkgs == "base" & !is.na(Pkgs)])  
  
## [1] "base"      "compiler"   "datasets"   "graphics"   "grDevices"  "grid"  
## [7] "methods"   "parallel"   "splines"    "stats"      "stats4"     "tcltk"  
## [13] "tools"     "utils"  
  
names(Pkgs[Pkgs == "recommended" & !is.na(Pkgs)])  
  
## [1] "boot"      "class"     "cluster"    "codetools"  "foreign"  
## [6] "KernSmooth" "lattice"   "MASS"      "Matrix"    "mgcv"  
## [11] "nlme"      "nnet"      "rpart"     "spatial"   "survival"
```

数据变形，分组统计聚合等，用以作为模型的输入，绘图的对象，操作的数据对象是数据框 (`data.frame`) 类型的，而且如果没有特别说明，文中出现的数据集都是 Base R 内置的，第三方 R 包或者来源于网上的

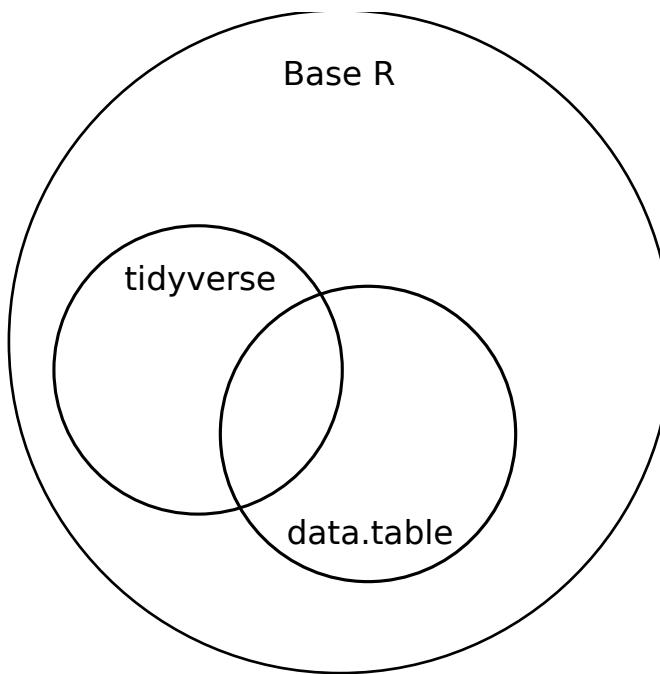


图 6.1: Tidyverse 和 Base R 的关系

数据集都会加以说明。

```
# 给定一个/些 R 包名, 返回该 R 包存放的位置
sapply(.libPaths(), function(pkg_path) {
  c("survival", "ggplot2") %in% .packages(T, lib.loc = pkg_path)
})
```

```
##           /home/runner/work/_temp/Library /opt/R/4.2.0/lib/R/library
## [1,]          FALSE                  TRUE
## [2,]          TRUE                  FALSE
```

## 6.1 查看数据

查看属性

```
str(iris)
```

```
## 'data.frame': 150 obs. of 5 variables:
##   $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##   $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##   $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##   $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##   $ Species     : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

查看部分数据集

```
head(iris, 5)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1       3.5         1.4         0.2   setosa
## 2          4.9       3.0         1.4         0.3   setosa
## 3          4.7       3.2         1.3         0.2   setosa
## 4          4.6       3.1         1.5         0.2   setosa
## 5          5.0       3.6         1.4         0.2 versicolor
```



```
## 1      5.1      3.5      1.4      0.2  setosa
## 2      4.9      3.0      1.4      0.2  setosa
## 3      4.7      3.2      1.3      0.2  setosa
## 4      4.6      3.1      1.5      0.2  setosa
## 5      5.0      3.6      1.4      0.2  setosa
```

```
tail(iris, 5)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 146      6.7      3.0      5.2      2.3  virginica
## 147      6.3      2.5      5.0      1.9  virginica
## 148      6.5      3.0      5.2      2.0  virginica
## 149      6.2      3.4      5.4      2.3  virginica
## 150      5.9      3.0      5.1      1.8  virginica
```

查看文件前（后）5行

```
head -n 5 test.csv
tail -n 5 test.csv
```

对象的类型，存储方式

```
class(iris)
```

```
## [1] "data.frame"
mode(iris)
```

```
## [1] "list"
typeof(iris)
```

```
## [1] "list"
```

查看对象在 R 环境中所占空间的大小

```
object.size(iris)
```

```
## 7256 bytes
```

```
object.size(letters)
```

```
## 1712 bytes
```

```
object.size(ls)
```

```
## 89880 bytes
```

```
format(object.size(library), units = "auto")
```

```
## [1] "1.8 Mb"
```

## 6.2 提取子集

```
subset(x, subset, select, drop = FALSE, ...)
```

参数 `subset` 代表行操作, `select` 代表列操作, 函数 `subset` 从数据框中提取部分数据

```
subset(iris, subset = Species == "virginica" & Sepal.Length > 7.5)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 106      7.6      3.0       6.6      2.1 virginica
## 118      7.7      3.8       6.7      2.2 virginica
## 119      7.7      2.6       6.9      2.3 virginica
## 123      7.7      2.8       6.7      2.0 virginica
## 132      7.9      3.8       6.4      2.0 virginica
## 136      7.7      3.0       6.1      2.3 virginica
```

```
# summary(iris$Sepal.Length)  mean(iris$Sepal.Length)
```

# 且的逻辑

```
# subset(iris, Species == "virginica" & Sepal.Length > 5.8)
subset(iris, Species == "virginica" &
      Sepal.Length == median(Sepal.Length))
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 102      5.8      2.7       5.1      1.9 virginica
## 115      5.8      2.8       5.1      2.4 virginica
## 143      5.8      2.7       5.1      1.9 virginica
```

# 在行的子集范围内

```
subset(iris, Species %in% c("virginica", "versicolor") &
      Sepal.Length == median(Sepal.Length))
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 68       5.8      2.7       4.1      1.0 versicolor
## 83       5.8      2.7       3.9      1.2 versicolor
## 93       5.8      2.6       4.0      1.2 versicolor
## 102      5.8      2.7       5.1      1.9 virginica
## 115      5.8      2.8       5.1      2.4 virginica
## 143      5.8      2.7       5.1      1.9 virginica
```

# 在列的子集内 先选中列

```
subset(iris, Sepal.Length == median(Sepal.Length),
      select = c("Sepal.Length", "Species"))
)
```

```
##   Sepal.Length   Species
## 15       5.8      setosa
## 68       5.8 versicolor
## 83       5.8 versicolor
## 93       5.8 versicolor
```



```
## 102      5.8  virginica
## 115      5.8  virginica
## 143      5.8  virginica
```

高级操作：加入正则表达式筛选

```
## sometimes requiring a logical 'subset' argument is a nuisance
nm <- rownames(state.x77)
start_with_M <- nm %in% grep("^M", nm, value = TRUE)
subset(state.x77, start_with_M, Illiteracy:Murder)
```

```
##          Illiteracy Life Exp Murder
## Maine      0.7    70.39   2.7
## Maryland   0.9    70.22   8.5
## Massachusetts 1.1    71.83   3.3
## Michigan   0.9    70.63  11.1
## Minnesota  0.6    72.96   2.3
## Mississippi 2.4    68.09  12.5
## Missouri   0.8    70.69   9.3
## Montana    0.6    70.56   5.0
```

# 简化

```
subset(state.x77, subset = grepl("^M", rownames(state.x77)), select = Illiteracy:Murder)
```

```
##          Illiteracy Life Exp Murder
## Maine      0.7    70.39   2.7
## Maryland   0.9    70.22   8.5
## Massachusetts 1.1    71.83   3.3
## Michigan   0.9    70.63  11.1
## Minnesota  0.6    72.96   2.3
## Mississippi 2.4    68.09  12.5
## Missouri   0.8    70.69   9.3
## Montana    0.6    70.56   5.0
```

# 继续简化

```
subset(state.x77, grepl("^M", rownames(state.x77)), Illiteracy:Murder)
```

```
##          Illiteracy Life Exp Murder
## Maine      0.7    70.39   2.7
## Maryland   0.9    70.22   8.5
## Massachusetts 1.1    71.83   3.3
## Michigan   0.9    70.63  11.1
## Minnesota  0.6    72.96   2.3
## Mississippi 2.4    68.09  12.5
## Missouri   0.8    70.69   9.3
## Montana    0.6    70.56   5.0
```

## 注意

警告：这是一个为了交互使用打造的便捷函数。对于编程，最好使用标准的子集函数，如 `[`，特别地，参数 `subset` 的非标准计算 (non-standard evaluation)<sup>a</sup> 可能带来意想不到的后果。

<sup>a</sup>Thomas Lumley (2003) Standard nonstandard evaluation rules. <https://developer.r-project.org/nonstandard-eval.pdf>

## C

使用索引 `[`

```
iris[iris$Species == "virginica" & iris$Sepal.Length == 5.8, ]  
  
##      Sepal.Length Sepal.Width Petal.Length Petal.Width   Species  
## 102          5.8        2.7        5.1        1.9  virginica  
## 115          5.8        2.8        5.1        2.4  virginica  
## 143          5.8        2.7        5.1        1.9  virginica  
  
iris[iris$Species == "virginica" &  
  iris$Sepal.Length == median(iris$Sepal.Length), ]  
  
##      Sepal.Length Sepal.Width Petal.Length Petal.Width   Species  
## 102          5.8        2.7        5.1        1.9  virginica  
## 115          5.8        2.8        5.1        2.4  virginica  
## 143          5.8        2.7        5.1        1.9  virginica  
  
iris[  
  iris$Species == "virginica" &  
  iris$Sepal.Length == median(iris$Sepal.Length),  
  c("Sepal.Length", "Species")  
]  
  
##      Sepal.Length   Species  
## 102          5.8  virginica  
## 115          5.8  virginica  
## 143          5.8  virginica  
  
iris[iris$Species == "setosa" & iris$Sepal.Length > 5.5, grepl("Sepal", colnames(iris))]  
  
##      Sepal.Length Sepal.Width  
## 15          5.8        4.0  
## 16          5.7        4.4  
## 19          5.7        3.8  
  
subset(iris,  
  subset = Species == "setosa" & Sepal.Length > 5.5,  
  select = grepl("Sepal", colnames(iris))  
)  
  
##      Sepal.Length Sepal.Width  
## 15          5.8        4.0  
## 16          5.7        4.4  
## 19          5.7        3.8
```

选择操作是针对数据框的列 (变量/特征/字段)

```
library(data.table)
mtcars$cars <- rownames(mtcars)
mtcars_df <- as.data.table(mtcars)

mtcars_df[, .(mpg, disp)] |> head()

##      mpg disp
## 1: 21.0 160
## 2: 21.0 160
## 3: 22.8 108
## 4: 21.4 258
## 5: 18.7 360
## 6: 18.1 225
```

dplyr 版

```
mtcars |>
  dplyr::select(mpg, disp) |>
  head()

##      mpg disp
## 1: 21.0 160
## 2: 21.0 160
## 3: 22.8 108
## 4: 21.4 258
## 5: 18.7 360
## 6: 18.1 225
```

## 6.3 数据重塑

重复测量数据的变形 Reshape Grouped Data，将宽格式 wide 的数据框变长格式 long 的，反之也行。reshape 还支持正则表达式

```
str(Indometh)
```

```
## Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame': 66 obs. of 3 variables:
##   $ Subject: Ord.factor w/ 6 levels "1"<"4"<"2"<"5"<...: 1 1 1 1 1 1 1 1 1 ...
##   $ time   : num  0.25 0.5 0.75 1 1.25 2 3 4 5 6 ...
##   $ conc   : num  1.5 0.94 0.78 0.48 0.37 0.19 0.12 0.11 0.08 0.07 ...
##   - attr(*, "formula")=Class 'formula' language conc ~ time | Subject
##   .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
##   - attr(*, "labels")=List of 2
##     ..$ x: chr "Time since drug administration"
##     ..$ y: chr "Indomethacin concentration"
##   - attr(*, "units")=List of 2
##     ..$ x: chr "(hr)"
##     ..$ y: chr "(mcg/ml)"
```

```
summary(Indometh)

##  Subject      time        conc
## 1:11   Min.   :0.250   Min.   :0.0500
## 4:11   1st Qu.:0.750   1st Qu.:0.1100
## 2:11   Median :2.000   Median :0.3400
## 5:11   Mean    :2.886   Mean    :0.5918
## 6:11   3rd Qu.:5.000   3rd Qu.:0.8325
## 3:11   Max.    :8.000   Max.    :2.7200

# 长的变宽
wide <- reshape(Indometh,
  v.names = "conc", idvar = "Subject",
  timevar = "time", direction = "wide"
)
wide[, 1:6]
```

```
##   Subject conc.0.25 conc.0.5 conc.0.75 conc.1 conc.1.25
## 1       1     1.50    0.94     0.78    0.48    0.37
## 12      2     2.03    1.63     0.71    0.70    0.64
## 23      3     2.72    1.49     1.16    0.80    0.80
## 34      4     1.85    1.39     1.02    0.89    0.59
## 45      5     2.05    1.04     0.81    0.39    0.30
....
```

```
# 宽的变长
reshape(wide, direction = "long")
```

```
##      Subject time conc
## 1.0.25      1 0.25 1.50
## 2.0.25      2 0.25 2.03
## 3.0.25      3 0.25 2.72
## 4.0.25      4 0.25 1.85
## 5.0.25      5 0.25 2.05
....
```

宽的格式变成长的格式 <https://stackoverflow.com/questions/2185252> 或者长的格式变成宽的格式 <https://stackoverflow.com/questions/5890584>

```
set.seed(45)
dat <- data.frame(
  name = rep(c("Orange", "Apple"), each=4),
  numbers = rep(1:4, 2),
  value = rnorm(8))
dat

##   name numbers      value
## 1 Orange      1  0.3407997
## 2 Orange      2 -0.7033403
```



```
## 3 Orange      3 -0.3795377
## 4 Orange      4 -0.7460474
## 5 Apple       1 -0.8981073
## 6 Apple       2 -0.3347941
## 7 Apple       3 -0.5013782
## 8 Apple       4 -0.1745357

reshape(dat, idvar = "name", timevar = "numbers", direction = "wide")

##      name  value.1  value.2  value.3  value.4
## 1 Orange 0.3407997 -0.7033403 -0.3795377 -0.7460474
## 5 Apple -0.8981073 -0.3347941 -0.5013782 -0.1745357

## times need not be numeric
df <- data.frame(id = rep(1:4, rep(2,4)),
                  visit = I(rep(c("Before","After"), 4)),
                  x = rnorm(4), y = runif(4))
df

##   id visit      x      y
## 1  1 Before  1.8090374 0.89106978
## 2  1 After  -0.2301050 0.06920426
## 3  2 Before -1.1304182 0.94623103
## 4  2 After   0.2159889 0.74850150
## 5  3 Before  1.8090374 0.89106978
## 6  3 After  -0.2301050 0.06920426
## 7  4 Before -1.1304182 0.94623103
## 8  4 After   0.2159889 0.74850150

reshape(df, timevar = "visit", idvar = "id", direction = "wide")

##   id x.Before y.Before x.After y.After
## 1 1  1.809037 0.8910698 -0.2301050 0.06920426
## 3 2 -1.130418 0.9462310  0.2159889 0.74850150
## 5 3  1.809037 0.8910698 -0.2301050 0.06920426
## 7 4 -1.130418 0.9462310  0.2159889 0.74850150

## warns that y is really varying
reshape(df, timevar = "visit", idvar = "id", direction = "wide", v.names = "x")

## Warning in reshapeWide(data, idvar = idvar, timevar = timevar, varying =
## varying, : some constant variables (y) are really varying

##   id      y x.Before x.After
## 1 1 0.8910698 1.809037 -0.2301050
## 3 2 0.9462310 -1.130418 0.2159889
## 5 3 0.8910698 1.809037 -0.2301050
## 7 4 0.9462310 -1.130418 0.2159889
```

更加复杂的例子，gambia 数据集，重塑的效果是使得个体水平的长格式变为村庄水平的宽格式



```
# data(gambia, package = "geoR")
# 在线下载数据集
gambia <- read.table(
  file =
  paste("http://www.leg.ufpr.br/lib/exe/fetch.php",
  "pessoais:paulojus:mbgbook:datasets:gambia.txt",
  sep = "/",
  ), header = TRUE
)
head(gambia)
# Building a "village-level" data frame
ind <- paste("x", gambia[, 1], "y", gambia[, 2], sep = "")
village <- gambia[!duplicated(ind), c(1:2, 7:8)]
village$prev <- as.vector(tapply(gambia$pos, ind, mean))
head(village)
```

## 6.4 数据转换

transform 对数据框中的某些列做计算，取对数，将计算的结果单存一列加到数据框中

```
transform(iris[1:6, ], scale.sl = (max(Sepal.Length) - Sepal.Length) / (max(Sepal.Length) - min(Sepal.Length)))
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species scale.sl
## 1          5.1        3.5       1.4        0.2  setosa  0.375
## 2          4.9        3.0       1.4        0.2  setosa  0.625
## 3          4.7        3.2       1.3        0.2  setosa  0.875
## 4          4.6        3.1       1.5        0.2  setosa  1.000
## 5          5.0        3.6       1.4        0.2  setosa  0.500
## 6          5.4        3.9       1.7        0.4  setosa  0.000
```

验证一下 scale.sl 变量的第一个值

```
(max(iris$Sepal.Length) - 5.1) / (max(iris$Sepal.Length) - min(iris$Sepal.Length))
## [1] 0.7777778
```

注意

Warning: This is a convenience function intended for use interactively. For programming it is better to use the standard subsetting arithmetic functions, and in particular the non-standard evaluation of argument transform can have unanticipated consequences.

## 6.5 按列排序

在数据框内，根据 (order) 某一列或几列对行进行排序 (sort)，根据鸢尾花 (iris) 的类别 (Species) 对萼片 (sepal) 的长度进行排序，其余的列随之变化



```
# 先对花瓣的宽度排序，再对花瓣的长度排序
head(iris[order(iris$Species, iris$Petal.Width, iris$Petal.Length), ])
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 14      4.3       3.0      1.1       0.1  setosa
## 13      4.8       3.0      1.4       0.1  setosa
## 38      4.9       3.6      1.4       0.1  setosa
## 10      4.9       3.1      1.5       0.1  setosa
## 33      5.2       4.1      1.5       0.1  setosa
## 23      4.6       3.6      1.0       0.2  setosa
```

sort/ordered 排序，默认是升序

```
dd <- data.frame(
  b = factor(c("Hi", "Med", "Hi", "Low"),
  levels = c("Low", "Med", "Hi"), ordered = TRUE
),
  x = c("A", "D", "A", "C"), y = c(8, 3, 9, 9),
  z = c(1, 1, 1, 2)
)
str(dd)
```

```
## 'data.frame': 4 obs. of 4 variables:
## $ b: Ord.factor w/ 3 levels "Low" < "Med" < "Hi": 3 2 3 1
## $ x: chr "A" "D" "A" "C"
## $ y: num 8 3 9 9
## $ z: num 1 1 1 2
dd[order(~dd[,4], dd[,1]), ]
```

```
##   b x y z
## 4 Low C 9 2
## 2 Med D 3 1
## 1 Hi A 8 1
## 3 Hi A 9 1
```

根据变量 z

```
dd[order(dd$z, dd$b), ]
```

```
##   b x y z
## 2 Med D 3 1
## 1 Hi A 8 1
## 3 Hi A 9 1
## 4 Low C 9 2
```

## 6.6 数据拆分

数据拆分通常是按找某一个分类变量分组，分完组就是计算，计算完就把结果按照原来的分组方式合并

```
## Notice that assignment form is not used since a variable is being added
g <- airquality$Month
l <- split(airquality, g) # 分组
l <- lapply(l, transform, Oz.Z = scale(Ozone)) # 计算: 按月对 Ozone 标准化
aq2 <- unsplit(l, g) # 合并
head(aq2)
```

```
##   Ozone Solar.R Wind Temp Month Day      Oz.Z
## 1     41      190  7.4   67      5   1  0.7822293
## 2     36      118  8.0   72      5   2  0.5572518
## 3     12      149 12.6   74      5   3 -0.5226399
## 4     18      313 11.5   62      5   4 -0.2526670
## 5     NA      NA 14.3   56      5   5       NA
## 6     28      NA 14.9   66      5   6  0.1972879
```

tapply 自带分组的功能，按月份 Month 对 Ozone 中心标准化，其返回一个列表

```
with(airquality, tapply(Ozone, Month, scale))
```

```
## $`5`
##      [,1]
## [1,] 0.78222929
## [2,] 0.55725184
## [3,] -0.52263993
## [4,] -0.25266698
## [5,]       NA
## [6,] 0.19728792
## [7,] -0.02768953
## [8,] -0.20767149
....
```

上面的过程等价于

```
do.call("rbind", lapply(split(airquality, airquality$Month), transform, Oz.Z = scale(Ozone)))

##   Ozone Solar.R Wind Temp Month Day      Oz.Z
## 5.1     41      190  7.4   67      5   1  0.782229293
## 5.2     36      118  8.0   72      5   2  0.557251841
## 5.3     12      149 12.6   74      5   3 -0.522639926
## 5.4     18      313 11.5   62      5   4 -0.252666984
## 5.5     NA      NA 14.3   56      5   5       NA
## 5.6     28      NA 14.9   66      5   6  0.197287919
## 5.7     23      299  8.6   65      5   7 -0.027689532
## 5.8     19      99 13.8   59      5   8 -0.207671494
## 5.9      8      19 20.1   61      5   9 -0.702621887
```



....

由于上面对 Ozone 正态标准化，所以标准化后的 Oz.Z 再按月分组计算方差自然每个月都是 1，而均值都是 0。

```
with(aq2, tapply(Oz.Z, Month, sd, na.rm = TRUE))

## 5 6 7 8 9
## 1 1 1 1 1

with(aq2, tapply(Oz.Z, Month, mean, na.rm = TRUE))

##          5          6          7          8          9
## -4.240273e-17 1.052760e-16 5.841432e-17 5.898060e-17 2.571709e-17
```

循着这个思路，我们可以用 tapply 实现分组计算，上面函数 sd 和 mean 完全可以用自定义的更加复杂的函数替代

cut 函数可以将连续型变量划分为分类变量

```
set.seed(2019)
Z <- stats::rnorm(10)
cut(Z, breaks = -6:6)

##  [1] (0,1]  (-1,0]  (-2,-1]  (0,1]  (-2,-1]  (0,1]  (-1,0]  (0,1]  (-2,-1]
## [10] (-1,0]
## 12 Levels: (-6,-5] (-5,-4] (-4,-3] (-3,-2] (-2,-1] (-1,0] (0,1] (1,2] ... (5,6]
# labels = FALSE 返回每个数所落的区间位置
cut(Z, breaks = -6:6, labels = FALSE)

##  [1] 7 6 5 7 5 7 6 7 5 6
```

我们还可以指定参数 dig.lab 设置分组的精度，ordered 将分组变量看作是有序的，breaks 传递单个数字时，表示分组数，而不是断点

```
cut(Z, breaks = 3, dig.lab = 4, ordered = TRUE)

##  [1] (0.06396,0.9186]  (-0.7881,0.06396]  (-1.643,-0.7881]  (0.06396,0.9186]
##  [5] (-1.643,-0.7881]  (0.06396,0.9186]  (-0.7881,0.06396]  (0.06396,0.9186]
##  [9] (-1.643,-0.7881]  (-0.7881,0.06396]
## Levels: (-1.643,-0.7881] < (-0.7881,0.06396] < (0.06396,0.9186]
```

此时，统计每组的频数，如图 6.2

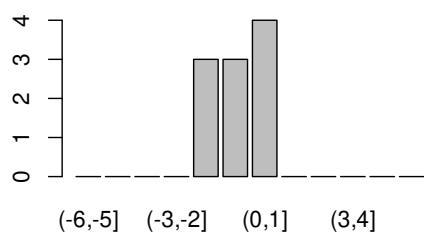
```
# 条形图
plot(cut(Z, breaks = -6:6))

# 直方图
hist(Z, breaks = -6:6)
```

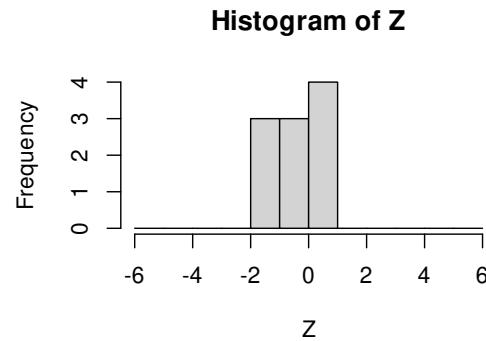
在指定分组数的情况下，我们还想获取分组的断点

```
labs <- levels(cut(Z, 3))
labs

## [1] "(-1.64,-0.788]" "(-0.788,0.064]" "(0.064,0.919]"
```



(a) 条形图



(b) 直方图

图 6.2: 连续型变量分组统计

用正则表达式抽取断点

```
cbind(
  lower = as.numeric(sub("\\\\(.+),.*", "\\\1", labs)),
  upper = as.numeric(sub("[^,]*,([^,]*)\\\\]", "\\\1", labs))
)

##      lower   upper
## [1,] -1.640 -0.788
## [2,] -0.788  0.064
## [3,]  0.064  0.919
```

更多相关函数可以参考 `findInterval` 和 `embed`

`tabulate` 和 `table` 有所不同，它表示排列，由 0 和 1 组成的一个长度为 5 数组，其中 1 有 3 个，则排列组合为

```
combn(5, 3, tabulate, nbins = 5)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    1    1    1    1    1    1    0    0    0    0
## [2,]    1    1    1    0    0    0    1    1    1    0
## [3,]    1    0    0    1    1    0    1    1    0    1
## [4,]    0    1    0    1    0    1    1    0    1    1
## [5,]    0    0    1    0    1    1    0    1    1    1
```

## 6.7 数据合并

`merge` 合并两个数据框

```
authors <- data.frame(
  ## I(*) : use character columns of names to get sensible sort order
  surname = I(c("Tukey", "Venables", "Tierney", "Ripley", "McNeil")),
  nationality = c("US", "Australia", "US", "UK", "Australia"),
```

```
deceased = c("yes", rep("no", 4))
)
authorN <- within(authors, {
  name <- surname
  rm(surname)
})
books <- data.frame(
  name = I(c(
    "Tukey", "Venables", "Tierney",
    "Ripley", "Ripley", "McNeil", "R Core"
  )),
  title = c(
    "Exploratory Data Analysis",
    "Modern Applied Statistics ...",
    "LISP-STAT",
    "Spatial Statistics", "Stochastic Simulation",
    "Interactive Data Analysis",
    "An Introduction to R"
  ),
  other.author = c(
    NA, "Ripley", NA, NA, NA, NA,
    "Venables & Smith"
  )
)

authors

##   surname nationality deceased
## 1   Tukey          US      yes
## 2 Venables    Australia      no
## 3  Tierney          US      no
## 4   Ripley          UK      no
## 5   McNeil    Australia      no

authorN

##   nationality deceased     name
## 1          US      yes  Tukey
## 2    Australia      no Venables
## 3          US      no  Tierney
## 4          UK      no  Ripley
## 5    Australia      no  McNeil

books

##      name                  title  other.author
## 1   Tukey  Exploratory Data Analysis      <NA>
## 2 Venables Modern Applied Statistics ...      Ripley
```



```
## 3 Tierney           LISP-STAT      <NA>
## 4 Ripley            Spatial Statistics <NA>
## 5 Ripley            Stochastic Simulation <NA>
## 6 McNeil             Interactive Data Analysis <NA>
## 7 R Core              An Introduction to R Venables & Smith
```



默认找到同名的列，然后是同名的行合并，多余的没有匹配到的就丢掉

```
merge(authors, books)
```

```
##      name nationality deceased          title other.author
## 1  McNeil    Australia      no  Interactive Data Analysis      <NA>
## 2  Ripley      UK          no   Spatial Statistics      <NA>
## 3  Ripley      UK          no  Stochastic Simulation      <NA>
## 4 Tierney      US          no      LISP-STAT      <NA>
## 5  Tukey      US          yes Exploratory Data Analysis      <NA>
## 6 Venables  Australia      no Modern Applied Statistics ...    Ripley
```

还可以指定合并的列，先按照 surname 合并，留下 surname

```
merge(authors, books, by.x = "surname", by.y = "name")
```

```
##      surname nationality deceased          title other.author
## 1  McNeil    Australia      no  Interactive Data Analysis      <NA>
## 2  Ripley      UK          no   Spatial Statistics      <NA>
## 3  Ripley      UK          no  Stochastic Simulation      <NA>
## 4 Tierney      US          no      LISP-STAT      <NA>
## 5  Tukey      US          yes Exploratory Data Analysis      <NA>
## 6 Venables  Australia      no Modern Applied Statistics ...    Ripley
```

留下的是 name

```
merge(books, authors, by.x = "name", by.y = "surname")
```

```
##      name          title other.author nationality deceased
## 1  McNeil  Interactive Data Analysis      <NA>    Australia      no
## 2  Ripley   Spatial Statistics      <NA>        UK      no
## 3  Ripley  Stochastic Simulation      <NA>        UK      no
## 4 Tierney      LISP-STAT      <NA>        US      no
## 5  Tukey  Exploratory Data Analysis      <NA>        US     yes
## 6 Venables Modern Applied Statistics ...    Ripley    Australia      no
```

为了比较清楚地观察几种合并的区别，这里提供对应的动画展示 <https://github.com/gadenbuie/tidyexplain>

(inner, outer, left, right, cross) join 共 5 种合并方式详情请看 <https://stackoverflow.com/questions/1299871>

cbind 和 rbind 分别是按列和行合并数据框

## 6.8 数据去重

单个数值型向量去重，此时和 unique 函数作用一样

```
(x <- c(9:20, 1:5, 3:7, 0:8))

## [1]  9 10 11 12 13 14 15 16 17 18 19 20  1  2  3  4  5  3  4  5  6  7  0  1  2
## [26]  3  4  5  6  7  8

## extract unique elements
x[!duplicated(x)]
```

```
## [1]  9 10 11 12 13 14 15 16 17 18 19 20  1  2  3  4  5  6  7  0  8

unique(x)
```

```
## [1]  9 10 11 12 13 14 15 16 17 18 19 20  1  2  3  4  5  6  7  0  8
```

数据框类型数据中，去除重复的行，这个重复可以是多个变量对应的向量

```
set.seed(2019)
df <- data.frame(
  x = sample(0:1, 10, replace = T),
  y = sample(0:1, 10, replace = T),
  z = 1:10
)
df
```

```
##   x y  z
## 1 0 0  1
## 2 0 1  2
## 3 1 0  3
## 4 0 0  4
## 5 0 1  5
## 6 0 1  6
## 7 1 0  7
## 8 0 1  8
## 9 0 0  9
## 10 1 0 10

df[!duplicated(df[, c("x", "y")]), ]
```

```
##   x y  z
## 1 0 0  1
## 2 0 1  2
## 3 1 0  3
```

提示

去掉字段 cyl 和 gear 有重复的记录, data.table 方式

```
mtcars_df[!duplicated(mtcars_df, by = c("cyl", "gear"))][,.(mpg, cyl, gear)]  
##      mpg cyl gear  
## 1: 21.0   6   4  
## 2: 22.8   4   4  
## 3: 21.4   6   3  
## 4: 18.7   8   3  
## 5: 21.5   4   3  
## 6: 26.0   4   5  
## 7: 15.8   8   5  
## 8: 19.7   6   5
```

dplyr 方式

```
mtcars |>  
  dplyr::distinct(cyl, gear, .keep_all = TRUE) |>  
  dplyr::select(mpg, cyl, gear)  
  
##      mpg cyl gear  
## Mazda RX4      21.0   6   4  
## Datsun 710     22.8   4   4  
## Hornet 4 Drive 21.4   6   3  
## Hornet Sportabout 18.7   8   3  
## Toyota Corona   21.5   4   3  
## Porsche 914-2    26.0   4   5  
## Ford Pantera L   15.8   8   5  
## Ferrari Dino    19.7   6   5
```

dplyr 的去重操作不需要拷贝一个新的数据对象 mtcars\_df, 并且可以以管道的方式将后续的选择操作连接起来, 代码更加具有可读性。

```
mtcars_df[!duplicated(mtcars_df[, c("cyl", "gear")]), c("mpg", "cyl", "gear")]  
##      mpg cyl gear  
## 1: 21.0   6   4  
## 2: 22.8   4   4  
## 3: 21.4   6   3  
## 4: 18.7   8   3  
## 5: 21.5   4   3  
## 6: 26.0   4   5  
## 7: 15.8   8   5  
## 8: 19.7   6   5
```

Base R 和 data.table 提供的 duplicated() 函数和 [ 函数一起实现去重的操作, 选择操作放在 [ 实现, [ 其实是一个函数

```
x <- 2:4  
x[1]  
## [1] 2  
`[  
  (x, 1)  
## [1] 2
```

## 6.9 数据缺失

缺失数据操作

```
data("airquality")
head(airquality)

##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5    1
## 2    36     118  8.0   72     5    2
## 3    12     149 12.6   74     5    3
## 4    18     313 11.5   62     5    4
## 5    NA      NA 14.3   56     5    5
## 6    28      NA 14.9   66     5    6
```

对缺失值的处理默认是 `na.action = na.omit`

```
# Ozone 最高的那天
aggregate(data = airquality, Ozone ~ Month, max)

##   Month Ozone
## 1     5    115
## 2     6     71
## 3     7    135
## 4     8    168
## 5     9     96

# 每月 Ozone, Solar.R, Wind, Temp 平均值
aggregate(data = airquality, Ozone ~ Month, mean)
```

```
##   Month     Ozone
## 1     5 23.61538
## 2     6 29.44444
## 3     7 59.11538
## 4     8 59.96154
## 5     9 31.44828
```

缺失值处理

```
library(DataExplorer)
plot_missing(airquality)
```

查看包含缺失的记录，不完整的记录

```
airquality[!complete.cases(airquality), ]

##   Ozone Solar.R Wind Temp Month Day
## 5    NA      NA 14.3   56     5    5
## 6    28      NA 14.9   66     5    6
## 10   NA     194  8.6   69     5   10
## 11   7      NA  6.9   74     5   11
```



```
## 25     NA    66 16.6  57    5  25
## 26     NA   266 14.9  58    5  26
## 27     NA    NA  8.0   57    5  27
## 32     NA   286  8.6   78    6  1
## 33     NA   287  9.7   74    6  2
## 34     NA   242 16.1   67    6  3
## 35     NA   186  9.2   84    6  4
## 36     NA   220  8.6   85    6  5
## 37     NA   264 14.3   79    6  6
## 39     NA   273  6.9   87    6  8
## 42     NA   259 10.9   93    6 11
## 43     NA   250  9.2   92    6 12
## 45     NA   332 13.8   80    6 14
## 46     NA   322 11.5   79    6 15
## 52     NA   150  6.3   77    6 21
## 53     NA    59  1.7   76    6 22
## 54     NA    91  4.6   76    6 23
## 55     NA   250  6.3   76    6 24
## 56     NA   135  8.0   75    6 25
## 57     NA   127  8.0   78    6 26
## 58     NA    47 10.3   73    6 27
## 59     NA   98 11.5   80    6 28
## 60     NA    31 14.9   77    6 29
## 61     NA   138  8.0   83    6 30
## 65     NA   101 10.9   84    7  4
## 72     NA   139  8.6   82    7 11
## 75     NA   291 14.9   91    7 14
## 83     NA   258  9.7   81    7 22
## 84     NA   295 11.5   82    7 23
## 96     78     NA  6.9   86    8  4
## 97     35     NA  7.4   85    8  5
## 98     66     NA  4.6   87    8  6
## 102    NA    222  8.6   92    8 10
## 103    NA   137 11.5   86    8 11
## 107    NA    64 11.5   79    8 15
## 115    NA   255 12.6   75    8 23
## 119    NA   153  5.7   88    8 27
## 150    NA   145 13.2   77    9 27
```

Ozone 和 Solar.R 同时包含缺失值的行

```
airquality[is.na(airquality$Ozone) & is.na(airquality$Solar.R), ]
```

```
##     Ozone Solar.R Wind Temp Month Day
## 5     NA     NA 14.3   56    5    5
## 27    NA     NA  8.0   57    5   27
```



## 6.10 数据聚合

分组求和 <https://stackoverflow.com/questions/1660124>

主要是分组统计

```
apropos("apply")  
  
## [1] "apply"      "dendrapply"  "eapply"      "frollapply"  "kernapply"  
## [6] "lapply"      "mapply"      "rapply"      "sapply"      "tapply"  
## [11] "vapply"  
  
# 分组求和 colSums colMeans max  
unique(iris$Species)  
  
## [1] setosa      versicolor  virginica  
## Levels: setosa versicolor virginica  
  
# 分类求和  
# colSums(iris[iris$Species == "setosa", -5])  
# colSums(iris[iris$Species == "virginica", -5])  
colSums(iris[iris$Species == "versicolor", -5])  
  
## Sepal.Length Sepal.Width Petal.Length Petal.Width  
##      296.8       138.5      213.0       66.3  
  
# apply(iris[iris$Species == "setosa", -5], 2, sum)  
# apply(iris[iris$Species == "setosa", -5], 2, mean)  
# apply(iris[iris$Species == "setosa", -5], 2, min)  
# apply(iris[iris$Species == "setosa", -5], 2, max)  
apply(iris[iris$Species == "setosa", -5], 2, quantile)  
  
##      Sepal.Length Sepal.Width Petal.Length Petal.Width  
## 0%          4.3       2.300      1.000       0.1  
## 25%         4.8       3.200      1.400       0.2  
## 50%         5.0       3.400      1.500       0.2  
## 75%         5.2       3.675      1.575       0.3  
## 100%        5.8       4.400      1.900       0.6  
  
aggregate: Compute Summary Statistics of Data Subsets  
  
# 按分类变量 Species 分组求和  
# aggregate(subset(iris, select = -Species), by = list(iris[, "Species"]), FUN = sum)  
aggregate(iris[, -5], list(iris[, 5]), sum)  
  
##      Group.1 Sepal.Length Sepal.Width Petal.Length Petal.Width  
## 1      setosa      250.3      171.4       73.1      12.3  
## 2  versicolor      296.8      138.5      213.0       66.3  
## 3  virginica      329.4      148.7      277.6      101.3  
  
# 先确定位置，假设有很多分类变量  
ind <- which("Species" == colnames(iris))
```



# 分组统计

```
aggregate(iris[, -ind], list(iris[, ind]), sum)

##      Group.1 Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1      setosa      250.3      171.4       73.1       12.3
## 2  versicolor     296.8      138.5      213.0       66.3
## 3  virginica      329.4      148.7      277.6      101.3
```

按照 Species 划分的类别，分组计算，使用公式表示形式，右边一定是分类变量，否则会报错误或者警告，输出奇怪的结果，请读者尝试运行 `aggregate(Species ~ Sepal.Length, data = iris, mean)`。公式法表示分组计算，~ 左手边可以做加 + 减 - 乘 \* 除 / 取余 %% 等数学运算。下面以数据集 iris 为例，只对 Sepal.Length 按 Species 分组计算

```
aggregate(Sepal.Length ~ Species, data = iris, mean)
```

```
##      Species Sepal.Length
## 1      setosa      5.006
## 2  versicolor     5.936
## 3  virginica      6.588
```

与上述分组统计结果一样的命令，在大数据集上，与 aggregate 相比，tapply 要快很多，by 是 tapply 的包裹，处理速度差不多。读者可以构造伪随机数据集验证。

```
# tapply(iris$Sepal.Length, list(iris$Species), mean)
```

```
with(iris, tapply(Sepal.Length, Species, mean))
```

```
##      setosa versicolor  virginica
##      5.006     5.936     6.588
```

```
by(iris$Sepal.Length, iris$Species, mean)
```

```
## iris$Species: setosa
## [1] 5.006
## -----
## iris$Species: versicolor
## [1] 5.936
## -----
## iris$Species: virginica
## [1] 6.588
```

对所有变量按 Species 分组计算

```
aggregate(. ~ Species, data = iris, mean)
```

```
##      Species Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1      setosa      5.006      3.428      1.462      0.246
## 2  versicolor     5.936      2.770      4.260      1.326
## 3  virginica      6.588      2.974      5.552      2.026
```

对变量 Sepal.Length 和 Sepal.Width 求和后，按 Species 分组计算

```
aggregate(Sepal.Length + Sepal.Width ~ Species, data = iris, mean)

##      Species Sepal.Length + Sepal.Width
## 1      setosa          8.434
## 2 versicolor          8.706
## 3 virginica          9.562
```

对多个分类变量做分组计算，在数据集 ChickWeight 中 Chick 和 Diet 都是数字编码的分类变量，其中 Chick 是有序的因子变量，Diet 是无序的因子变量，而 Time 是数值型的变量，表示小鸡出生的天数。

# 查看数据

```
str(ChickWeight)
```

```
## Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame': 578 obs. of 4 variables:
## $ weight: num 42 51 59 64 76 93 106 125 149 171 ...
## $ Time  : num 0 2 4 6 8 10 12 14 16 18 ...
## $ Chick : Ord.factor w/ 50 levels "18"<"16"<"15"<...: 15 15 15 15 15 15 15 15 15 15 ...
## $ Diet  : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
## - attr(*, "formula")=Class 'formula' language weight ~ Time | Chick
## .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
## - attr(*, "outer")=Class 'formula' language ~Diet
## .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
## - attr(*, "labels")=List of 2
## .. $ x: chr "Time"
## .. $ y: chr "Body weight"
## - attr(*, "units")=List of 2
## .. $ x: chr "(days)"
## .. $ y: chr "(gm)"
```

查看数据集 ChickWeight 的前几行

```
head(ChickWeight)
```

```
##   weight Time Chick Diet
## 1     42     0     1     1
## 2     51     2     1     1
## 3     59     4     1     1
## 4     64     6     1     1
## 5     76     8     1     1
## ...
##
```

```
str(ChickWeight)
```

```
## Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame': 578 obs. of 4 variables:
## $ weight: num 42 51 59 64 76 93 106 125 149 171 ...
## $ Time  : num 0 2 4 6 8 10 12 14 16 18 ...
## $ Chick : Ord.factor w/ 50 levels "18"<"16"<"15"<...: 15 15 15 15 15 15 15 15 15 15 ...
## $ Diet  : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
## - attr(*, "formula")=Class 'formula' language weight ~ Time | Chick
```

....

对于数据集 ChickWeight 中的有序变量 Chick, aggregate 会按照既定顺序返回分组计算的结果

```
aggregate(weight ~ Chick, data = ChickWeight, mean)
```

```
##   Chick   weight
## 1     18 37.00000
## 2     16 49.71429
## 3     15 60.12500
## 4     13 67.83333
## 5      9 81.16667
```

....

```
aggregate(weight ~ Diet, data = ChickWeight, mean)
```

```
##   Diet   weight
## 1     1 102.6455
## 2     2 122.6167
## 3     3 142.9500
## 4     4 135.2627
```

分类变量没有用数字编码, 以 CO2 数据集为例, 该数据集描述草植对二氧化碳的吸收情况, Plant 是具有 12 个水平的有序的因子变量, Type 表示植物的源头分别是魁北克 (Quebec) 和密西西比 (Mississippi), Treatment 表示冷却 (chilled) 和不冷却 (nonchilled) 两种处理方式, conc 表示周围环境中二氧化碳的浓度, uptake 表示植物吸收二氧化碳的速率。

```
# 查看数据集
```

```
head(CO2)
```

```
##   Plant   Type Treatment conc uptake
## 1   Qn1 Quebec nonchilled   95   16.0
## 2   Qn1 Quebec nonchilled  175   30.4
## 3   Qn1 Quebec nonchilled  250   34.8
## 4   Qn1 Quebec nonchilled  350   37.2
## 5   Qn1 Quebec nonchilled  500   35.3
## 6   Qn1 Quebec nonchilled  675   39.2
```

```
str(CO2)
```

```
## Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame':  84 obs. of  5 variables:
## $ Plant      : Ord.factor w/ 12 levels "Qn1"<"Qn2"<"Qn3"<...: 1 1 1 1 1 1 1 2 2 2 ...
## $ Type       : Factor w/ 2 levels "Quebec","Mississippi": 1 1 1 1 1 1 1 1 1 1 ...
## $ Treatment: Factor w/ 2 levels "nonchilled","chilled": 1 1 1 1 1 1 1 1 1 1 ...
## $ conc       : num  95 175 250 350 500 675 1000 95 175 250 ...
## $ uptake     : num  16 30.4 34.8 37.2 35.3 39.2 39.7 13.6 27.3 37.1 ...
## - attr(*, "formula")=Class 'formula' language uptake ~ conc | Plant
## ... .- attr(*, ".Environment")=<environment: R_EmptyEnv>
## - attr(*, "outer")=Class 'formula' language ~Treatment * Type
## ... .- attr(*, ".Environment")=<environment: R_EmptyEnv>
```



```
## - attr(*, "labels")=List of 2
##   ..$ x: chr "Ambient carbon dioxide concentration"
##   ..$ y: chr "CO2 uptake rate"
## - attr(*, "units")=List of 2
##   ..$ x: chr "(uL/L)"
##   ..$ y: chr "(umol/m^2 s)"
```

对单个变量分组统计

```
aggregate(uptake ~ Plant, data = CO2, mean)
```

```
##      Plant    uptake
## 1     Qn1 33.22857
## 2     Qn2 35.15714
## 3     Qn3 37.61429
## 4     Qc1 29.97143
## 5     Qc3 32.58571
## 6     Qc2 32.70000
## 7     Mn3 24.11429
## 8     Mn2 27.34286
## 9     Mn1 26.40000
## 10    Mc2 12.14286
## 11    Mc3 17.30000
## 12    Mc1 18.00000
```

```
aggregate(uptake ~ Type, data = CO2, mean)
```

```
##      Type    uptake
## 1     Quebec 33.54286
## 2 Mississippi 20.88333
```

```
aggregate(uptake ~ Treatment, data = CO2, mean)
```

```
##      Treatment    uptake
## 1 nonchilled 30.64286
## 2 chilled 23.78333
```

对多个变量分组统计，查看二氧化碳吸收速率 uptake 随类型 Type 和处理方式 Treatment

```
aggregate(uptake ~ Type + Treatment, data = CO2, mean)
```

```
##      Type Treatment    uptake
## 1     Quebec nonchilled 35.33333
## 2 Mississippi nonchilled 25.95238
## 3     Quebec     chilled 31.75238
## 4 Mississippi     chilled 15.81429
```

```
tapply(CO2$uptake, list(CO2$Type, CO2$Treatment), mean)
```

```
##      nonchilled     chilled
## Quebec      35.33333 31.75238
## Mississippi 25.95238 15.81429
```



```
by(CO2$uptake, list(CO2$Type, CO2$Treatment), mean)
```

```
## : Quebec
## : nonchilled
## [1] 35.33333
## -----
## : Mississippi
## : nonchilled
## [1] 25.95238
## -----
## : Quebec
## : chilled
## [1] 31.75238
## -----
## : Mississippi
## : chilled
## [1] 15.81429
```

在这个例子中 tapply 和 by 的输出结果的表示形式不一样, aggregate 返回一个 data.frame 数据框, tapply 返回一个表格 table, by 返回特殊的数据类型 by。

Function by is an object-oriented wrapper for tapply applied to data frames.

```
# 分组求和
# by(iris[, 1], INDICES = list(iris$Species), FUN = sum)
# by(iris[, 2], INDICES = list(iris$Species), FUN = sum)
by(iris[, 3], INDICES = list(iris$Species), FUN = sum)
```

```
## : setosa
## [1] 73.1
## -----
## : versicolor
## [1] 213
## -----
## : virginica
## [1] 277.6
by(iris[1:3], INDICES = list(iris$Species), FUN = sum)
```

```
## : setosa
## [1] 494.8
## -----
## : versicolor
## [1] 648.3
## -----
## : virginica
## [1] 755.7
```

```
by(iris[1:3], INDICES = list(iris$Species), FUN = summary)
```

```
## : setosa
##   Sepal.Length   Sepal.Width   Petal.Length
##   Min.   :4.300   Min.   :2.300   Min.   :1.000
##   1st Qu.:4.800   1st Qu.:3.200   1st Qu.:1.400
##   Median :5.000   Median :3.400   Median :1.500
##   Mean   :5.006   Mean   :3.428   Mean   :1.462
##   3rd Qu.:5.200   3rd Qu.:3.675   3rd Qu.:1.575
##   Max.   :5.800   Max.   :4.400   Max.   :1.900
## -----
## : versicolor
##   Sepal.Length   Sepal.Width   Petal.Length
##   Min.   :4.900   Min.   :2.000   Min.   :3.00
##   1st Qu.:5.600   1st Qu.:2.525   1st Qu.:4.00
##   Median :5.900   Median :2.800   Median :4.35
##   Mean   :5.936   Mean   :2.770   Mean   :4.26
##   3rd Qu.:6.300   3rd Qu.:3.000   3rd Qu.:4.60
##   Max.   :7.000   Max.   :3.400   Max.   :5.10
## -----
## : virginica
##   Sepal.Length   Sepal.Width   Petal.Length
##   Min.   :4.900   Min.   :2.200   Min.   :4.500
##   1st Qu.:6.225   1st Qu.:2.800   1st Qu.:5.100
##   Median :6.500   Median :3.000   Median :5.550
##   Mean   :6.588   Mean   :2.974   Mean   :5.552
##   3rd Qu.:6.900   3rd Qu.:3.175   3rd Qu.:5.875
##   Max.   :7.900   Max.   :3.800   Max.   :6.900
```

```
by(iris, INDICES = list(iris$Species), FUN = summary)
```

```
## : setosa
##   Sepal.Length   Sepal.Width   Petal.Length   Petal.Width
##   Min.   :4.300   Min.   :2.300   Min.   :1.000   Min.   :0.100
##   1st Qu.:4.800   1st Qu.:3.200   1st Qu.:1.400   1st Qu.:0.200
##   Median :5.000   Median :3.400   Median :1.500   Median :0.200
##   Mean   :5.006   Mean   :3.428   Mean   :1.462   Mean   :0.246
##   3rd Qu.:5.200   3rd Qu.:3.675   3rd Qu.:1.575   3rd Qu.:0.300
##   Max.   :5.800   Max.   :4.400   Max.   :1.900   Max.   :0.600
##   Species
##   setosa   :50
##   versicolor: 0
##   virginica : 0
## 
## 
## 
```

```
## -----
## : versicolor
##   Sepal.Length   Sepal.Width   Petal.Length   Petal.Width      Species
##   Min.   :4.900   Min.   :2.000   Min.   :3.00   Min.   :1.000   setosa   : 0
##   1st Qu.:5.600   1st Qu.:2.525   1st Qu.:4.00   1st Qu.:1.200   versicolor:50
##   Median :5.900   Median :2.800   Median :4.35   Median :1.300   virginica : 0
##   Mean    :5.936   Mean    :2.770   Mean    :4.26   Mean    :1.326
##   3rd Qu.:6.300   3rd Qu.:3.000   3rd Qu.:4.60   3rd Qu.:1.500
##   Max.    :7.000   Max.    :3.400   Max.    :5.10   Max.    :1.800
## -----
## : virginica
##   Sepal.Length   Sepal.Width   Petal.Length   Petal.Width
##   Min.   :4.900   Min.   :2.200   Min.   :4.500   Min.   :1.400
##   1st Qu.:6.225   1st Qu.:2.800   1st Qu.:5.100   1st Qu.:1.800
##   Median :6.500   Median :3.000   Median :5.550   Median :2.000
##   Mean    :6.588   Mean    :2.974   Mean    :5.552   Mean    :2.026
##   3rd Qu.:6.900   3rd Qu.:3.175   3rd Qu.:5.875   3rd Qu.:2.300
##   Max.    :7.900   Max.    :3.800   Max.    :6.900   Max.    :2.500
##   Species
##   setosa   : 0
##   versicolor: 0
##   virginica :50
## 
## 
##
```

Group Averages Over Level Combinations of Factors 分组平均

```
str(warpbreaks)
```

```
## 'data.frame': 54 obs. of 3 variables:
##   $ breaks : num 26 30 54 25 70 52 51 26 67 18 ...
##   $ wool   : Factor w/ 2 levels "A","B": 1 1 1 1 1 1 1 1 1 ...
##   $ tension: Factor w/ 3 levels "L","M","H": 1 1 1 1 1 1 1 1 2 ...
```

```
head(warpbreaks)
```

```
##   breaks wool tension
## 1     26    A       L
## 2     30    A       L
## 3     54    A       L
## 4     25    A       L
## 5     70    A       L
## 6     52    A       L
```

```
ave(warpbreaks$breaks, warpbreaks$wool)
```

```
## [1] 31.03704 31.03704 31.03704 31.03704 31.03704 31.03704 31.03704 31.03704
## [9] 31.03704 31.03704 31.03704 31.03704 31.03704 31.03704 31.03704 31.03704
```



```
## [17] 31.03704 31.03704 31.03704 31.03704 31.03704 31.03704 31.03704 31.03704
## [25] 31.03704 31.03704 31.03704 25.25926 25.25926 25.25926 25.25926 25.25926
## [33] 25.25926 25.25926 25.25926 25.25926 25.25926 25.25926 25.25926 25.25926
## [41] 25.25926 25.25926 25.25926 25.25926 25.25926 25.25926 25.25926 25.25926
## [49] 25.25926 25.25926 25.25926 25.25926 25.25926 25.25926 25.25926 25.25926

with(warpbreaks, ave(breaks, tension, FUN = function(x) mean(x, trim = 0.1)))

## [1] 35.6875 35.6875 35.6875 35.6875 35.6875 35.6875 35.6875 35.6875 35.6875
## [10] 26.3125 26.3125 26.3125 26.3125 26.3125 26.3125 26.3125 26.3125 26.3125
## [19] 21.0625 21.0625 21.0625 21.0625 21.0625 21.0625 21.0625 21.0625 21.0625
## [28] 35.6875 35.6875 35.6875 35.6875 35.6875 35.6875 35.6875 35.6875 35.6875
## [37] 26.3125 26.3125 26.3125 26.3125 26.3125 26.3125 26.3125 26.3125 26.3125
## [46] 21.0625 21.0625 21.0625 21.0625 21.0625 21.0625 21.0625 21.0625 21.0625

# 分组求和
with(warpbreaks, ave(breaks, tension, FUN = function(x) sum(x)))

## [1] 655 655 655 655 655 655 655 655 655 475 475 475 475 475 475 475 475 475 475 390
## [20] 390 390 390 390 390 390 390 390 390 655 655 655 655 655 655 655 655 655 475 475
## [39] 475 475 475 475 475 475 390 390 390 390 390 390 390 390 390 390 390 390 390

# 分组求和
with(iris, ave(Sepal.Length, Species, FUN = function(x) sum(x)))

## [1] 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3
## [13] 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3
## [25] 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3
## [37] 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3 250.3
## [49] 250.3 250.3 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8
## [61] 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8
## [73] 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8
## [85] 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8 296.8
## [97] 296.8 296.8 296.8 296.8 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4
## [109] 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4
## [121] 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4
## [133] 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4
## [145] 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4 329.4
```

## 6.11 表格统计

介绍操作表格的 table, addmargins, prop.table, xtabs, margin.table, ftable 等函数

table 多个分类变量分组计数统计

- 介绍 warpbreaks 和 airquality 纽约空气质量监测数据集二维的数据框
- UCBAdmissions 1973 年加州大学伯克利分校的院系录取数据集 3 维的列联表
- Titanic 4 维的列联表数据泰坦尼克号幸存者数据集



```
with(warpbreaks, table(wool, tension))  
##      tension  
## wool L M H  
##      A 9 9 9  
##      B 9 9 9
```

以 iris 数据集为例, table 的第一个参数是自己制造的第二个分类变量, 原始分类变量是 Species

```
with(iris, table(Sepal.check = Sepal.Length > 7, Species))  
  
##             Species  
## Sepal.check setosa versicolor virginica  
##      FALSE     50        50        38  
##      TRUE      0         0        12  
  
with(iris, table(Sepal.check = Sepal.Length > mean(Sepal.Length), Species))  
  
##             Species  
## Sepal.check setosa versicolor virginica  
##      FALSE     50        24         6  
##      TRUE      0         26        44
```

以 airquality 数据集为例, 看看月份中臭氧含量比较高的几天

```
aiq.tab <- with(airquality, table(Oz.high = Ozone > 80, Month))  
aiq.tab
```

```
##             Month  
## Oz.high  5   6   7   8   9  
##      FALSE 25   9  20  19  27  
##      TRUE   1   0   6   7   2
```

对表格按行和列求和, 即求表格的边际, 查看总体情况

```
addmargins(aiq.tab, 1:2)  
  
##             Month  
## Oz.high   5   6   7   8   9   Sum  
##      FALSE 25   9  20  19  27  100  
##      TRUE   1   0   6   7   2   16  
##      Sum    26   9  26  26  29  116
```

臭氧含量超 80 的天数在每个月的占比, addmargins 的第二个参数 1 表示对列求和

```
aiq.prop <- prop.table(aiq.tab, 2)  
aiq.prop  
  
##             Month  
## Oz.high      5       6       7       8       9  
##      FALSE 0.96153846 1.00000000 0.76923077 0.73076923 0.93103448  
##      TRUE  0.03846154 0.00000000 0.23076923 0.26923077 0.06896552
```

```
aiq.marprop <- addmargins(aiq.prop, 1)
aiq.marprop

##          Month
## Oz.high      5       6       7       8       9
##  FALSE 0.96153846 1.00000000 0.76923077 0.73076923 0.93103448
##  TRUE  0.03846154 0.00000000 0.23076923 0.26923077 0.06896552
##  Sum   1.00000000 1.00000000 1.00000000 1.00000000 1.00000000
```

转换成百分比，将小数四舍五入转化为百分数，保留两位小数点

```
round(100 * aiq.marprop, 2)

##          Month
## Oz.high      5       6       7       8       9
##  FALSE 96.15 100.00 76.92 73.08 93.10
##  TRUE  3.85  0.00 23.08 26.92  6.90
##  Sum   100.00 100.00 100.00 100.00 100.00
pairs(airquality, panel = panel.smooth, main = "airquality data")
```

以 UCBAdmissions 数据集为例，使用 xtabs 函数把数据组织成列联表，先查看数据的内容

```
UCBAdmissions

## , , Dept = A
##
##          Gender
## Admit      Male Female
## Admitted   512    89
## Rejected   313    19
.....
UCBA2DF <- as.data.frame(UCBAdmissions)
UCBA2DF
```

```
##          Admit Gender Dept Freq
## 1  Admitted   Male    A  512
## 2  Rejected   Male    A  313
## 3  Admitted Female   A   89
## 4  Rejected Female   A   19
## 5  Admitted   Male    B 353
....
```

接着将 UCBA2DF 数据集转化为表格的形式

```
UCBA2DF.tab <- xtabs(Freq ~ Gender + Admit + Dept, data = UCBA2DF)
ftable(UCBA2DF.tab)

##          Dept   A   B   C   D   E   F
## Gender Admit
## Male  Admitted   512 353 120 138  53  22
```

```
##           Rejected      313 207 205 279 138 351
## Female Admitted      89  17 202 131  94  24
##           Rejected      19   8 391 244 299 317
```

将录取性别和院系进行对比

```
prop.table(margin.table(UCBA2DF.tab, c(1, 3)), 1)
```

```
##           Dept
## Gender          A          B          C          D          E          F
##   Male  0.30657748 0.20810108 0.12077295 0.15496098 0.07097733 0.13861018
##   Female 0.05885559 0.01362398 0.32316076 0.20435967 0.21416894 0.18583106
```

男生倾向于申请院系 A 和 B，女生倾向于申请院系 C 到 F，院系 A 和 B 是最容易录取的。

## 6.12 索引访问

which 与引用 [ 性能比较，在区间  $[0, 1]$  上生成 10 万个服从均匀分布的随机数，随机抽取其中  $\frac{1}{4}$ 。

```
n <- 100000
x <- runif(n)
i <- logical(n)
i[sample(n, n / 4)] <- TRUE
microbenchmark::microbenchmark(x[i], x[which(i)], times = 1000)
```

**TODO:** 使用 subset 函数与 [ 比较

## 6.13 多维数组

多维数组的行列是怎么定义的 ?array 轴的概念，画个图表示数组

```
array(1:27, c(3, 3, 3))
```

```
## , , 1
##
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
##
## , , 2
##
##      [,1] [,2] [,3]
## [1,]   10   13   16
## [2,]   11   14   17
## [3,]   12   15   18
##
```



```
## , , 3
##
##      [,1] [,2] [,3]
## [1,]    19   22   25
## [2,]    20   23   26
## [3,]    21   24   27
```

垂直于 Z 轴的平面去截三维立方体，3 代表 z 轴，得到三个截面（二维矩阵）

```
asplit(array(1:27, c(3, 3, 3)), 3)
```

```
## [[1]]
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
##
## [[2]]
##      [,1] [,2] [,3]
## [1,]   10   13   16
## [2,]   11   14   17
## [3,]   12   15   18
##
## [[3]]
##      [,1] [,2] [,3]
## [1,]   19   22   25
## [2,]   20   23   26
## [3,]   21   24   27
```

对每个二维矩阵按列求和

```
lapply(asplit(array(1:27, c(3, 3, 3)), 3), apply, 2, sum)
```

```
## [[1]]
## [1] 6 15 24
##
## [[2]]
## [1] 33 42 51
##
## [[3]]
## [1] 60 69 78
```

asplit 和 lapply 组合处理多维数组的计算问题

[三维数组的矩阵运算 abind](#) 包提供更多的数组操作，如合并，替换

数组操作 aperm 数组转置 Array Transposition

asplit 数组拆分其后接 lapply 或者 vapply

apply 数组计算



rray 包 <https://github.com/r-lib/ray>

## 6.14 其它操作

成对的数据操作有 `list` 与 `unlist`、`stack` 与 `unstack`、`class` 与 `unclass`、`attach` 与 `detach` 以及 `with` 和 `within`，它们在数据操作过程中有时会起到一定的补充作用。

### 6.14.1 列表属性

```
# 创建列表
list(...)

# 转化列表
as.list(x, ...)

## S3 method for class 'environment'
as.list(x, all.names = FALSE, sorted = FALSE, ...)

# 检查列表
is.list(x)
is.pairlist(x)

# 转化列表
alist(...)
```

`list` 函数用来构造、转化和检查 R 列表对象。下面创建一个临时列表对象 `tmp`，它包含两个元素 `A` 和 `B`，两个元素都是向量，前者是数值型，后者是字符型

```
(tmp <- list(A = c(1, 2, 3), B = c("a", "b")))

## $A
## [1] 1 2 3
##
## $B
## [1] "a" "b"

unlist(x, recursive = TRUE, use.names = TRUE)
```

`unlist` 函数将给定的列表对象 `x` 简化为原子向量 (atomic vector)，我们发现简化之后变成一个字符型向量

```
unlist(tmp)

##  A1  A2  A3  B1  B2
## "1" "2" "3" "a" "b"

unlist(tmp, use.names = FALSE)

## [1] "1" "2" "3" "a" "b"

unlist 的逆操作是 relist
```

### 6.14.2 堆叠向量

```
stack(x, ...)
## Default S3 method:
stack(x, drop = FALSE, ...)

## S3 method for class 'data.frame'
stack(x, select, drop = FALSE, ...)

unstack(x, ...)
## Default S3 method:
unstack(x, form, ...)
## S3 method for class 'data.frame'
unstack(x, form, ...)
```

stack 与 unstack 将多个向量堆在一起组成一个向量

```
# 查看数据集 PlantGrowth
class(PlantGrowth)

## [1] "data.frame"
head(PlantGrowth)

##   weight group
## 1  4.17  ctrl
## 2  5.58  ctrl
## 3  5.18  ctrl
## 4  6.11  ctrl
## 5  4.50  ctrl
## 6  4.61  ctrl

# 检查默认的公式
formula(PlantGrowth)

## weight ~ group

# 根据公式解除堆叠
# 下面等价于 unstack(PlantGrowth, form = weight ~ group)
(pg <- unstack(PlantGrowth))

##   ctrl trt1 trt2
## 1  4.17 4.81 6.31
## 2  5.58 4.17 5.12
## 3  5.18 4.41 5.54
## 4  6.11 3.59 5.50
## 5  4.50 5.87 5.37
## 6  4.61 3.83 5.29
## 7  5.17 6.03 4.92
## 8  4.53 4.89 6.15
```

```
## 9 5.33 4.32 5.80  
## 10 5.14 4.69 5.26
```

现在再将变量 pg 堆叠起来，还可以指定要堆叠的列

stack(pg)

```
##      values    ind
## 1      4.17  ctrl
## 2      5.58  ctrl
## 3      5.18  ctrl
## 4      6.11  ctrl
## 5      4.50  ctrl
.....
stack(pg, select
```

```
##      values    ind
## 1      4.81 trt1
## 2      4.17 trt1
## 3      4.41 trt1
## 4      3.59 trt1
## 5      5.87 trt1
....
```

形式上和 `reshape` 有一些相似之处，数据框可以由长变宽或由宽变长

### 6.14.3 属性转化

```
class(x)
class(x) <- value
unclass(x)
inherits(x, what, which = FALSE)
```

```
oldClass(x)  
oldClass(x) <- value
```

`class` 和 `unclass` 函数用来查看、设置类属性和取消类属性，常用于面向对象的编程设计中。

```
class(iris)
```

```
## [1] "data.frame"
```

```
class(iris$Species)
```

```
## [1] "factor"
```

```
unclass(iris$Species)
```



#### 6.14.4 绑定环境

```
attach(what,
  pos = 2L, name = deparse(substitute(what), backtick = FALSE),
  warn.conflicts = TRUE
)
detach(name,
  pos = 2L, unload = FALSE, character.only = FALSE,
  force = FALSE
)
```

attach 和 detach 是否绑定数据框的列名，不推荐操作，推荐使用 with

```
attach(iris)  
head(Species)
```

```
## [1] setosa setosa setosa setosa setosa setosa setosa  
## Levels: setosa versicolor virginica  
detach(iris)
```

### 6.14.5 数据环境

```
with(data, expr, ...)  
within(data, expr, ...)  
## S3 method for class 'list'  
within(data, expr, keepAttrs = TRUE, ...)
```

**data** 参数 `data` 用来构造表达式计算的环境。其默认值可以是一个环境，列表，数据框。在 `within` 函数中 `data` 参数只能是列表或数据框。

`expr` 参数 `expr` 包含要计算的表达式。在 `within` 中通常包含一个复合表达式，比如

```
{  
  a <- somefun()  
  b <- otherfun()  
  ...  
  rm(unused1, temp)  
}
```

`with` 和 `within` 计算一组表达式，计算的环境是由数据构造的，后者可以修改原始的数据



```
with(mtcars, mpg[cyl == 8 & disp > 350])  
## [1] 18.7 14.3 10.4 10.4 14.7 19.2 15.8
```

和下面计算的结果一样，但是更加简洁漂亮

```
mtcars$mpg[mtcars$cyl == 8 & mtcars$disp > 350]  
## [1] 18.7 14.3 10.4 10.4 14.7 19.2 15.8
```

`within` 函数可以修改原数据环境中的多个变量，比如删除、修改和添加等

```
# 原数据集 airquality  
head(airquality)
```

```
##   Ozone Solar.R Wind Temp Month Day  
## 1    41     190  7.4   67     5    1  
## 2    36     118  8.0   72     5    2  
## 3    12     149 12.6   74     5    3  
## 4    18     313 11.5   62     5    4  
## 5    NA      NA 14.3   56     5    5  
## 6    28      NA 14.9   66     5    6
```

```
aq <- within(airquality, {  
  lOzone <- log(Ozone) # 取对数  
  Month <- factor(month.abb[Month]) # 字符串型转因子型  
  cTemp <- round((Temp - 32) * 5 / 9, 1) # 从华氏温度到摄氏温度转化  
  S.cT <- Solar.R / cTemp # 使用新创建的变量  
  rm(Day, Temp)  
})  
# 修改后的数据集  
head(aq)
```

```
##   Ozone Solar.R Wind Month      S.cT cTemp    lOzone  
## 1    41     190  7.4   May  9.793814 19.4 3.713572  
## 2    36     118  8.0   May  5.315315 22.2 3.583519  
## 3    12     149 12.6   May  6.394850 23.3 2.484907  
## 4    18     313 11.5   May 18.742515 16.7 2.890372  
## 5    NA      NA 14.3   May       NA 13.3       NA  
## 6    28      NA 14.9   May       NA 18.9 3.332205
```

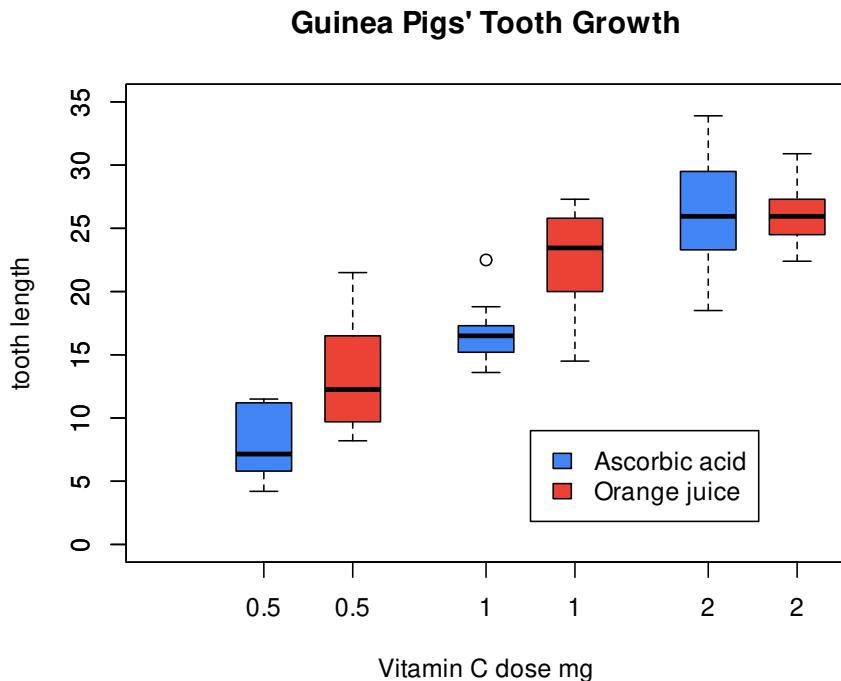
下面再举一个复杂的绘图例子，这个例子来自 `boxplot` 函数

```
with(ToothGrowth, {  
  boxplot(len ~ dose,  
    boxwex = 0.25, at = 1:3 - 0.2,  
    subset = (supp == "VC"), col = "#4285f4",  
    main = "Guinea Pigs' Tooth Growth",  
    xlab = "Vitamin C dose mg",  
    ylab = "tooth length", ylim = c(0, 35)  
})
```

```

boxplot(len ~ dose,
  add = TRUE, boxwex = 0.25, at = 1:3 + 0.2,
  subset = supp == "OJ", col = "#EA4335"
)
legend(2, 9, c("Ascorbic acid", "Orange juice"),
  fill = c("#4285f4", "#EA4335")
)
})

```



将 `boxplot` 函数的 `subset` 参数单独提出来，调用数据子集选择函数 `subset`，这里 `with` 中只包含一个表达式，所以也可以不用大括号

```

with(
  subset(ToothGrowth, supp == "VC"),
  boxplot(len ~ dose,
  boxwex = 0.25, at = 1:3 - 0.2,
  col = "#4285f4", main = "Guinea Pigs' Tooth Growth",
  xlab = "Vitamin C dose mg",
  ylab = "tooth length", ylim = c(0, 35)
)
)
with(
  subset(ToothGrowth, supp == "OJ"),
  boxplot(len ~ dose,
  add = TRUE, boxwex = 0.25, at = 1:3 + 0.2,
  col = "#EA4335"
)
)

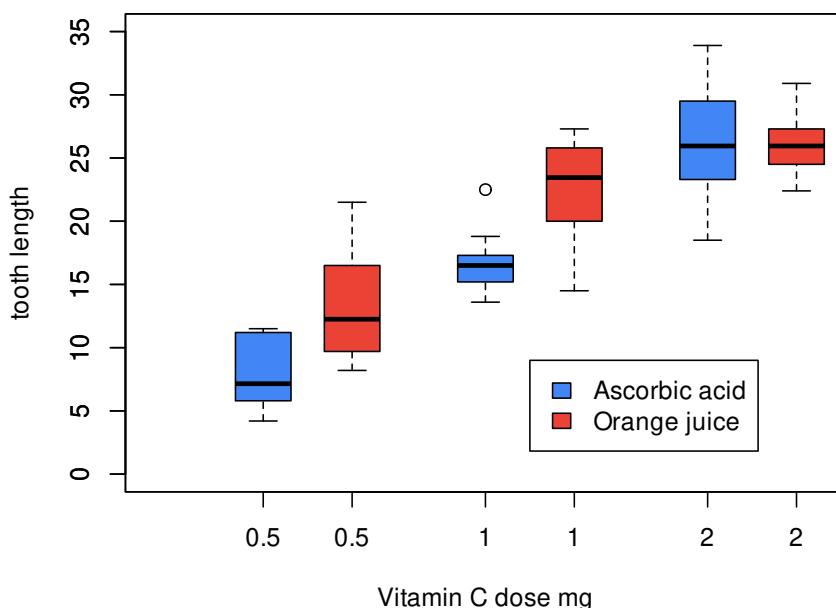
```

```

)
legend(2, 9, c("Ascorbic acid", "Orange juice"),
  fill = c("#4285f4", "#EA4335")
)

```

Guinea Pigs' Tooth Growth



可以作为数据变换 `transform` 的一种替代，它也比较像 `dplyr` 包的 `mutate` 函数

```

within(mtcars[1:5,1:3],{
  disp.cc <- disp * 2.54^3
  disp.l <- disp.cc / 1e3
})

##          mpg cyl disp   disp.l   disp.cc
## Mazda RX4     21.0   6 160 2.621930 2621.930
## Mazda RX4 Wag 21.0   6 160 2.621930 2621.930
## Datsun 710    22.8   4 108 1.769803 1769.803
## Hornet 4 Drive 21.4   6 258 4.227863 4227.863
## Hornet Sportabout 18.7   8 360 5.899343 5899.343

```

# 只能使用已有的列，刚生成的列不能用

```

# transform(
#   mtcars[1:5, 1:3],
#   disp.cc = disp * 2.54^3,
#   disp.l = disp.cc / 1e3
# )
transform(
  mtcars[1:5, 1:3],
  disp.cc = disp * 2.54^3
)

```

```
)
```

```
##          mpg cyl disp disp.cc
## Mazda RX4     21.0   6 160 2621.930
## Mazda RX4 Wag 21.0   6 160 2621.930
## Datsun 710    22.8   4 108 1769.803
## Hornet 4 Drive 21.4   6 258 4227.863
## Hornet Sportabout 18.7   8 360 5899.343
```

`transform` 只能使用已有的列，变换中间生成的列不能用，所以相比于 `transform` 函数，`within` 显得更为灵活

## 6.15 apply 族

表 6.1: apply 函数

函数	输入	输出
<code>apply()</code>	矩阵、数据框	向量
<code>lapply()</code>	向量、列表	列表
<code>sapply()</code>	向量、列表	向量、矩阵
<code>mapply()</code>	多个向量	列表
<code>tapply()</code>	数据框、数组	向量
<code>vapply()</code>	列表	矩阵
<code>eapply()</code>	列表	列表
<code>rapply()</code>	嵌套列表	嵌套列表

除此之外，还有 `dendrapply()` 专门处理层次聚类或分类回归树型结构，而函数 `kernapply()` 用于时间序列的平滑处理

```
# Reproduce example 10.4.3 from Brockwell and Davis (1991) [@Brockwell_1991_Time]
spectrum(sunspot.year, kernel = kernel("daniell", c(11, 7, 3)), log = "no")
```

将函数应用到多个向量，返回一个列表，生成四组服从正态分布  $\mathcal{N}(\mu_i, \sigma_i)$  的随机数，它们的均值和方差依次是  $\mu_i = \sigma_i = 1 \dots 4$

```
means <- 1:4
sds <- 1:4
set.seed(2020)
samples <- mapply(rnorm,
  mean = means, sd = sds,
  MoreArgs = list(n = 10), SIMPLIFY = FALSE
)
samples

## [[1]]
## [1] 1.37697212 1.30154837 -0.09802317 -0.13040590 -1.79653432 1.72057350
```

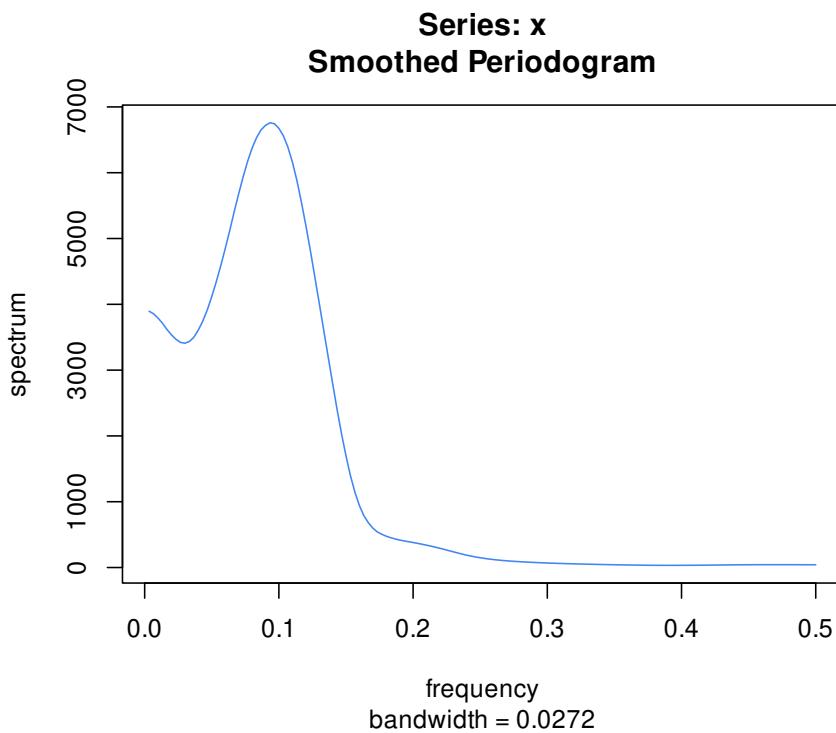


图 6.3: 太阳黑子的频谱

```
## [7] 1.93912102 0.77062225 2.75913135 1.11736679
##
## [[2]]
## [1] 0.2937544 3.8185184 4.3927459 1.2568322 1.7534795 5.6000862
## [7] 5.4079918 -4.0775292 -2.5779499 2.1166070
##
## [[3]]
## [1] 9.523096 6.294548 3.954661 2.780557 5.502806 3.596252 6.893524 5.810155
## [9] 2.557700 3.331296
##
## [[4]]
## [1] 0.7499813 1.0251913 8.3813803 13.7414948 5.5524739 5.1625107
## [7] 2.8576069 4.3040589 1.7588056 5.7887535
```

我们借用图6.4来看一下 mapply 的效果，多组随机数生成非常有助于快速模拟。

```
par(mfrow = c(2, 2), mar = c(2, 2, 2, 2))
invisible(lapply(samples, function(x) {
  plot(x, pch = 16, col = "grey")
  abline(h = mean(x), lwd = 2, col = "darkorange")
}))
```

分别计算每个样本的平均值

```
sapply(samples, mean)
```

```
## [1] 0.8960372 1.7984536 5.0244596 4.9322257
```

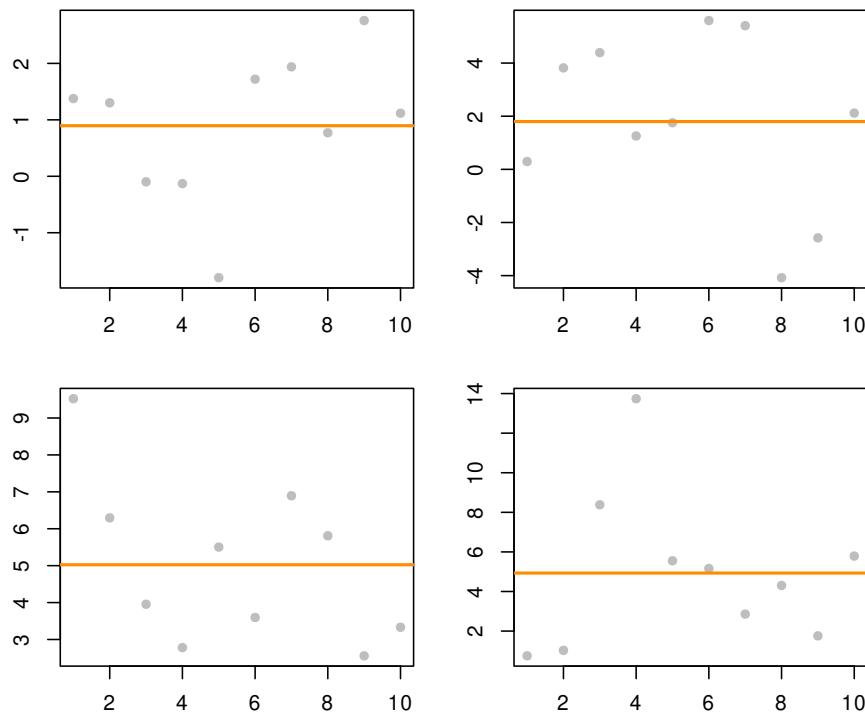


图 6.4: lapply 函数

分别计算每个样本的 1, 2, 3 分位点

```
lapply(samples, quantile, probs = 1:3 / 4)
```

```
## [[1]]
##      25%      50%      75%
## 0.1191382 1.2094576 1.6346732
##
## [[2]]
##      25%      50%      75%
## 0.5345238 1.9350433 4.2491890
##
## [[3]]
##      25%      50%      75%
## 3.397535 4.728734 6.173450
##
## [[4]]
##      25%      50%      75%
## 2.033506 4.733285 5.729684
```

仅用 `sapply()` 函数替换上面的 `lapply()`，我们可以得到一个矩阵，值得注意的是函数 `quantile()` 和 `fivenum()` 算出来的结果有一些差异

```
sapply(samples, quantile, probs = 1:3 / 4)

##      [,1]      [,2]      [,3]      [,4]
## 25% 0.1191382 0.5345238 3.397535 2.033506
```

湘  
潭  
書  
院  
C

```
## 50% 1.2094576 1.9350433 4.728734 4.733285
## 75% 1.6346732 4.2491890 6.173450 5.729684
vapply(samples, fivenum, c(Min. = 0, "1st Qu." = 0, Median = 0, "3rd Qu." = 0, Max. = 0))
## [,1]      [,2]      [,3]      [,4]
## Min.    -1.79653432 -4.0775292 2.557700  0.7499813
## 1st Qu. -0.09802317  0.2937544 3.331296  1.7588056
## Median   1.20945758  1.9350433 4.728734  4.7332848
## 3rd Qu.  1.72057350  4.3927459 6.294548  5.7887535
## Max.     2.75913135  5.6000862 9.523096 13.7414948
```

vapply 和 sapply 类似，但是预先指定返回值类型，这样可以更加安全，有时也更快。

以数据集 presidents 为例，它是一个 ts 对象类型的时间序列数据，记录了 1945 年至 1974 年每个季度美国总统的支持率，这组数据中存在缺失值，以 NA 表示。支持率的变化趋势见图 6.5。

```
plot(presidents)
```

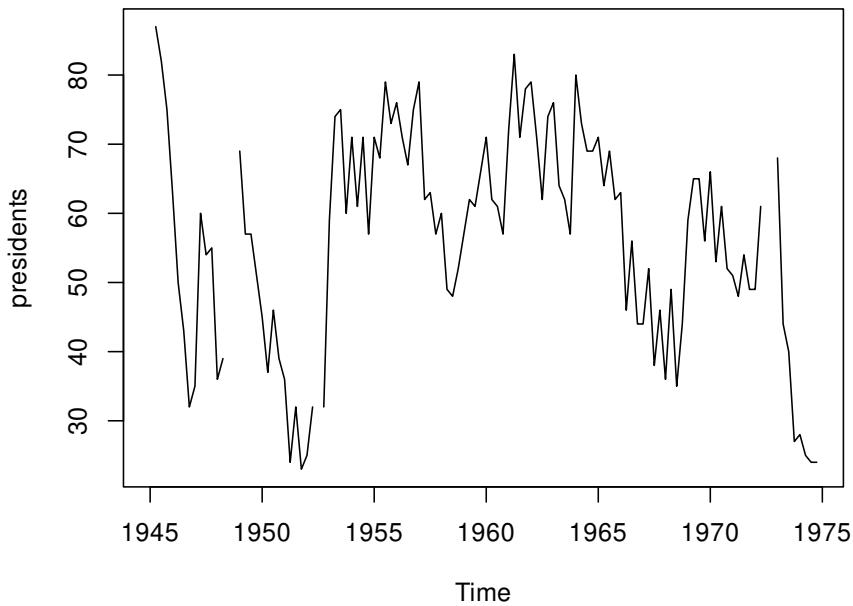


图 6.5: 1945-1974 美国总统的支持率

计算这 30 年每个季度的平均支持率

```
tapply(presidents, cycle(presidents), mean, na.rm = TRUE)
```

```
##      1      2      3      4
## 58.44828 56.43333 57.22222 53.07143
```

cycle() 函数计算序列中每个观察值在周期中的位置，presidents 的周期为 4，根据位置划分组，然后分组求平均，也可以化作如下计算步骤，虽然看起来复杂，但是数据操作的过程很清晰，不再看起来像是一个黑箱。

tapply 函数来做分组求和

```
# 一个变量分组求和
tapply(warpbreaks$breaks, warpbreaks[, 3, drop = FALSE], sum)
```

```
## tension
##   L   M   H
## 655 475 390
```

# 两个变量分组计数

```
with(warpbreaks, table(wool, tension))
```

```
##      tension
## wool L M H
##   A 9 9 9
##   B 9 9 9
```

# 两个变量分组求和

```
dat <- aggregate(breaks ~ wool + tension, data = warpbreaks, sum) |>
  reshape(v.names = "breaks", idvar = "wool", timevar = "tension", direction = "wide", sep = "")
`colnames<-`(dat, gsub(pattern = "(breaks)", x = colnames(dat), replacement = ""))

```

```
##   wool   L   M   H
## 1   A 401 216 221
## 2   B 254 259 169
```

## 6.16 with 选项

注意 data.table 与 Base R 不同的地方

```
# https://github.com/Rdatatable/data.table/issues/4513
# https://d.cosx.org/d/421532-dataratable-base-r
library(data.table)
iris <- as.data.table(iris)

iris[Species == "setosa" & Sepal.Length > 5.5, grepl("Sepal", colnames(iris))]
```

```
## [1] TRUE TRUE FALSE FALSE FALSE
```

需要使用 with = FALSE 选项

```
iris[Species == "setosa" & Sepal.Length > 5.5,
  grepl("Sepal", colnames(iris)),
  with = FALSE
]

##   Sepal.Length Sepal.Width
## 1:          5.8          4.0
## 2:          5.7          4.4
```

```
## 3:      5.7      3.8
```

不使用 with 选项, 用函数 mget() 将字符串转变量

```
iris[
  Species == "setosa" & Sepal.Length > 5.5,
  mget(grep("Sepal", colnames(iris), value = TRUE))
]

##      Sepal.Length Sepal.Width
## 1:      5.8      4.0
## 2:      5.7      4.4
## 3:      5.7      3.8
```

更加 data.table 风格的方式见

```
iris[Species == "setosa" & Sepal.Length > 5.5, .SD, .SDcols = patterns("Sepal")]
```

```
##      Sepal.Length Sepal.Width
## 1:      5.8      4.0
## 2:      5.7      4.4
## 3:      5.7      3.8
```

with 还可以这样用, 直接修改、添加一列

```
df <- expand.grid(x = 1:10, y = 1:10)
df$z <- with(df, x^2 + y^2)
df <- subset(df, z < 100)
df <- df[sample(nrow(df)), ]
head(df)

##      x y  z
## 7  7 1 50
## 8  8 1 65
## 65 5 7 74
## 14 4 2 20
## 37 7 4 65
## 5  5 1 26

library(ggplot2)
ggplot(df, aes(x, y, z = z)) +
  geom_contour()
```

## 6.17 分组聚合

```
methods("aggregate")

## [1] aggregate.data.frame  aggregate.default*  aggregate.formula*
## [4] aggregate.ts
## see '?methods' for accessing help and source code
```

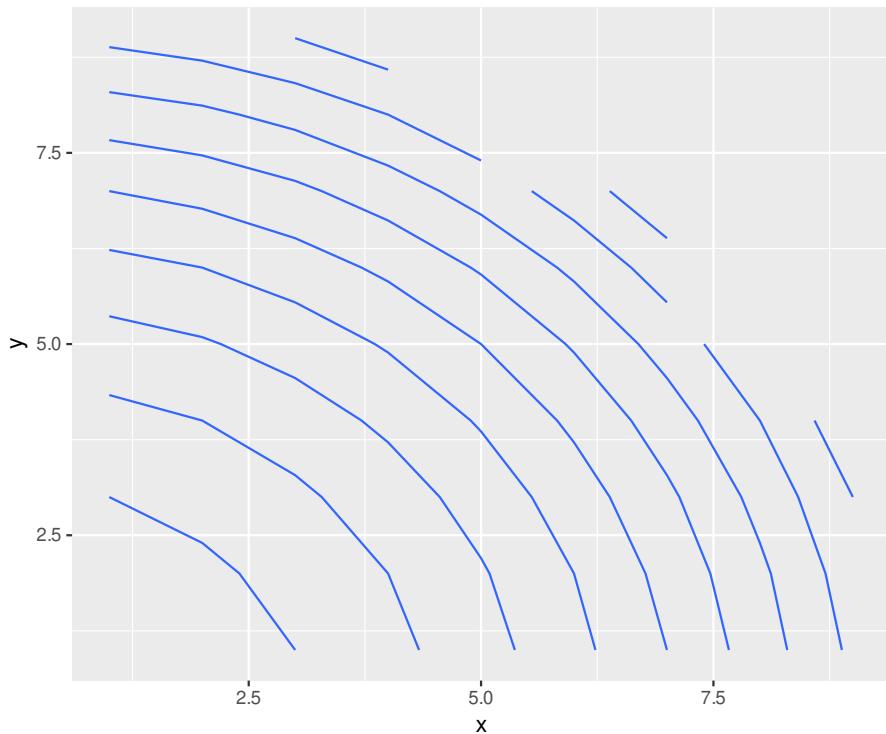


图 6.6: with 操作

```
args("aggregate.data.frame")

## function (x, by, FUN, ..., simplify = TRUE, drop = TRUE)
## NULL

args("aggregate.ts")

## function (x, nfrequency = 1, FUN = sum, ndeltat = 1, ts.eps = getOption("ts.eps"),
##         ...)
## NULL

# getAnywhere(aggregate.formula)
```

按 Species 分组，对 Sepal.Length 中大于平均值的数取平均

```
aggregate(Sepal.Length ~ Species, iris, function(x) mean(x[x > mean(x)]))

##      Species Sepal.Length
## 1      setosa     5.313636
## 2  versicolor     6.375000
## 3  virginica     7.159091

library(data.table)

dt <- data.table(
  x = rep(1:3, each = 3), y = rep(1:3, 3),
  z = rep(c("A", "B", "C"), 3), w = rep(c("a", "b", "a"), each = 3)
)
```



```
dt[, .(x_sum = sum(x), y_sum = sum(y)), by = .(z, w)]  
##      z w x_sum y_sum  
## 1: A a     4     2  
## 2: B a     4     4  
## 3: C a     4     6  
## 4: A b     2     1  
## 5: B b     2     2  
## 6: C b     2     3  
  
dt[, .(x_sum = sum(x), y_sum = sum(y)), by = mget(c("z", "w"))]  
##      z w x_sum y_sum  
## 1: A a     4     2  
## 2: B a     4     4  
## 3: C a     4     6  
## 4: A b     2     1  
## 5: B b     2     2  
## 6: C b     2     3
```

shiny 前端传递字符串向量，借助 `mget()` 函数根据选择的变量分组统计计算，只有一个变量可以使用 `get()` 传递变量给 `data.table`

```
library(shiny)  
  
ui <- fluidPage(  
  fluidRow(  
    column(  
      6,  
      selectInput("input_vars",  
                 label = "变量", # 给筛选框取名  
                 choices = c(z = "z", w = "w"), # 待选的值  
                 selected = "z", # 指定默认值  
                 multiple = TRUE # 允许多选  
      ),  
      DT::dataTableOutput("output_table")  
    )  
  )  
)  
  
library(data.table)  
library(magrittr)  
  
dt <- data.table(  
  x = rep(1:3, each = 3), y = rep(1:3, 3),  
  z = rep(c("A", "B", "C"), 3), w = rep(c("a", "b", "a"), each = 3)  
)
```

```
server <- function(input, output, session) {  
  output$output_table <- DT::renderDataTable(  
  {  
    dt[, .(x_sum = sum(x), y_sum = sum(y)), by = mget(input$input_vars)] |>  
    DT::datatable()  
  },  
  server = FALSE  
)  
}  
  
# 执行  
shinyApp(ui = ui, server = server)
```

## 6.18 合并操作

```
dat1 <- data.frame(x = c(0, 0, 10, 10, 20, 20, 30, 30), y = c(1, 1, 2, 2, 3, 3, 4, 4))  
dat2 <- data.frame(x = c(0, 10, 20, 30), z = c(3, 4, 5, 6))  
  
data.frame(dat1, z = dat2$z[match(dat1$x, dat2$x)])  
  
##      x y z  
## 1    0 1 3  
## 2    0 1 3  
## 3   10 2 4  
## 4   10 2 4  
## 5   20 3 5  
## 6   20 3 5  
## 7   30 4 6  
## 8   30 4 6  
  
merge(dat1, dat2)  
  
##      x y z  
## 1    0 1 3  
## 2    0 1 3  
## 3   10 2 4  
## 4   10 2 4  
## 5   20 3 5  
## 6   20 3 5  
## 7   30 4 6  
## 8   30 4 6
```

保留两个数据集中的所有行

表 6.2: 不同生长环境下植物的干重

group	1	2	3	4	5	6	7	8	9	10
ctrl	4.17	5.58	5.18	6.11	4.50	4.61	5.17	4.53	5.33	5.14
trt1	4.81	4.17	4.41	3.59	5.87	3.83	6.03	4.89	4.32	4.69
trt2	6.31	5.12	5.54	5.50	5.37	5.29	4.92	6.15	5.80	5.26

## 6.19 长宽转换

```
args("reshape")

## function (data, varying = NULL, v.names = NULL, timevar = "time",
##        idvar = "id", ids = 1L:NROW(data), times = seq_along(varying[[1L]]),
##        drop = NULL, direction, new.row.names = NULL, sep = ".",
##        split = if (sep == "") {
##            list(regexp = "[A-Za-z][0-9]", include = TRUE)
##        } else {
##            list(regexp = sep, include = FALSE, fixed = TRUE)
##        })
## NULL
```

PlantGrowth 数据集的重塑操作也可以使用内置的函数 `reshape()` 实现

```
PlantGrowth$id <- rep(1:10, 3)
dat <- reshape(
  data = PlantGrowth, idvar = "group", v.names = "weight",
  timevar = "id", direction = "wide",
  sep = ""
)
knitr::kable(dat,
  caption = "不同生长环境下植物的干重", row.names = FALSE,
  col.names = gsub("(weight)", "", names(dat)),
  align = "c"
)
```

或者，我们也可以使用 `tidyverse` 包提供的 `pivot_wider()` 函数

```
tidyverse::pivot_wider(
  data = PlantGrowth, id_cols = id,
  names_from = group, values_from = weight
)
```

```
## # A tibble: 10 x 4
##       id  ctrl  trt1  trt2
##   <int> <dbl> <dbl> <dbl>
## 1     1   4.17  4.81  6.31
## 2     2   5.58  4.17  5.12
## 3     3   5.18  4.41  5.54
```

```
## 4     4 6.11 3.59 5.5
## 5     5 4.5  5.87 5.37
## 6     6 4.61 3.83 5.29
## 7     7 5.17 6.03 4.92
## 8     8 4.53 4.89 6.15
## 9     9 5.33 4.32 5.8
## 10    10 5.14 4.69 5.26
```

或者，我们还可以使用 **data.table** 包提供的 **dcast()** 函数，用于将长格式的数据框重塑为宽格式的

```
PlantGrowth_DT <- as.data.table(PlantGrowth)
# 纵
dcast(PlantGrowth_DT, id ~ group, value.var = "weight")
```

```
##     id ctrl trt1 trt2
## 1:  1 4.17 4.81 6.31
## 2:  2 5.58 4.17 5.12
## 3:  3 5.18 4.41 5.54
## 4:  4 6.11 3.59 5.50
## 5:  5 4.50 5.87 5.37
## 6:  6 4.61 3.83 5.29
## 7:  7 5.17 6.03 4.92
## 8:  8 4.53 4.89 6.15
## 9:  9 5.33 4.32 5.80
## 10: 10 5.14 4.69 5.26
```

```
# 横
dcast(PlantGrowth_DT, group ~ id, value.var = "weight")
```

```
##     group    1    2    3    4    5    6    7    8    9    10
## 1:  ctrl 4.17 5.58 5.18 6.11 4.50 4.61 5.17 4.53 5.33 5.14
## 2:  trt1 4.81 4.17 4.41 3.59 5.87 3.83 6.03 4.89 4.32 4.69
## 3:  trt2 6.31 5.12 5.54 5.50 5.37 5.29 4.92 6.15 5.80 5.26
```

## 6.20 对符合条件的列操作

```
# 数值型变量的列的位置
which(sapply(iris, is.numeric))

## Sepal.Length Sepal.Width Petal.Length Petal.Width
##           1          2          3          4

iris[, sapply(iris, is.numeric), with = F][Sepal.Length > 7.5]

##     Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1:         7.6       3.0       6.6       2.1
## 2:         7.7       3.8       6.7       2.2
## 3:         7.7       2.6       6.9       2.3
```

```
## 4:      7.7      2.8      6.7      2.0
## 5:      7.9      3.8      6.4      2.0
## 6:      7.7      3.0      6.1      2.3
class(iris)
```

C

```
## [1] "data.table" "data.frame"
```

用 Base R 提供的管道符号 |> 将 `data.table` 数据操作与 `ggplot2` 数据可视化连接起来

```
library(ggplot2)
iris |>
  subset(Species == "setosa" & Sepal.Length > 5.5) |>
  # 行过滤
  # subset(select = grep("Sepal", colnames(iris), value = TRUE)) |> # 列过滤
  subset(select = grepl("Sepal", colnames(iris))) |>
  ggplot(aes(x = Sepal.Length, y = Sepal.Width)) + # 绘图
  geom_point()
```

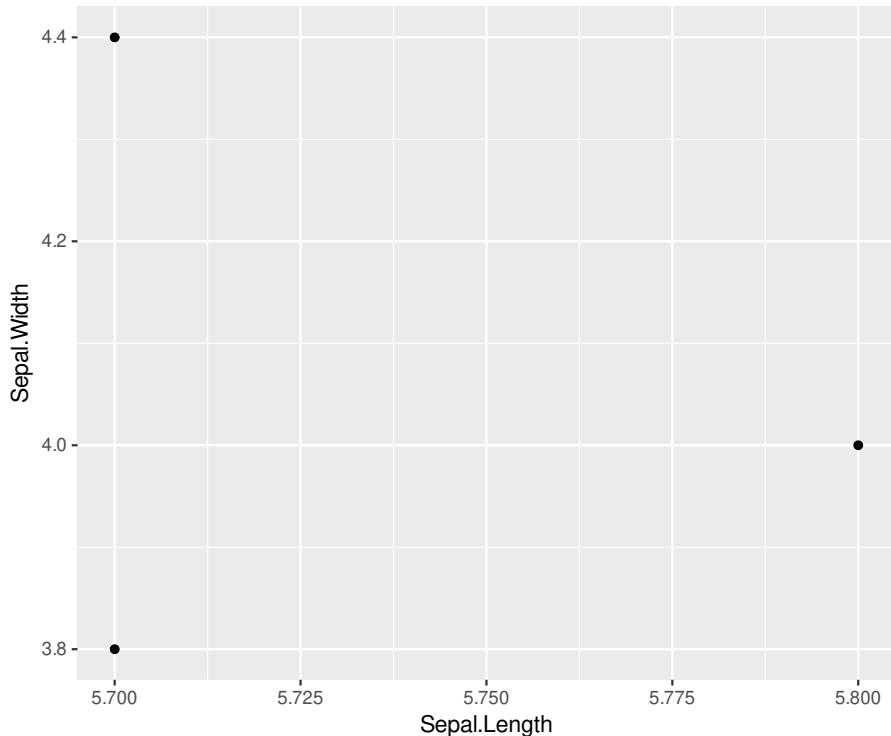


图 6.7: 管道连接数据操作和可视化

## 6.21 CASE WHEN 和 fcase

`CASE WHEN` 是 SQL 中的条件判断语句, `data.table` 中的函数 `fcase()` 可与之等价。值得注意的是, `fcase()` 需要 `data.table` 版本 1.13.0 及以上。

```

dat <- data.table(
  weights = c(56.8, 57.2, 46.3, 38.5),
  gender = c("1", "0", "", "0")
)
# 1 表示男, 0表示女, 空表示未知
transform(dat, gender_cn = fcase(
  gender == "1", "男",
  gender == "0", "女",
  gender == "", "未知"
))

##      weights gender gender_cn
## 1:    56.8      1      男
## 2:    57.2      0      女
## 3:    46.3      ""    未知
## 4:    38.5      0      女

```

## 6.22 数据操作实战

Toby Dylan Hocking 在 useR! 2020 大会上分享的幻灯片 <https://github.com/tdhock/r-devel-emails>

## 6.23 高频数据操作

以数据集 dat 为例介绍常用的数据操作

```

set.seed(2020)
dat <- data.frame(
  num_a = rep(seq(4), each = 4), num_b = rep(seq(4), times = 4),
  group_a = sample(x = letters[1:3], size = 16, replace = T),
  group_b = sample(x = LETTERS[1:3], size = 16, replace = T)
)
dat <- as.data.table(dat)
dat

##      num_a num_b group_a group_b
## 1:      1      1      c      B
## 2:      1      2      b      B
## 3:      1      3      a      B
## 4:      1      4      a      C
## 5:      2      1      b      B
## 6:      2      2      b      C
## 7:      2      3      a      B
## 8:      2      4      a      A
## 9:      3      1      b      C

```



```
## 10:   3   2   b   B
## 11:   3   3   b   B
## 12:   3   4   a   B
## 13:   4   1   b   C
## 14:   4   2   c   B
## 15:   4   3   b   C
## 16:   4   4   a   C
```

### 6.23.1 循环合并

- 问题来源 [Faster version of Reduce\(merge, list\(DT1,DT2,DT3,...\)\) called mergelist \(a la rbindlist\)](#)

### 6.23.2 分组计数

```
dat[, .(length(num_a)), by = .(group_a)] # dat[, .N , by = .(group_a)]
```

```
##   group_a V1
## 1:      c  2
## 2:      b  8
## 3:      a  6
```

```
dat[, .(length(num_a)), by = .(group_b)]
```

```
##   group_b V1
## 1:      B  9
## 2:      C  6
## 3:      A  1
```

```
dat[, .(length(num_a)), by = .(group_a, group_b)]
```

```
##   group_a group_b V1
## 1:      c      B  2
## 2:      b      B  4
## 3:      a      B  3
## 4:      a      C  2
## 5:      b      C  4
## 6:      a      A  1
```

### 6.23.3 分组抽样

以 group\_a 为组别, a、b、c 分别有 6、8、2 条记录

```
# 无放回的抽样
dt_sample_1 <- dat[, .SD[sample(x = .N, size = 2, replace = FALSE)], by = group_a]
# 有放回的随机抽样
dt_sample_2 <- dat[, .SD[sample(x = .N, size = 3, replace = TRUE)], by = group_a]
```

可能存在该组样本不平衡，有的组的样本量不足你想要的样本量。每个组无放回地抽取 4 个样本，如果该组样本量不足 4，则全部抽取全部样本量。

```
dat[, .SD[sample(x = .N, size = min(4, .N))], by = group_a]
```

```
##      group_a num_a num_b group_b
## 1:      c     1     1      B
## 2:      c     4     2      B
## 3:      b     3     2      B
## 4:      b     2     2      C
## 5:      b     2     1      B
## 6:      b     3     3      B
## 7:      a     1     3      B
## 8:      a     2     3      B
## 9:      a     2     4      A
## 10:     a     1     4      C
```

还可以按照指定的比例抽取样本量<sup>1</sup>

#### 6.23.4 分组排序

data.table 包的分组排序问题 <https://d.cosx.org/d/421650-ddatatable/3>

```
dat[with(dat, order(-ave(num_a, group_a, FUN = max), -num_a)), ]
```

```
##      num_a num_b group_a group_b
## 1:     4     1      b      C
## 2:     4     2      c      B
## 3:     4     3      b      C
## 4:     4     4      a      C
## 5:     3     1      b      C
## 6:     3     2      b      B
## 7:     3     3      b      B
## 8:     3     4      a      B
## 9:     2     1      b      B
## 10:    2     2      b      C
## 11:    2     3      a      B
## 12:    2     4      a      A
## 13:    1     1      c      B
## 14:    1     2      b      B
## 15:    1     3      a      B
## 16:    1     4      a      C
```

# num\_a 降序排列，然后对 group\_a 升序排列

```
dat[with(dat, order(-num_a, group_a)), ]
```

```
##      num_a num_b group_a group_b
```

<sup>1</sup><https://stackoverflow.com/questions/18258690/take-randomly-sample-based-on-groups>



```
## 1: 4 4 a C
## 2: 4 1 b C
## 3: 4 3 b C
## 4: 4 2 c B
## 5: 3 4 a B
## 6: 3 1 b C
## 7: 3 2 b B
## 8: 3 3 b B
## 9: 2 3 a B
## 10: 2 4 a A
## 11: 2 1 b B
## 12: 2 2 b C
## 13: 1 3 a B
## 14: 1 4 a C
## 15: 1 2 b B
## 16: 1 1 c B
```

# 简写

```
dat[order(-num_a, group_a)]
```

```
## num_a num_b group_a group_b
## 1: 4 4 a C
## 2: 4 1 b C
## 3: 4 3 b C
## 4: 4 2 c B
## 5: 3 4 a B
## 6: 3 1 b C
## 7: 3 2 b B
## 8: 3 3 b B
## 9: 2 3 a B
## 10: 2 4 a A
## 11: 2 1 b B
## 12: 2 2 b C
## 13: 1 3 a B
## 14: 1 4 a C
## 15: 1 2 b B
## 16: 1 1 c B
```

`setorder()` 函数直接修改原始数据记录的排序

```
setorder(dat, -num_a, group_a)
```

参考多个列分组排序<sup>2</sup>

<sup>2</sup><https://stackoverflow.com/questions/1296646/how-to-sort-a-dataframe-by-multiple-columns>

## 提示

如果数据集 dat 包含缺失值, 考虑去掉缺失值

```
dat[, .(length(!is.na(num_a))), by = .(group_a)]  
##   group_a V1  
## 1:      c  2  
## 2:      b  8  
## 3:      a  6
```

如果数据集 dat 包含重复值, 考虑去掉重复值

```
dat[, .(length(unique(num_a))), by = .(group_a)]  
##   group_a V1  
## 1:      c  2  
## 2:      b  4  
## 3:      a  4
```

按 Species 分组, 对 Sepal.Length 降序排列, 取 Top 3

```
iris <- as.data.table(iris)  
iris[order(-Sepal.Length), .SD[1:3], by = "Species"]  
  
##           Species Sepal.Length Sepal.Width Petal.Length Petal.Width  
## 1:  virginica      7.9       3.8       6.4       2.0  
## 2:  virginica      7.7       3.8       6.7       2.2  
## 3:  virginica      7.7       2.6       6.9       2.3  
## 4: versicolor      7.0       3.2       4.7       1.4  
## 5: versicolor      6.9       3.1       4.9       1.5  
## 6: versicolor      6.8       2.8       4.8       1.4  
## 7:  setosa         5.8       4.0       1.2       0.2  
## 8:  setosa         5.7       4.4       1.5       0.4  
## 9:  setosa         5.7       3.8       1.7       0.3
```

对 iris 各个列排序

```
dat <- head(iris)  
ind <- do.call(what = "order", args = dat[, c(5, 1, 2, 3)])  
dat[ind, ]  
  
##           Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 1:          4.6       3.1       1.5       0.2  setosa  
## 2:          4.7       3.2       1.3       0.2  setosa  
## 3:          4.9       3.0       1.4       0.2  setosa  
## 4:          5.0       3.6       1.4       0.2  setosa  
## 5:          5.1       3.5       1.4       0.2  setosa  
## 6:          5.4       3.9       1.7       0.4  setosa
```

按 Species 分组, 对 Sepal.Length 降序排列, 取 Top 3

```
iris = as.data.table(iris)  
iris[order(-Sepal.Length), .SD[1:3], by = "Species"]
```

表 6.3: iris 数据集原顺序 (左) 和新顺序 (右)

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
4.3	3.0	1.1	0.1	setosa
4.4	2.9	1.4	0.2	setosa
4.4	3.0	1.3	0.2	setosa
4.4	3.2	1.3	0.2	setosa
4.5	2.3	1.3	0.3	setosa
4.6	3.1	1.5	0.2	setosa

```
##          Species Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1:  virginica     7.9       3.8       6.4       2.0
## 2:  virginica     7.7       3.8       6.7       2.2
## 3:  virginica     7.7       2.6       6.9       2.3
## 4: versicolor     7.0       3.2       4.7       1.4
## 5: versicolor     6.9       3.1       4.9       1.5
## 6: versicolor     6.8       2.8       4.8       1.4
## 7:    setosa       5.8       4.0       1.2       0.2
## 8:    setosa       5.7       4.4       1.5       0.4
## 9:    setosa       5.7       3.8       1.7       0.3
```

对 iris 各个列排序, 依次对第 5、1、2、3 列升序排列

```
ind <- do.call(what = "order", args = iris[,c(5,1,2,3)])
head(iris[ind, ])
```

```
##          Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1:        4.3       3.0       1.1       0.1  setosa
## 2:        4.4       2.9       1.4       0.2  setosa
## 3:        4.4       3.0       1.3       0.2  setosa
## 4:        4.4       3.2       1.3       0.2  setosa
## 5:        4.5       2.3       1.3       0.3  setosa
## 6:        4.6       3.1       1.5       0.2  setosa
```

# 第七章 高级数据操作

```
library(data.table)
```

介绍 data.table 处理数据的方式，对标 dplyr 的基本操作

## 7.1 基础介绍

```
# 用一个真实的数据集替换，让每一个操作都有实际含义和价值 mtcars
```

```
DT <- data.table(  
  x = rep(c("b", "a", "c"), each = 3),  
  v = c(1, 1, 1, 2, 2, 1, 1, 2, 2),  
  y = c(1, 3, 6), a = 1:9, b = 9:1  
)  
DT
```

```
##      x v y a b  
## 1: b 1 1 1 9  
## 2: b 1 3 2 8  
## 3: b 1 6 3 7  
## 4: a 2 1 4 6  
## 5: a 2 3 5 5  
## 6: a 1 6 6 4  
## 7: c 1 1 7 3  
## 8: c 2 3 8 2  
## 9: c 2 6 9 1
```

```
# 分组求和
```

```
DT[, sum(v), by = .(y %% 2)]
```

```
##      y V1  
## 1: 1  9  
## 2: 0  4  
DT[, sum(v), by = .(bool = y %% 2)]
```

```
##      bool V1  
## 1:     1  9  
## 2:     0  4
```



```
DT[, .SD[2], by = x] # 每组第二行
```

```
##      x v y a b
## 1: b 1 3 2 8
## 2: a 2 3 5 5
## 3: c 2 3 8 2
```

```
DT[, tail(.SD, 2), by = x] # 每组最后两行
```

```
##      x v y a b
## 1: b 1 3 2 8
## 2: b 1 6 3 7
## 3: a 2 3 5 5
## 4: a 1 6 6 4
## 5: c 2 3 8 2
## 6: c 2 6 9 1
```

# 除了 x 列外, 所有列都按 x 分组求和

```
DT[, lapply(.SD, sum), by = x]
```

```
##      x v y a b
## 1: b 3 10 6 24
## 2: a 5 10 15 15
## 3: c 5 10 24 6
```

# 各个列都按 x 分组取最小

```
DT[, .SD[which.min(v)], by = x] # 分组嵌套查询
```

```
##      x v y a b
## 1: b 1 1 1 9
## 2: a 1 6 6 4
## 3: c 1 1 7 3
```

```
DT[, list(MySum = sum(v), MyMin = min(v), MyMax = max(v)), by = .(x, y %% 2)] # 表达式嵌套
```

```
##      x y MySum MyMin MyMax
## 1: b 1     2     1     1
## 2: b 0     1     1     1
## 3: a 1     4     2     2
## 4: a 0     1     1     1
## 5: c 1     3     1     2
## 6: c 0     2     2     2
```

```
DT[, .(a = .(a), b = .(b)), by = x] # 按 x 分组, 将 a, b 两列的值列出来
```

```
##      x     a     b
## 1: b 1,2,3 9,8,7
## 2: a 4,5,6 6,5,4
## 3: c 7,8,9 3,2,1
```

```
DT[, .(seq = min(a):max(b)), by = x] # 列操作不仅仅是聚合
```

```
##      x seq
## 1: b  1
## 2: b  2
## 3: b  3
## 4: b  4
## 5: b  5
## 6: b  6
## 7: b  7
## 8: b  8
## 9: b  9
## 10: a 4
## 11: a 5
## 12: a 6
## 13: c 7
## 14: c 6
## 15: c 5
## 16: c 4
## 17: c 3
```

# 按 x 分组对 v 求和, 然后过滤出和小于 20 的行

```
DT[, sum(v), by = x][V1 < 20] # 组合查询
```

```
##      x V1
## 1: b  3
## 2: a  5
## 3: c  5
```

```
DT[, sum(v), by = x][order(-V1)] # 对结果排序
```

```
##      x V1
## 1: a  5
## 2: c  5
## 3: b  3
```

```
DT[, c(.N, lapply(.SD, sum)), by = x] # 计算每一组的和, 每一组的观测数
```

```
##      x N v  y  a  b
## 1: b 3 3 10  6 24
## 2: a 3 5 10 15 15
## 3: c 3 5 10 24  6
```

# 两个复杂的操作, 还没弄清楚这个技术存在的意义

```
DT[, 
  {
    tmp <- mean(y)
    .(a = a - tmp, b = b - tmp)
```



```
  },
  by = x
] # anonymous lambda in 'j', j accepts any valid

##      x          a          b
## 1: b -2.3333333 5.6666667
## 2: b -1.3333333 4.6666667
## 3: b -0.3333333 3.6666667
## 4: a  0.6666667 2.6666667
## 5: a  1.6666667 1.6666667
## 6: a  2.6666667 0.6666667
## 7: c  3.6666667 -0.3333333
## 8: c  4.6666667 -1.3333333
## 9: c  5.6666667 -2.3333333

# using rleid, get max(y) and min of all cols in .SDcols for each consecutive run of 'v'
DT[, c(.y = max(y)), lapply(.SD, min)), by = rleid(v), .SDcols = v:b]

##      rleid y v y a b
## 1:      1 6 1 1 1 7
## 2:      2 3 2 1 4 5
## 3:      3 6 1 1 6 3
## 4:      4 6 2 3 8 1
```

### 7.1.1 过滤

```
mtcars_df <- as.data.table(mtcars)
```

过滤 cyl = 6 并且 gear = 4 的记录

```
mtcars_df[cyl == 6 & gear == 4]
```

```
##      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## 1: 21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## 2: 21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## 3: 19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## 4: 17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
```

过滤操作是针对数据框的行（记录）

```
mtcars_df[cyl == 6 & gear == 4, .(mpg, disp)]
```

```
##      mpg  disp
## 1: 21.0 160.0
## 2: 21.0 160.0
## 3: 19.2 167.6
## 4: 17.8 167.6
```



```
subset(x = mtcars_df, subset = cyl == 6 & gear == 4, select = c(mpg, disp))

##      mpg  disp
## 1: 21.0 160.0
## 2: 21.0 160.0
## 3: 19.2 167.6
## 4: 17.8 167.6

mtcars |>
  dplyr::filter(cyl == 6 & gear == 4) |>
  dplyr::select(mpg, disp)

##      mpg  disp
## Mazda RX4     21.0 160.0
## Mazda RX4 Wag 21.0 160.0
## Merc 280      19.2 167.6
## Merc 280C     17.8 167.6
```

## 7.1.2 变换

根据已有的列生成新的列，或者修改已有的列，一次只能修改一列

```
mtcars_df[, mean_mpg := mean(mpg)][, mean_disp := mean(disp)]
mtcars_df[1:6, ]

##      mpg cyl disp  hp drat    wt  qsec vs am gear carb mean_mpg mean_disp
## 1: 21.0   6 160 110 3.90 2.620 16.46  0  1     4    4 20.09062  230.7219
## 2: 21.0   6 160 110 3.90 2.875 17.02  0  1     4    4 20.09062  230.7219
## 3: 22.8   4 108  93 3.85 2.320 18.61  1  1     4    1 20.09062  230.7219
## 4: 21.4   6 258 110 3.08 3.215 19.44  1  0     3    1 20.09062  230.7219
## 5: 18.7   8 360 175 3.15 3.440 17.02  0  0     3    2 20.09062  230.7219
## 6: 18.1   6 225 105 2.76 3.460 20.22  1  0     3    1 20.09062  230.7219

mtcars_df[, .(mean_mpg = mean(mpg), mean_disp = mean(disp))]

##      mean_mpg mean_disp
## 1: 20.09062 230.7219

# mtcars_df[, .(mean_mpg := mean(mpg), mean_disp := mean(disp))] # 报错
# 正确的姿势
mtcars_df[, `:=`(mean_mpg = mean(mpg), mean_disp = mean(disp))][, .(mpg, disp, mean_mpg, mean_disp)]

##      mpg disp mean_mpg mean_disp
## 1: 21.0 160 20.09062  230.7219
## 2: 21.0 160 20.09062  230.7219
## 3: 22.8 108 20.09062  230.7219
## 4: 21.4 258 20.09062  230.7219
## 5: 18.7 360 20.09062  230.7219
## 6: 18.1 225 20.09062  230.7219
```



```
mtcars |>
  dplyr::summarise(mean_mpg = mean(mpg), mean_disp = mean(disp))

##   mean_mpg mean_disp
## 1 20.09062 230.7219
```

④

```
mtcars |>
  dplyr::mutate(mean_mpg = mean(mpg), mean_disp = mean(disp)) |>
  dplyr::select(mpg, disp, mean_mpg, mean_disp) |>
  head()
```

```
##          mpg disp mean_mpg mean_disp
## Mazda RX4     21.0 160 20.09062 230.7219
## Mazda RX4 Wag 21.0 160 20.09062 230.7219
## Datsun 710    22.8 108 20.09062 230.7219
## Hornet 4 Drive 21.4 258 20.09062 230.7219
## Hornet Sportabout 18.7 360 20.09062 230.7219
## Valiant       18.1 225 20.09062 230.7219
```

### 7.1.3 聚合

分组统计多个分组变量

```
dcast(mtcars_df, cyl ~ gear, value.var = "mpg", fun = mean)
```

```
##      cyl     3     4     5
## 1:     4 21.50 26.925 28.2
## 2:     6 19.75 19.750 19.7
## 3:     8 15.05     NA 15.4
```

```
tapply(mtcars$mpg, list(mtcars$cyl, mtcars$gear), mean)
```

```
##      3     4     5
## 4 21.50 26.925 28.2
## 6 19.75 19.750 19.7
## 8 15.05     NA 15.4
```

```
mtcars_df[, .(mean_mpg = mean(mpg)), by = .(cyl, gear)]
```

```
##      cyl gear mean_mpg
## 1:     6     4 19.750
## 2:     4     4 26.925
## 3:     6     3 19.750
## 4:     8     3 15.050
## 5:     4     3 21.500
## 6:     4     5 28.200
## 7:     8     5 15.400
## 8:     6     5 19.700
```



```
aggregate(data = mtcars_df, mpg ~ cyl + gear, FUN = mean)

##   cyl gear   mpg
## 1   4   3 21.500
## 2   6   3 19.750
## 3   8   3 15.050
## 4   4   4 26.925
## 5   6   4 19.750
## 6   4   5 28.200
## 7   6   5 19.700
## 8   8   5 15.400

mtcars |>
  dplyr::group_by(cyl, gear) |>
  dplyr::summarise(mean_mpg = mean(mpg))

## # A tibble: 8 x 3
## # Groups:   cyl [3]
##   cyl   gear mean_mpg
##   <dbl> <dbl>     <dbl>
## 1     4     3     21.5
## 2     4     4     26.9
## 3     4     5     28.2
## 4     6     3     19.8
## 5     6     4     19.8
## 6     6     5     19.7
## 7     8     3     15.0
## 8     8     5     15.4
```

#### 7.1.4 命名

修改列名，另存一份生效

```
sub_mtcars_df <- mtcars_df[, .(mean_mpg = mean(mpg)), by = .(cyl, gear)]
setNames(sub_mtcars_df, c("cyl", "gear", "ave_mpg"))

##   cyl gear ave_mpg
## 1   4   3 19.750
## 2   6   3 26.925
## 3   8   3 15.050
## 4   4   4 21.500
## 5   6   4 19.750
## 6   4   5 28.200
## 7   8   5 15.400
## 8   6   5 19.700
```



```
# 注意 sub_mtcars_df 并没有修改列名
sub_mtcars_df
```

```
##      cyl gear mean_mpg
## 1:     6     4 19.750
## 2:     4     4 26.925
## 3:     6     3 19.750
## 4:     8     3 15.050
## 5:     4     3 21.500
## 6:     4     5 28.200
## 7:     8     5 15.400
## 8:     6     5 19.700
```

修改列名并直接起作用，在原来的数据集上生效

```
setnames(sub_mtcars_df, old = c("mean_mpg"), new = c("ave_mpg"))
# sub_mtcars_df 已经修改了列名
sub_mtcars_df
```

```
##      cyl gear ave_mpg
## 1:     6     4 19.750
## 2:     4     4 26.925
## 3:     6     3 19.750
## 4:     8     3 15.050
## 5:     4     3 21.500
## 6:     4     5 28.200
## 7:     8     5 15.400
## 8:     6     5 19.700
```

修改列名最好使用 **data.table** 包的函数 `setnames()` 明确指出了要修改的列名，

### 7.1.5 排序

按照某（些）列从大到小或从小到大的顺序排列，先按 cyl 升序，然后按 gear 降序

```
mtcars_df[, .(mpg, cyl, gear)][cyl == 4][order(cyl, -gear)]
```

```
##      mpg cyl gear
## 1: 26.0   4    5
## 2: 30.4   4    5
## 3: 22.8   4    4
## 4: 24.4   4    4
## 5: 22.8   4    4
## 6: 32.4   4    4
## 7: 30.4   4    4
## 8: 33.9   4    4
## 9: 27.3   4    4
## 10: 21.4  4    4
```

```
## 11: 21.5 4 3
mtcars |>
  dplyr::select(mpg, cyl, gear) |>
  dplyr::filter(cyl == 4) |>
  dplyr::arrange(cyl, desc(gear))

##          mpg cyl gear
## Porsche 914-2 26.0 4 5
## Lotus Europa 30.4 4 5
## Datsun 710 22.8 4 4
## Merc 240D 24.4 4 4
## Merc 230 22.8 4 4
## Fiat 128 32.4 4 4
## Honda Civic 30.4 4 4
## Toyota Corolla 33.9 4 4
## Fiat X1-9 27.3 4 4
## Volvo 142E 21.4 4 4
## Toyota Corona 21.5 4 3
```

### 7.1.6 变形

melt 宽的变长的

```
DT <- data.table(
  i_1 = c(1:5, NA),
  i_2 = c(NA, 6, 7, 8, 9, 10),
  f_1 = factor(sample(c(letters[1:3], NA), 6, TRUE)),
  f_2 = factor(c("z", "a", "x", "c", "x", "x"), ordered = TRUE),
  c_1 = sample(c(letters[1:3], NA), 6, TRUE),
  d_1 = as.Date(c(1:3, NA, 4:5), origin = "2013-09-01"),
  d_2 = as.Date(6:1, origin = "2012-01-01")
)

DT[, .(i_1, i_2, f_1, f_2)]

##    i_1 i_2 f_1 f_2
## 1:  1  NA  b   z
## 2:  2   6  b   a
## 3:  3   7  c   x
## 4:  4   8  a   c
## 5:  5   9  b   x
## 6:  NA  10  b   x

melt(DT, id = 1:2, measure = c("f_1", "f_2"))

##    i_1 i_2 variable value
## 1:  1  NA     f_1      b
```

```
## 2: 2 6 f_1 b
## 3: 3 7 f_1 c
## 4: 4 8 f_1 a
## 5: 5 9 f_1 b
## 6: NA 10 f_1 b
## 7: 1 NA f_2 z
## 8: 2 6 f_2 a
## 9: 3 7 f_2 x
## 10: 4 8 f_2 c
## 11: 5 9 f_2 x
## 12: NA 10 f_2 x
```

dcast 长的变宽的

```
sleep <- as.data.table(sleep)
dcast(sleep, group ~ ID, value.var = "extra")
```

```
##      group 1 2 3 4 5 6 7 8 9 10
## 1:     1 0.7 -1.6 -0.2 -1.2 -0.1 3.4 3.7 0.8 0.0 2.0
## 2:     2 1.9  0.8  1.1  0.1 -0.1 4.4 5.5 1.6 4.6 3.4
```

# 如果有多个值

```
dcast(mtcars_df, cyl ~ gear, value.var = "mpg")
```

```
##      cyl 3 4 5
## 1:     4 1 8 2
## 2:     6 2 4 1
## 3:     8 12 0 2
```

```
dcast(mtcars_df, cyl ~ gear, value.var = "mpg", fun = mean)
```

```
##      cyl 3 4 5
## 1:     4 21.50 26.925 28.2
## 2:     6 19.75 19.750 19.7
## 3:     8 15.05     NaN 15.4
```

tidyR 包提供数据变形的函数 `tidyR::pivot_longer()` 和 `tidyR::pivot_wider()` 相比于 Base R 提供的 `reshape()` 和 `data.table` 提供的 `melt()` 和 `dcast()` 更加形象的命名

```
tidyR::pivot_wider(data = sleep, names_from = "ID", values_from = "extra")
```

```
## # A tibble: 2 x 11
##   group `1` `2` `3` `4` `5` `6` `7` `8` `9` `10`
##   <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1     0.7  -1.6 -0.2 -1.2 -0.1  3.4  3.7  0.8  0   2
## 2 2     1.9   0.8  1.1  0.1 -0.1  4.4  5.5  1.6  4.6  3.4
```

```
reshape(data = sleep, v.names = "extra", idvar = "group", timevar = "ID", direction = "wide")
```

```
##      group extra.1 extra.2 extra.3 extra.4 extra.5 extra.6 extra.7 extra.8
## 1:     1 0.7    -1.6   -0.2    -1.2   -0.1    3.4    3.7    0.8
## 2:     2 1.9     0.8    1.1     0.1   -0.1    4.4    5.5    1.6
```

```
## extra.9 extra.10
## 1: 0.0 2.0
## 2: 4.6 3.4

• idvar 分组变量
• timevar 组内编号
• v.names 个体观察值
• sep 新的列名是由参数 v.names (extra) 和参数值 timevar (ID) 拼接起来的, 默认 sep = "." 推荐使用下划线来做分割 sep = "_"
```

```
head(ToothGrowth)
```

```
## len supp dose
## 1 4.2 VC 0.5
## 2 11.5 VC 0.5
## 3 7.3 VC 0.5
## 4 5.8 VC 0.5
## 5 6.4 VC 0.5
## 6 10.0 VC 0.5

ToothGrowth$time <- rep(1:10, 6)
reshape(ToothGrowth,
       v.names = "len", idvar = c("supp", "dose"),
       timevar = "time", direction = "wide"
)

## supp dose len.1 len.2 len.3 len.4 len.5 len.6 len.7 len.8 len.9 len.10
## 1 VC 0.5 4.2 11.5 7.3 5.8 6.4 10.0 11.2 11.2 5.2 7.0
## 11 VC 1.0 16.5 16.5 15.2 17.3 22.5 17.3 13.6 14.5 18.8 15.5
## 21 VC 2.0 23.6 18.5 33.9 25.5 26.4 32.5 26.7 21.5 23.3 29.5
## 31 OJ 0.5 15.2 21.5 17.6 9.7 14.5 10.0 8.2 9.4 16.5 9.7
## 41 OJ 1.0 19.7 23.3 23.6 26.4 20.0 25.2 25.8 21.2 14.5 27.3
## 51 OJ 2.0 25.5 26.4 22.4 24.5 24.8 30.9 26.4 27.3 29.4 23.0
```

以数据集 ToothGrowth 为例, 变量 supp (大组), dose (小组) 和 time (组内个体编号) 一起决定唯一的一个数据 len, 特别适合纵向数据的变形操作

### 7.1.7 分组

分组切片, 取每组第一个和最后一个值

```
Loblolly |>
  dplyr::group_by(Seed) |>
  dplyr::arrange(height, age, Seed) |>
  dplyr::slice(1, dplyr::n())
```

```
## # A tibble: 28 x 3
## # Groups:   Seed [14]
##   height   age Seed
```



```
##      <dbl> <dbl> <ord>
## 1    3.93     3 329
## 2    56.4     25 329
## 3    4.12     3 327
## 4    56.8     25 327
## 5    4.38     3 325
## 6    58.5     25 325
## 7    3.91     3 307
## 8    59.1     25 307
## 9    3.46     3 331
## 10   59.5     25 331
## # ... with 18 more rows
```

dplyr::slice() 和函数 slice.index() 有关系吗?

### 7.1.8 合并

合并操作对应于数据库中的连接操作, dplyr 包的哲学就来源于对数据库操作的进一步抽象, data.table 包的 merge 函数就对应为 dplyr 包的 join 函数

data.table::merge 和 dplyr::join

给出一个表格, 数据操作, data.table 实现, dplyr 实现

```
dt1 <- data.table(A = letters[1:10], X = 1:10, key = "A")
dt2 <- data.table(A = letters[5:14], Y = 1:10, key = "A")
merge(dt1, dt2) # 内连接
```

```
##      A  X  Y
## 1: e  5  1
## 2: f  6  2
## 3: g  7  3
## 4: h  8  4
## 5: i  9  5
## 6: j 10  6
```

参数 key 的作用相当于建立一个索引, 通过它实现更快的数据操作速度

key = c("x", "y", "z") 或者 key = "x,y,z" 其中 x,y,z 是列名

```
data(band_members, band_instruments, package = "dplyr")
band_members
```

```
## # A tibble: 3 x 2
##   name  band
##   <chr> <chr>
## 1 Mick  Stones
## 2 John  Beatles
## 3 Paul  Beatles
```



```
band_instruments

## # A tibble: 3 x 2
##   name  plays
##   <chr> <chr>
## 1 John  guitar
## 2 Paul  bass
## 3 Keith guitar

dplyr::inner_join(band_members, band_instruments)

## # A tibble: 2 x 3
##   name  band   plays
##   <chr> <chr>  <chr>
## 1 John  Beatles guitar
## 2 Paul  Beatles bass
```

list 列表里每个元素都是 data.frame 时，最适合用 data.table::rbindlist 合并

```
# 合并列表 https://recology.info/2018/10/limiting-dependencies/
function(x) {
  tibble::as_tibble((x <- data.table::setDF(
    data.table::rbindlist(x, use.names = TRUE, fill = TRUE, idcol = "id"))
  ))
}

## function(x) {
##   tibble::as_tibble((x <- data.table::setDF(
##     data.table::rbindlist(x, use.names = TRUE, fill = TRUE, idcol = "id"))
##   ))
## }
```

## 7.2 高频操作

以面向问题的方式介绍 Base R 提供的数据操作，然后过渡到 data.table，它是加强版的 Base R。

表 7.1: 单表的操作

base	dplyr
df[order(x), , drop = FALSE]	arrange(df, x)
df[!duplicated(x), , drop = FALSE], unique()	distinct(df, x)
df[x & !is.na(x), , drop = FALSE], subset()	filter(df, x)
df\$z <- df\$x + df\$y, transform()	mutate(df, z = x + y)
df\$x	pull(df, x)
N/A	rename(df, y = x)

base	dplyr
<code>df[c("x", "y")], subset()</code>	<code>select(df, x, y)</code>
<code>df[grepl(names(df), "^x")]</code>	<code>select(df, starts_with("x"))</code>
<code>mean(df\$x)</code>	<code>summarise(df, mean(x))</code>
<code>df[c(1, 2, 5), , drop = FALSE]</code>	<code>slice(df, c(1, 2, 5))</code>

表 7.2: 两表的操作

base	dplyr
<code>merge(df1, df2)</code>	<code>inner_join(df1, df2)</code>
<code>merge(df1, df2, all.x = TRUE)</code>	<code>left_join(df1, df2)</code>
<code>merge(df1, df2, all.y = TRUE)</code>	<code>right_join(df1, df2)</code>
<code>merge(df1, df2, all = TRUE)</code>	<code>full_join(df1, df2)</code>
<code>df1[df1\$x %in% df2\$x, , drop = FALSE]</code>	<code>semi_join(df1, df2)</code>
<code>df1[!df1\$x %in% df2\$x, , drop = FALSE]</code>	<code>anti_join(df1, df2)</code>

```
class(mtcars)
## [1] "data.frame"

library(data.table)
mtcars <- as.data.table(mtcars)
class(mtcars)

## [1] "data.table" "data.frame"
```

### 7.2.1 选择多列

```
# base
mtcars[, c("cyl", "gear")] |> head(3)

##      cyl gear
## 1:     6     4
## 2:     6     4
## 3:     4     4

# data.table
mtcars[, c("cyl", "gear")] |> head(3)

##      cyl gear
## 1:     6     4
## 2:     6     4
## 3:     4     4

# dplyr
dplyr::select(mtcars, cyl, gear) |> head(3)
```

```
##      cyl gear
## 1:   6   4
## 2:   6   4
## 3:   4   4
```

反选多列，选择除了 cyl 和 gear 的列

```
## 或者 mtcars[, setdiff(names(mtcars), c("cyl", "gear"))]
mtcars[, !(names(mtcars) %in% c("cyl", "gear"))] |> head(3)
```

```
## [1] TRUE FALSE TRUE
subset(mtcars, select = -c(cyl, gear)) |> head(3)
```

```
##      mpg disp  hp drat    wt  qsec vs am carb
## 1: 21.0 160 110 3.90 2.620 16.46 0  1   4
## 2: 21.0 160 110 3.90 2.875 17.02 0  1   4
## 3: 22.8 108  93 3.85 2.320 18.61 1  1   1
```

## 7.2.2 过滤多行

```
# base
mtcars[mtcars$cyl == 6 & mtcars$gear == 4, ]
```

```
##      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## 1: 21.0   6 160.0 110 3.90 2.620 16.46 0  1   4   4
## 2: 21.0   6 160.0 110 3.90 2.875 17.02 0  1   4   4
## 3: 19.2   6 167.6 123 3.92 3.440 18.30 1  0   4   4
## 4: 17.8   6 167.6 123 3.92 3.440 18.90 1  0   4   4
subset(mtcars, subset = cyl == 6 & gear == 4)
```

```
##      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## 1: 21.0   6 160.0 110 3.90 2.620 16.46 0  1   4   4
## 2: 21.0   6 160.0 110 3.90 2.875 17.02 0  1   4   4
## 3: 19.2   6 167.6 123 3.92 3.440 18.30 1  0   4   4
## 4: 17.8   6 167.6 123 3.92 3.440 18.90 1  0   4   4
```

```
# data.table
mtcars[cyl == 6 & gear == 4, ]
```

```
##      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## 1: 21.0   6 160.0 110 3.90 2.620 16.46 0  1   4   4
## 2: 21.0   6 160.0 110 3.90 2.875 17.02 0  1   4   4
## 3: 19.2   6 167.6 123 3.92 3.440 18.30 1  0   4   4
## 4: 17.8   6 167.6 123 3.92 3.440 18.90 1  0   4   4
```

```
# dplyr
dplyr::filter(mtcars, cyl == 6 & gear == 4)
```

```
##      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
```

```
## 1: 21.0   6 160.0 110 3.90 2.620 16.46 0 1 4 4
## 2: 21.0   6 160.0 110 3.90 2.875 17.02 0 1 4 4
## 3: 19.2   6 167.6 123 3.92 3.440 18.30 1 0 4 4
## 4: 17.8   6 167.6 123 3.92 3.440 18.90 1 0 4 4
```

### 7.2.3 去重多行

```
# base
mtcars[!duplicated(mtcars[, c("cyl", "gear")])]
```

```
##   mpg cyl  disp  hp drat   wt  qsec vs am gear carb
## 1: 21.0   6 160.0 110 3.90 2.620 16.46 0 1 4 4
## 2: 22.8   4 108.0  93 3.85 2.320 18.61 1 1 4 1
## 3: 21.4   6 258.0 110 3.08 3.215 19.44 1 0 3 1
## 4: 18.7   8 360.0 175 3.15 3.440 17.02 0 0 3 2
## 5: 21.5   4 120.1  97 3.70 2.465 20.01 1 0 3 1
## 6: 26.0   4 120.3  91 4.43 2.140 16.70 0 1 5 2
## 7: 15.8   8 351.0 264 4.22 3.170 14.50 0 1 5 4
## 8: 19.7   6 145.0 175 3.62 2.770 15.50 0 1 5 6
```

```
# data.table
mtcars[!duplicated(mtcars, by = c("cyl", "gear"))], ]
```

```
##   mpg cyl  disp  hp drat   wt  qsec vs am gear carb
## 1: 21.0   6 160.0 110 3.90 2.620 16.46 0 1 4 4
## 2: 22.8   4 108.0  93 3.85 2.320 18.61 1 1 4 1
## 3: 21.4   6 258.0 110 3.08 3.215 19.44 1 0 3 1
## 4: 18.7   8 360.0 175 3.15 3.440 17.02 0 0 3 2
## 5: 21.5   4 120.1  97 3.70 2.465 20.01 1 0 3 1
## 6: 26.0   4 120.3  91 4.43 2.140 16.70 0 1 5 2
## 7: 15.8   8 351.0 264 4.22 3.170 14.50 0 1 5 4
## 8: 19.7   6 145.0 175 3.62 2.770 15.50 0 1 5 6
```

```
unique(mtcars, by = c("cyl", "gear"))
```

```
##   mpg cyl  disp  hp drat   wt  qsec vs am gear carb
## 1: 21.0   6 160.0 110 3.90 2.620 16.46 0 1 4 4
## 2: 22.8   4 108.0  93 3.85 2.320 18.61 1 1 4 1
## 3: 21.4   6 258.0 110 3.08 3.215 19.44 1 0 3 1
## 4: 18.7   8 360.0 175 3.15 3.440 17.02 0 0 3 2
## 5: 21.5   4 120.1  97 3.70 2.465 20.01 1 0 3 1
## 6: 26.0   4 120.3  91 4.43 2.140 16.70 0 1 5 2
## 7: 15.8   8 351.0 264 4.22 3.170 14.50 0 1 5 4
## 8: 19.7   6 145.0 175 3.62 2.770 15.50 0 1 5 6
```

```
# dplyr
dplyr::distinct(mtcars, cyl, gear, .keep_all = TRUE)
```



```
##      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## 1: 21.0   6 160.0 110 3.90 2.620 16.46  0  1     4     4
## 2: 22.8   4 108.0  93 3.85 2.320 18.61  1  1     4     1
## 3: 21.4   6 258.0 110 3.08 3.215 19.44  1  0     3     1
## 4: 18.7   8 360.0 175 3.15 3.440 17.02  0  0     3     2
## 5: 21.5   4 120.1  97 3.70 2.465 20.01  1  0     3     1
## 6: 26.0   4 120.3  91 4.43 2.140 16.70  0  1     5     2
## 7: 15.8   8 351.0 264 4.22 3.170 14.50  0  1     5     4
## 8: 19.7   6 145.0 175 3.62 2.770 15.50  0  1     5     6
```

### 7.2.4 合并操作

在数据库的操作中，合并又称为连接

#### 7.2.4.1 左合并

```
# dplyr::inner_join()
# dplyr::left_join()
# dplyr::right_join()
# dplyr::full_join()
```

#### 7.2.4.2 右合并

### 7.2.5 新添多列

```
mtcars[cyl == 6, `:=` (disp_mean = mean(disp), hp_mean = mean(hp))][cyl == 6, .(cyl, disp, hp, disp_mean, hp_mean)]
##      cyl  disp  hp disp_mean  hp_mean
## 1:   6 160.0 110 183.3143 122.2857
## 2:   6 160.0 110 183.3143 122.2857
## 3:   6 258.0 110 183.3143 122.2857
## 4:   6 225.0 105 183.3143 122.2857
## 5:   6 167.6 123 183.3143 122.2857
## 6:   6 167.6 123 183.3143 122.2857
## 7:   6 145.0 175 183.3143 122.2857
```

### 7.2.6 删减多列

删除列就是将该列的值清空，置为 NULL，下面将新添的两个列删除，根据列名的特点用正则表达式匹配

```
mtcars[, colnames(mtcars)[grep("mean$", colnames(mtcars))] := NULL]
```



## 7.2.7 篩选多列

按照某一规则筛选多列

```
library(data.table)
iris <- as.data.table(iris)
iris[, head(.SD, 6), .SDcols = function(x) is.numeric(x)]
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
## 1:	5.1	3.5	1.4	0.2
## 2:	4.9	3.0	1.4	0.2
## 3:	4.7	3.2	1.3	0.2
## 4:	4.6	3.1	1.5	0.2
## 5:	5.0	3.6	1.4	0.2
## 6:	5.4	3.9	1.7	0.4

## 7.2.8 修改多列类型

```
mtcars[, (c("cyl", "disp")) := lapply(.SD, as.integer), .SDcols = c("cyl", "disp")]
str(mtcars)
```

```
## Classes 'data.table' and 'data.frame': 32 obs. of 11 variables:
## $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
## $ cyl : int 6 6 4 6 8 6 8 4 4 6 ...
## $ disp: int 160 160 108 258 360 225 360 146 140 167 ...
## $ hp : num 110 110 93 110 175 105 245 62 95 123 ...
## $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
## $ wt : num 2.62 2.88 2.32 3.21 3.44 ...
## $ qsec: num 16.5 17 18.6 19.4 17 ...
## $ vs : num 0 0 1 1 0 1 0 1 1 1 ...
## $ am : num 1 1 1 0 0 0 0 0 0 0 ...
## $ gear: num 4 4 4 3 3 3 3 4 4 4 ...
## $ carb: num 4 4 1 1 2 1 4 2 2 4 ...
## - attr(*, ".internal.selfref")=<externalptr>
## - attr(*, "index")= int(0)
```

## 7.2.9 取每组第一行

先将 mtcars 按 cyl 升序, gear 降序排列, 然后按 cyl, gear 和 am 分组取第一行

```
mtcars[order(cyl, -gear)][, head(.SD, 1), by = list(cyl, gear, am)]
```

```
##      cyl gear am mpg disp  hp drat    wt  qsec vs carb
## 1: 4     5   1 26.0 120  91 4.43 2.140 16.70 0     2
## 2: 4     4   1 22.8 108  93 3.85 2.320 18.61 1     1
## 3: 4     4   0 24.4 146  62 3.69 3.190 20.00 1     2
## 4: 4     3   0 21.5 120  97 3.70 2.465 20.01 1     1
```



```
## 5:   6   5   1 19.7  145 175 3.62 2.770 15.50  0   6
## 6:   6   4   1 21.0  160 110 3.90 2.620 16.46  0   4
## 7:   6   4   0 19.2  167 123 3.92 3.440 18.30  1   4
## 8:   6   3   0 21.4  258 110 3.08 3.215 19.44  1   1
## 9:   8   5   1 15.8  351 264 4.22 3.170 14.50  0   4
## 10:  8   3   0 18.7  360 175 3.15 3.440 17.02  0   2
```

# 或者

```
mtcars[order(cyl, -gear)][, .SD[1], by = list(cyl, gear, am)]
```

```
##   cyl gear am  mpg disp  hp drat    wt  qsec vs carb
## 1:   4   5   1 26.0 120  91 4.43 2.140 16.70  0   2
## 2:   4   4   1 22.8 108  93 3.85 2.320 18.61  1   1
## 3:   4   4   0 24.4 146  62 3.69 3.190 20.00  1   2
## 4:   4   3   0 21.5 120  97 3.70 2.465 20.01  1   1
## 5:   6   5   1 19.7 145 175 3.62 2.770 15.50  0   6
## 6:   6   4   1 21.0 160 110 3.90 2.620 16.46  0   4
## 7:   6   4   0 19.2 167 123 3.92 3.440 18.30  1   4
## 8:   6   3   0 21.4 258 110 3.08 3.215 19.44  1   1
## 9:   8   5   1 15.8 351 264 4.22 3.170 14.50  0   4
## 10:  8   3   0 18.7 360 175 3.15 3.440 17.02  0   2
```

### 7.2.10 计算环比同比

以数据集 AirPassengers 为例，重新整理后见表 7.3

```
library(magrittr)
dat <- data.frame(
  year = rep(1949:1960, each = 12),
  month = month.abb, num = AirPassengers
) %>%
  reshape(.,
  v.names = "num", idvar = "year", timevar = "month",
  direction = "wide", sep = "")
) %>%
  setNames(., gsub(pattern = "(num)", replacement = "", x = colnames(.)))

rownames(dat) <- subset(dat, select = year, drop = TRUE)
air_passengers <- subset(dat, select = -year)

knitr:::kable(air_passengers,
  caption = "1949-1960年国际航班乘客数量变化",
  align = "c", row.names = TRUE
)
```

横向计算环比，如 1949 年 2 月相比 1 月增长多少、3 月相比 2 月增长多少，以此类推，就是计算环比？纵向计算同比，如 1950 年 1 月相比 1949 年 1 月增长多少、1951 年相比 1950 年 1 月增长多少？

表 7.3: 1949-1960 年国际航班乘客数量变化

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1949	112	118	132	129	121	135	148	148	136	119	104	118
1950	115	126	141	135	125	149	170	170	158	133	114	140
1951	145	150	178	163	172	178	199	199	184	162	146	166
1952	171	180	193	181	183	218	230	242	209	191	172	194
1953	196	196	236	235	229	243	264	272	237	211	180	201
1954	204	188	235	227	234	264	302	293	259	229	203	229
1955	242	233	267	269	270	315	364	347	312	274	237	278
1956	284	277	317	313	318	374	413	405	355	306	271	306
1957	315	301	356	348	355	422	465	467	404	347	305	336
1958	340	318	362	348	363	435	491	505	404	359	310	337
1959	360	342	406	396	420	472	548	559	463	407	362	405
1960	417	391	419	461	472	535	622	606	508	461	390	432

```

# 环比横向/同比纵向
mom <- function(x) diff(x, lag = 1) / x[-length(x)] # month to month
# 格式化输出
format_mom <- function(x) formatC(mom(x), format = "f", digits = 4)

library(formattable)
# 同比变化
air_passengers %>%
  apply(., 2, format_mom) %>%
  as.data.frame() %>%
  formattable(., list(
    Jan = color_tile("white", "pink"),
    Feb = color_tile("white", "springgreen4"),
    Mar = percent
  ))

library(DT)
datatable(air_passengers)

```

### 7.2.11 合并多个数据框

将所有列都保留，以 `full_join()` 方式合并

```

df1 <- iris[1:10, c(1, 5)]
df2 <- iris[11:15, c(1, 2, 5)]
df3 <- iris[16:30, c(1, 3, 5)]
all_dfs <- list(df1, df2, df3)
# base
Reduce(function(x, y, ...) merge(x, y, ..., all = TRUE), all_dfs)

```

```
##      Sepal.Length Species Sepal.Width Petal.Length
## 1:          4.3  setosa       NA        1.0
## 2:          4.4  setosa       NA        1.0
## 3:          4.6  setosa       NA        1.0
## 4:          4.6  setosa       NA        1.0
## 5:          4.7  setosa       NA        1.6
## 6:          4.8  setosa      3.4        1.9
## 7:          4.8  setosa      3.0        1.9
## 8:          4.9  setosa       NA        NA
## 9:          4.9  setosa       NA        NA
## 10:         5.0  setosa       NA        1.6
## 11:         5.0  setosa       NA        1.6
## 12:         5.0  setosa       NA        1.6
## 13:         5.0  setosa       NA        1.6
## 14:         5.1  setosa       NA        1.4
## 15:         5.1  setosa       NA        1.5
## 16:         5.1  setosa       NA        1.5
## 17:         5.1  setosa       NA        1.7
## 18:         5.2  setosa       NA        1.5
## 19:         5.2  setosa       NA        1.4
## 20:         5.4  setosa      3.7        1.3
## 21:         5.4  setosa      3.7        1.7
## 22:         5.7  setosa       NA        1.5
## 23:         5.7  setosa       NA        1.7
## 24:         5.8  setosa      4.0        NA
##      Sepal.Length Species Sepal.Width Petal.Length
# dplyr
Reduce(function(x, y, ...) dplyr::full_join(x, y, ...), all_dfs)

##      Sepal.Length Species Sepal.Width Petal.Length
## 1:          5.1  setosa       NA        1.4
## 2:          5.1  setosa       NA        1.5
## 3:          5.1  setosa       NA        1.5
## 4:          5.1  setosa       NA        1.7
## 5:          4.9  setosa       NA        NA
## 6:          4.7  setosa       NA        1.6
## 7:          4.6  setosa       NA        1.0
## 8:          5.0  setosa       NA        1.6
## 9:          5.0  setosa       NA        1.6
## 10:         5.4  setosa      3.7        1.3
## 11:         5.4  setosa      3.7        1.7
## 12:         4.6  setosa       NA        1.0
## 13:         5.0  setosa       NA        1.6
## 14:         5.0  setosa       NA        1.6
## 15:         4.4  setosa       NA        NA
```

```
## 16:      4.9  setosa      NA      NA
## 17:      4.8  setosa     3.4     1.9
## 18:      4.8  setosa     3.0     1.9
## 19:      4.3  setosa     3.0      NA
## 20:      5.8  setosa     4.0      NA
## 21:      5.7  setosa      NA     1.5
## 22:      5.7  setosa      NA     1.7
## 23:      5.2  setosa      NA     1.5
## 24:      5.2  setosa      NA     1.4
## Sepal.Length Species Sepal.Width Petal.Length
```

合并完应该有 30 行，为啥只有 24 行？这是因为 `merge()` 函数对主键 `key` 相同的记录会合并，要想不合并，需要调用 `rbindlist()` 函数 <https://d.cosx.org/d/421235>

`rbind()` 列数相同的两个 `data.frame` 按行合并，`cbind()` 行数相同的两个 `data.frame` 按列合并，`merge()` 对行、列数没有要求

```
rbindlist(all_dfs, fill = TRUE)
```

```
## Sepal.Length Species Sepal.Width Petal.Length
## 1:      5.1  setosa      NA      NA
## 2:      4.9  setosa      NA      NA
## 3:      4.7  setosa      NA      NA
## 4:      4.6  setosa      NA      NA
## 5:      5.0  setosa      NA      NA
## 6:      5.4  setosa      NA      NA
## 7:      4.6  setosa      NA      NA
## 8:      5.0  setosa      NA      NA
## 9:      4.4  setosa      NA      NA
## 10:     4.9  setosa      NA      NA
## 11:     5.4  setosa     3.7      NA
## 12:     4.8  setosa     3.4      NA
## 13:     4.8  setosa     3.0      NA
## 14:     4.3  setosa     3.0      NA
## 15:     5.8  setosa     4.0      NA
## 16:     5.7  setosa      NA     1.5
## 17:     5.4  setosa      NA     1.3
## 18:     5.1  setosa      NA     1.4
## 19:     5.7  setosa      NA     1.7
## 20:     5.1  setosa      NA     1.5
## 21:     5.4  setosa      NA     1.7
## 22:     5.1  setosa      NA     1.5
## 23:     4.6  setosa      NA     1.0
## 24:     5.1  setosa      NA     1.7
## 25:     4.8  setosa      NA     1.9
## 26:     5.0  setosa      NA     1.6
## 27:     5.0  setosa      NA     1.6
```



```
## 28:      5.2  setosa      NA     1.5
## 29:      5.2  setosa      NA     1.4
## 30:      4.7  setosa      NA     1.6
##      Sepal.Length Species Sepal.Width Petal.Length
# dplyr
dplyr::bind_rows(all_dfs)

##      Sepal.Length Species Sepal.Width Petal.Length
## 1:      5.1  setosa      NA      NA
## 2:      4.9  setosa      NA      NA
## 3:      4.7  setosa      NA      NA
## 4:      4.6  setosa      NA      NA
## 5:      5.0  setosa      NA      NA
## 6:      5.4  setosa      NA      NA
## 7:      4.6  setosa      NA      NA
## 8:      5.0  setosa      NA      NA
## 9:      4.4  setosa      NA      NA
## 10:     4.9  setosa      NA      NA
## 11:     5.4  setosa     3.7      NA
## 12:     4.8  setosa     3.4      NA
## 13:     4.8  setosa     3.0      NA
## 14:     4.3  setosa     3.0      NA
## 15:     5.8  setosa     4.0      NA
## 16:     5.7  setosa      NA     1.5
## 17:     5.4  setosa      NA     1.3
## 18:     5.1  setosa      NA     1.4
## 19:     5.7  setosa      NA     1.7
## 20:     5.1  setosa      NA     1.5
## 21:     5.4  setosa      NA     1.7
## 22:     5.1  setosa      NA     1.5
## 23:     4.6  setosa      NA     1.0
## 24:     5.1  setosa      NA     1.7
## 25:     4.8  setosa      NA     1.9
## 26:     5.0  setosa      NA     1.6
## 27:     5.0  setosa      NA     1.6
## 28:     5.2  setosa      NA     1.5
## 29:     5.2  setosa      NA     1.4
## 30:     4.7  setosa      NA     1.6
##      Sepal.Length Species Sepal.Width Petal.Length
```

### 7.2.12 分组聚合多个指标

<https://stackoverflow.com/questions/24151602/calculate-multiple-aggregations-with-lapply-sd>

```
# base
aggregate(
  data = mtcars, cbind(mpg, hp) ~ cyl,
  FUN = function(x) c(mean = mean(x), median = median(x))
)
##   cyl mpg.mean mpg.median hp.mean hp.median
## 1   4 26.66364 26.00000 82.63636 91.00000
## 2   6 19.74286 19.70000 122.28571 110.00000
## 3   8 15.10000 15.20000 209.21429 192.50000
```

```
# 数据一致性 https://d.cosx.org/d/420763-base-r
with(
  aggregate(cbind(mpg, hp) ~ cyl, mtcars,
  FUN = function(x) c(mean = mean(x), median = median(x))
  ),
  cbind.data.frame(cyl, mpg, hp)
)
```

```
##   cyl      mean median      mean median
## 1   4 26.66364 26.0 82.63636 91.0
## 2   6 19.74286 19.7 122.28571 110.0
## 3   8 15.10000 15.2 209.21429 192.5
```

```
# data.table
mtcars[, as.list(unlist(lapply(.SD, function(x) {
  list(
    mean = mean(x),
    median = median(x)
  )
}))), by = "cyl", .SDcols = c("mpg", "hp")]
]
```

```
##   cyl mpg.mean mpg.median hp.mean hp.median
## 1:   6 19.74286      19.7 122.28571     110.0
## 2:   4 26.66364      26.0 82.63636     91.0
## 3:   8 15.10000      15.2 209.21429     192.5
```

```
# dplyr
mtcars |>
  dplyr::group_by(cyl) |>
  dplyr::summarise(
    mean_mpg = mean(mpg), mean_hp = mean(hp),
    median_mpg = median(mpg), median_hp = median(hp)
  )
```

```
## # A tibble: 3 x 5
```



```
##      cyl mean_mpg median_hp median_mpg median_hp
## 1      4    26.7     82.6      26.7     82.6
## 2      6    19.7    122.0      19.7    122.0
## 3      8    15.1    209.0      15.1    209.0
```

### 7.2.13 重命名多个列

```
tmp <- aggregate(
  data = mtcars, cbind(mpg, hp) ~ cyl,
  FUN = median
)
tmp <- as.data.table(tmp)
setnames(tmp, old = c("mpg", "hp"), new = c("median_mpg", "median_hp"))
tmp

##      cyl median_mpg median_hp
## 1:     4      26.0      91.0
## 2:     6      19.7     110.0
## 3:     8      15.2     192.5
```

### 7.2.14 对多个列依次排序

<https://stackoverflow.com/questions/1296646/how-to-sort-a-dataframe-by-multiple-columns>

```
# base
tmp[order(median_mpg, -median_hp), ]

##      cyl median_mpg median_hp
## 1:     8      15.2     192.5
## 2:     6      19.7     110.0
## 3:     4      26.0      91.0

# data.table
setorder(tmp, median_mpg, -median_hp)
# dplyr
dplyr::arrange(tmp, median_mpg, desc(median_hp))

##      cyl median_mpg median_hp
## 1:     8      15.2     192.5
## 2:     6      19.7     110.0
## 3:     4      26.0      91.0
```



## 7.2.15 重排多个列的位置

```
# https://stackoverflow.com/questions/19619666/change-column-position-of-data-table
setcolorder(tmp, c("median_mpg", setdiff(names(tmp), "median_mpg")))
tmp

## median_mpg cyl median_hp
## 1:      15.2   8     192.5
## 2:      19.7   6     110.0
## 3:      26.0   4     91.0

# dplyr
dplyr::select(tmp, "median_mpg", setdiff(names(tmp), "median_mpg"))

## median_mpg cyl median_hp
## 1:      15.2   8     192.5
## 2:      19.7   6     110.0
## 3:      26.0   4     91.0
```

## 7.2.16 整理回归结果

```
dat <- split(iris, iris$Species)
mod <- lapply(dat, function(x) lm(Petal.Length ~ Sepal.Length, x))
mod <- lapply(mod, function(x) coef(summary(x)))
mod <- Map(function(x, y) {
  x <- as.data.frame(x)
  x$Species <- y
  x
}, mod, names(dat))
mod <- do.call(rbind, mod)
mod

##                               Estimate Std. Error    t value    Pr(>|t|) Species
## setosa.(Intercept)      0.8030518 0.34387807  2.3352806 2.375647e-02  setosa
## setosa.Sepal.Length     0.1316317 0.06852690  1.9208760 6.069778e-02  setosa
## versicolor.(Intercept)  0.1851155 0.51421351  0.3599974 7.204283e-01 versicolor
## versicolor.Sepal.Length 0.6864698 0.08630708  7.9538056 2.586190e-10 versicolor
## virginica.(Intercept)   0.6104680 0.41710685  1.4635770 1.498279e-01 virginica
## virginica.Sepal.Length  0.7500808 0.06302606 11.9011203 6.297786e-16 virginica
```

```
# 管道操作
split(iris, iris$Species) %>%
  lapply(., function(x) coef(summary(lm(Petal.Length ~ Sepal.Length, x)))) %>%
  Map(function(x, y) {
    x <- as.data.frame(x)
    x$Species <- y
    x
  })
```

```

}, ., levels(iris$Species)) %>%
  do.call(rbind, .)

##                                     Estimate Std. Error    t value    Pr(>|t|) Species
## setosa.(Intercept)      0.8030518 0.34387807  2.3352806 2.375647e-02  setosa
## setosa.Sepal.Length     0.1316317 0.06852690  1.9208760 6.069778e-02  setosa
## versicolor.(Intercept) 0.1851155 0.51421351  0.3599974 7.204283e-01 versicolor
## versicolor.Sepal.Length 0.6864698 0.08630708  7.9538056 2.586190e-10 versicolor
## virginica.(Intercept)   0.6104680 0.41710685  1.4635770 1.498279e-01 virginica
## virginica.Sepal.Length  0.7500808 0.06302606 11.9011203 6.297786e-16 virginica

# dplyr 操作, 需要 dplyr >= 1.0.0
iris %>%
  dplyr::group_by(Species) %>%
  dplyr::summarise(broom::tidy(lm(Petal.Length ~ Sepal.Length)))

```

```

## # A tibble: 6 x 6
## # Groups:   Species [3]
##   Species   term      estimate std.error statistic  p.value
##   <fct>     <chr>      <dbl>     <dbl>      <dbl>    <dbl>
## 1 setosa   (Intercept)  0.803     0.344      2.34    2.38e- 2
## 2 setosa   Sepal.Length 0.132     0.0685     1.92    6.07e- 2
## 3 versicolor (Intercept) 0.185     0.514      0.360   7.20e- 1
## 4 versicolor Sepal.Length 0.686     0.0863     7.95   2.59e-10
## 5 virginica (Intercept)  0.610     0.417      1.46    1.50e- 1
## 6 virginica Sepal.Length  0.750     0.0630    11.9    6.30e-16

```

### 7.2.17 := 和 .()

```

mtcars[, mpg_rate := round(mpg / sum(mpg) * 100, digits = 2), by = .(cyl, vs, am)]
mtcars[, .(mpg_rate, mpg, cyl, vs, am)]
```

```

##   mpg_rate  mpg cyl vs am
## 1: 34.04 21.0   6  0  1
## 2: 34.04 21.0   6  0  1
## 3: 11.48 22.8   4  1  1
## 4: 27.97 21.4   6  1  0
## 5: 10.35 18.7   8  0  0
## 6: 23.66 18.1   6  1  0
## 7:  7.92 14.3   8  0  0
## 8: 35.52 24.4   4  1  0
## 9: 33.19 22.8   4  1  0
## 10: 25.10 19.2   6  1  0
## 11: 23.27 17.8   6  1  0
## 12:  9.08 16.4   8  0  0
## 13:  9.58 17.3   8  0  0

```

```
## 14: 8.42 15.2 8 0 0
## 15: 5.76 10.4 8 0 0
## 16: 5.76 10.4 8 0 0
## 17: 8.14 14.7 8 0 0
## 18: 16.31 32.4 4 1 1
## 19: 15.31 30.4 4 1 1
## 20: 17.07 33.9 4 1 1
## 21: 31.30 21.5 4 1 0
## 22: 8.58 15.5 8 0 0
## 23: 8.42 15.2 8 0 0
## 24: 7.36 13.3 8 0 0
## 25: 10.63 19.2 8 0 0
## 26: 13.75 27.3 4 1 1
## 27: 100.00 26.0 4 0 1
## 28: 15.31 30.4 4 1 1
## 29: 51.30 15.8 8 0 1
## 30: 31.93 19.7 6 0 1
## 31: 48.70 15.0 8 0 1
## 32: 10.78 21.4 4 1 1
##     mpg_rate mpg cyl vs am
mtcars[, .(mpg_rate = round(mpg / sum(mpg) * 100, digits = 2)), by = .(cyl, vs, am)]
```

```
##     cyl vs am mpg_rate
## 1: 6 0 1 34.04
## 2: 6 0 1 34.04
## 3: 6 0 1 31.93
## 4: 4 1 1 11.48
## 5: 4 1 1 16.31
## 6: 4 1 1 15.31
## 7: 4 1 1 17.07
## 8: 4 1 1 13.75
## 9: 4 1 1 15.31
## 10: 4 1 1 10.78
## 11: 6 1 0 27.97
## 12: 6 1 0 23.66
## 13: 6 1 0 25.10
## 14: 6 1 0 23.27
## 15: 8 0 0 10.35
## 16: 8 0 0 7.92
## 17: 8 0 0 9.08
## 18: 8 0 0 9.58
## 19: 8 0 0 8.42
## 20: 8 0 0 5.76
## 21: 8 0 0 5.76
## 22: 8 0 0 8.14
```

```
## 23: 8 0 0 8.58
## 24: 8 0 0 8.42
## 25: 8 0 0 7.36
## 26: 8 0 0 10.63
## 27: 4 1 0 35.52
## 28: 4 1 0 33.19
## 29: 4 1 0 31.30
## 30: 4 0 1 100.00
## 31: 8 0 1 51.30
## 32: 8 0 1 48.70
## cyl vs am mpg_rate
```

### 7.2.18 去掉含有缺失值的记录

```
airquality[complete.cases(airquality), ] |> head()
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5    1
## 2    36     118  8.0   72     5    2
## 3    12     149 12.6   74     5    3
## 4    18     313 11.5   62     5    4
## 7    23     299  8.6   65     5    7
## 8    19      99 13.8   59     5    8
```

# 或着

```
airquality[!apply(airquality, 1, anyNA), ] |> head()
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5    1
## 2    36     118  8.0   72     5    2
## 3    12     149 12.6   74     5    3
## 4    18     313 11.5   62     5    4
## 7    23     299  8.6   65     5    7
## 8    19      99 13.8   59     5    8
```

### 7.2.19 集合操作

match 和 %in% <https://d.cosx.org/d/421314>

```
`%nin%` <- Negate("%in%")
# `%in%` <- function(x, table) match(x, table, nomatch = 0) > 0 # %in% 函数的定义
x <- letters[1:5]
y <- letters[3:8]

x %in% y
```



```
## [1] FALSE FALSE  TRUE  TRUE  TRUE  
x %nin% y
```

```
## [1] TRUE  TRUE FALSE FALSE FALSE
```

返回一个逻辑向量，x 中的元素匹配到了就返回 TRUE，否则 FALSE，%nin% 是 %in% 的取反效果

```
match(x, y)
```

```
## [1] NA NA  1  2  3
```

x 在 y 中的匹配情况，匹配到了，就返回在 y 中匹配的位置，没有匹配到就返回 NA

```
setdiff(x, y)
```

```
## [1] "a" "b"
```

```
intersect(x, y)
```

```
## [1] "c" "d" "e"
```

```
union(x, y)
```

```
## [1] "a" "b" "c" "d" "e" "f" "g" "h"
```

## 7.2.20 对数值向量按既定分组计数

此数据处理过程陆续使用了 transform()、cut() 和 aggregate() 三个函数

```
# 对数值向量按既定分组计数  
dat <- data.frame(y = 1:12)  
dat <- transform(dat, x = cut(y, breaks = c(0, 6, 9, 15)))  
dat <- aggregate(data = dat, y ~ x, FUN = length)
```

```
ggplot(data = dat, aes(x = x, y = y)) +  
  geom_col()  
  
data.frame(y = 1:12) %>%  
  transform(x = cut(y, breaks = c(0, 6, 9, 15))) %>%  
  aggregate(data = ., y ~ x, FUN = length) %>%  
  ggplot(data = ., aes(x = x, y = y)) +  
  geom_col()
```

对数值向量按分位数分组计数

```
dat <- data.frame(y = 1:12)  
dat <- transform(dat, x = cut(  
  x = y,  
  breaks = quantile(y, prob = seq(0, 1, 0.25), na.rm = TRUE)  
))  
  
# dat <- transform(dat, x = cut(
```

```

#   x = y,
#   breaks = quantile(y, prob = seq(0, 1, 0.25)),
#   include.lowest = T
# ))

dat1 <- aggregate(data = dat, y ~ x, FUN = length)

ggplot(data = dat1, aes(x = x, y = y)) +
  geom_col()

```

### 7.2.21 分组排序

按变量 a 分组计算，之后按变量 b 降序排列

```

dat <- aggregate(data = iris, cbind(Sepal.Width, Sepal.Length) ~ Species, FUN = mean)
# 按 Species 降序排列
dat[order(dat$Species, decreasing = T), ]

##      Species Sepal.Width Sepal.Length
## 3  virginica     2.974      6.588
## 2  versicolor    2.770      5.936
## 1    setosa       3.428      5.006

```

### 7.2.22 分组获取 Top 值

分组按既定规律取数，比如按 Species 分组取 Top 6

```

# 分组取前6个
do.call("rbind.data.frame", lapply(base::split(x = iris, ~Species), head))

# 分组取 Top 6
do.call(rbind, lapply(split(iris, iris$Species),
  FUN = function(x) head(x[order(x$Sepal.Length, decreasing = T), ], 6)
))

##      Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1:          5.8       4.0       1.2       0.2  setosa
## 2:          5.7       4.4       1.5       0.4  setosa
## 3:          5.7       3.8       1.7       0.3  setosa
## 4:          5.5       4.2       1.4       0.2  setosa
## 5:          5.5       3.5       1.3       0.2  setosa
## 6:          5.4       3.9       1.7       0.4  setosa
## 7:          7.0       3.2       4.7       1.4 versicolor
## 8:          6.9       3.1       4.9       1.5 versicolor
## 9:          6.8       2.8       4.8       1.4 versicolor
## 10:         6.7       3.1       4.4       1.4 versicolor
## 11:         6.7       3.0       5.0       1.7 versicolor

```

```
## 12:      6.7      3.1      4.7      1.5 versicolor
## 13:      7.9      3.8      6.4      2.0  virginica
## 14:      7.7      3.8      6.7      2.2  virginica
## 15:      7.7      2.6      6.9      2.3  virginica
## 16:      7.7      2.8      6.7      2.0  virginica
## 17:      7.7      3.0      6.1      2.3  virginica
## 18:      7.6      3.0      6.6      2.1  virginica
```

### 7.2.23 分组抽样

```
# 分组抽样
do.call(rbind, lapply(split(iris, iris$Species),
  FUN = function(x) x[sample(1:nrow(x), size = 6), ]
))
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1:      5.1        3.8        1.9        0.4  setosa
## 2:      4.3        3.0        1.1        0.1  setosa
## 3:      4.5        2.3        1.3        0.3  setosa
## 4:      4.9        3.0        1.4        0.2  setosa
## 5:      5.0        3.0        1.6        0.2  setosa
## 6:      4.6        3.6        1.0        0.2  setosa
## 7:      5.6        2.7        4.2        1.3 versicolor
## 8:      5.8        2.7        4.1        1.0 versicolor
## 9:      5.7        2.6        3.5        1.0 versicolor
## 10:     6.3        2.5        4.9        1.5 versicolor
## 11:     6.7        3.1        4.4        1.4 versicolor
## 12:     5.7        2.8        4.5        1.3 versicolor
## 13:     6.4        3.2        5.3        2.3  virginica
## 14:     6.5        3.0        5.5        1.8  virginica
## 15:     5.8        2.8        5.1        2.4  virginica
## 16:     6.1        2.6        5.6        1.4  virginica
## 17:     6.3        2.8        5.1        1.5  virginica
## 18:     7.3        2.9        6.3        1.8  virginica
```

### 7.2.24 分组计算分位数

```
# 分组计算分位数, 如何分组呢
do.call(rbind, lapply(iris[, sapply(iris, class) == "numeric"], quantile))

##          0% 25% 50% 75% 100%
## Sepal.Length 1   1   1   1   1
## Sepal.Width  1   1   1   1   1
## Petal.Length 1   1   1   1   1
```



```
## Petal.Width  1   1   1   1   1
## Species      0   0   0   0   0

aggregate(data = iris, cbind(Sepal.Length, Sepal.Width) ~ Species, FUN = quantile)

##      Species Sepal.Length.0% Sepal.Length.25% Sepal.Length.50% Sepal.Length.75%
## 1      setosa      4.300        4.800        5.000        5.200
## 2  versicolor     4.900        5.600        5.900        6.300
## 3  virginica      4.900        6.225        6.500        6.900
## Sepal.Length.100% Sepal.Width.0% Sepal.Width.25% Sepal.Width.50%
## 1            5.800        2.300        3.200        3.400
## 2            7.000        2.000        2.525        2.800
## 3            7.900        2.200        2.800        3.000
## Sepal.Width.75% Sepal.Width.100%
## 1            3.675        4.400
## 2            3.000        3.400
## 3            3.175        3.800

# 对 Sepal.Length 按 Species 分组计算分位数
do.call("rbind", tapply(iris$Sepal.Length, iris$Species, quantile))

##          0%   25%   50%   75%   100%
## setosa    4.3 4.800 5.0 5.2  5.8
## versicolor 4.9 5.600 5.9 6.3  7.0
## virginica 4.9 6.225 6.5 6.9  7.9

# 分组取平均 mean / 中位数 median
aggregate(data = iris, . ~ Species, FUN = mean)

##      Species Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1      setosa      5.006      3.428      1.462      0.246
## 2  versicolor     5.936      2.770      4.260      1.326
## 3  virginica      6.588      2.974      5.552      2.026
```

### 7.2.25 计算日粒度的 DoD/WoW/MoM/YoY

截止写作时间，data.table 提供的滑动窗口聚合统计函数 `frollmean()`、`frollsum()` 和 `frollapply()` 还处于实验阶段。`shift` 提供漂移功能，向前前置 `lead` 或向后延迟 `lag`。

#### 移动平均、求和和计算

```
dat <- data.frame(dt = seq(
  from = as.Date("2021-01-01"),
  to = Sys.Date(), by = "1 day"
))

dat <- within(dat, {
  uv = round(1000 * runif(n = nrow(dat)))
  uv_dod_d = ifelse(nrow(dat) <= 1, NA, c(NA, diff(uv, lag = 1)))
})
```



```
  uv_wow_d = ifelse(nrow(dat) <= 7, NA, c(rep(NA, 7), diff(uv, lag = 7)))
  uv_mom_d = ifelse(nrow(dat) <= 30, NA, c(rep(NA, 30), diff(uv, lag = 30)))
  uv_yoy_d = ifelse(nrow(dat) <= 365, NA, c(rep(NA, 365), diff(uv, lag = 365)))
}
```



## 7.3 运行环境

```
sessionInfo()

## R version 4.2.0 (2022-04-22)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.4 LTS
##
## Matrix products: default
## BLAS:    /usr/lib/x86_64-linux-gnublas/libblas.so.3.9.0
## LAPACK: /usr/lib/x86_64-linux-gnulapack/liblapack.so.3.9.0
##
## locale:
## [1] LC_CTYPE=en_US.UTF-8          LC_NUMERIC=C
## [3] LC_TIME=en_US.UTF-8          LC_COLLATE=en_US.UTF-8
## [5] LC_MONETARY=en_US.UTF-8       LC_MESSAGES=en_US.UTF-8
## [7] LC_PAPER=en_US.UTF-8          LC_NAME=C
## [9] LC_ADDRESS=C                  LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8   LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats      graphics   grDevices  utils      datasets   methods   base
##
## other attached packages:
## [1] magrittr_2.0.3   data.table_1.14.2
##
## loaded via a namespace (and not attached):
## [1] knitr_1.39       sysfonts_0.8.8   tidyselect_1.1.2 R6_2.5.1
## [5] rlang_1.0.2       fastmap_1.1.0    fansi_1.0.3     stringr_1.4.0
## [9] dplyr_1.0.9       tools_4.2.0      broom_0.8.0     xfun_0.31
## [13] utf8_1.2.2       DBI_1.1.2       cli_3.3.0      htmltools_0.5.2
## [17] ellipsis_0.3.2   assertthat_0.2.1  yaml_2.3.5     digest_0.6.29
## [21] tibble_3.1.7     lifecycle_1.0.1   crayon_1.5.1   bookdown_0.26
## [25] tidyverse_1.2.0   purrr_0.3.4     vctrs_0.4.1    curl_4.3.2
## [29] glue_1.6.2       evaluate_0.15   rmarkdown_2.14  stringi_1.7.6
## [33] compiler_4.2.0   pillar_1.7.0    backports_1.4.1 generics_0.1.2
## [37] pkgconfig_2.0.3
```

# 第八章 并行化操作

向量化运算、并行运算和分布式运算

- `future` 在 R 语言中提供统一的并行和分布式处理框架
- `future.apply` 可以替代 base R 提供的 `apply` 族函数
- `future.batchtools` 使用 `batchtools` 实现并行和分布式处理
- `batchtools` `Map` 函数的并行实现，用于高性能计算系统和分布式处理，可以单机多核并行也可以多机并行，还提供了一种抽象的机制去定义大规模计算机实验。
- `multidplyr` 是 `dplyr` 的后端，多核环境下实现数据分块，提高并行处理性能
- `disk.frame` 是基于磁盘的超出内存容量的快速并行数据操作框架
- `parallelMap` R package to interface some popular parallelization back-ends with a unified interface
- `big.data.table` 基于 `data.table` 的分布式并行计算

## 8.1 apply

apply 家族和 `do.call`

## 8.2 MapReduce

高阶函数，简单来说，就是参数为函数，返回值也是函数。Base R 提供了 `Reduce`、`Filter`、`Find`、`Map`、`Negate` 和 `Position` 等常用函数，此外还有 `*apply` 族。

与 `purrr::map` 比较

在 R 语言里玩转 `apply`、`Map()` 和 `Reduce()`<sup>1</sup>，下面分别以提取合并多张 XLSX 表格<sup>2</sup>，分组计算<sup>3</sup> 和 子集操作<sup>4</sup> 为例，从函数式编程到 MapReduce<sup>5</sup>，制作数据透视表<sup>6</sup>，用于数据处理的函数式编程和单元测试 Functional programming and unit testing for data munging with R 特别是第三章 <https://b-rodrigues.github.io/fput/>，然后是函数式编程与数据建模 Modeling data with functional programming in R<sup>7</sup>

<sup>1</sup><https://stackoverflow.com/questions/3505701/grouping-functions-tapply-by-aggregate-and-the-apply-family>

<sup>2</sup><https://trinkerrstuff.wordpress.com/2018/02/14/easily-make-multi-tabbed-xlsx-files-with-openxlsx/>

<sup>3</sup><https://statcompute.wordpress.com/2018/09/03/playing-map-and-reduce-in-r-by-group-calculation/>

<sup>4</sup><https://statcompute.wordpress.com/2018/09/08/playing-map-and-reduce-in-r-subsetting/>

<sup>5</sup><https://cartesianfaith.com/2015/09/17/from-functional-programming-to-mapreduce-in-r/>

<sup>6</sup><https://digitheadslabnotebook.blogspot.com/2010/01/pivot-tables-in-r.html>

<sup>7</sup><https://cartesianfaith.files.wordpress.com/2015/12/rowe-modeling-data-with-functional-programming-in-r.pdf>



```
add <- function(x) Reduce("+", x)
add(list(1, 2, 3))

## [1] 6

add_accuml <- function(x) Reduce("+", x, accumulate = TRUE)
add_accuml(list(1, 2, 3))

## [1] 1 3 6
```

### 8.3 parallel

并行计算小抄 将共享内存的 R 包整理在一起

```
library(parallel)
```

### 8.4 Rmpi

Rmpi 由卡尔顿大学的 Hao Yu 开发和维护

首先安装 openmpi-devel 开发环境（以 Fedora 30 为例）

```
yum install -y openmpi-devel
echo "export ORTED=/usr/lib64/openmpi/bin" >> ~/.bashrc
# 或者
echo "PATH=/usr/lib64/openmpi/bin:$PATH; export PATH" | tee -a ~/.bashrc
source ~/.bashrc
```

然后进入 R 安装 R 包 Rmpi

```
install.packages('Rmpi')
```

使用 Rmpi 包生成两组服从均匀分布的随机数

```
# 加载 R 包
library(Rmpi)
# 检测可用的逻辑 CPU 核心数
parallel::detectCores()
# 虚拟机分配四个逻辑CPU核
# 1个 master 2个 worker 主机 cloud
mpi.spawn.Rslaves(nslaves=2)

#          2 slaves are spawned successfully. 0 failed.
# master (rank 0, comm 1) of size 3 is running on: cloud
# slave1 (rank 1, comm 1) of size 3 is running on: cloud
# slave2 (rank 2, comm 1) of size 3 is running on: cloud
```

调用 mpi.apply 函数

```
set.seed(1234)
mpi.apply(c(10, 20), runif)

[[1]]
[1] 0.33684269 0.84638494 0.82776590 0.23707947 0.07593769 0.27981368
[7] 0.45307675 0.02878214 0.32807421 0.92854275

[[2]]
[1] 0.63474442 0.04025071 0.01996498 0.01922093 0.41258827 0.84150414
[7] 0.74705002 0.07635368 0.32807392 0.94570363 0.89187667 0.67069020
[13] 0.92996997 0.22486589 0.22118236 0.15807970 0.65619450 0.16473730
[19] 0.85833484 0.11416449
```

用完要关闭

```
mpi.close.Rslaves()
```

pbdMPI 包处于活跃维护状态，是 [pbdR 项目](#) 的核心组件，能够以分布式计算的方式轻松处理 TB 级数据<sup>8</sup>

[Rhpc](#) 包同样基于 MPI 方式，但是集 Rmpi 和 snow 两个包的优点于一身，在保持 apply 编程风格的同时，能够提供更好的高性能计算环境，支持长向量，能够处理一些大数据。

## 8.5 gpuR

Charles Determan 开发的 [gpuR](#) 基于 OpenCL 加速，目前处于活跃维护状态。而 Charles Determan 开发的另一个 [gpuRcuda](#) 包是基于 CUDA 加速

[赵鹏](#) 的博客 [ParallelR](#) 关注基于 CUDA 的 GPU 加速

此外还有 [gpuTools](#)

```
library(gpuR)
set.seed(2019)
gpuA <- gpuMatrix(rnorm(16), nrow = 4, ncol = 4)
gpuA
```

```
An object of class "fgpuMatrix"
Slot "address":
<pointer: 0x0000000000fbe9760>
```

```
Slot ".context_index":
[1] 1
```

```
Slot ".platform_index":
[1] 1
```

```
Slot ".platform":
```

<sup>8</sup>2016 年国际 R 语言大会上的介绍<https://github.com/snoweye/user2016.demo> 和 2018 年 JSM 会上的介绍 [https://github.com/RBigData/R\\_JSM2018](https://github.com/RBigData/R_JSM2018)



```
[1] "Intel(R) OpenCL"

Slot ".device_index":
[1] 1

Slot ".device":
[1] "Intel(R) HD Graphics 4600"

gpuB <- gpuA %*% gpuA
print(gpuB)
```

Source: gpuR Matrix [4 x 4]

```
[,1]      [,2]      [,3]      [,4]
[1,] 2.61787200 -1.274909 -2.150301 -2.0073860
[2,] -0.02231596  1.566433  0.986027  0.7339008
[3,] -0.12862393  1.848340  3.261899  1.6919358
[4,] -1.90084898 -1.863014 -1.312350 -0.2553876
```

## 8.6 运行环境

```
xfun::session_info()

## R version 4.2.0 (2022-04-22)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.4 LTS
##
## Locale:
##   LC_CTYPE=en_US.UTF-8      LC_NUMERIC=C
##   LC_TIME=en_US.UTF-8       LC_COLLATE=en_US.UTF-8
##   LC_MONETARY=en_US.UTF-8   LC_MESSAGES=en_US.UTF-8
##   LC_PAPER=en_US.UTF-8      LC_NAME=C
##   LC_ADDRESS=C              LC_TELEPHONE=C
##   LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## Package version:
##   base64enc_0.1.3 bookdown_0.26   bslib_0.3.1    cli_3.3.0
##   compiler_4.2.0  curl_4.3.2      digest_0.6.29  evaluate_0.15
##   fastmap_1.1.0   fs_1.5.2       glue_1.6.2     graphics_4.2.0
##   grDevices_4.2.0  highr_0.9     htmltools_0.5.2 jquerylib_0.1.4
##   jsonlite_1.8.0   knitr_1.39    magrittr_2.0.3  methods_4.2.0
##   R6_2.5.1        rappdirs_0.3.3   rlang_1.0.2    rmarkdown_2.14
##   sass_0.4.1      stats_4.2.0     stringi_1.7.6  stringr_1.4.0
##   sysfonts_0.8.8   tinytex_0.39   tools_4.2.0    utils_4.2.0
##   xfun_0.31       yaml_2.3.5
```

[create-an-empty-data-frame pipe-r](#)

## 第九章 净土化操作

```
library(dplyr)
```

## 9.1 常用操作

dplyr 由 Hadley Wickham 主要由开发和维护，是 Rstudio 公司开源的用于数据处理的一大利器，该包号称「数据操作的语法」，与 ggplot2 的「图形语法」对应，也就是说数据处理那一套已经建立完整的和 SQL 一样的功能。它们都遵循同样的处理逻辑，只不过一个用 SQL 写，一个用 R 语言写，处理效率差不多，R 语言写的 SQL 会被翻译为 SQL 语句，再传至数据库查询，当然它也支持内存内的数据操作。目前 dplyr 以 dbplyr 为后端支持的数据库有：MySQL、PostgreSQL、SQLite 等，完整的支持列表请看 [这里](#)，连接特定数据库，都是基于 DBI，DBI 即 Database Interface，是使用 C/C++ 开发的底层数据库接口，是一个统一的关系型数据库连接框架，需要根据不同的具体的数据库进行实例化，才可使用。

dplyr 常用的函数是 7 个: arrange 排序 filter 过滤行 select 选择列 mutate 变换 summarise 汇总 group\_by 分组 distinct 去重

以 `ggplot2` 包自带的钻石数据集 `diamonds` 为例介绍

### 9.1.1 查看

除了直接打印数据集的前几行, `tibble` 包还提供 `glimpse` 函数查看数据集, 而 Base R 默认查看方式是调用 `str` 函数

```
## $ z      <dbl> 2.43, 2.31, 2.31, 2.63, 2.75, 2.48, 2.47, 2.53, 2.49, 2.39, 2.~
```

表 9.1: dplyr 定义的数据对象类型

类型	含义
int	整型 integer
dbl	(单) 双精度浮点类型
chr	字符(串)类型
dttm	data-time 类型
lgl	布尔类型
fctr	因子类型 factor
date	日期类型

表 9.1 中 dttm 和 date 类型代指 lubridate 包指定的日期对象 POSIXct、POSIXlt、Date、chron、yearmon、yearqtr、zoo、zooreg、timeDate、xts、its、ti、jul、timeSeries 和 fts。

### 9.1.2 篩选

按条件篩选数据的子集，按行篩选

```
diamonds |> filter(cut == "Ideal", carat >= 3)

## # A tibble: 4 x 10
##   carat cut   color clarity depth table price     x     y     z
##   <dbl> <ord> <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1   3.22 Ideal  I     I1     62.6    55 12545  9.49  9.42  5.92
## 2   3.5   Ideal  H     I1     62.8    57 12587  9.65  9.59  6.03
## 3   3.01 Ideal  J     SI2    61.7    58 16037  9.25  9.2   5.69
## 4   3.01 Ideal  J     I1     65.4    60 16538  8.99  8.93  5.86
```

先按行，再按列篩选

```
diamonds |>
  filter(carat >= 3, color == "I") |>
  select(cut, carat)

## # A tibble: 16 x 2
##   cut      carat
##   <ord>    <dbl>
## 1 Premium  3.01
## 2 Fair     3.02
## 3 Good     3
## 4 Ideal    3.22
## 5 Premium  4.01
## 6 Very Good 3.04
## 7 Very Good 4
## 8 Premium  3.67
```



```
## 9 Premium 3
## 10 Fair 3
## 11 Premium 3.01
## 12 Fair 3.01
## 13 Fair 3.01
## 14 Good 3.01
## 15 Good 3.01
## 16 Premium 3.04
```

### 9.1.3 排序

`arrange` 默认升序排列，按钻石重量升序，按价格降序

```
diamonds |>
  filter(cut == "Ideal", carat >= 3) |>
  arrange(carat, desc(price))
```

```
## # A tibble: 4 x 10
##   carat cut   color clarity depth table price     x     y     z
##   <dbl> <ord> <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  3.01 Ideal J     I1     65.4     60 16538  8.99  8.93  5.86
## 2  3.01 Ideal J     SI2    61.7     58 16037  9.25  9.2   5.69
## 3  3.22 Ideal I     I1     62.6     55 12545  9.49  9.42  5.92
## 4  3.5   Ideal H     I1     62.8     57 12587  9.65  9.59  6.03
```

### 9.1.4 聚合

分组求和，求平均，计数

```
diamonds |>
  filter(carat > 3, color == "I") |>
  group_by(cut, clarity) |>
  summarise(sum_carat = sum(carat), mean_carat = mean(carat), n_count = n())
```

```
## # A tibble: 8 x 5
## # Groups:   cut [5]
##   cut      clarity sum_carat mean_carat n_count
##   <ord>    <ord>     <dbl>      <dbl>     <int>
## 1 Fair      SI2      6.02      3.01      2
## 2 Fair      I1       3.02      3.02      1
## 3 Good     SI2      6.02      3.01      2
## 4 Very Good I1       4         4         1
## 5 Very Good SI2      3.04      3.04      1
## 6 Premium   I1      10.7      3.56      3
## 7 Premium   SI2      6.05      3.02      2
## 8 Ideal     I1       3.22      3.22      1
```

### 9.1.5 合并

按行合并

```
set.seed(2018)
one <- diamonds |>
  filter(color == "I") |>
  sample_n(5)
two <- diamonds |>
  filter(color == "J") |>
  sample_n(5)
# 按行合并数据框 one 和 two
bind_rows(one, two)

## # A tibble: 10 x 10
##   carat cut     color clarity depth table price     x     y     z
##   <dbl> <ord>   <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1 0.42  Ideal    I     VVS1    62.5  57    884  4.77  4.8   2.99
## 2 0.3   Ideal    I     VVS2    62.5  53.6   532  4.29  4.33  2.69
## 3 2.02  Good     I     VS1     57.9  63   17533  8.13  8.21  4.73
## 4 0.9   Premium  I     VS2     61.9  58    3398  6.18  6.23  3.84
## 5 1.98  Very Good I     VS2     62.7  60   15083  7.9   7.96  4.98
## 6 1.51  Very Good J     VVS2    62.6  63    8706  7.29  7.24  4.55
## 7 0.7   Very Good J     SI1     61.7  57    1979  5.65  5.69  3.5
## 8 1.16  Premium  J     VS2     62.2  59    4702  6.74  6.69  4.18
## 9 1.5   Premium  J     VVS2    61.8  60    8760  7.36  7.33  4.54
## 10 1.51  Premium J     SI1     60.4  62    6680  7.42  7.32  4.45
```

按列合并

```
set.seed(2018)
three <- diamonds |>
  select(carat, color) |>
  sample_n(5)
four <- diamonds |>
  select(carat, color) |>
  sample_n(5)
bind_cols(three, four)

## # A tibble: 5 x 4
##   carat...1 color...2 carat...3 color...4
##   <dbl> <ord>     <dbl> <ord>
## 1 0.33  H        0.52  F
## 2 1.09  F        0.51  F
## 3 1.52  I        0.5   G
## 4 0.95  G        0.38  E
## 5 0.35  E        0.51  J
```

### 9.1.6 变换

添加一列，新的列或者改变原来的列

```
diamonds |>
  filter(carat > 3, color == "I") |>
  select(cut, carat) |>
  mutate(vol = if_else(carat > 3.5, "A", "B"))
```

```
## # A tibble: 13 x 3
##   cut      carat vol
##   <ord>    <dbl> <chr>
## 1 Premium  3.01  B
## 2 Fair     3.02  B
## 3 Ideal    3.22  B
## 4 Premium  4.01  A
## 5 Very Good 3.04  B
## 6 Very Good 4     A
## 7 Premium  3.67  A
## 8 Premium  3.01  B
## 9 Fair     3.01  B
## 10 Fair    3.01  B
## 11 Good    3.01  B
## 12 Good    3.01  B
## 13 Premium 3.04  B
```

### 9.1.7 去重

数据去重在 dplyr 中的实现<sup>1</sup>。

```
set.seed(123)
df <- data.frame(
  x = sample(0:1, 10, replace = T),
  y = sample(0:1, 10, replace = T),
  z = 1:10
)
df
```

```
##   x y z
## 1 0 1 1
## 2 0 1 2
## 3 0 1 3
## 4 1 0 4
## 5 0 1 5
## 6 1 0 6
## 7 1 1 7
```

<sup>1</sup><https://stackoverflow.com/questions/22959635/>

```
## 8 1 0 8
## 9 0 0 9
## 10 0 0 10
```

去掉列重复的数据点 (x, y)

```
df |>
  group_by(x, y) |>
  filter(row_number(z) == 1)
```

```
## # A tibble: 4 x 3
## # Groups:   x, y [4]
##       x     y     z
##   <int> <int> <int>
## 1     0     1     1
## 2     1     0     4
## 3     1     1     7
## 4     0     0     9
```

# 此处不对，没有了 z

```
df |>
  distinct(x, y)
```

```
##   x y
## 1 0 1
## 2 1 0
## 3 1 1
## 4 0 0
```

# 应该为

```
df |>
  distinct(x, y, .keep_all = TRUE)
```

```
##   x y z
## 1 0 1 1
## 2 1 0 4
## 3 1 1 7
## 4 0 0 9
```

## 9.2 高频问题

常用的数据操作包含

1. 创建空的数据框或者说初始化一个数据框，
2. 按指定的列对数据框排序，
3. 选择特定的一些列，复杂情况是可能需要正则表达式从列名或者值中筛选
4. 合并两个数据框，分为 (inner outer left right) 四种情况



5. 宽格式和长格式互相转换，即重塑操作 `reshape`，单独的 `tidyR` 包操作，是 `reshape2` 包的进化版，提供 `spread` 和 `gather` 两个主要函数

### 9.2.1 初始化数据框



创建空的数据框，就是不包含任何行、记录

```
empty_df <- data.frame(  
  Doubles = double(),  
  Ints = integer(),  
  Factors = factor(),  
  Logicals = logical(),  
  Characters = character(),  
  stringsAsFactors = FALSE  
)  
str(empty_df)  
  
## 'data.frame': 0 obs. of 5 variables:  
## $ Doubles : num  
## $ Ints    : int  
## $ Factors : Factor w/ 0 levels:  
## $ Logicals: logi  
## $ Characters: chr
```

如果数据框 `df` 包含数据，现在要依据它创建一个空的数据框

```
empty_df = df[FALSE,]
```

还可以使用 `structure` 构造一个数据框，并且我们发现它的效率更高

```
s <- function() structure(list(  
  Date = as.Date(character()),  
  File = character(),  
  User = character(),  
,  
  class = "data.frame")  
d <- function() data.frame(  
  Date = as.Date(character()),  
  File = character(),  
  User = character(),  
  stringsAsFactors = FALSE)  
)  
microbenchmark::microbenchmark(s(), d())  
  
## Unit: microseconds  
##  expr   min     q1     mean median     uq     max neval  
##  s() 23.0 26.45 64.859 31.85 38.0 2973.0    100  
##  d() 247.8 252.80 287.072 254.90 261.3 2850.3    100
```

### 9.2.2 移除缺失记录

只要行中包含缺失值，我们就把这样的行移除出去

```
airquality[complete.cases(airquality), ] |> head()
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1     41     190  7.4   67     5    1
## 2     36     118  8.0   72     5    2
## 3     12     149 12.6   74     5    3
## 4     18     313 11.5   62     5    4
## 7     23     299  8.6   65     5    7
## 8     19      99 13.8   59     5    8
```

### 9.2.3 数据类型转化

```
str(PlantGrowth)
```

```
## 'data.frame': 30 obs. of 2 variables:
## $ weight: num 4.17 5.58 5.18 6.11 4.5 4.61 5.17 4.53 5.33 5.14 ...
## $ group : Factor w/ 3 levels "ctrl","trt1",...: 1 1 1 1 1 1 1 1 1 1 ...
bob <- PlantGrowth
i <- sapply(bob, is.factor)
bob[i] <- lapply(bob[i], as.character)
str(bob)
```

```
## 'data.frame': 30 obs. of 2 variables:
## $ weight: num 4.17 5.58 5.18 6.11 4.5 4.61 5.17 4.53 5.33 5.14 ...
## $ group : chr "ctrl" "ctrl" "ctrl" "ctrl" ...
```

### 9.2.4 跨列分组求和

输入是一个数据框 `data.frame`，按照其中某一变量分组，然后计算任意数量的变量的行和和列和。

空气质量数据集 `airquality` 按月份 `Month` 分组，然后求取满足条件的列的和

```
Reduce(rbind, lapply(unique(airquality$Month), function(gv) {
  subdta <- subset(airquality, subset = Month == gv)
  data.frame(
    Colsum = as.numeric(
      colSums(subdta[, grepl("[mM]", names(airquality))], na.rm = TRUE)
    ),
    Month = gv
  )
}))

##   Colsum Month
```

```
## 1    2032    5
## 2     155    5
## 3    2373    6
## 4     180    6
## 5    2601    7
## 6     217    7
## 7    2603    8
## 8     248    8
## 9    2307    9
## 10    270    9
```

什么是函数式编程，R语言环境下的函数式编程是如何操作的

### 9.3 管道操作

Stefan Milton Bache 开发了 `magrittr` 包实现管道操作，增加代码的可读性和维护性，但是这个 R 包的名字取的太奇葩，因为 [记不住](#)，它其实是一个复杂的[法语发音](#)，中式英语就叫它马格里特吧！这下应该好记多了吧！

我要查看是否需要新添加一个 R 包依赖，假设该 R 包是 `reticulate` 没有出现在 `DESCRIPTION` 文件中，但是可能已经被其中某（个）些 R 包依赖了

```
"reticulate" %in% sort(unique(unlist(tools::package_dependencies(desc::desc_get_deps())$package, recursive = TRUE)))
```

## [1] TRUE

安装 `pkg` 的依赖

```
pkg <- c(
  "bookdown",
  "e1071",
  "formatR",
  "lme4",
  "mvtnorm",
  "prettydoc", "psych",
  "reticulate", "rstan", "rstanarm", "rtricles",
  "svglite",
  "TMB", "glmmTMB"
)
# 获取 pkg 的所有依赖
dep_pkg <- tools::package_dependencies(pkg, recursive = TRUE)
# 将列表 list 合并为向量 vector
merge_pkg <- Reduce("c", dep_pkg, accumulate = FALSE)
# 所有未安装的 R 包
miss_pkg <- setdiff(unique(merge_pkg), unique(.packages(TRUE)))
# 除了 pkg 外，未安装的 R 包，安装 pkg 的依赖
sort(setdiff(miss_pkg, pkg))
```

```
## [1] "mnormt"
```

转化为管道操作，增加可读性

再举一个关于数据模拟的例子

模拟 0-1 序列，

```
set.seed(2019)
binom_sample <- function(n) {
  sum(sample(x = c(0,1), size = n, prob = c(0.8, 0.2), replace = TRUE))/n
}
# 频率估计概率
one_prob <- sapply(10^(seq(8)), binom_sample)
# 估计的误差
one_abs <- abs(one_prob - 0.2)
one_abs
```

```
## [1] 1.000e-01 1.000e-02 1.100e-02 4.400e-03 1.460e-03 3.980e-04 4.700e-06
## [8] 9.552e-05
```

似然估计

## 第二部分

### 统计图形

## 介绍

统计图形

## 第十章 图形基础

```
library(survival)
library(lattice)
library(nlme)
library(MASS)
library(RColorBrewer)
library(latticeExtra)
library(shape)
library(splines)
library(mgcv)
library(maps)
library(mapproj)
```

数据可视化是一种重要的数据分析手段, R 提供了两套图形系统, 分别是 `graphics` 包提供的基础绘图系统和 `grid` 包提供的栅格绘图系统, 后者主要以两个 R 包为大家所熟知, 一个是 `lattice` 包, 另一个是 `ggplot2` 包。

Base 图形系统的扩展包 `basetheme` 可以设置主题, `prettyB` 和 `gridGraphics`

为了方便记忆函数 `par` 的各个参数, Paul Murrell 整理了一份 [助记符](#), 此外, LaTeX 宏包 `geometry` 对版面设置有很多专业的说明

### 10.1 绘图基本要素

#### 10.1.1 点线

点和线是最常见的画图元素, 在 `plot` 函数中, 分别用参数 `pch` 和 `lty` 来设定类型, 点的大小、线的宽度分别用参数 `cex` 和 `lwd` 来指定, 颜色由参数 `col` 设置。参数 `type` 不同的值设置如下, `p` 显示点, `l` 绘制线, `b` 同时绘制空心点, 并用线连接, `c` 只有线, `o` 在线上绘制点, `s` 和 `S` 点线连接绘制阶梯图, `h` 绘制类似直方图一样的垂线, 最后 `n` 表示什么也不画。

点 `points`、线 `grid` 背景线 `abline` `lines` `rug` 刻度线 (线段 `segments`、箭头 `arrows`)、

```
## ----- Showing all the extra & some char graphics symbols -----
pchShow <-
  function(extras = c("*", ".", "o", "0", "0", "+", "-", "|", "%", "#"),
          cex = 2, ## good for both .Device=="postscript" and "x11"
          col = "red3", bg = "gold", coltext = "brown", cextext = 1.2,
```



```
main = paste(
  "plot symbols : points (... pch = *, cex =",
  "cex, )"
)) {
nex <- length(extras)
np <- 26 + nex
ipch <- 0:(np - 1)
k <- floor(sqrt(np))
dd <- c(-1, 1) / 2
rx <- dd + range(ix <- ipch %% k)
ry <- dd + range(iy <- 3 + (k - 1) - ipch %% k)
pch <- as.list(ipch) # list with integers & strings
if (nex > 0) pch[26 + 1:nex] <- as.list(extras)
plot(rx, ry, type = "n", axes = FALSE, xlab = "", ylab = "", main = main)
abline(v = ix, h = iy, col = "lightgray", lty = "dotted")
for (i in 1:np) {
  pc <- pch[[i]]
  ## 'col' symbols with a 'bg'-colored interior (where available) :
  points(ix[i], iy[i], pch = pc, col = col, bg = bg, cex = cex)
  if (cextext > 0) {
    text(ix[i] - 0.3, iy[i], pc, col = coltext, cex = cextext)
  }
}
}

pchShow()

## ----- test code for various pch specifications -----
# Try this in various font families (including Hershey)
# and locales. Use sign = -1 asserts we want Latin-1.
# Standard cases in a MBCS locale will not plot the top half.
TestChars <- function(sign = 1, font = 1, ...) {
  MB <- l10n_info()$MBCS
  r <- if (font == 5) {
    sign <- 1
    c(32:126, 160:254)
  } else if (MB) 32:126 else 32:255
  if (sign == -1) r <- c(32:126, 160:255)
  par(pty = "s")
  plot(c(-1, 16), c(-1, 16),
    type = "n", xlab = "", ylab = "",
    xaxs = "i", yaxs = "i",
    main = sprintf("sign = %d, font = %d", sign, font)
  )
  grid(17, 17, lty = 1)
```

**plot symbols : points (... pch = \*, cex =**

图 10.1: 不同的 pch 参数值

```
mtext(paste("MBCS:", MB))
for (i in r) try(points(i %% 16, i %/% 16, pch = sign * i, font = font, ...))
}
TestChars()

try(TestChars(sign = -1))

TestChars(font = 5) # Euro might be at 160 (0+10*16).

# macOS has apple at 240 (0+15*16).
try(TestChars(-1, font = 2)) # bold

x <- 0:12
y <- sin(pi / 5 * x)
par(mfrow = c(3, 3), mar = .1 + c(2, 2, 3, 1))
for (tp in c("p", "l", "b", "c", "o", "h", "s", "S", "n")) {
  plot(y ~ x, type = tp, main = paste0("plot(*, type = \"", tp, "\")"))
  if (tp == "S") {
    lines(x, y, type = "s", col = "red", lty = 2)
    mtext("lines(*, type = \"s\", ...)", col = "red", cex = 0.8)
  }
}
```

颜色 col 连续型和离散型

线帽/端和字体的样式

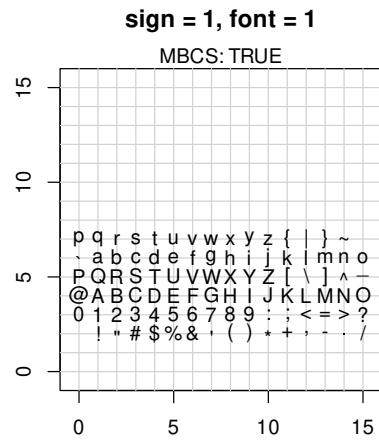


图 10.2: pch 支持的字符

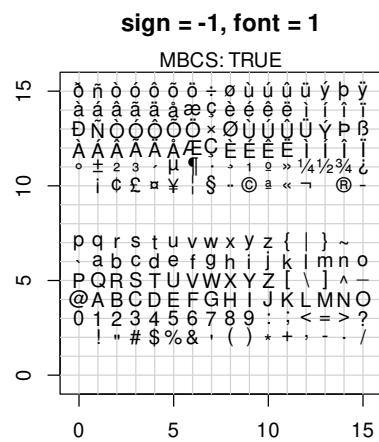


图 10.3: pch 支持的字符

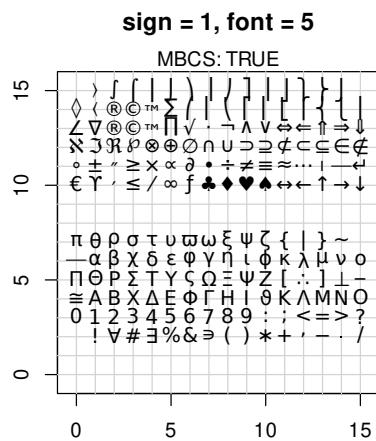


图 10.4: pch 支持的字符

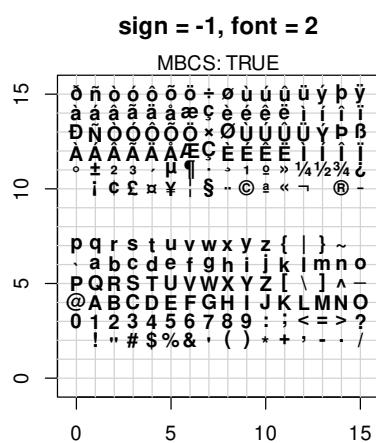


图 10.5: pch 支持的字符

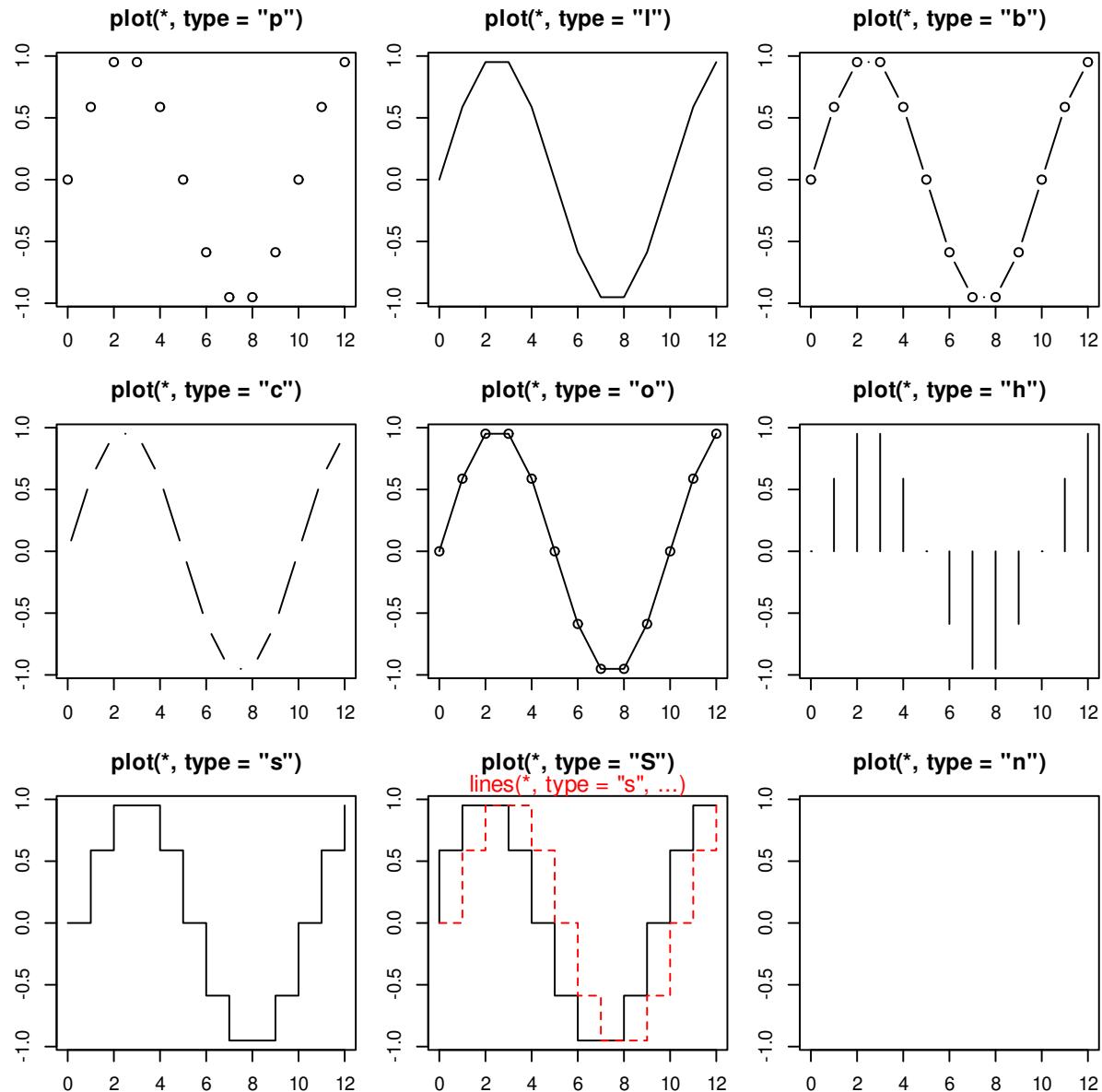


图 10.6: 不同的 type 参数值



```
# 合并为一个图 三条粗横线 横线上三种字形
plot(c(1, 20), c(1, 20), type = "n", ann = FALSE)
lines(x = c(5, 15), y = c(5, 5), lwd = 15, lend = "round")
text(10, 5, "Hello, Helvetica", cex = 1.5, family = "sans", pos = 1, offset = 1.5)
text(5, 5, "sans", cex = 1.5, family = "sans", pos = 2, offset = .5)
text(15, 5, "lend = round", pos = 4, offset = .5)

lines(x = c(5, 15), y = c(10, 10), lwd = 15, lend = "butt")
text(10, 10, "Hello, Helvetica", cex = 1.5, family = "mono", pos = 1, offset = 1.5)
text(5, 10, "mono", cex = 1.5, family = "mono", pos = 2, offset = .5)
text(15, 10, "lend = butt", pos = 4, offset = .5)

lines(x = c(5, 15), y = c(15, 15), lwd = 15, lend = "square")
text(10, 15, "Hello, Helvetica", cex = 1.5, family = "serif", pos = 1, offset = 1.5)
text(5, 15, "serif", cex = 1.5, family = "serif", pos = 2, offset = .5)
text(15, 15, "lend = square", pos = 4, offset = .5)
```

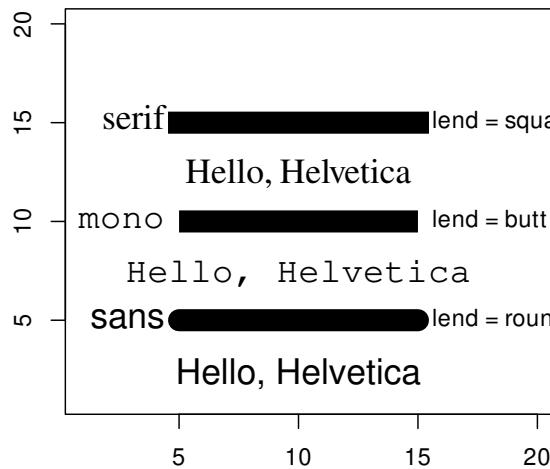


图 10.7: 不同的线端样式

`lend`: 线端的样式, 可用一个整数或字符串指定:

- 0 或 “round” 圆形 (默认)
- 1 或 “butt” 对接形
- 2 或 “square” 方形

### 10.1.2 区域

矩形, 多边形, 曲线交汇出来的区域面 (矩形 `rect`, 多边形 `polygon`)、路径 `polypath` 面/多边形 `rect` 颜色填充



```
# From the manual
ch.col <- c(
  "rainbow(n, start=.7, end=.1)",
  "heat.colors(n)",
  "terrain.colors(n)",
  "topo.colors(n)",
  "cm.colors(n)"
) # 选择颜色
n <- 16
nt <- length(ch.col)
i <- 1:n
j <- n / nt
d <- j / 6
dy <- 2 * d
plot(i, i + d,
  type = "n",
  yaxt = "n",
  ylab = "",
  xlab = "",
  main = paste("color palettes; n=", n)
)
for (k in 1:nt) {
  rect(i - .5, (k - 1) * j + dy, i + .4, k * j,
    col = eval(parse(text = ch.col[k])))
} # 咬人的函数/字符串解析为/转函数
text(2 * j, k * j + dy / 4, ch.col[k])
}
```

clip(x1, x2, y1, y2) 在用户坐标中设置剪切区域

```
x <- rnorm(1000)
hist(x, xlim = c(-4, 4))
usr <- par("usr")
clip(usr[1], -2, usr[3], usr[4])
hist(x, col = "red", add = TRUE)
clip(2, usr[2], usr[3], usr[4])
hist(x, col = "blue", add = TRUE)
```

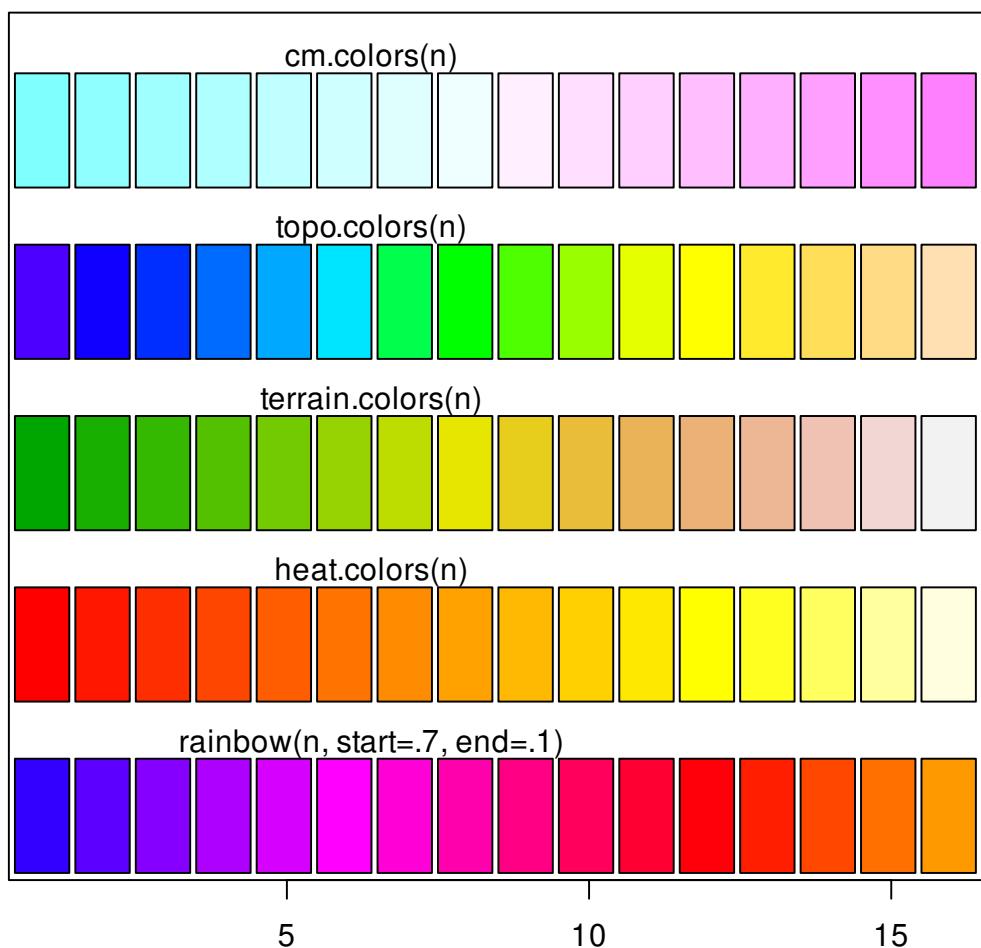
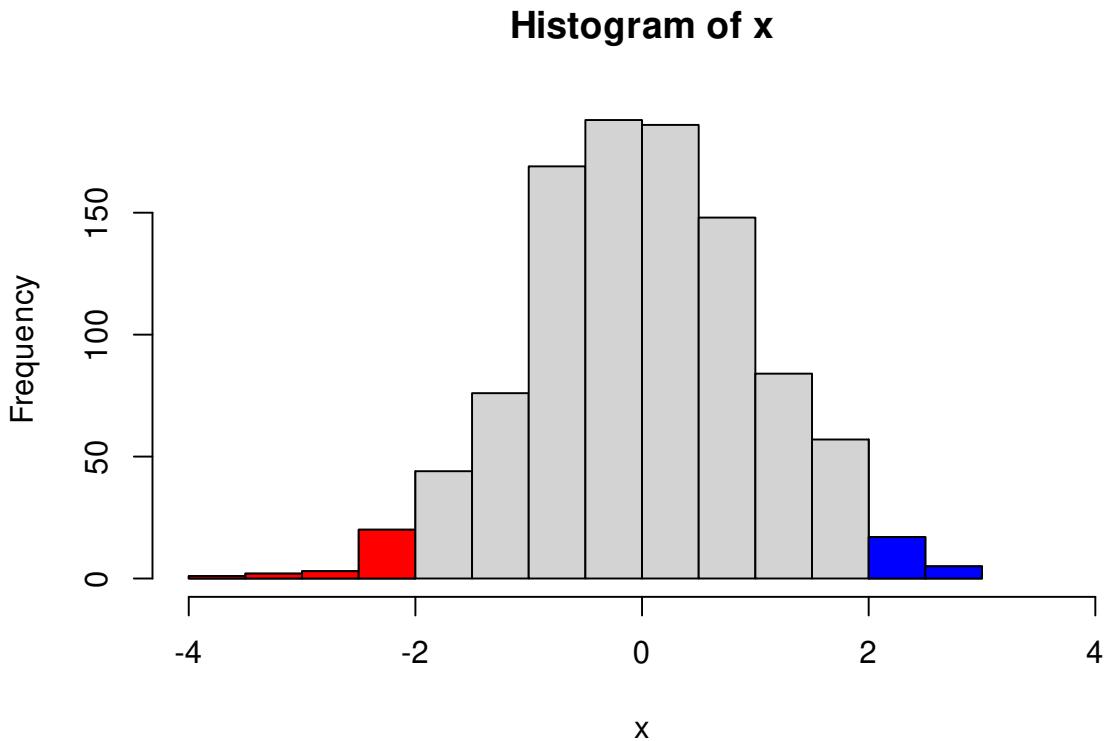
**color palettes; n= 16**

图 10.8: rect 函数画长方形



```
do.call("clip", as.list(usr)) # reset to plot region

my.col <- function(f, g, xmin, xmax, col, N = 200,
                    xlab = "", ylab = "", main = "") {
  x <- seq(xmin, xmax, length = N)
  fx <- f(x)
  gx <- g(x)
  plot(0, 0,
       type = "n",
       xlim = c(xmin, xmax),
       ylim = c(min(fx, gx), max(fx, gx)),
       xlab = xlab, ylab = ylab, main = main
  )
  polygon(c(x, rev(x)), c(fx, rev(gx)),
          col = "#EA4335", border = 0
  )
  lines(x, fx, lwd = 3, col = "#34A853")
  lines(x, gx, lwd = 3, col = "#4285f4")
}
my.col(function(x) x^2, function(x) x^2 + 10 * sin(x),
       -6, 6,
       main = "The \"polygon\" function"
)
```

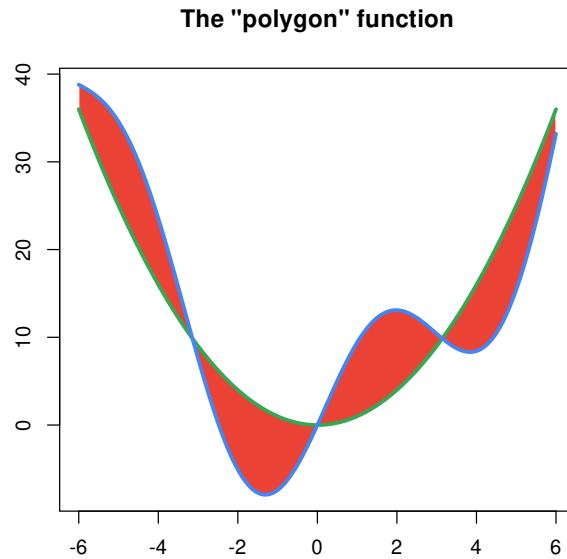


图 10.9: 区域重叠 polygon 函数

```
plot(0, 0,
  xlim = c(1, 5), ylim = c(-.5, 4),
  axes = F,
  xlab = "", ylab = ""
)
for (i in 0:4) {
  for (j in 1:5) {
    n <- 5 * i + j
    points(j, i,
      pch = n,
      cex = 3
    )
    text(j, i - .3, as.character(n))
  }
}
```

点、线、多边形和圆聚集在图 10.11 中

```
# https://jeroen.github.io/uros2018/#23
plot.new()
plot.window(xlim = c(0, 100), ylim = c(0, 100))
polygon(c(10, 40, 80), c(10, 80, 40), col = "hotpink")
text(40, 90, labels = "My drawing", col = "navyblue", cex = 3)
symbols(c(70, 80, 90), c(20, 50, 80),
  circles = c(10, 20, 10),
  bg = c("#4285f4", "#EA4335", "red"), add = TRUE, lty = "dashed"
)
```

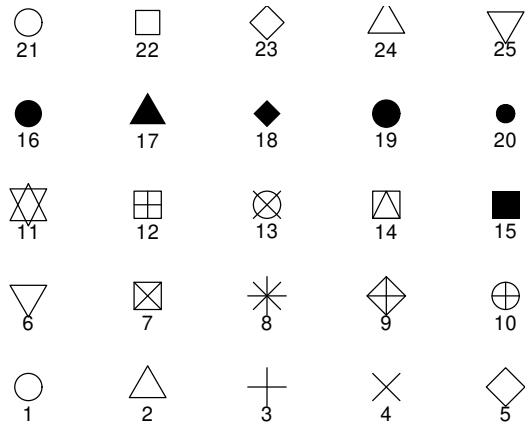


图 10.10: cex 支持的符号

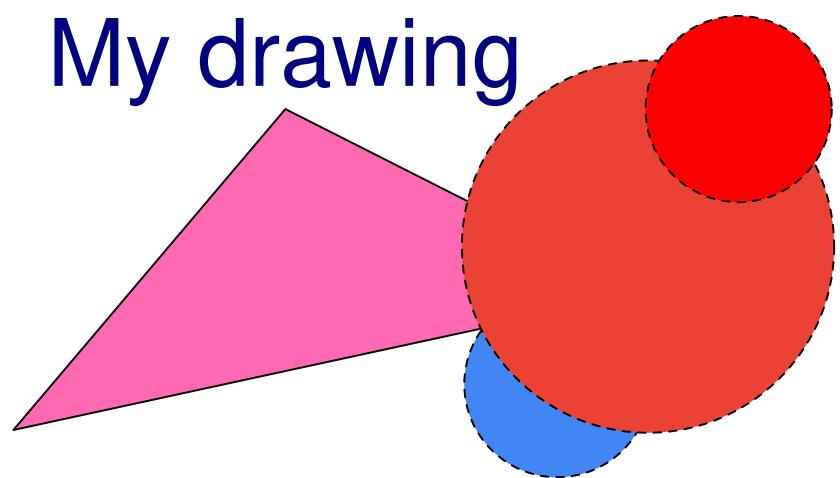


图 10.11: 多边形和符号元素

在介绍各种统计图形之前，先介绍几个绘图函数 `plot` 和 `text` 还有 `par` 参数设置，作为最简单的开始，尽量依次介绍其中的每个参数的含义并附上图形对比。

```

y <- x <- 1:4
plot(x, y, ann = F, col = "blue", pch = 16)
text(x, y,
  labels = c("1st", "2nd", "3rd", "4th"),
  col = "red", pos = c(3, 4, 4, 1), offset = 0.6
)
ahat <- "sigma"
# title(substitute(hat(a) == ahat, list(ahat = ahat)))
title(bquote(hat(a) == .(ahat)))

```

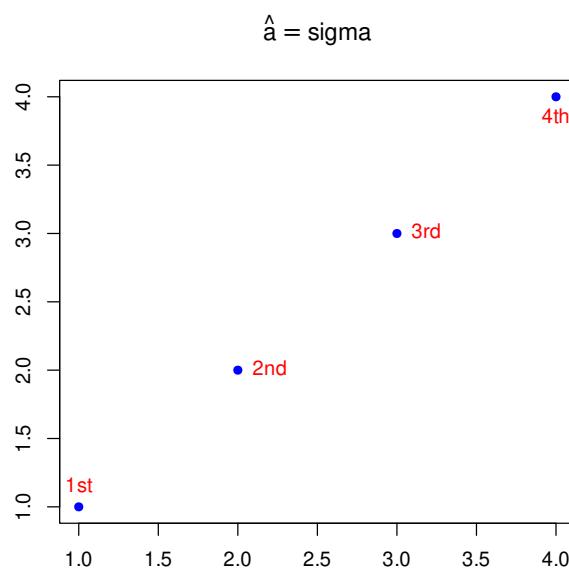


图 10.12: `pos` 位置参数

其中 `labels`, `pos` 都是向量化的参数

### 10.1.3 参考线

矩形网格线是用做背景参考线的，常常是淡灰色的细密虚线，`plot` 函数的 `panel.first` 参数和 `grid` 函数常用来画这种参考线

```

# modified from https://yihui.name/cn/2018/02/cohen-s-d/
n <- 30 # 样本量 (只是一个例子)
x <- seq(0, 12, 0.01)
par(mar = c(4, 4, 0.2, 0.1))
plot(x / sqrt(n), 2 * (1 - pt(x, n - 1)),
  xlab = expression(d = x / sqrt(n)),
  type = "l", panel.first = grid()

```

```
)  
abline(v = c(0.01, 0.2, 0.5, 0.8, 1.2, 2), lty = 2)
```

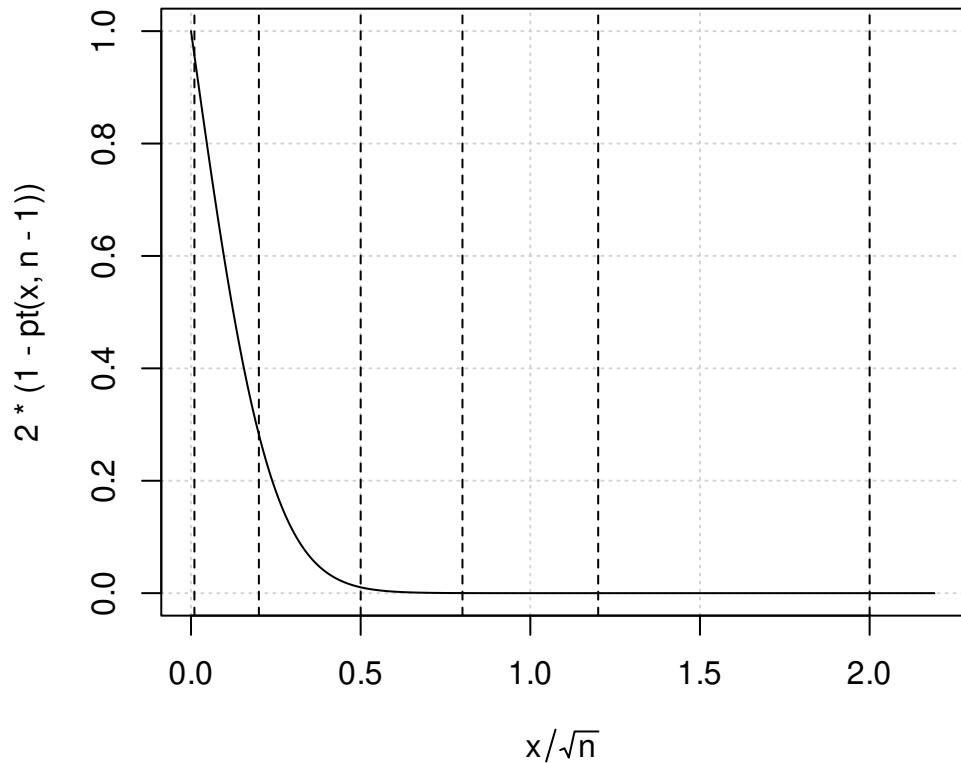
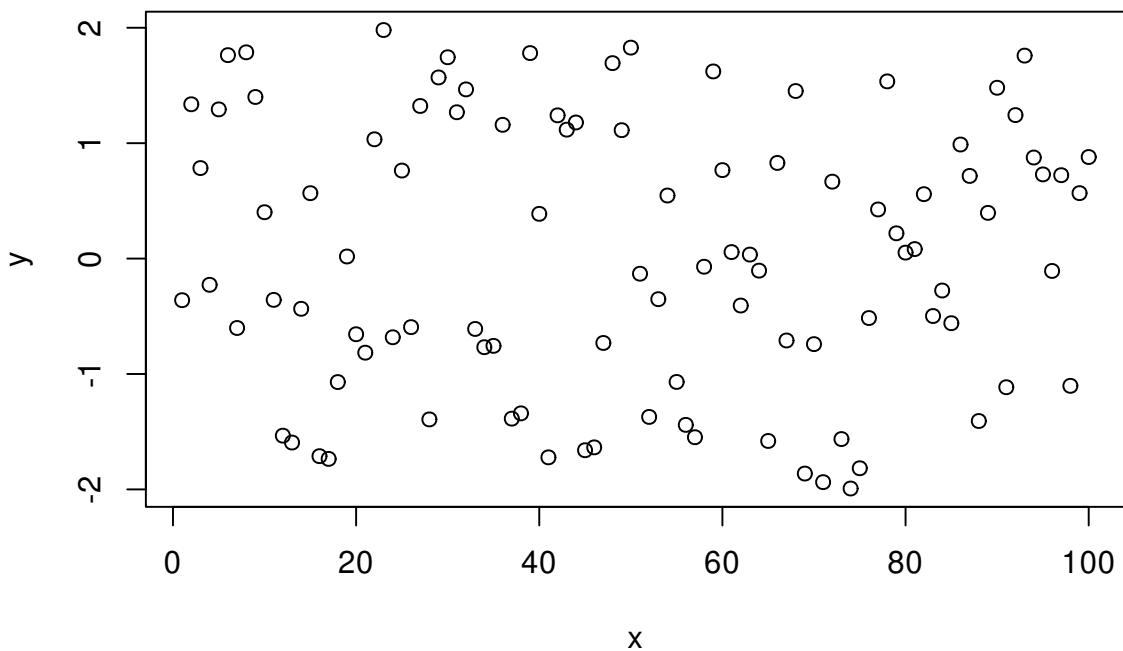


图 10.13: 添加背景参考线

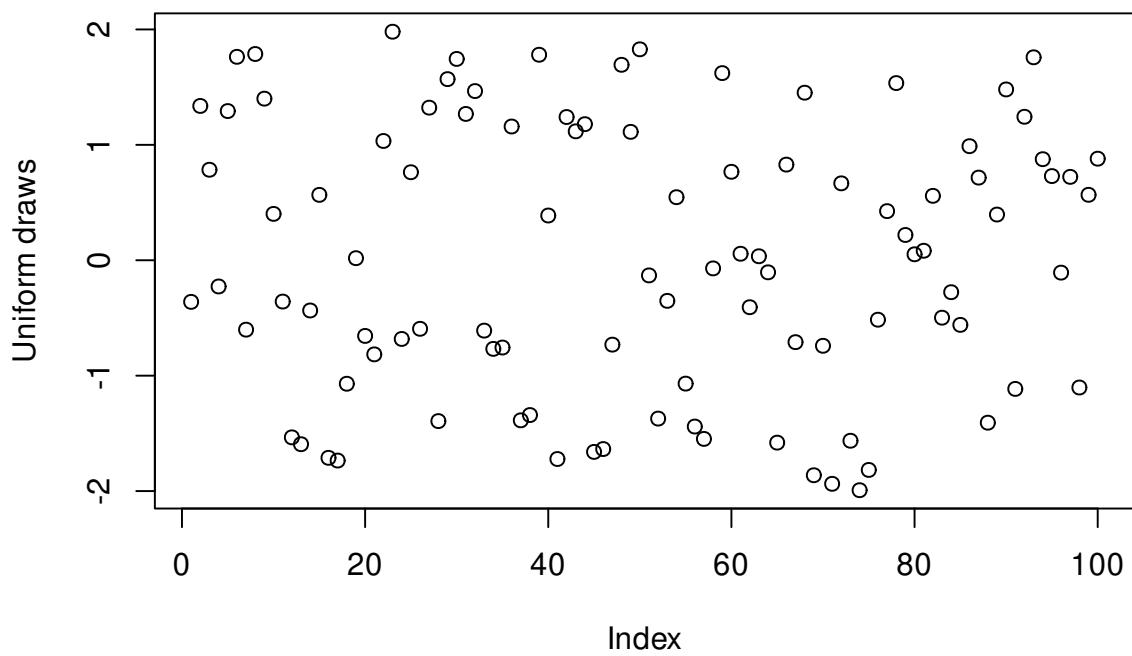
#### 10.1.4 坐标轴

图形控制参数默认设置下 `par` 通常的一幅图形，改变坐标轴标签是很简单的

```
x <- 1:100  
y <- runif(100, -2, 2)  
plot(x, y)
```



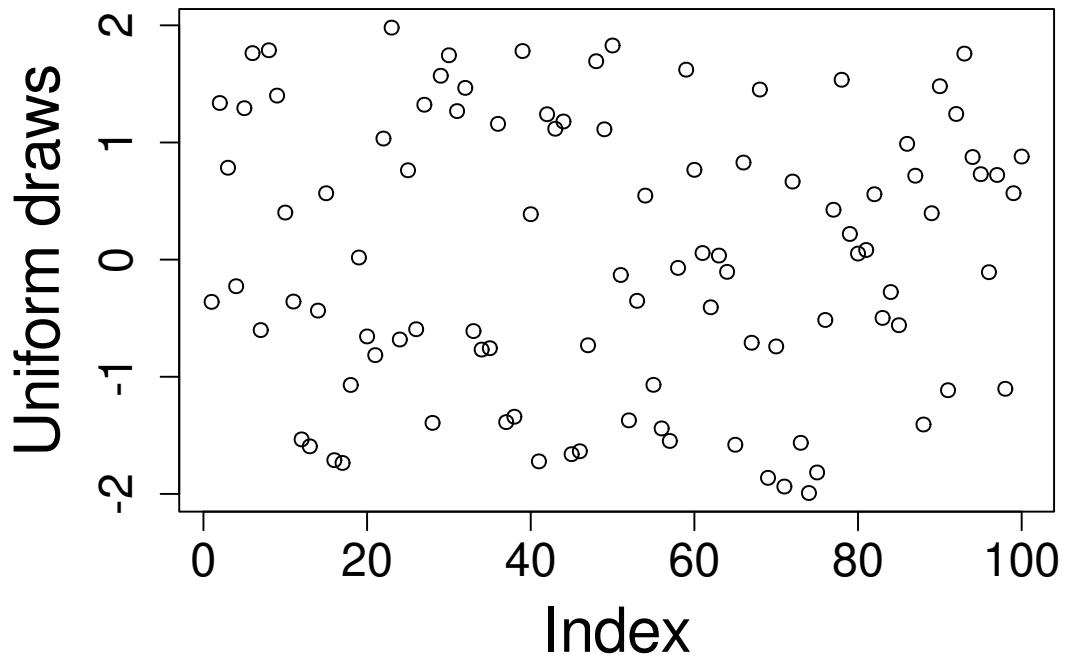
```
plot(x, y, xlab = "Index", ylab = "Uniform draws")
```



改变坐标轴标签和标题

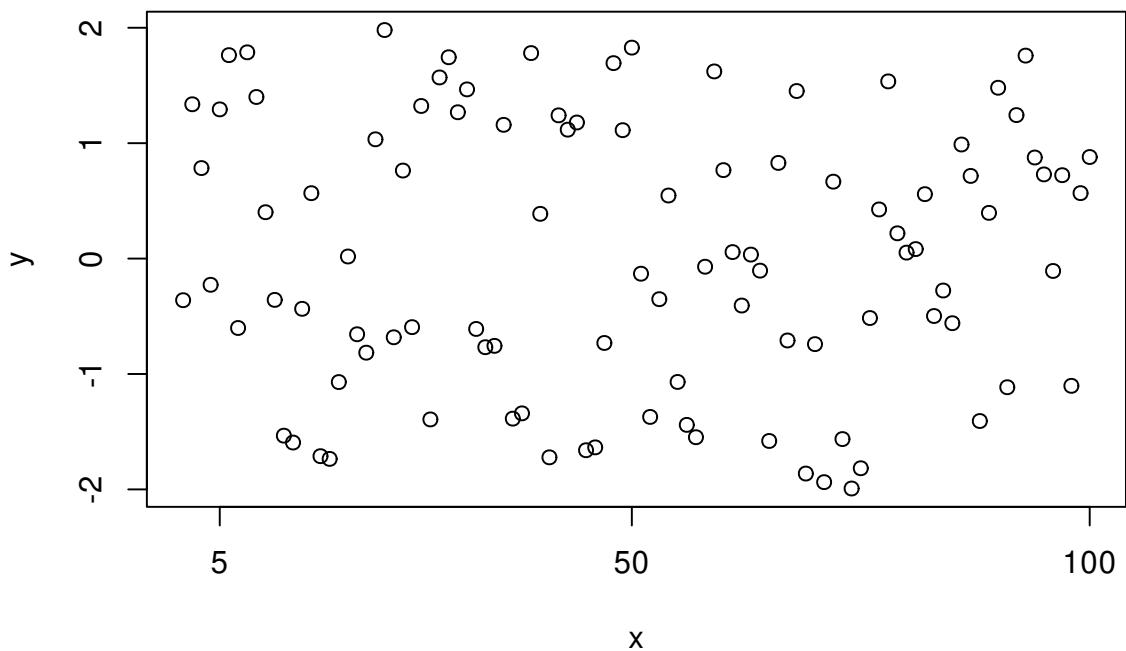
```
op <- par(no.readonly = TRUE) # 保存默认的 par 设置
par(cex.lab = 1.5, cex.axis = 1.3)
plot(x, y, xlab = "Index", ylab = "Uniform draws")

# 设置更大的坐标轴标签内容
par(mar = c(6, 6, 3, 3), cex.axis = 1.5, cex.lab = 2)
plot(x, y, xlab = "Index", ylab = "Uniform draws")
```



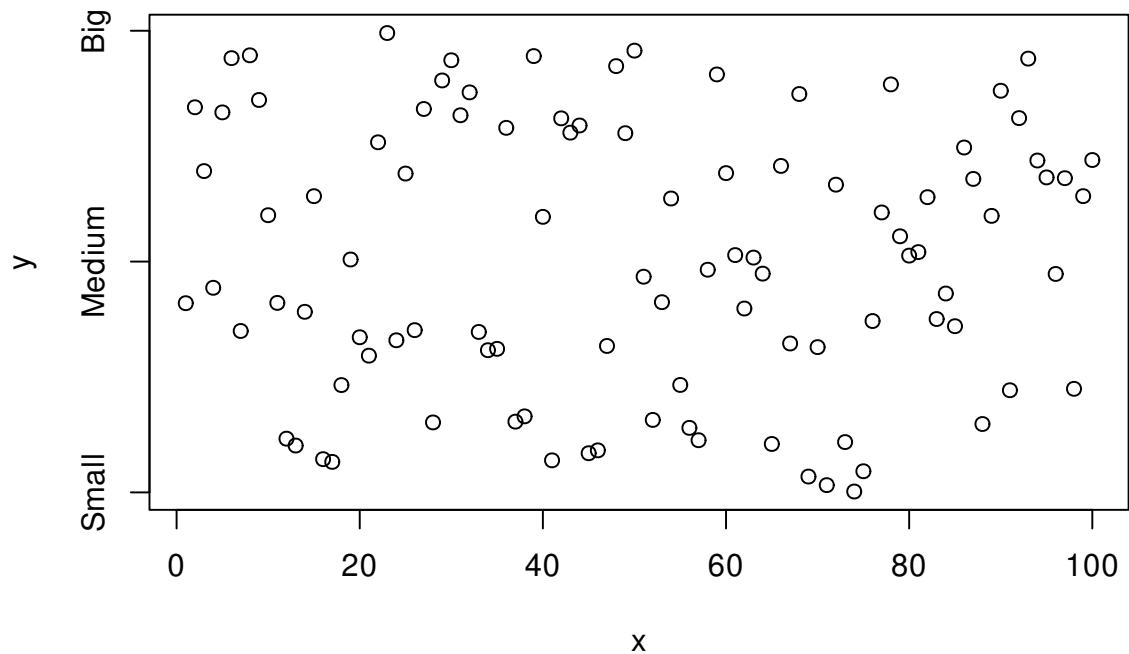
使用 `axis` 函数可以更加精细地控制坐标轴

```
par(op) # 恢复默认的 par 设置
plot(x, y, xaxt = "n") # 去掉 x 轴
axis(side = 1, at = c(5, 50, 100)) # 添加指定的刻度标签
```



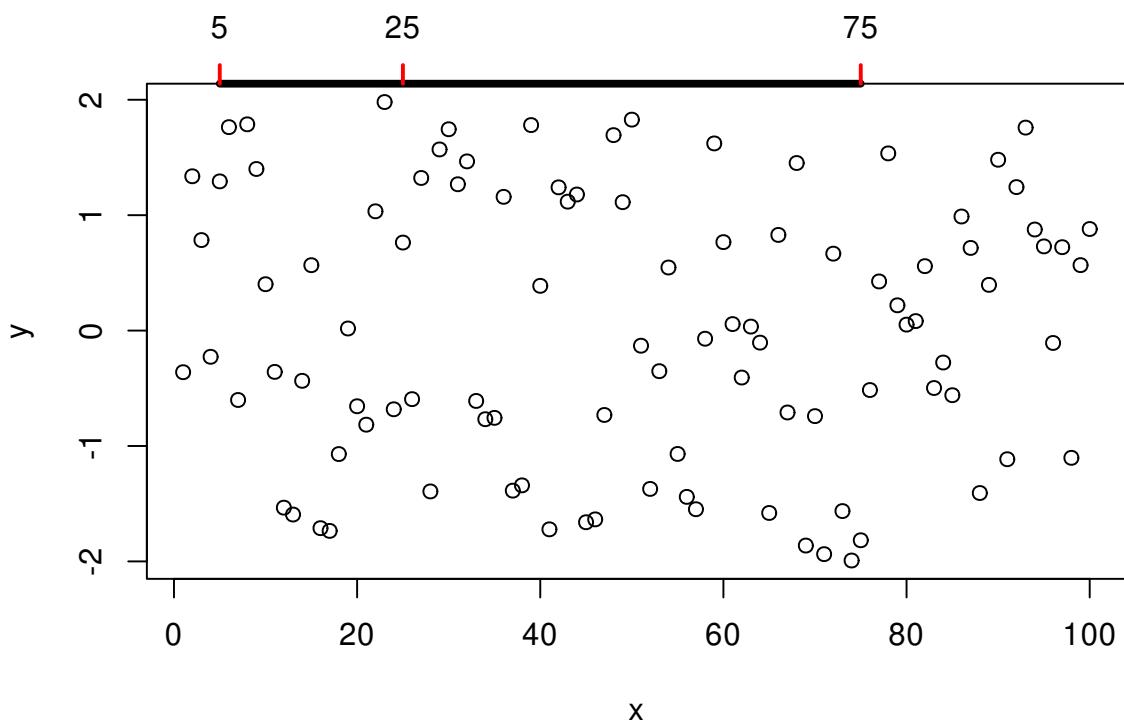
指定刻度标签的内容

```
plot(x, y, yaxt = "n")
axis(side = 2, at = c(-2, 0, 2), labels = c("Small", "Medium", "Big"))
```



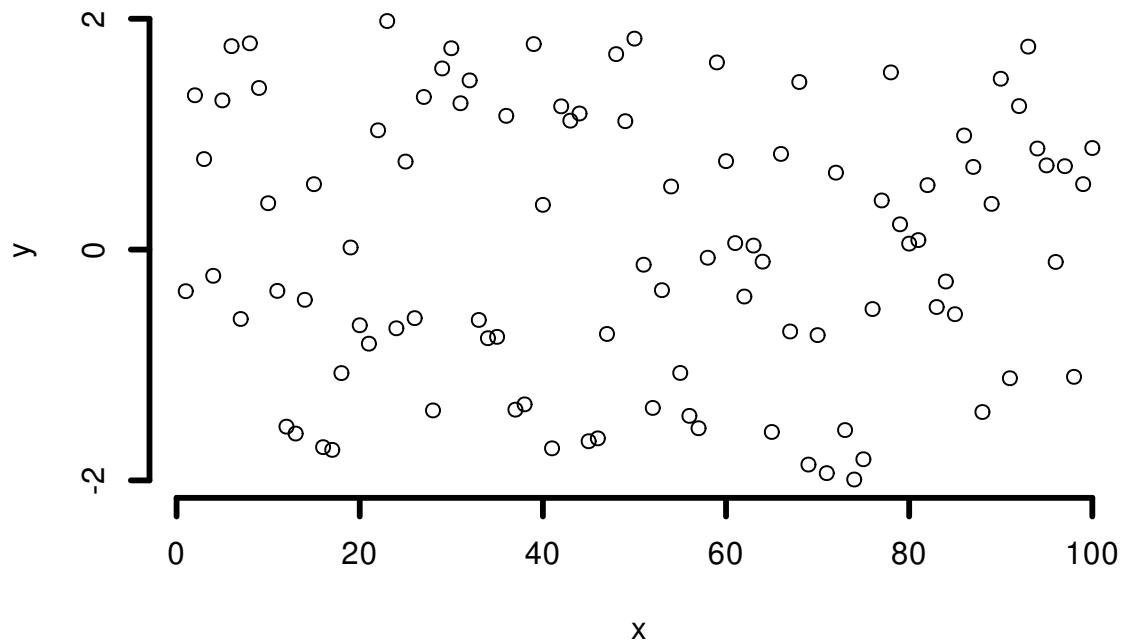
控制刻度线和轴线和刻度标签

```
plot(x, y)
axis(side = 3, at = c(5, 25, 75), lwd = 4, lwd.ticks = 2, col.ticks = "red")
```



还可以把 box 移除，绘图区域的边框去掉，只保留坐标轴

```
plot(x, y, bty = "n", xaxt = "n", yaxt = "n")
axis(side = 1, at = seq(0, 100, 20), lwd = 3)
axis(side = 2, at = seq(-2, 2, 2), lwd = 3)
```



```
# 双Y轴
N <- 200
x <- seq(-4, 4, length = N)
y1 <- sin(x)
y2 <- cos(x)
op <- par(mar = c(5, 4, 4, 4)) # Add some space in the right margin
# The default is c(5,4,4,2) + .1
xlim <- range(x)
ylim <- c(-1.1, 1.1)
plot(x, y1,
      col = "blue", type = "l",
      xlim = xlim, ylim = ylim,
      axes = F, xlab = "", ylab = "", main = "Title"
)
axis(1)
axis(2, col = "blue")
par(new = TRUE)
plot(x, y2,
      col = "red", type = "l",
      xlim = xlim, ylim = ylim,
      axes = F, xlab = "", ylab = "", main = ""
)
axis(4, col = "red")
mtext("First Y axis", 2, line = 2, col = "blue", cex = 1.2)
```

```
mtext("Second Y axis", 4, line = 2, col = "red", cex = 1.2)
```

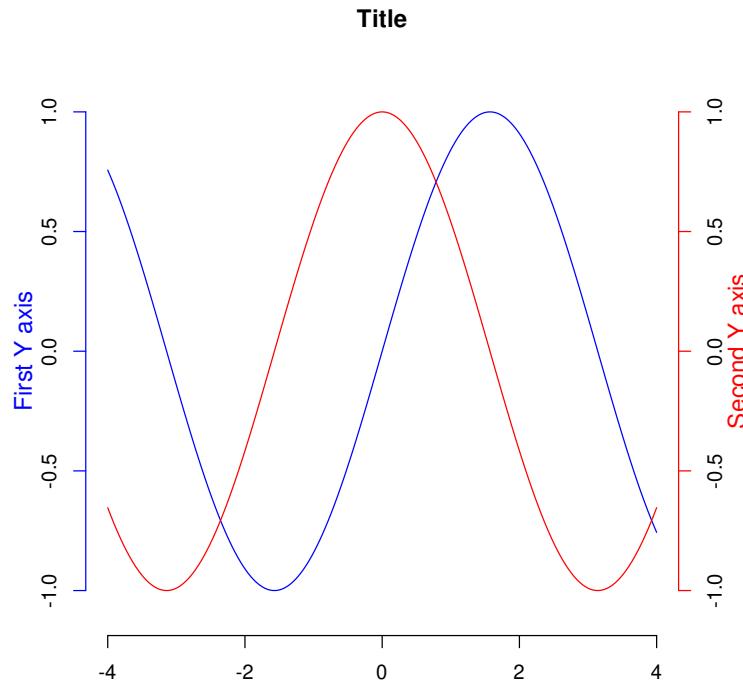


图 10.14: 两个 Y 轴

```
# 1,2,3,4 分别代表下左上右四个位置
```

调整坐标轴标签的距离

```
## Changing default gap between labels:
plot(c(0, 100), c(0, 50), type = "n", axes = FALSE, ann = FALSE)
title(quote("axis(1, .., gap.axis = f)," ~ ~ f >= 0))
axis(2, at = 5 * (0:10), las = 1, gap.axis = 1 / 4)
gaps <- c(4, 2, 1, 1 / 2, 1 / 4, 0.1, 0)
chG <- paste0(
  ifelse(gaps == 1, "default: ", ""),
  "gap.axis=", formatC(gaps)
)
jj <- seq_along(gaps)
linG <- -2.5 * (jj - 1)
for (j in jj) {
  isD <- gaps[j] == 1 # is default
  axis(1,
    at = 5 * (0:20), gap.axis = gaps[j], padj = -1, line = linG[j],
    col.axis = if (isD) "forest green" else 1, font.axis = 1 + isD
  )
}
```

```
mtext(chG,  
  side = 1, padj = -1, line = linG - 1 / 2, cex = 3 / 4,  
  col = ifelse(gaps == 1, "forest green", "blue3")  
)
```

axis(1, ..., gap.axis = f),  $f \geq 0$

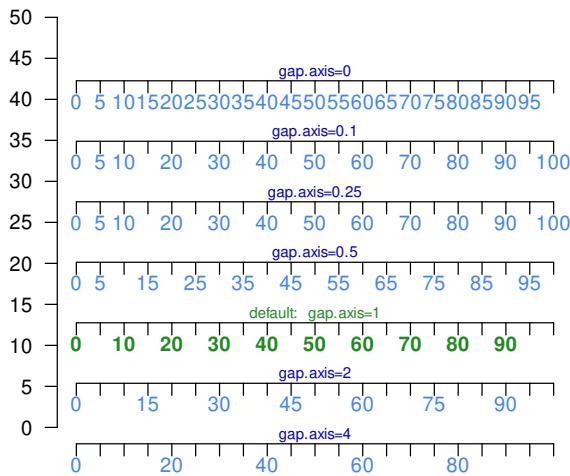


图 10.15: gap.axis 用法

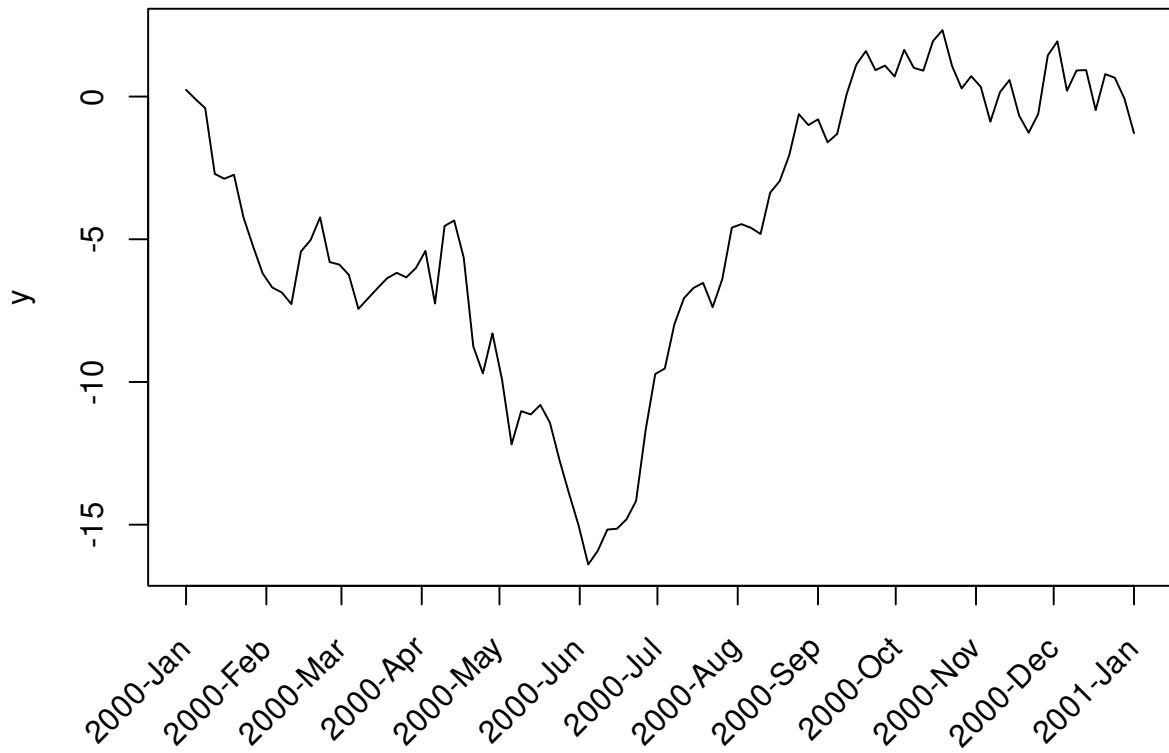
```
## now shrink the window (in x- and y-direction) and observe the axis labels drawn
```

旋转坐标轴标签

```
# Rotated axis labels in R plots  
# https://menugget.blogspot.com/2014/08/rotated-axis-labels-in-r-plots.html  
# Example data  
tmin <- as.Date("2000-01-01")  
tmax <- as.Date("2001-01-01")  
tlab <- seq(tmin, tmax, by = "month")  
lab <- format(tlab, format = "%Y-%b")  
set.seed(111)  
x <- seq(tmin, tmax, length.out = 100)  
y <- cumsum(rnorm(100))  
  
# Plot  
# png("plot_w_rotated_axis_labels.png", height = 3,  
#      width = 6, units = "in", res = 300)  
op <- par(mar = c(6, 4, 1, 1))  
plot(x, y, t = "l", xaxt = "n", xlab = "")  
axis(1, at = tlab, labels = FALSE)  
text(
```

```
x = tlab, y = par()$usr[3] - 0.1 * (par()$usr[4] - par()$usr[3]),
labels = lab, srt = 45, adj = 1, xpd = TRUE
)
```

C



```
par(op)
# dev.off()
```

旋转坐标轴标签的例子来自手册《R FAQ》的第 7 章第 27 个问题 [Hornik, 2020]，在基础图形中，旋转坐标轴标签需要 `text()` 而不是 `mttext()`，因为后者不支持 `par("srt")`

```
## Increase bottom margin to make room for rotated labels
par(mar = c(5, 4, .5, 2) + 0.1)
## Create plot with no x axis and no x axis label
plot(1:8, xaxt = "n", xlab = "")
## Set up x axis with tick marks alone
axis(1, labels = FALSE)
## Create some text labels
labels <- paste("Label", 1:8, sep = " ")
## Plot x axis labels at default tick marks
text(1:8, par("usr")[3] - 0.5,
  srt = 45, adj = 1,
  labels = labels, xpd = TRUE
)
## Plot x axis label at line 6 (of 7)
```

```
mtext(side = 1, text = "X Axis Label", line = 4)
```

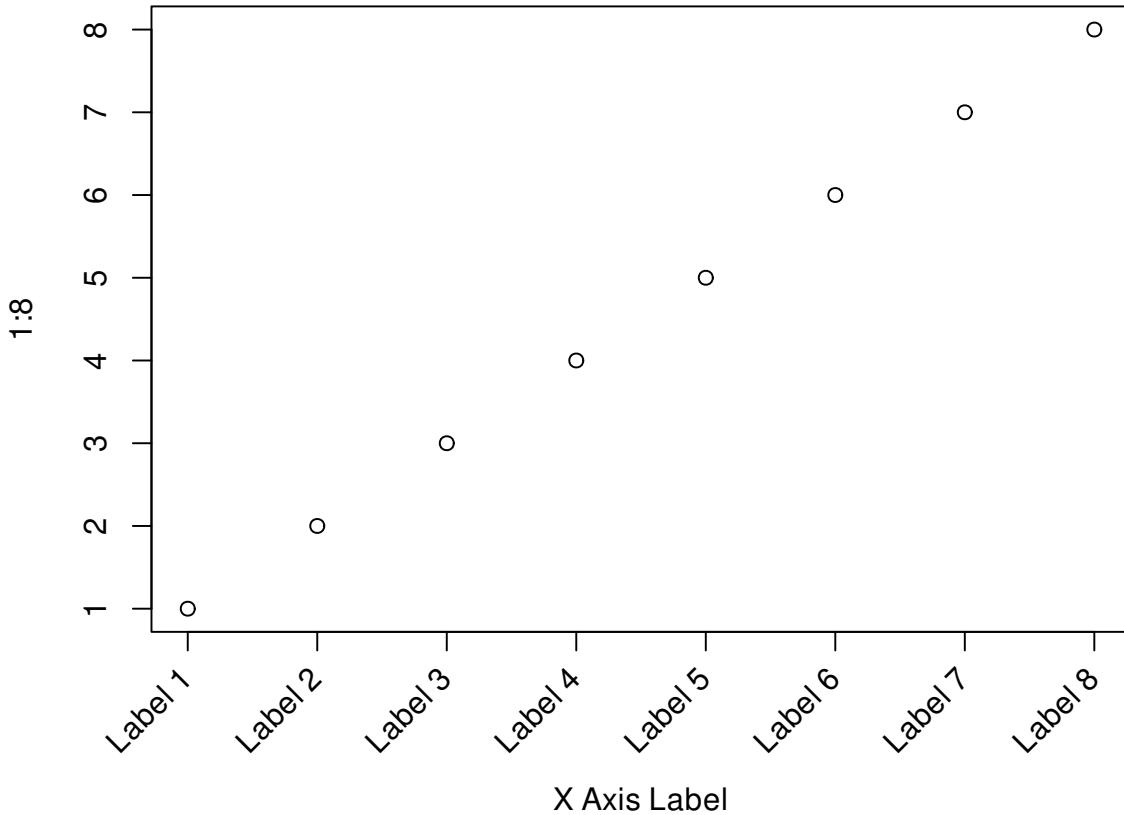


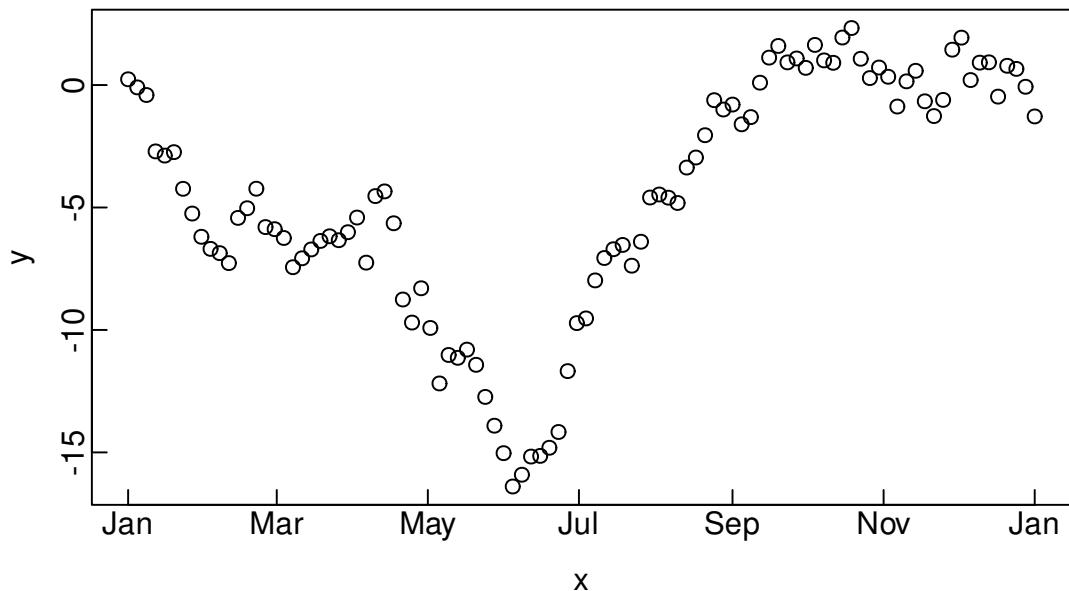
图 10.16: 旋转坐标轴标签

`srt = 45` 表示文本旋转角度, `xpd = TRUE` 允许文本越出绘图区域, `adj = 1` to place the right end of text at the tick marks; You can adjust the value of the 0.5 offset as required to move the axis labels up or down relative to the x axis. 详细地参考 [Murrell, 2003]

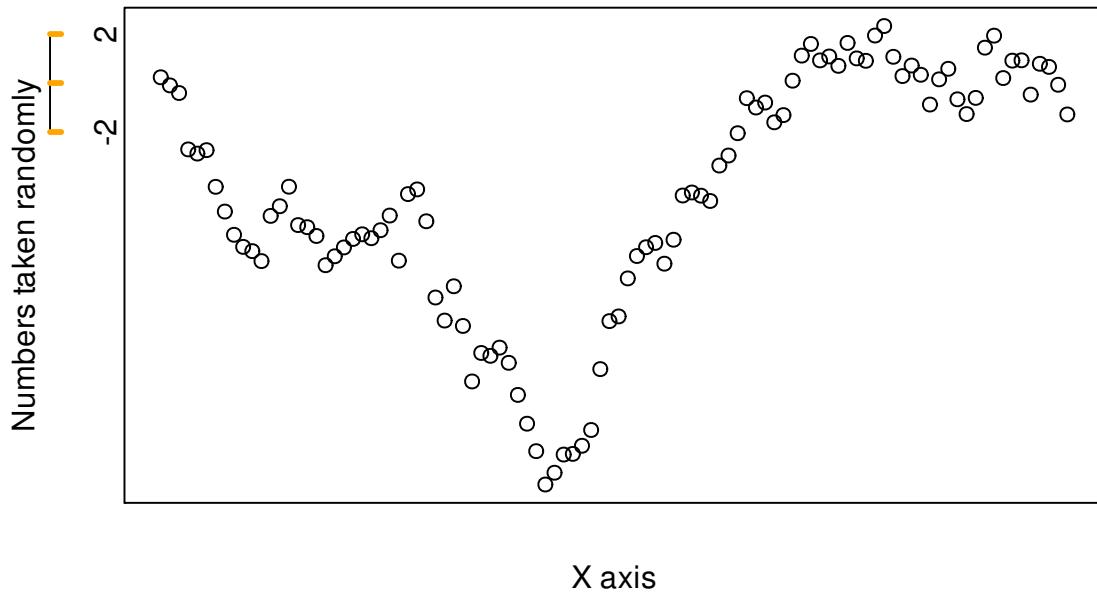
### 10.1.5 刻度线

通过 `par` 或 `axis` 函数实现刻度线的精细操作, `tcl` 控制刻度线的长度, 正值让刻度画在绘图区域内, 负值正好相反, 画在外面, `mgp` 参数有三个值, 第一个值控制绘图区域和坐标轴标题之间的行数, 第二个是绘图区域与坐标轴标签的行数, 第三个绘图区域与轴线的行数, 行数表示间距

```
par(tcl = 0.4, mgp = c(1.5, 0, 0))
plot(x, y)
```



```
# 又一个例子
par(op)
plot(x, y, xaxt = "n", yaxt = "n", xlab = "", ylab = "")
axis(side = 1, at = seq(5, 95, 30), tcl = 0.4, lwd.ticks = 3, mgp = c(0, 0.5, 0))
mtext(side = 1, text = "X axis", line = 1.5)
# mtext 设置坐标轴标签
axis(side = 2, at = seq(-2, 2, 2), tcl = 0.3, lwd.ticks = 3, col.ticks = "orange", mgp = c(0, 0, 2))
mtext(side = 2, text = "Numbers taken randomly", line = 2.2)
```



### 10.1.6 标题

添加多个标题

```
N <- 200
x <- runif(N, -4, 4)
y <- sin(x) + .5 * rnorm(N)
plot(x, y, xlab = "", ylab = "", main = "")
mtext("Subtitle", 3, line = .8)
mtext("Title", 3, line = 2, cex = 1.5)
mtext("X axis", 1, line = 2.5, cex = 1.5)
mtext("X axis subtitle", 1, line = 3.7)
```

### 10.1.7 注释

数学符号注释, 图10.18 自定义坐标轴 [Murrell and Ihaka, 2000]。

```
# 自定义坐标轴
plot(c(1, 1e6), c(-pi, pi),
      type = "n",
      axes = FALSE, ann = FALSE, log = "x"
)
axis(1,
      at = c(1, 1e2, 1e4, 1e6),
```

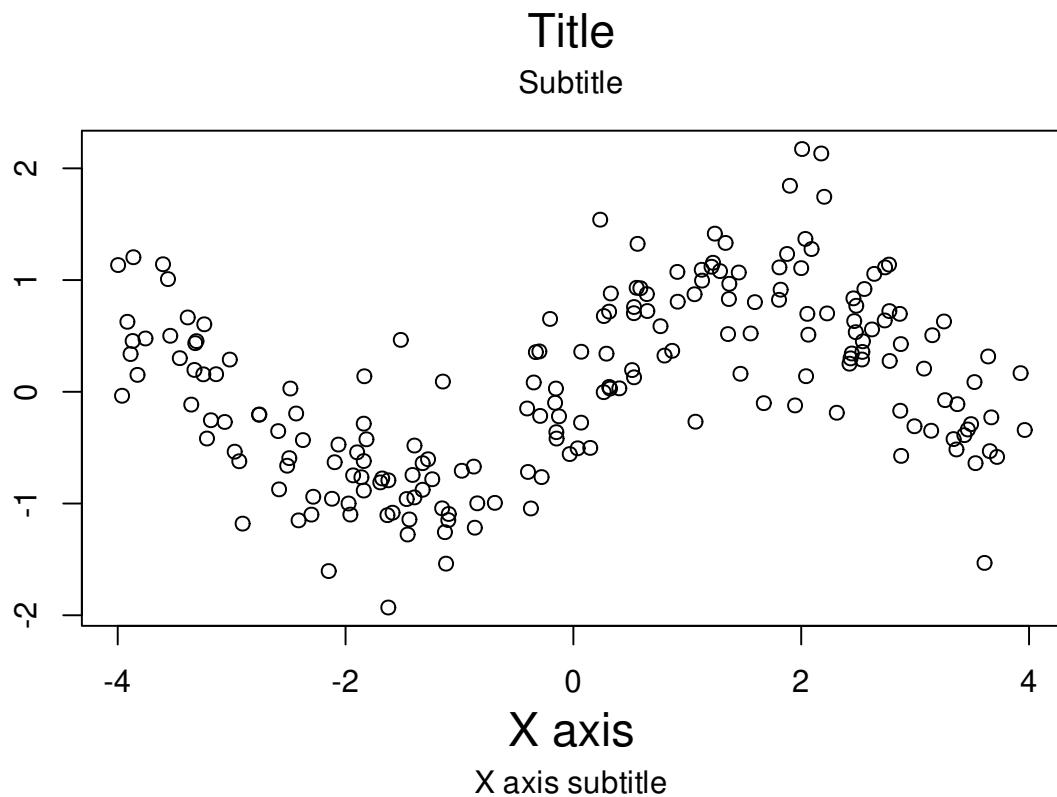


图 10.17: 图标题/子标题 x 轴标题/子标题

```
  labels = expression(1, 10^2, 10^4, 10^6)
)
axis(2,
  at = c(-pi, -pi / 2, 0, pi / 2, pi),
  labels = expression(-pi, -pi / 2, 0, pi / 2, pi)
)
text(1e3, 0, expression(italic("Customized Axes")))
box()
```

在标题中添加数学公式

```
x <- seq(-5, 5, length = 200)
y <- sqrt(1 + x^2)
plot(y ~ x,
  type = "l",
  ylab = expression(sqrt(1 + x^2))
)
title(main = expression(
  "graph of the function f"(x) == sqrt(1 + x^2)
))
```

修改参数使用 `substitute` 函数批量生成

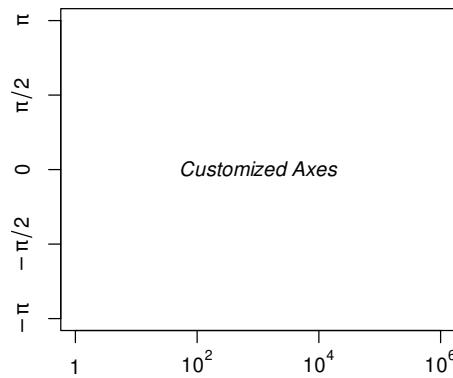


图 10.18: 创建自定义的坐标轴和刻度标签

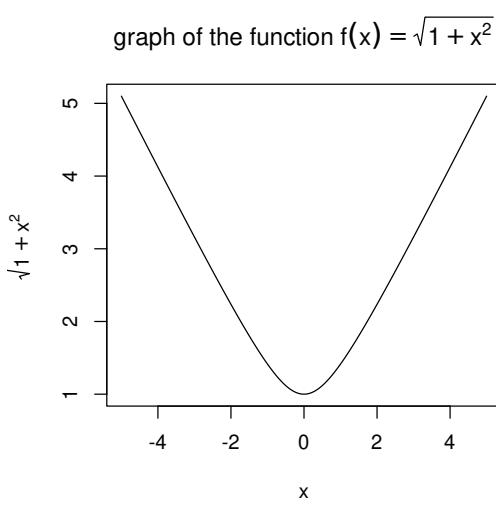


图 10.19: 标题含有数学公式

```

x <- seq(-5, 5, length = 200)
for (i in 1:4) { # 画四个图
  y <- sqrt(i + x^2)
  plot(y ~ x,
    type = "l",
    ylim = c(0, 6),
    ylab = substitute(
      expression(sqrt(i + x^2)),
      list(i = i)
    )
  )
  title(main = substitute(
    "graph of the function f"(x) == sqrt(i + x^2),
    list(i = i)
  ))
}

```

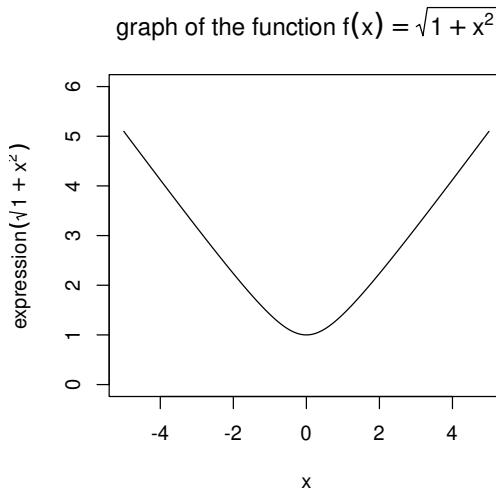


图 10.20: 批量生成函数图形

基础绘图函数，如 `plot` 标签 `xlab` 支持 `Unicode` 代码表示的希腊字母，常用字母表备查，公式环境下，也可以用在绘图中

表 10.1: 希腊字母表

希腊字母	LaTeX 代码	Unicode 代码	希腊字母	LaTeX 代码	Unicode 代码
$\alpha$	<code>\alpha</code>	<code>\u03b1</code>	$\mu$	<code>\mu</code>	<code>\u03bc</code>
$\beta$	<code>\beta</code>	<code>\u03b2</code>	$\nu$	<code>\nu</code>	<code>\u03bd</code>
$\gamma$	<code>\gamma</code>	<code>\u03b3</code>	$\xi$	<code>\xi</code>	<code>\u03be</code>
$\delta$	<code>\delta</code>	<code>\u03b4</code>	$\varphi$	<code>\varphi</code>	<code>\u03c6</code>
$\epsilon$	<code>\epsilon</code>	<code>\u03b5</code>	$\pi$	<code>\pi</code>	<code>\u03c0</code>
$\zeta$	<code>\zeta</code>	<code>\u03b6</code>	$\rho$	<code>\rho</code>	<code>\u03c1</code>

希腊字母	LaTeX 代码	Unicode 代码	希腊字母	LaTeX 代码	Unicode 代码
$\eta$	\eta	\u03B7	$v$	\upsilon	\u03C5
$\theta$	\theta	\u03B8	$\phi$	\phi	\u03C6
$\iota$	\iota	\u03B9	$\chi$	\chi	\u03C7
$\kappa$	\kappa	\u03BA	$\psi$	\psi	\u03C8
$\lambda$	\lambda	\u03BB	$\omega$	\omega	\u03C9
$\sigma$	\sigma	\u03C3	$\tau$	\tau	\u03C4

表 10.2: 数字上下标

上标数字	LaTeX 代码	Unicode 代码	下标数字	LaTeX 代码	Unicode 代码
0	\{}^0	\u2070	0	\{}_0	\u2080
1	\{}^1	\u00B9	1	\{}_1	\u2081
2	\{}^2	\u00B2	2	\{}_2	\u2082
3	\{}^3	\u00B3	3	\{}_3	\u2083
4	\{}^4	\u2074	4	\{}_4	\u2084
5	\{}^5	\u2075	5	\{}_5	\u2085
6	\{}^6	\u2076	6	\{}_6	\u2086
7	\{}^7	\u2077	7	\{}_7	\u2087
8	\{}^8	\u2078	8	\{}_8	\u2088
9	\{}^9	\u2079	9	\{}_9	\u2089
$n$	\{}^n	\u207F	$n$	\{}_n	-

其它字母, 请查看 [Unicode 字母表](#)

### 10.1.8 图例

```
x <- seq(-6, 6, length = 200)
y <- sin(x)
z <- cos(x)
plot(y ~ x,
  type = "l", lwd = 3,
  ylab = "", xlab = "angle", main = "Trigonometric functions"
)
abline(h = 0, lty = 3)
abline(v = 0, lty = 3)
lines(z ~ x, type = "l", lwd = 3, col = "red")
legend(-6, -1,
  yjust = 0,
  c("Sine", "Cosine"),
  lwd = 3, lty = 1, col = c(par("fg"), "red")
)
```

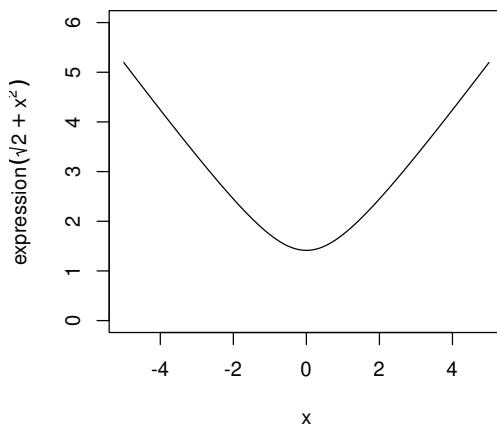
graph of the function  $f(x) = \sqrt{2 + x^2}$ 

图 10.21: 批量生成函数图形

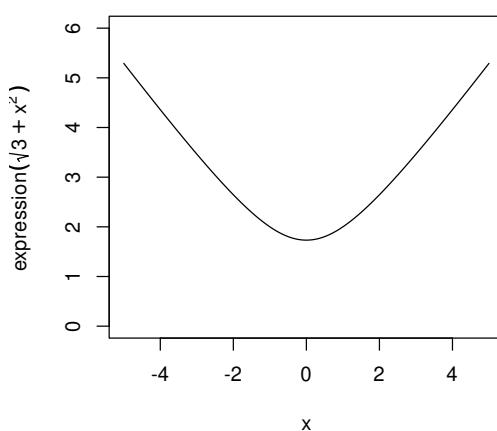
graph of the function  $f(x) = \sqrt{3 + x^2}$ 

图 10.22: 批量生成函数图形

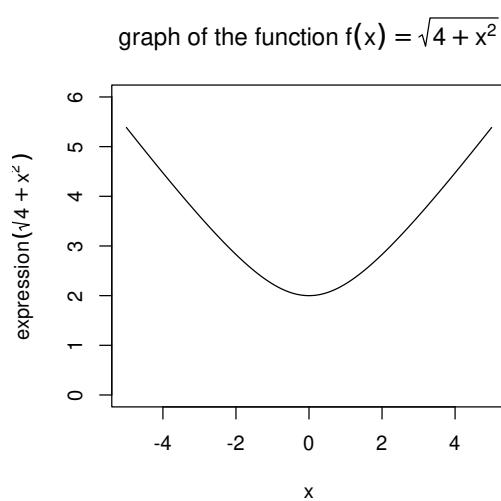


图 10.23: 批量生成函数图形

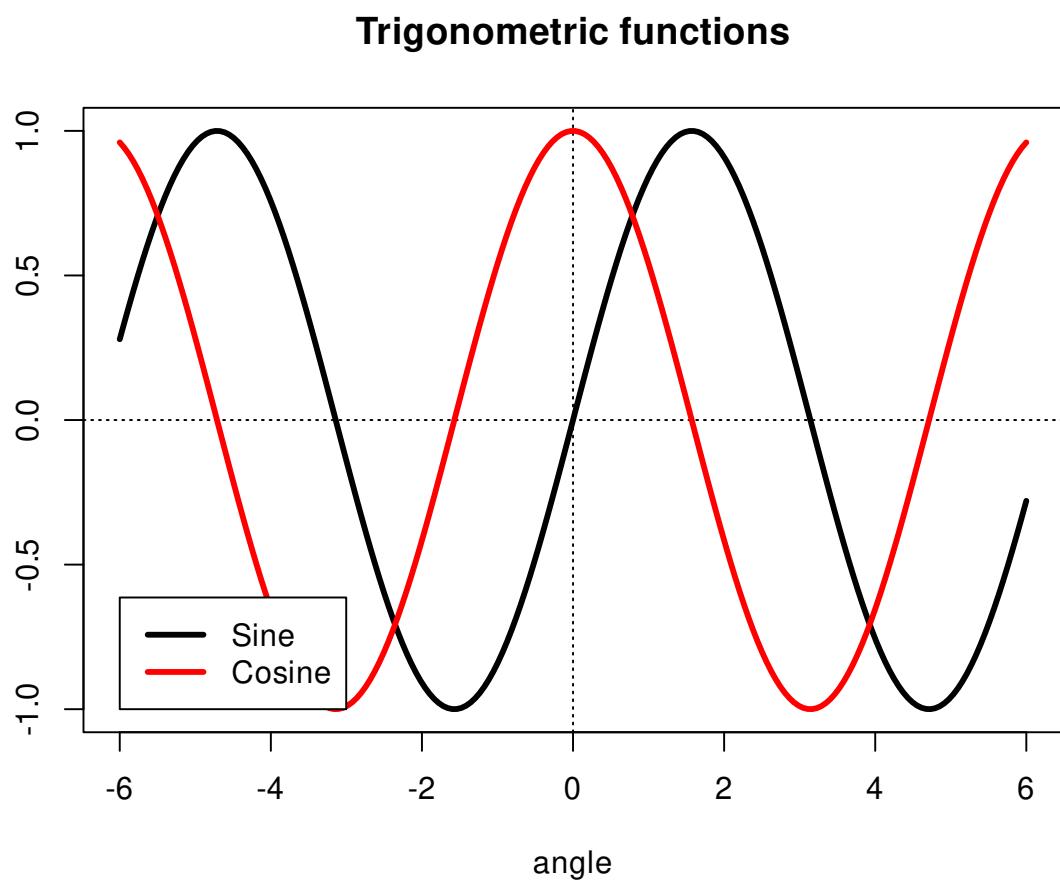


图 10.24: 三角函数添加图例

©

```
xmin <- par("usr")[1]
xmax <- par("usr")[2]
ymin <- par("usr")[3]
ymax <- par("usr")[4]

plot(y ~ x,
  type = "l", lwd = 3,
  ylab = "", xlab = "angle", main = "Trigonometric functions"
)
abline(h = 0, lty = 3)
abline(v = 0, lty = 3)
lines(z ~ x, type = "l", lwd = 3, col = "red")
legend("bottomleft",
  c("Sine", "Cosine"),
  lwd = 3, lty = 1, col = c(par("fg"), "red")
)
```

Trigonometric functions

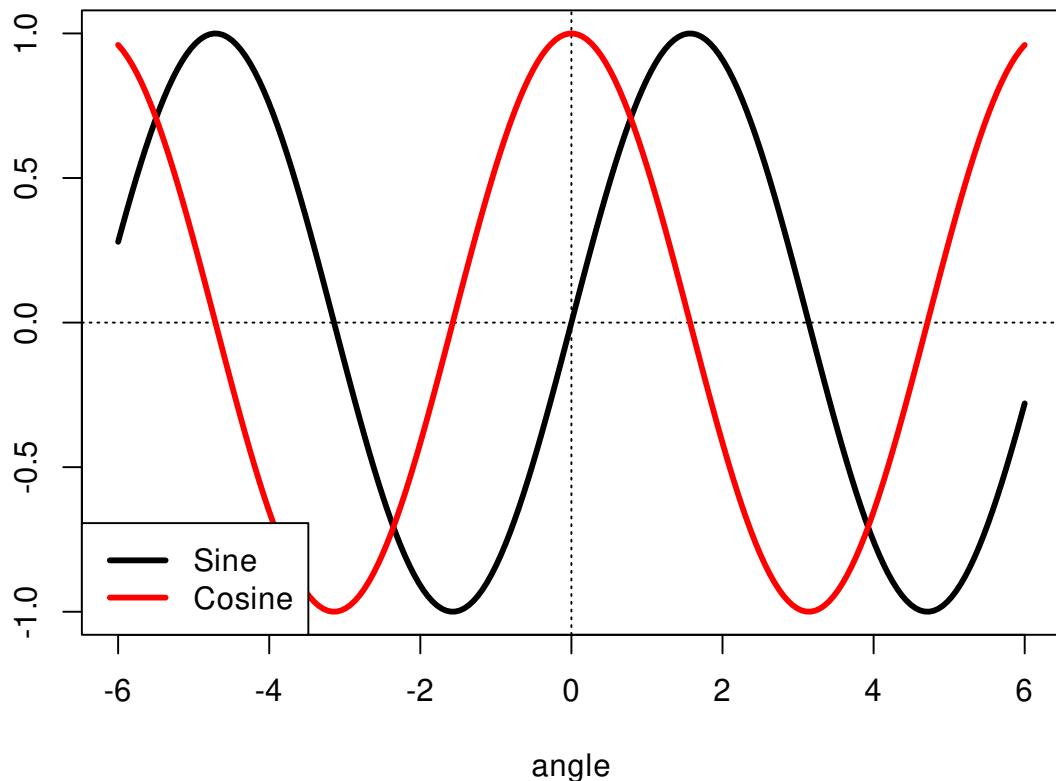


图 10.25: 设置图例的位置

```
plot(y ~ x,
  type = "l", lwd = 3,
```

```
ylab = "", xlab = "angle", main = "Trigonometric functions"
)
abline(h = 0, lty = 3)
abline(v = 0, lty = 3)
lines(z ~ x, type = "l", lwd = 3, col = "red")
legend("bottomleft",
  c("Sine", "Cosine"),
  inset = c(.03, .03),
  lwd = 3, lty = 1, col = c(par("fg"), "red")
)
```

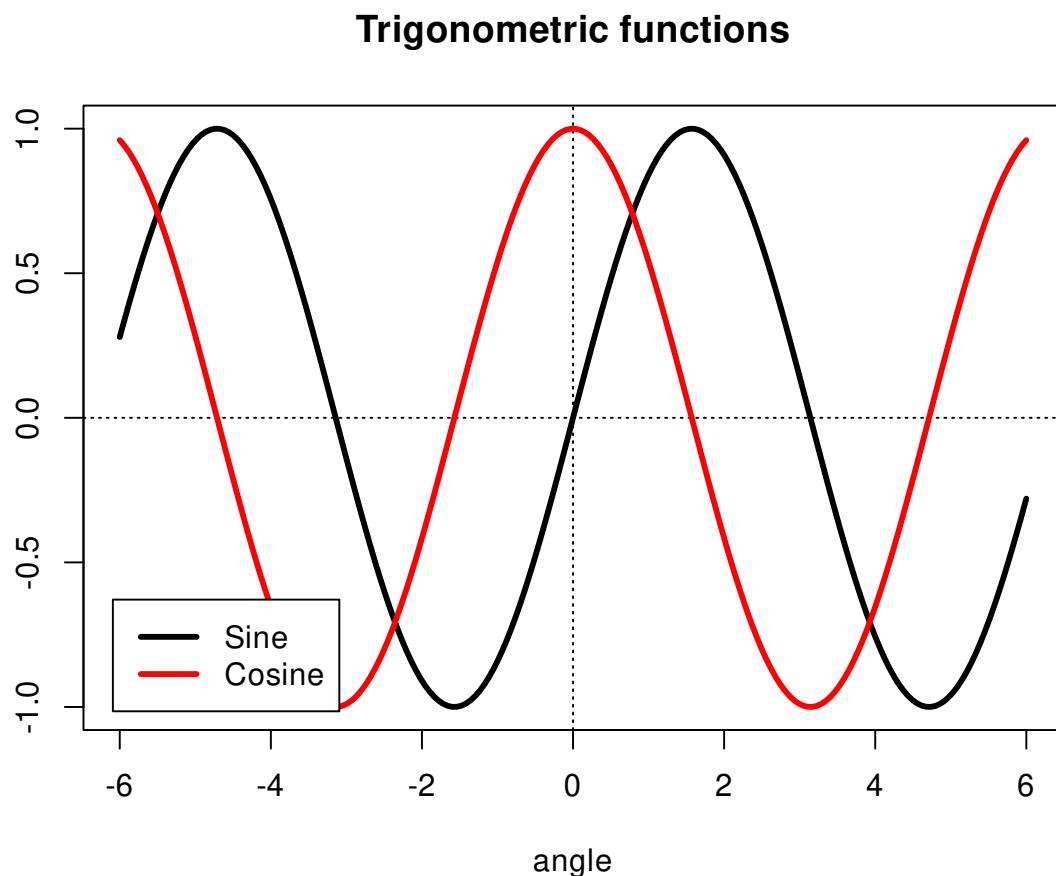


图 10.26: insert 函数微调图例位置

```
op <- par(no.readonly = TRUE)
plot(y ~ x,
  type = "l", lwd = 3,
  ylab = "", xlab = "angle", main = "Trigonometric functions"
)
abline(h = 0, lty = 3)
abline(v = 0, lty = 3)
lines(z ~ x, type = "l", lwd = 3, col = "red")
```

```
par(xpd = TRUE) # Do not clip to the drawing area 关键一行/允许出界
lambda <- .025
legend(par("usr")[1],
  (1 + lambda) * par("usr")[4] - lambda * par("usr")[3],
  c("Sine", "Cosine"),
  xjust = 0, yjust = 0,
  lwd = 3, lty = 1, col = c(par("fg"), "red")
)
```

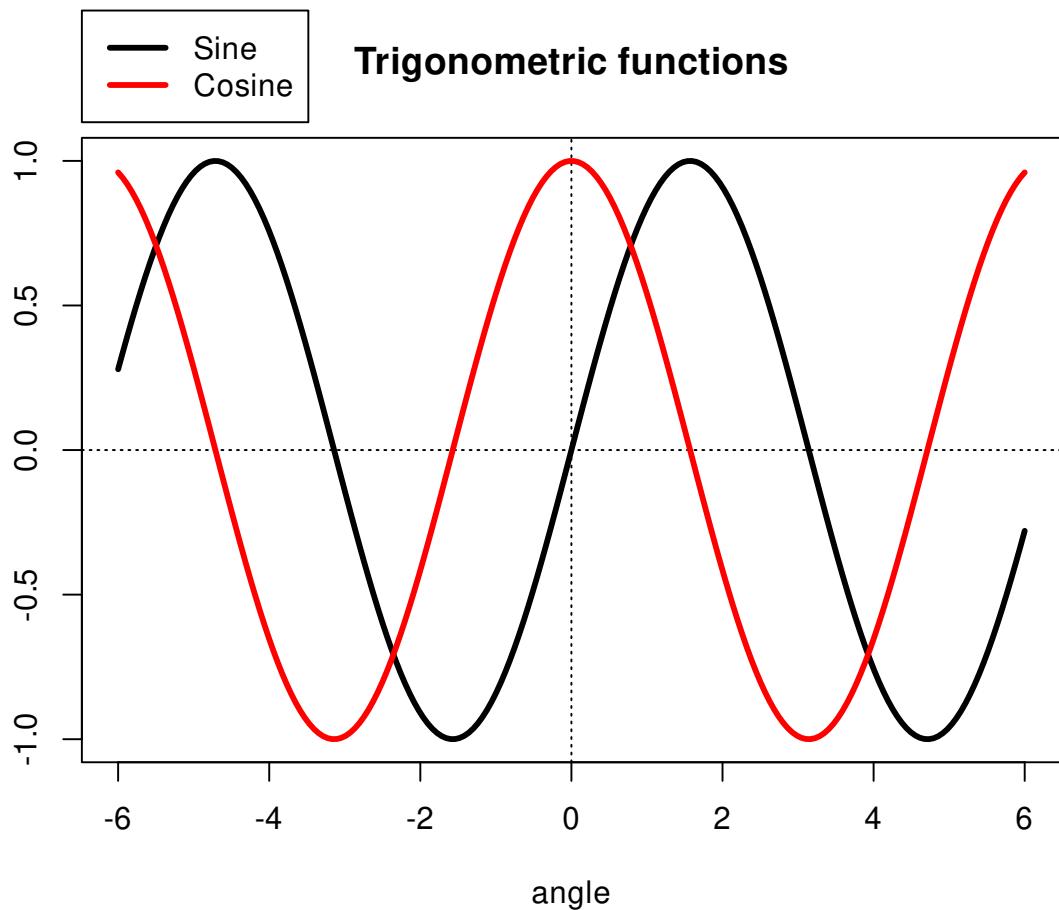


图 10.27: 将图例放在绘图区域外面

```
par(op)
```

Hmisc 包的 labcurve 函数可以在曲线上放置名称，而不是遥远的图例上

### 10.1.9 边空

边空分为内边空和外边空

line 第一行

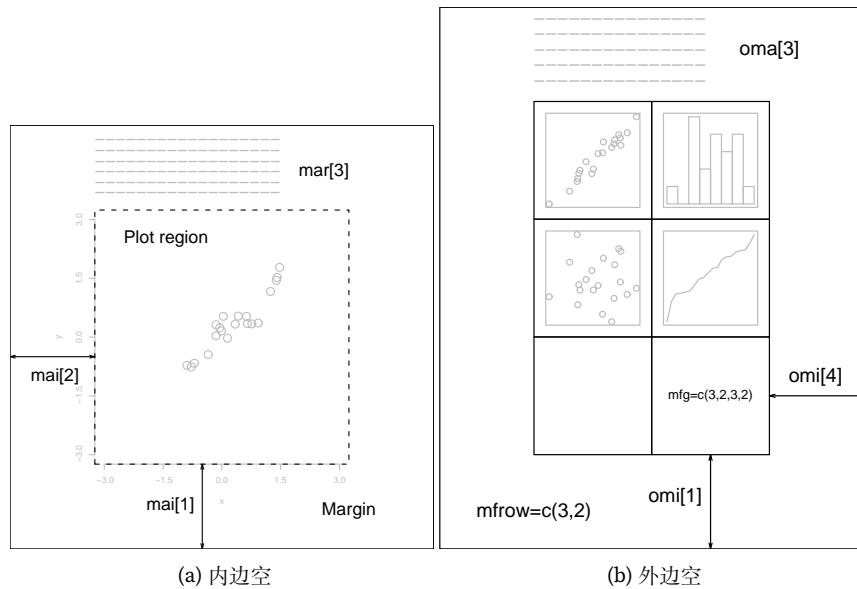


图 10.28: 边空

```

N <- 200
x <- runif(N, -4, 4)
y <- sin(x) + .5 * rnorm(N)
plot(x, y,
      xlab = "", ylab = "",
      main = paste(
        "The \"mtext\" function",
        paste(rep(" ", 60), collapse = ""))
      )
)
for (i in seq(from = 0, to = 1, by = 1)) {
  mtext(paste("Line", i), 3, line = i)
}

par
# 多图排列/分屏 page 47
# 最常用的是 par mflow mfcol 分别按行/列放置图形
op <- par(
  mflow = c(2, 2),
  oma = c(0, 0, 4, 0) # Outer margins
)
for (i in 1:4) {
  plot(runif(20), runif(20),
    main = paste("random plot (", i, ")"))
}
par(op)

```

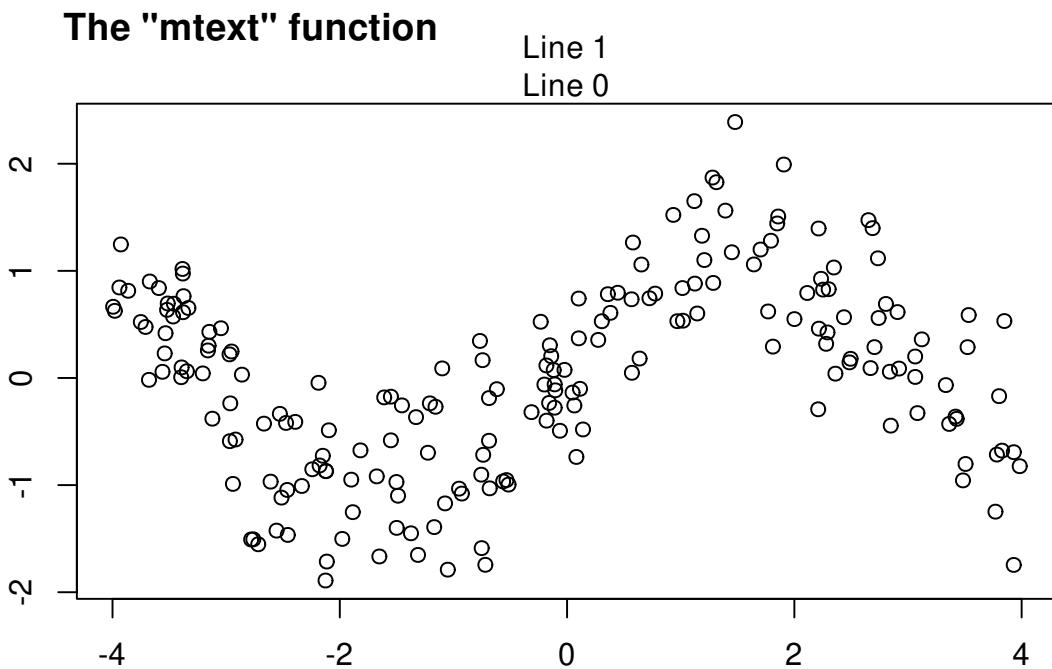


图 10.29: 外边空在图的边缘添加文字

```
mtext("Four plots, without enough room for this title",
  side = 3, font = 2, cex = 1.5, col = "red"
) # 总/大标题放不下
```

par 的 oma 用来设置外边空的大小，默认情形下没有外边空的

```
par()$oma
```

```
## [1] 0 0 0 0
```

我们可以自己设置外边空

```
op <- par(
  mfrow = c(2, 2),
  oma = c(0, 0, 3, 0) # Outer margins
)
for (i in 1:4) {
  plot(runif(20), runif(20),
    main = paste("random plot (", i, ")"),
    sep = ""))
}
par(op)
mtext("Four plots, with some room for this title",
  side = 3, line = 1.5, font = 1, cex = 1.5, col = "red")
```

## Four plots, without enough room for this title

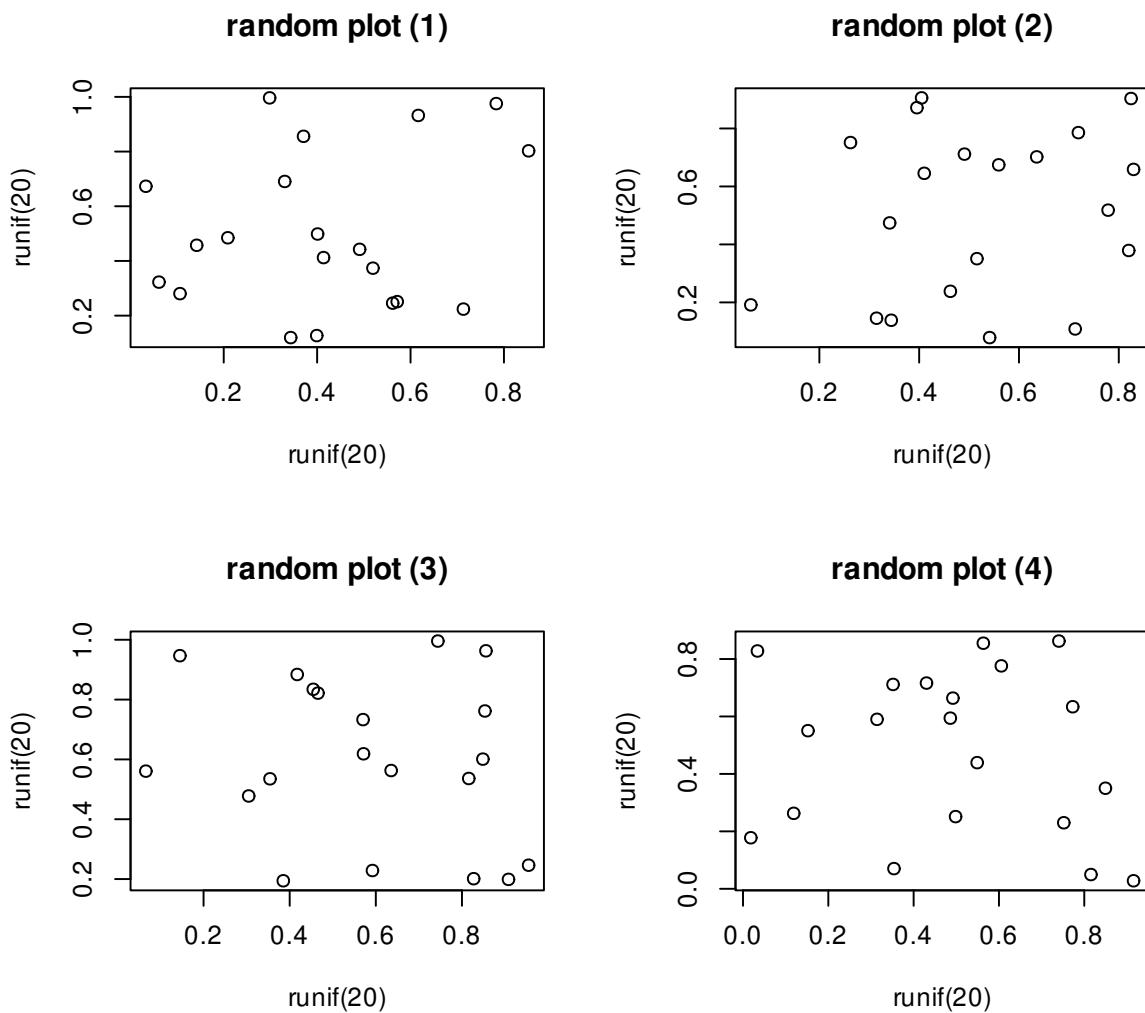


图 10.30: 多图排列共享一个大标题

```
)
```

## Four plots, with some room for this title

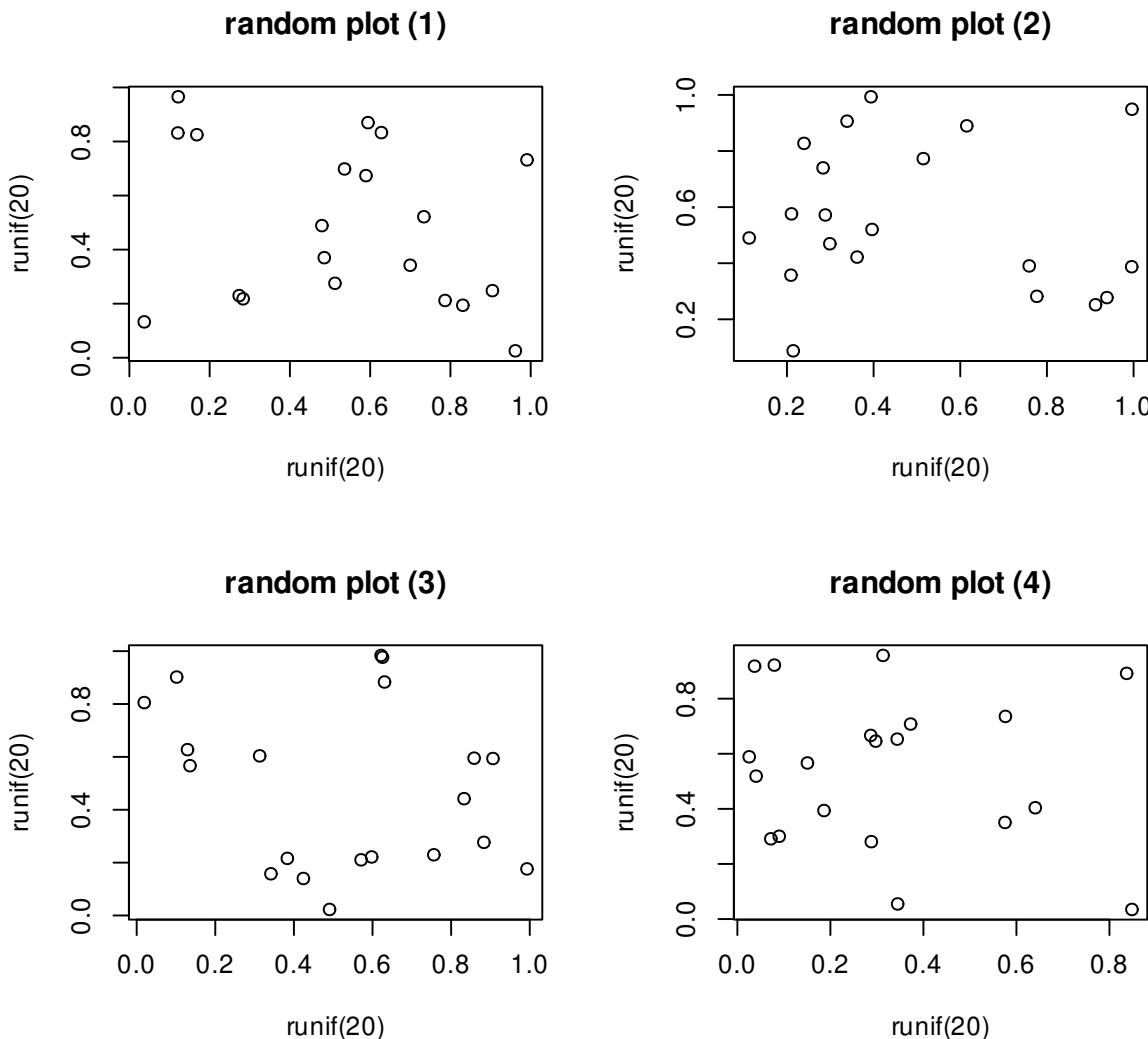


图 10.31: 设置外边空放置大标题

除了内边空还有外边空，内外边空用来放注释说明

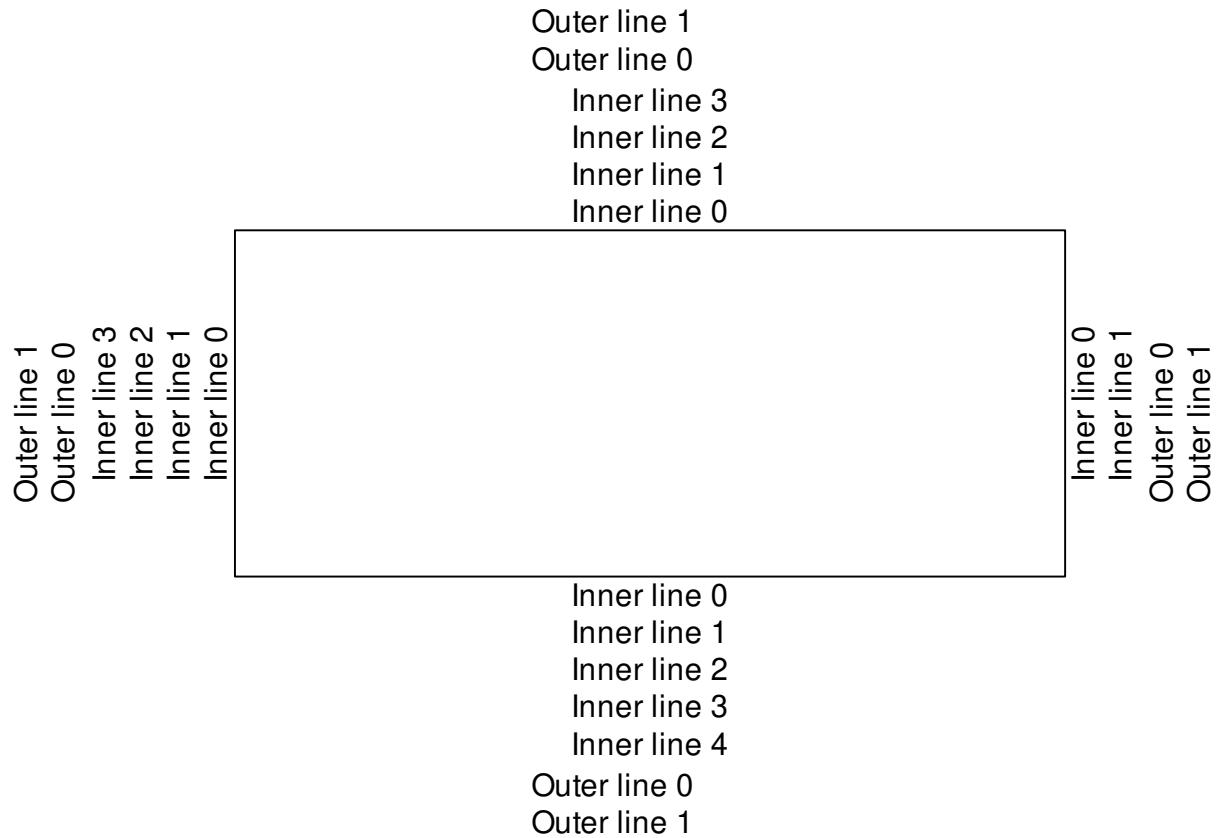
```
op <- par(no.readonly = TRUE)
par(oma = c(2, 2, 2, 2))
plot(1, 1, type = "n", xlab = "", ylab = "", xaxt = "n", yaxt = "n")
for (side in 1:4) {
  inner <- round(par()$mar[side], 0) - 1
  for (line in 0:inner) {
    mtext(text = paste0("Inner line ", line), side = side, line = line)
  }
  outer <- round(par()$oma[side], 0) - 1
  for (line in 0:outer) {
```

```

  mtext(text = paste0("Outer line ", line), side = side, line = line, outer = TRUE)
}

}

```

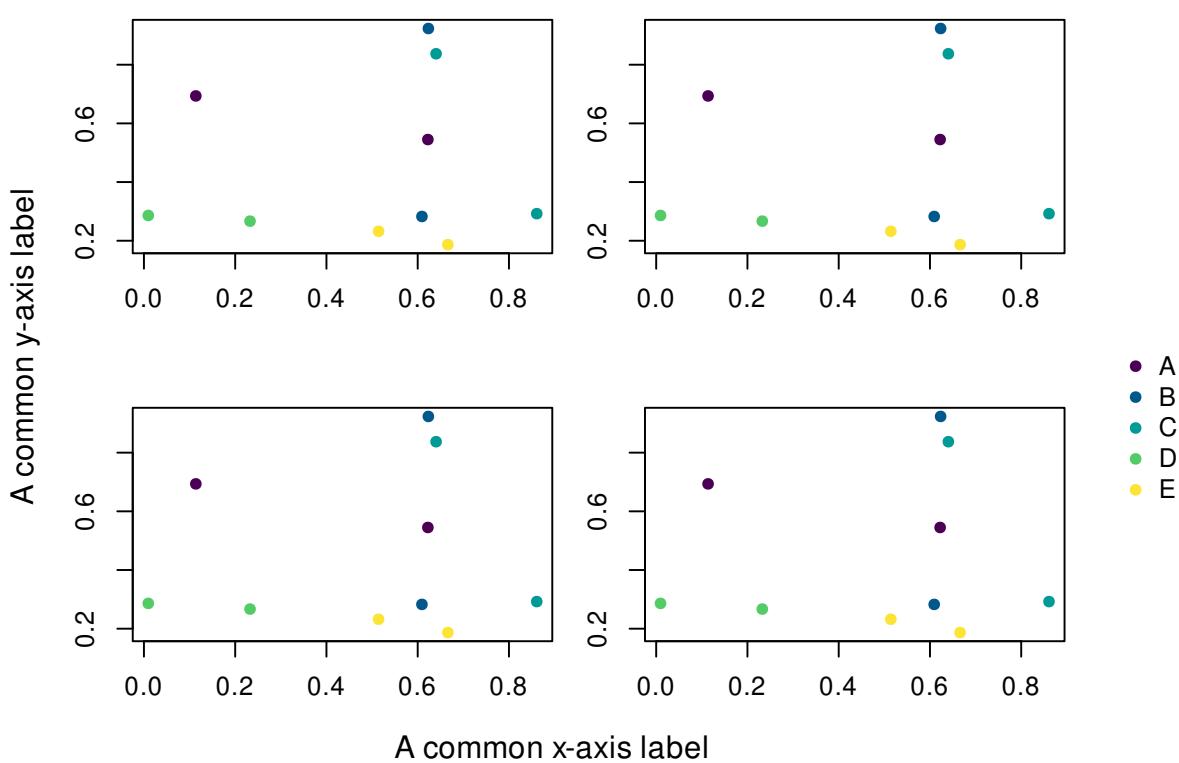


外边空可以用来放图例

```

set.seed(1234)
x <- runif(10)
y <- runif(10)
cols <- rep(hcl.colors(5), each = 2)
op <- par(oma = c(2, 2, 0, 4), mar = c(3, 3, 2, 0), mfrow = c(2, 2), pch = 16)
for (i in 1:4) {
  plot(x, y, col = cols, ylab = "", xlab = "")
}
mtext(text = "A common x-axis label", side = 1, line = 0, outer = TRUE)
mtext(text = "A common y-axis label", side = 2, line = 0, outer = TRUE)
legend(
  x = 1, y = 1.2, legend = LETTERS[1:5],
  col = unique(cols), pch = 16, bty = "n", xpd = NA
)

```

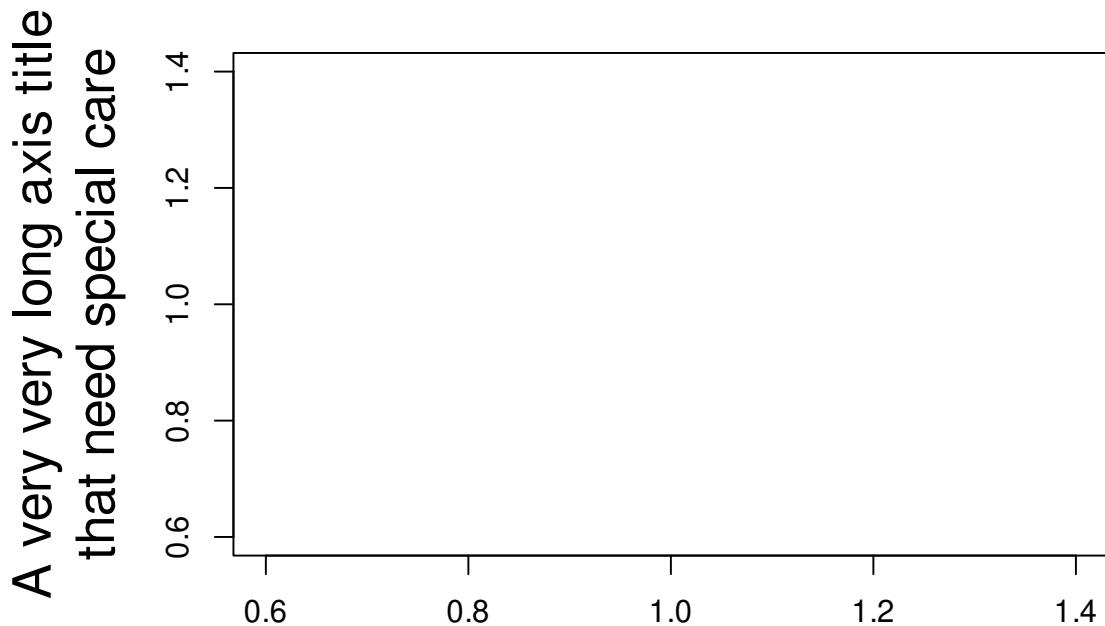


```
par(op)
```

坐标轴标签 xlab 和 ylab 的内容很长的时候需要内边空

```
par(cex.lab = 1.7)
plot(1, 1,
  ylab = "A very very long axis title\nthat need special care",
  xlab = "", type = "n"
)

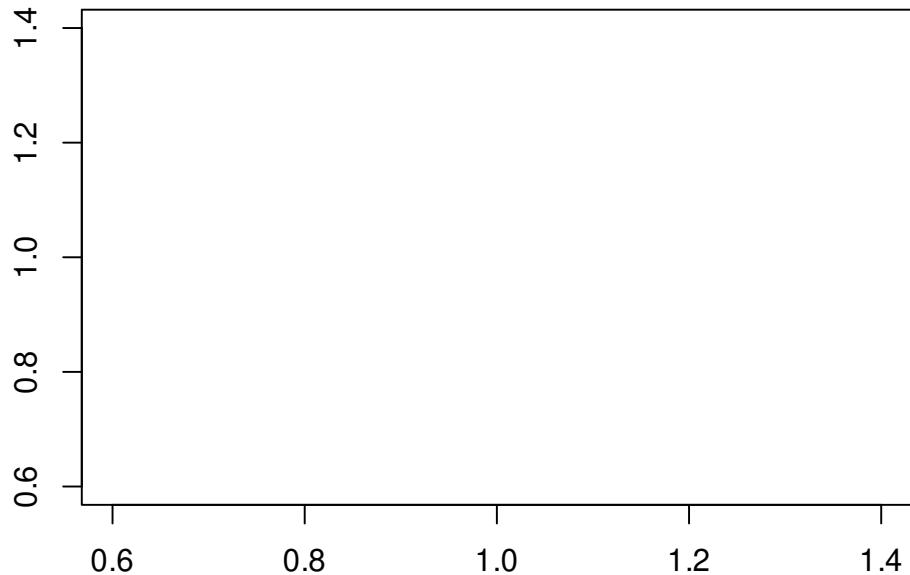
# 增加内边空的大小
par(mar = c(5, 7, 4, 2))
plot(1, 1,
  ylab = "A very very long axis title\nthat need special care",
  xlab = "", type = "n"
)
```



有时候，仅仅增加内边空还不够，坐标轴标签内容甚至可以出现在绘图区域外面，设置 `outer = TRUE`

```
par(oma = c(0, 4, 0, 0))
plot(1, 1, ylab = "", xlab = "", type = "n")
mtext(
  text = "A very very long axis title\nthat need special care",
  side = 2, line = 0, outer = TRUE, cex = 1.7
)
```

A very very long axis title  
that need special care



```
op <- par(  
  mfrow = c(2, 2),  
  oma = c(0, 0, 3, 0),  
  mar = c(3, 3, 4, 1) + .1 # Margins  
)  
for (i in 1:4) {  
  plot(runif(20), runif(20),  
    xlab = "", ylab = "",  
    main = paste("random plot (", i, ")", sep = ""))  
}  
par(op)  
mtext("Title",  
  side = 3, line = 1.5, font = 2, cex = 2, col = "red")  
)
```

### 10.1.10 图层

覆盖图形 add = T or par(new=TRUE)

```
plot(runif(5), runif(5),  
  xlim = c(0, 1), ylim = c(0, 1))  
points(runif(5), runif(5),
```

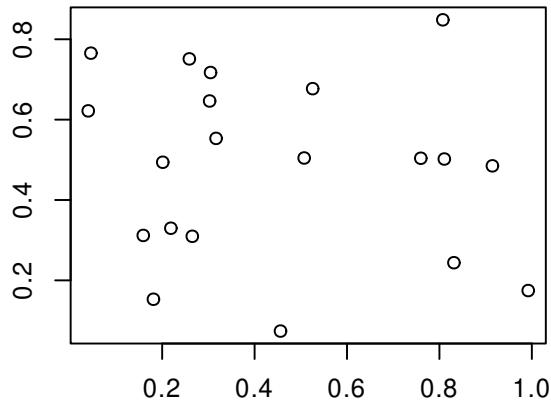
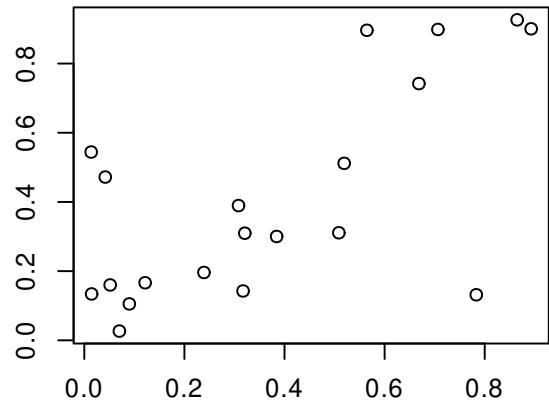
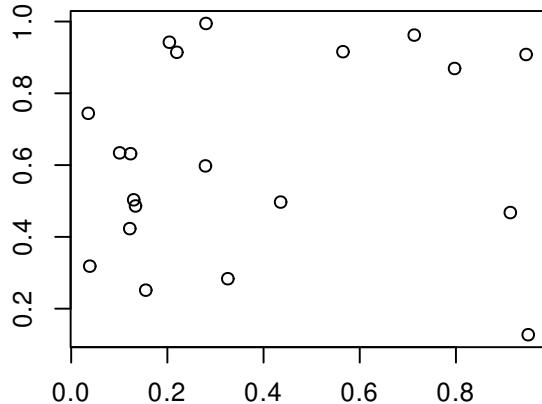
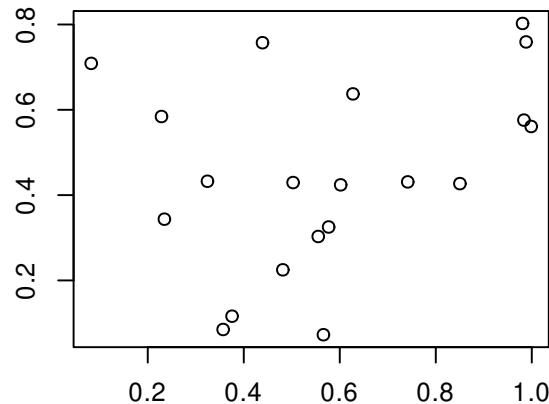
**Title****random plot (1)****random plot (2)****random plot (3)****random plot (4)**

图 10.32: 设置每个子图的边空 mar

```
  col = "#EA4335", pch = 16, cex = 3
)
lines(runif(5), runif(5), col = "red")
segments(runif(5), runif(5), runif(5), runif(5),
  col = "blue"
)
title(main = "Overlaying points, segments, lines...")
```

Overlaying points, segments, lines...

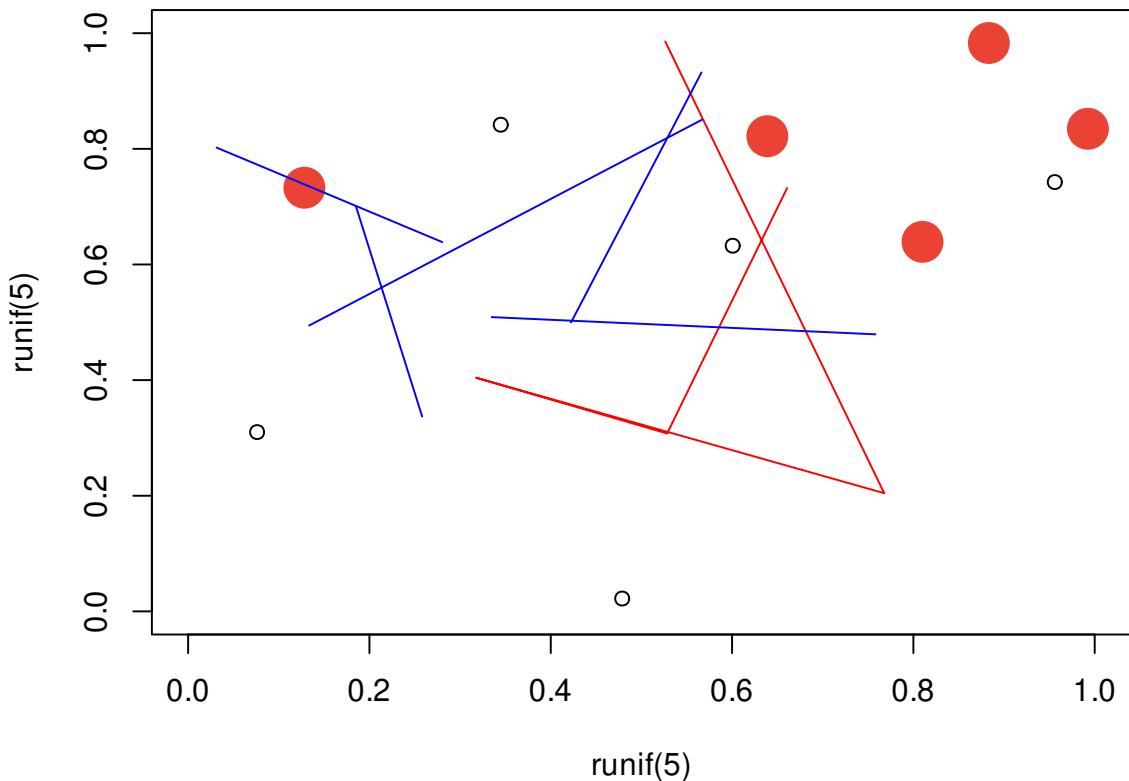


图 10.33: 添加图层

### 10.1.11 布局

layout 函数布局, 绘制复杂组合图形

```
op <- par(oma = c(0, 0, 3, 0))
layout(matrix(c(
  1, 1, 1,
  2, 3, 4,
  2, 3, 4
), nr = 3, byrow = TRUE))
hist(rnorm(n), col = "light blue")
```

```
hist(rnorm(n), col = "light blue")
hist(rnorm(n), col = "light blue")
hist(rnorm(n), col = "light blue")
mtext("The \"layout\" function",
      side = 3, outer = TRUE,
      font = 2, cex = 1.2
    )
```

### The "layout" function

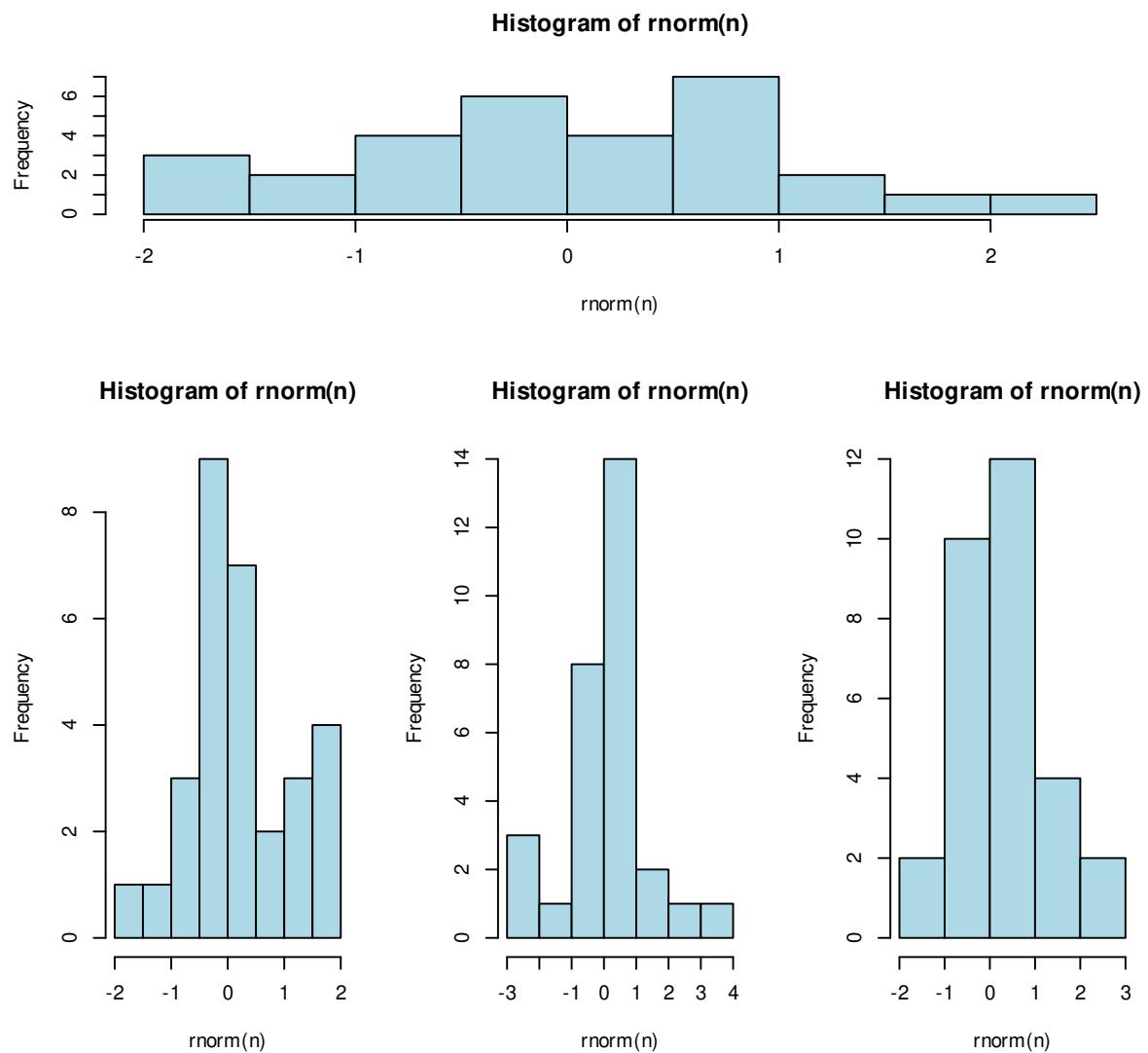


图 10.34: 更加复杂的组合图形

#### 10.1.12 组合

`par` 之 `fig` 参数很神奇, 使得多个图可以叠加在一起, 它接受一个数值向量 `c(x1, x2, y1, y2)`, 是图形设备显示区域中的绘图区域的 (NDC, normalized device coordinates) 坐标。

```
plot(1:12,
  type = "b", main = "'fg' : axes, ticks and box in gray",
  fg = gray(0.7), bty = "7", sub = R.version.string
)
par(fig = c(1, 6, 5, 10) / 10, new = T)
plot(6:10,
  type = "b", main = "",
  fg = gray(0.7), bty = "7", xlab = R.version.string
)
```

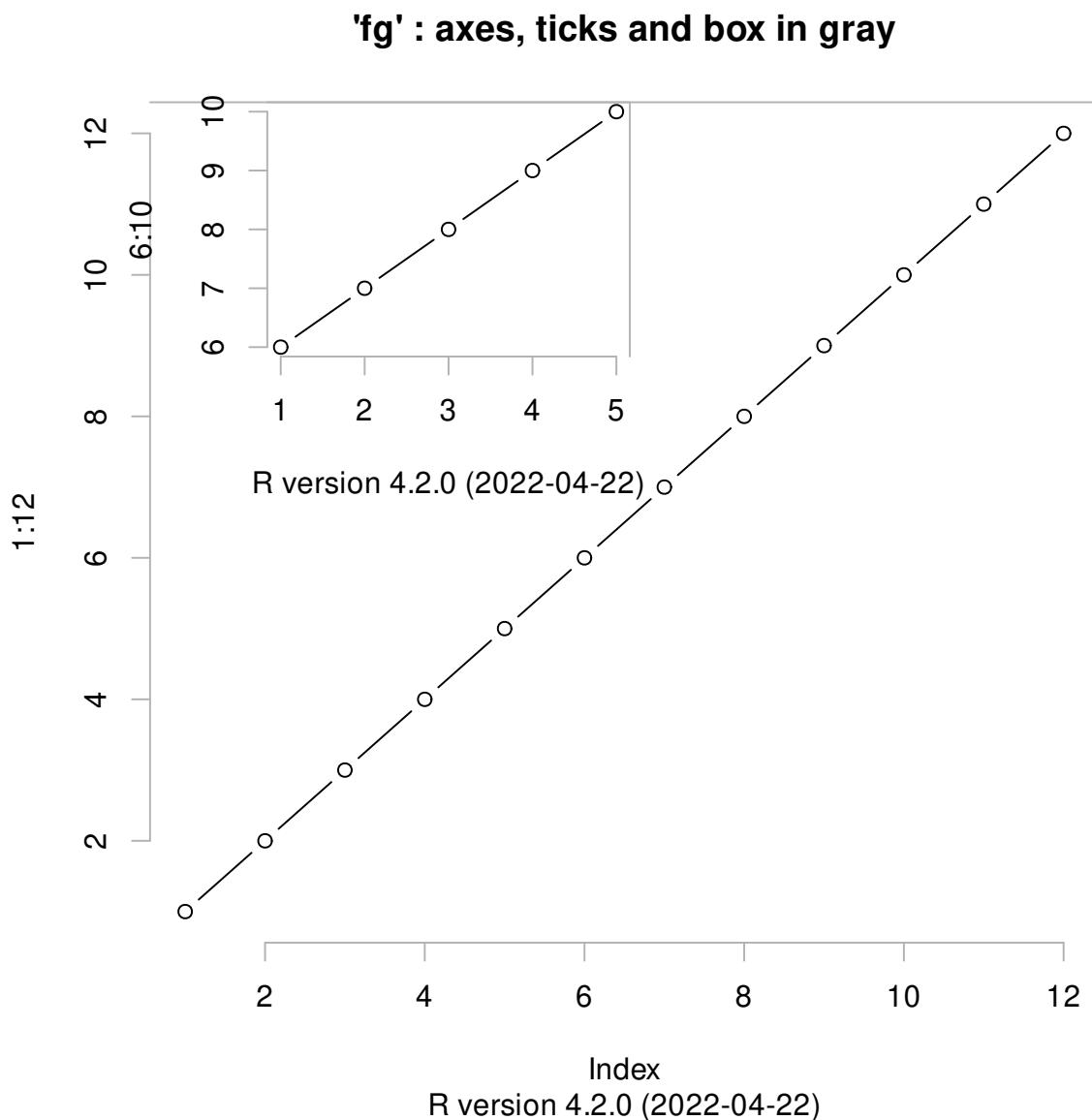


图 10.35: 多图叠加

`fig`参数控制图形的位置，用来绘制组合图形

```
n <- 1000
x <- rt(n, df = 10)
hist(x,
  col = "light blue",
  probability = TRUE, main = "",
  ylim = c(0, 1.2 * max(density(x)$y)))
)
lines(density(x),
  col = "red",
  lwd = 3
)
op <- par(
  fig = c(.02, .4, .5, .98),
  new = TRUE
)
qqnorm(x,
  xlab = "", ylab = "", main = "",
  axes = FALSE
)
qqline(x, col = "red", lwd = 2)
box(lwd = 2)
```

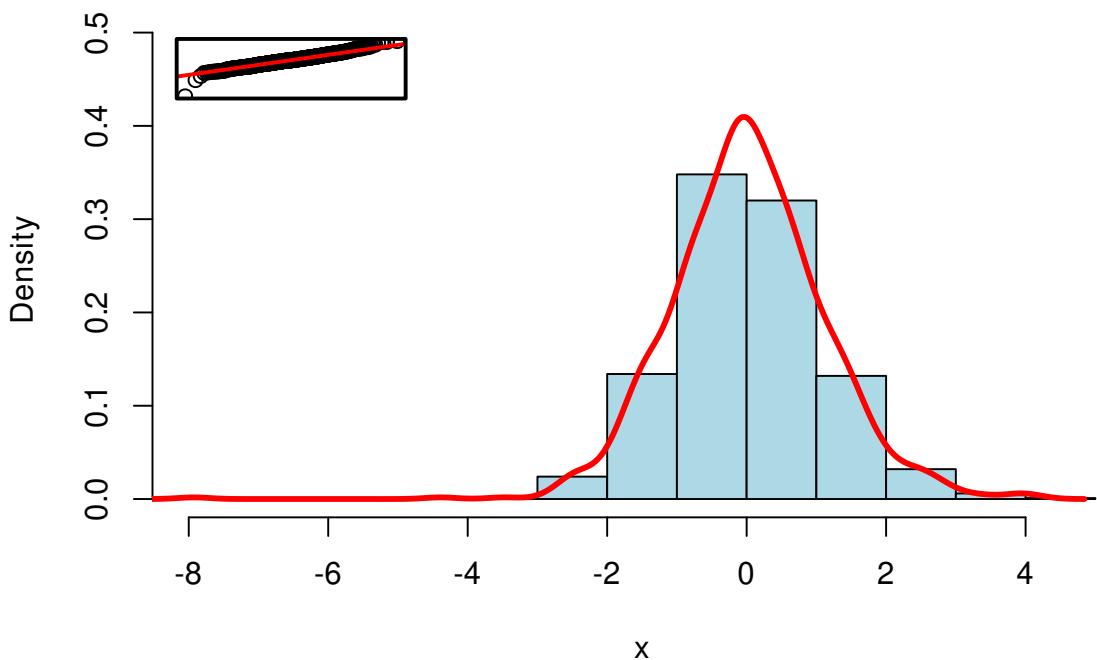


图 10.36: 组合图形



```
par(op)
```

### 10.1.13 分屏

split.screen 分屏组合

```
random.plot <- function() {  
  N <- 200  
  f <- sample(  
    list(  
      rnorm,  
      function(x) {  
        rt(x, df = 2)  
      },  
      rlnorm,  
      runif  
    ),  
    1  
  ) [[1]]  
  x <- f(N)  
  hist(x, col = "lightblue", main = "", xlab = "", ylab = "", axes = F)  
  axis(1)  
}  
op <- par(bg = "white", mar = c(2.5, 2, 1, 2))  
split.screen(c(2, 1))
```

```
## [1] 1 2
```

```
split.screen(c(1, 3), screen = 2)
```

```
## [1] 3 4 5
```

```
screen(1)  
random.plot()  
# screen(2); random.plot() # Screen 2 was split into three screens: 3, 4, 5  
screen(3)  
random.plot()  
screen(4)  
random.plot()  
screen(5)  
random.plot()  
  
close.screen(all = TRUE)  
par(op)
```

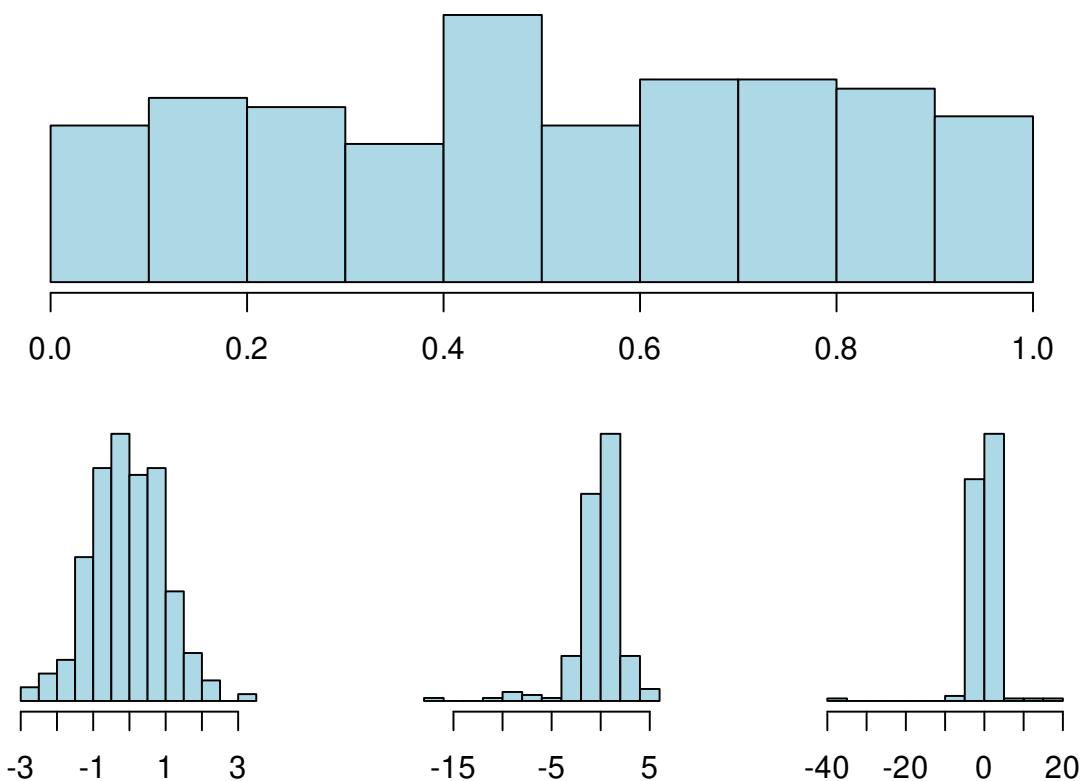


图 10.37: 分屏

### 10.1.14 交互

辅助绘图 identify locator

## 10.2 基础统计图形

按图的类型划分，最后在小结部分给出各图适用的数据类型

根据数据类型划分：对于一元数据，可用什么图来描述；多元数据呢，连续数据和离散数据（分类数据）

先找一个不重不漏的划分，指导原则是根据数据类型选择图，根据探索到的数据中的规律，选择图

其它 assocplot fourfoldplot sunflowerplot

### 10.2.1 条形图

#### 条形图

简单条形图

```
data(diamonds, package = "ggplot2") # 加载数据
par(mar = c(2, 5, 1, 1))
barCenters <- barplot(table(diamonds$cut),
  col = "lightblue", axes = FALSE,
```



```
  axisnames = FALSE, horiz = TRUE, border = "white"
)
text(
  y = barCenters, x = par("usr")[3],
  adj = 1, labels = names(table(diamonds$cut)), xpd = TRUE
)
axis(1,
  labels = seq(0, 25000, by = 5000), at = seq(0, 25000, by = 5000),
  las = 1, col = "gray"
)
grid()
```

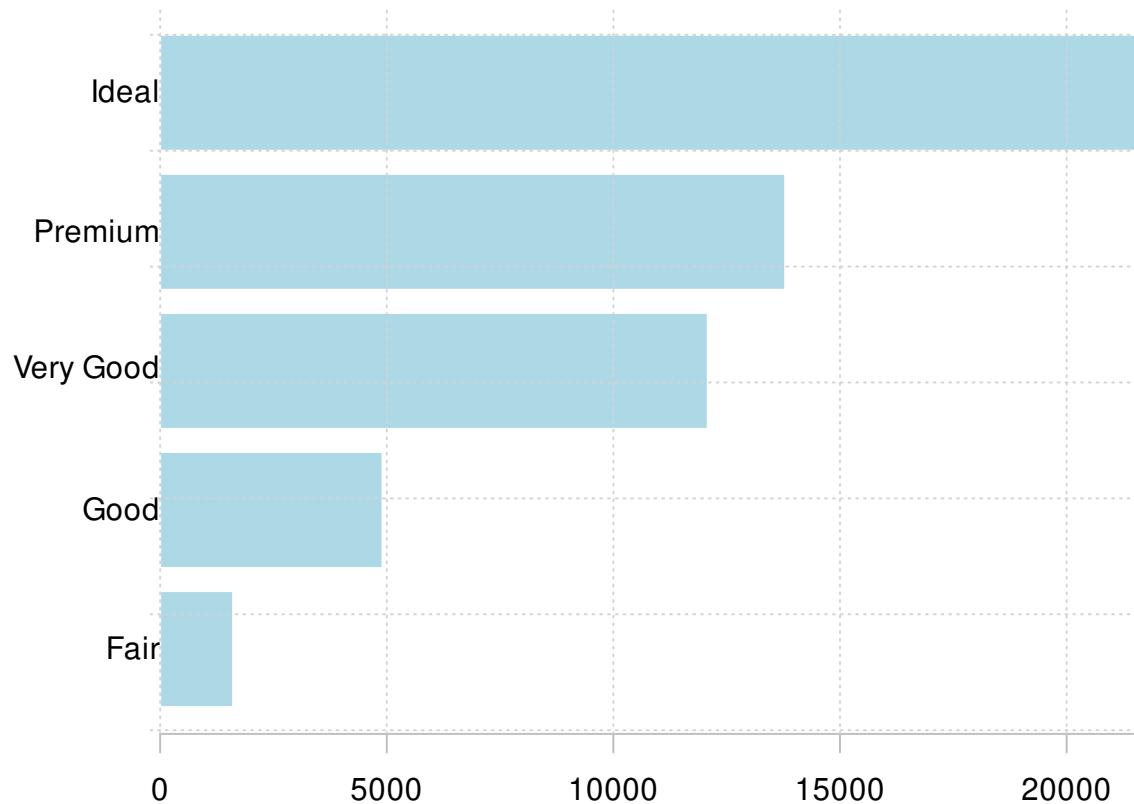


图 10.38: 条形图

### 简单柱形图

```
set.seed(123456)
barPois <- table(stats::rpois(1000, lambda = 5))
plot(barPois, col = "lightblue", type = "h", lwd = 10, main = "")
box(col = "gray")
```

### 复合条形图

```
par(mar = c(4.1, 2.1, 0.5, 4.5))
barplot(VADeaths,
  border = "white", horiz = FALSE, col = hcl.colors(5),
```

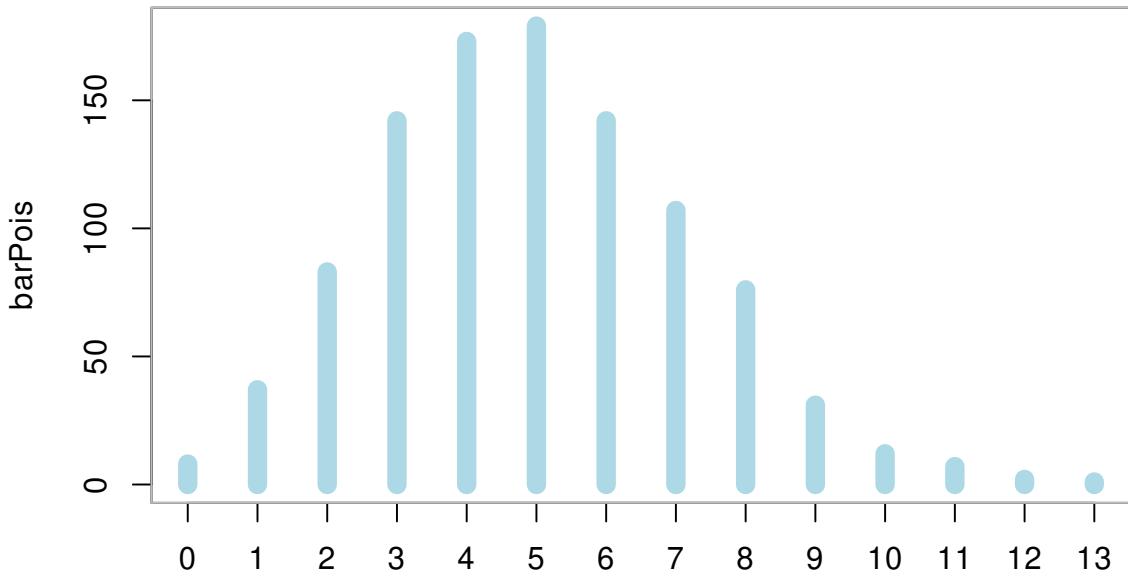


图 10.39: 柱形图

```
legend.text = rownames(VADeaths), xpd = TRUE, beside = TRUE,
cex.names = 0.9,
args.legend = list(
  x = "right", border = "white", title = "Age",
  box.col = NA, horiz = FALSE, inset = c(-.2, 0),
  xpd = TRUE
),
panel.first = grid(nx = 0, ny = 7)
)
```

堆积条形图

```
par(mar = c(4.1, 2.1, 0.5, 4.5))
barplot(VADeaths,
border = "white", horiz = FALSE, col = hcl.colors(5),
legend.text = rownames(VADeaths), xpd = TRUE, beside = FALSE,
cex.names = 0.9,
args.legend = list(
  x = "right", border = "white", title = "Age",
  box.col = NA, horiz = FALSE, inset = c(-.2, 0),
  xpd = TRUE
),
panel.first = grid(nx = 0, ny = 4)
```

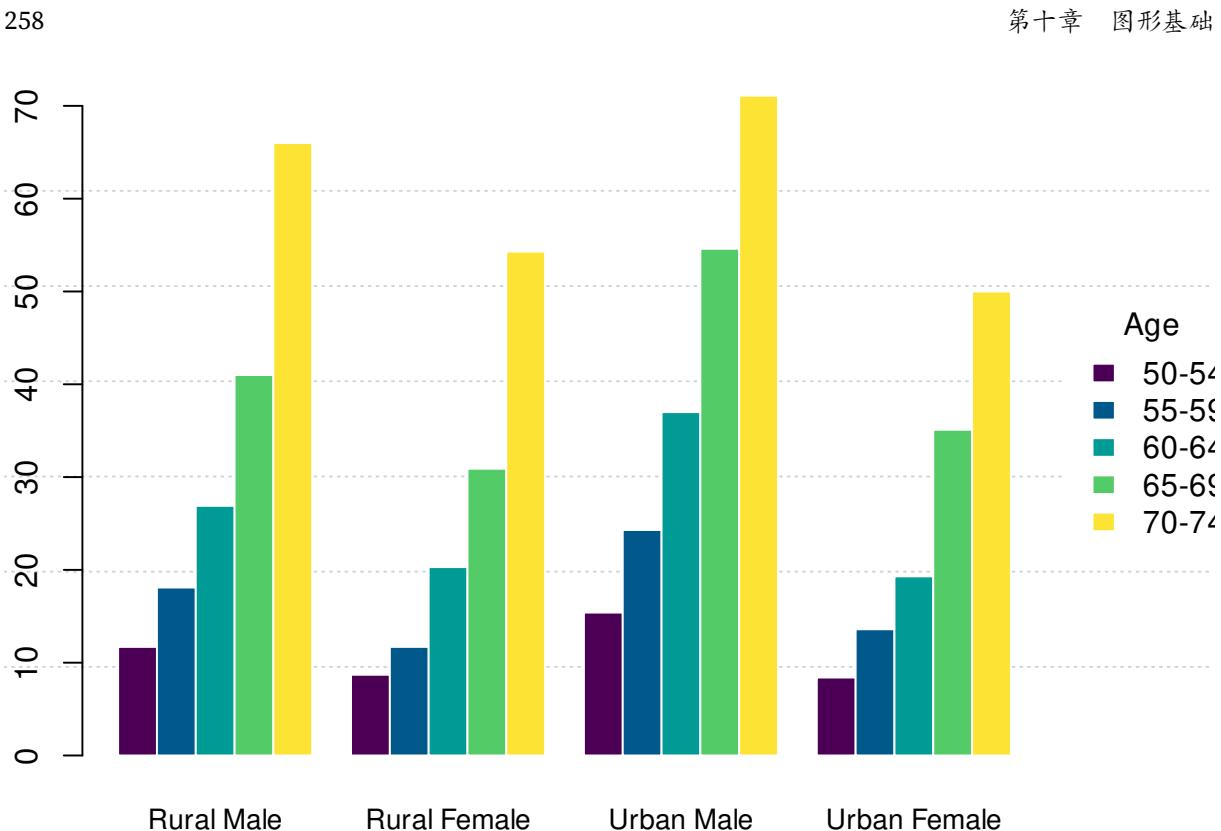


图 10.40: 复合条形图

- 堆积条形图 spineplot

### 简单条形图

```
barplot(  
  data = BOD, demand ~ Time, ylim = c(0, 20),  
  border = "white", horiz = FALSE, col = hcl.colors(1)  
)
```

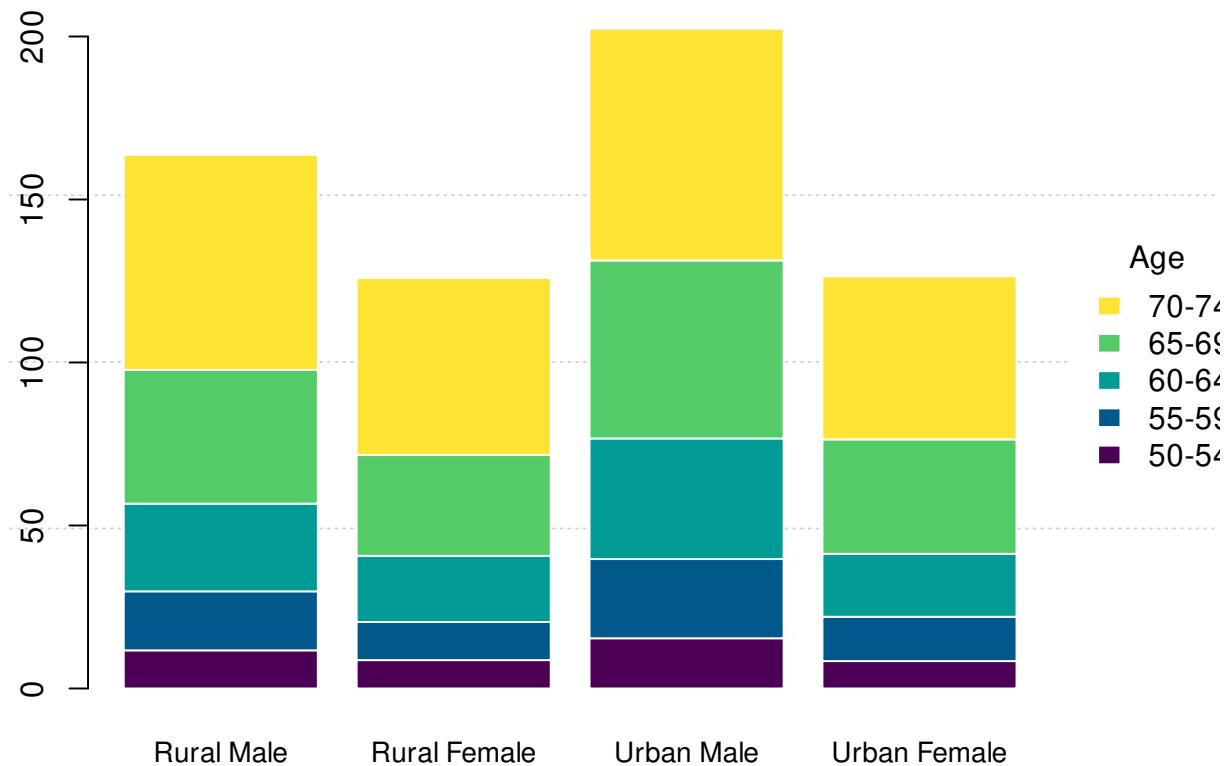
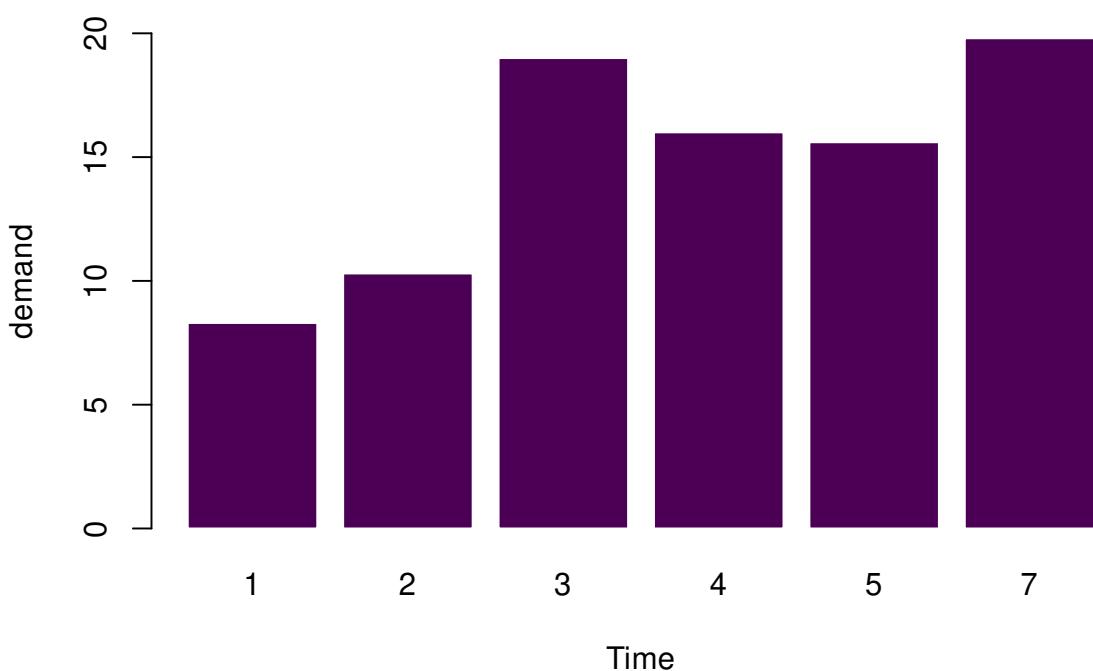
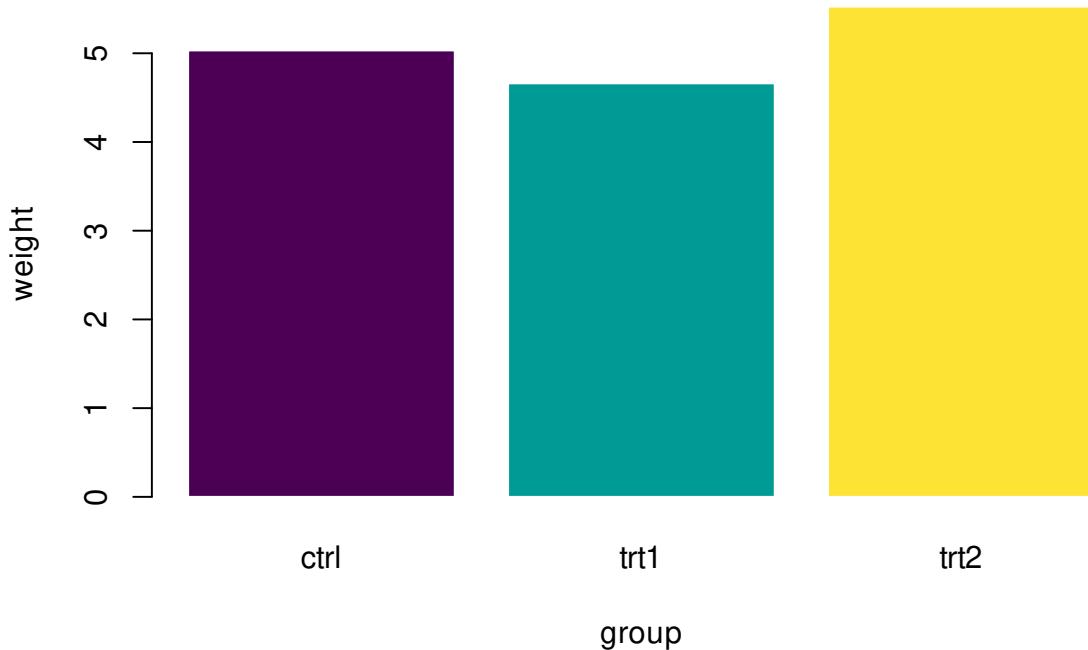


图 10.41: 堆积条形图



```
pg_mean <- aggregate(weight ~ group, data = PlantGrowth, mean)
barplot(
  data = pg_mean, weight ~ group,
  border = "white", horiz = FALSE, col = hcl.colors(3)
)
```

C

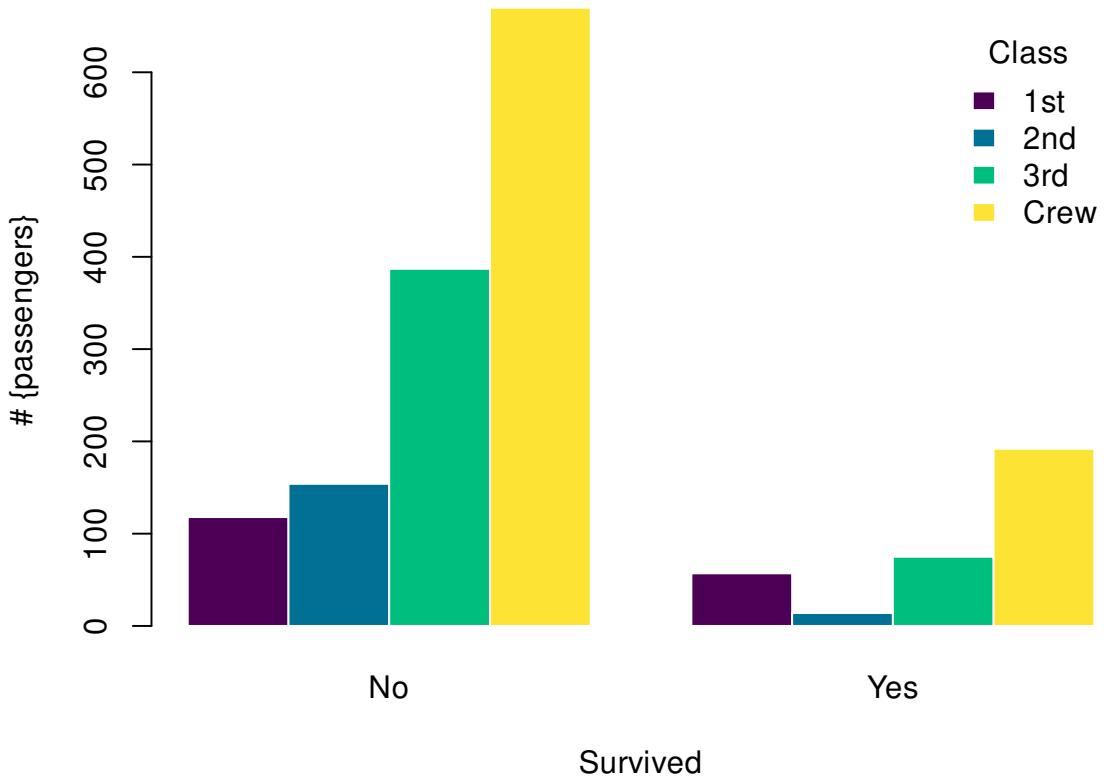


Titanic 数据集是 table 数据类型

简单条形图

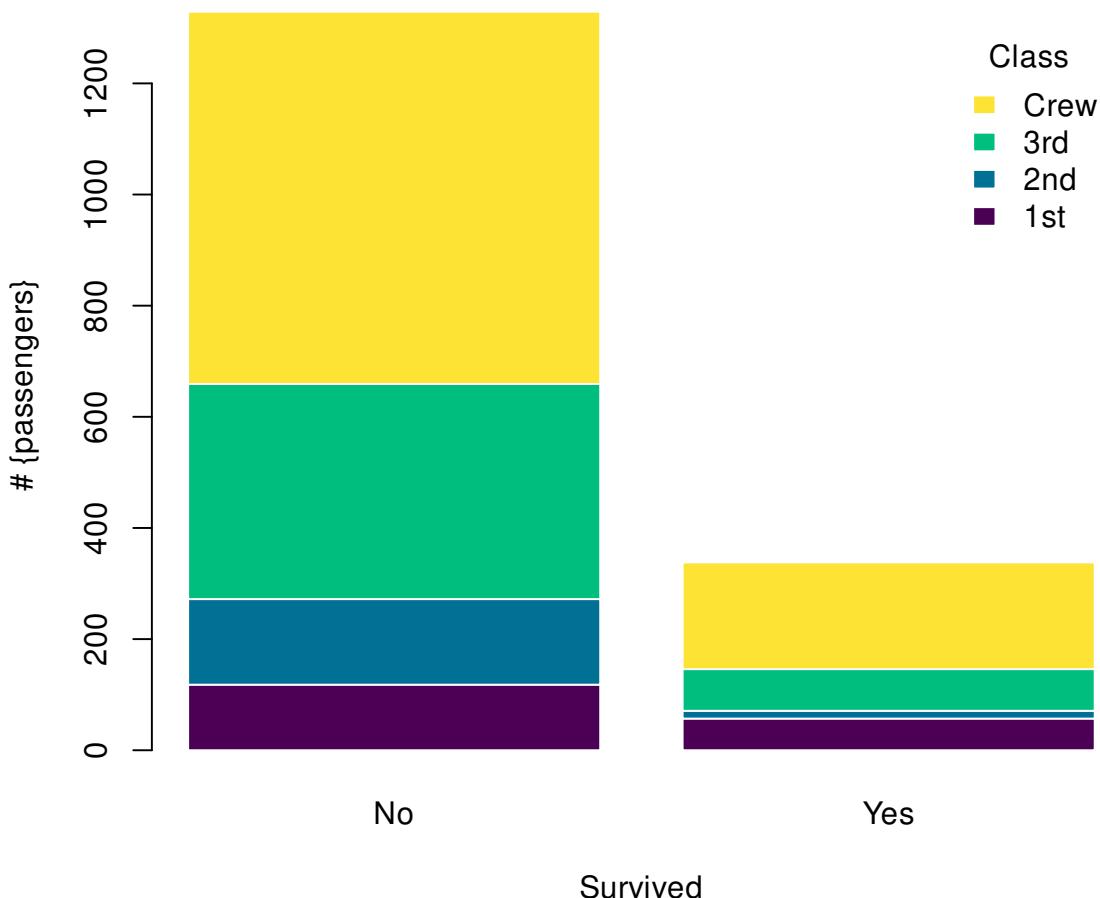
复合条形图

```
barplot(Freq ~ Class + Survived,
  data = Titanic,
  subset = Age == "Adult" & Sex == "Male",
  beside = TRUE,
  border = "white", horiz = FALSE, col = hcl.colors(4),
  args.legend = list(
    border = "white", title = "Class",
    box.col = NA, horiz = FALSE,
    xpd = TRUE
  ),
  ylab = "# {passengers}", legend = TRUE
)
```



堆积条形图

```
barplot(Freq ~ Class + Survived,
  data = Titanic,
  subset = Age == "Adult" & Sex == "Male",
  border = "white", horiz = FALSE, col = hcl.colors(4),
  args.legend = list(
    border = "white", title = "Class",
    box.col = NA, horiz = FALSE,
    xpd = TRUE
  ),
  ylab = "# {passengers}", legend = TRUE
)
```



### 10.2.2 直方图

```
set.seed(1234)
n <- 2^24
x <- runif(n, 0, 1)
delta <- 0.01
len <- diff(c(0, which(x < delta), n + 1)) - 1
ylim <- seq(0, 1800, by = 300)
xlim <- seq(0, 100, by = 20)
p <- hist(len[len < 101], breaks = -1:100 + 0.5, plot = FALSE)
plot(p, ann = FALSE, axes = FALSE, col = "lightblue", border = "white", main = "")
axis(1, labels = xlim, at = xlim, las = 1) # x 轴
axis(2, labels = ylim, at = ylim, las = 0) # y 轴
box(col = "gray")
with(faithful, plot(eruptions ~ waiting, pch = 16))
```

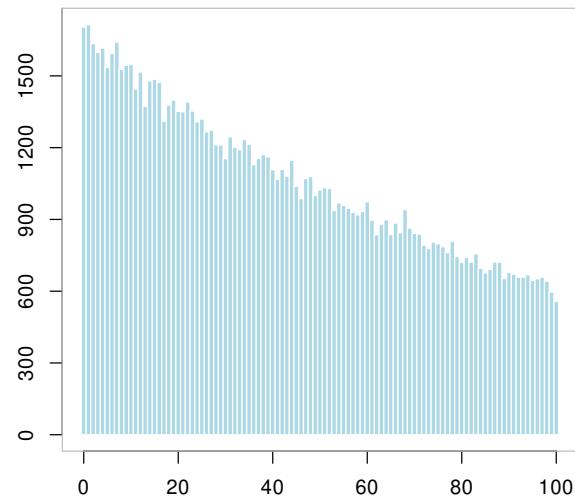
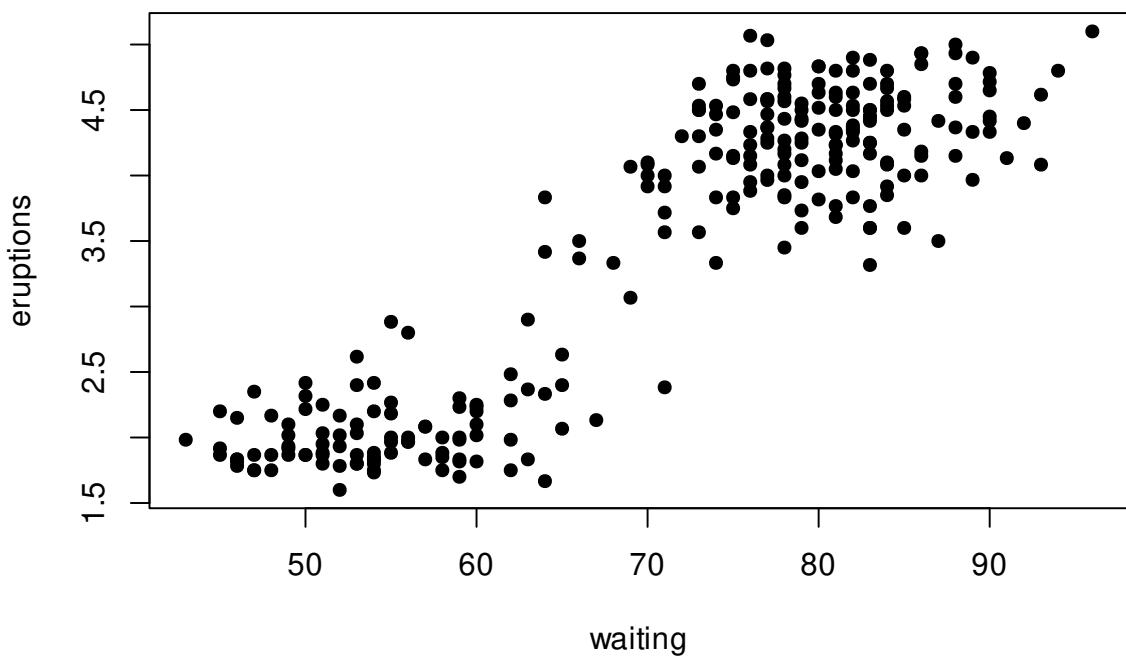
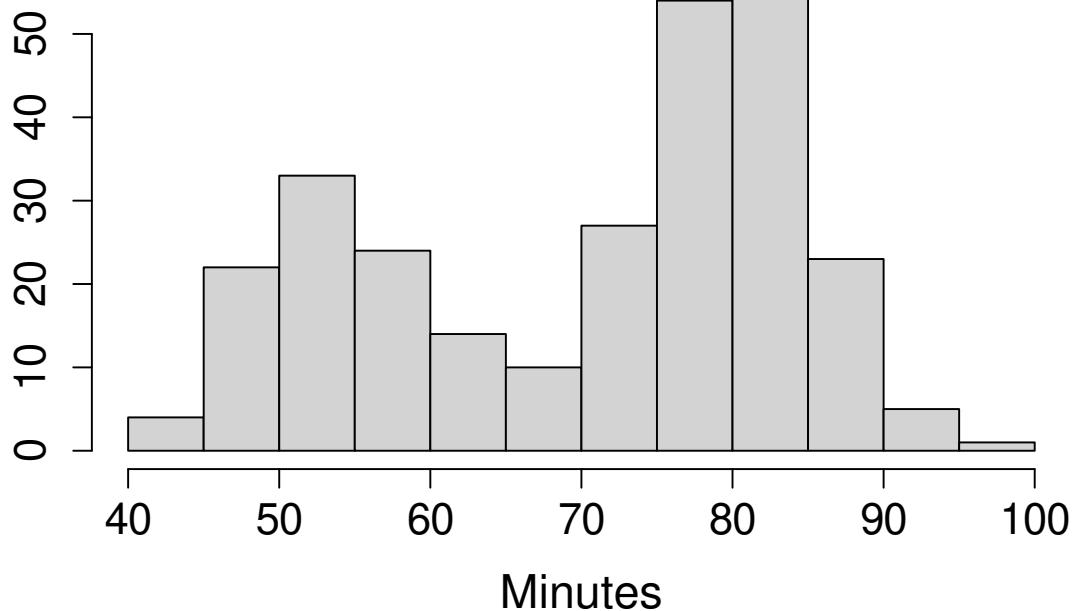


图 10.42: 直方图



```
with(faithful, hist(waiting,
  main = "Time between Old Faithful eruptions",
  xlab = "Minutes", ylab = "",
  cex.main = 1.5, cex.lab = 1.5, cex.axis = 1.4
))
```

## Time between Old Faithful eruptions



```
with(data = faithful, {  
  hist(eruptions, seq(1.6, 5.2, 0.2),  
    prob = TRUE,  
    main = "", col = "lightblue", border = "white"  
  )  
  lines(density(eruptions, bw = 0.1), col = "#EA4335")  
  rug(eruptions, col = "#EA4335") # 添加数据点  
})
```

```
hist(longley$Unemployed,  
  probability = TRUE,  
  col = "light blue", main = "")  
#  
# 添加密度估计  
lines(density(longley$Unemployed),  
  col = "red",  
  lwd = 3  
)
```

直方图有很多花样的，添加阴影线，angle 控制倾斜的角度

```
# hist(longley$Unemployed, density = 1, angle = 45)  
# hist(longley$Unemployed, density = 3, angle = 15)  
# hist(longley$Unemployed, density = 1, angle = 15)  
hist(longley$Unemployed, density = 3, angle = 45, main = "")
```

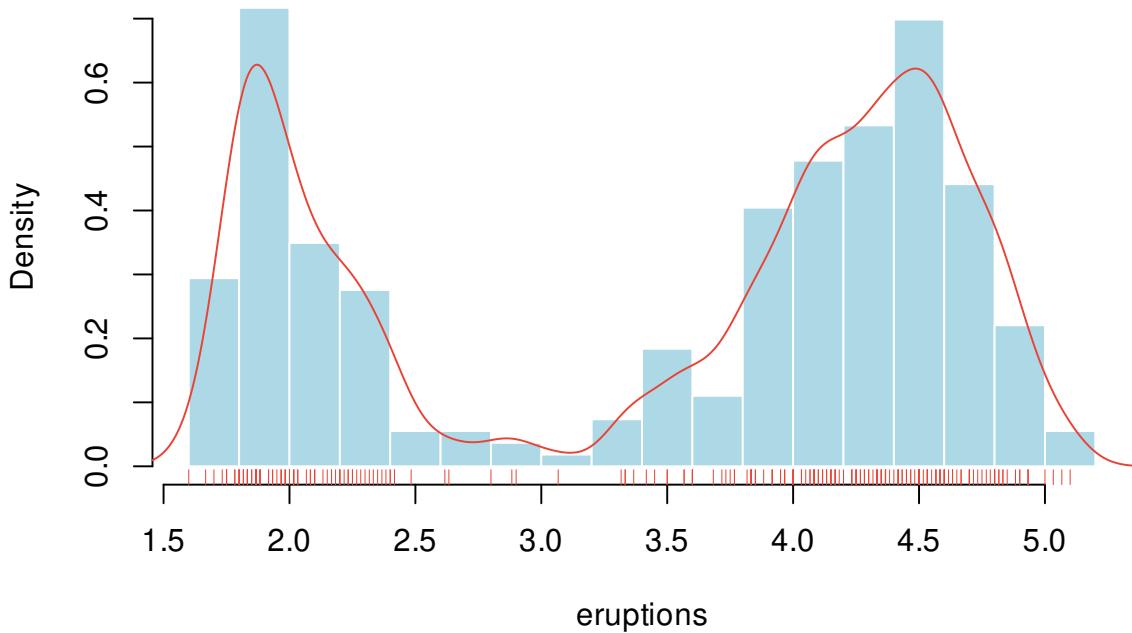


图 10.43: 老忠实泉间歇性喷水的时间间隔分布

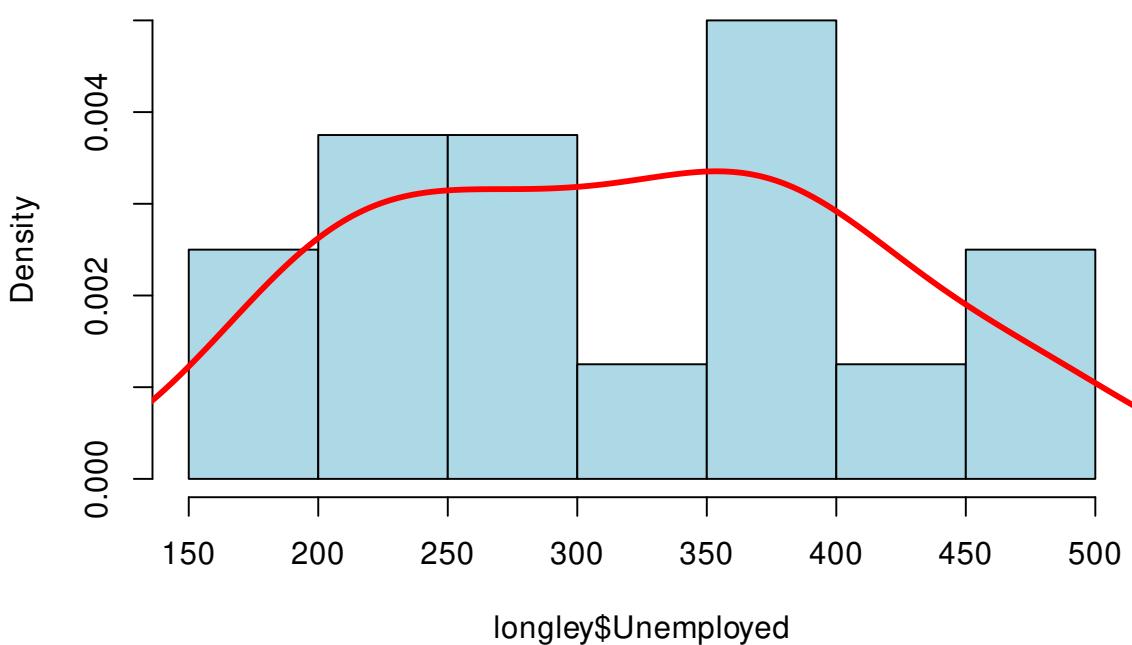


图 10.44: 概率密度分布

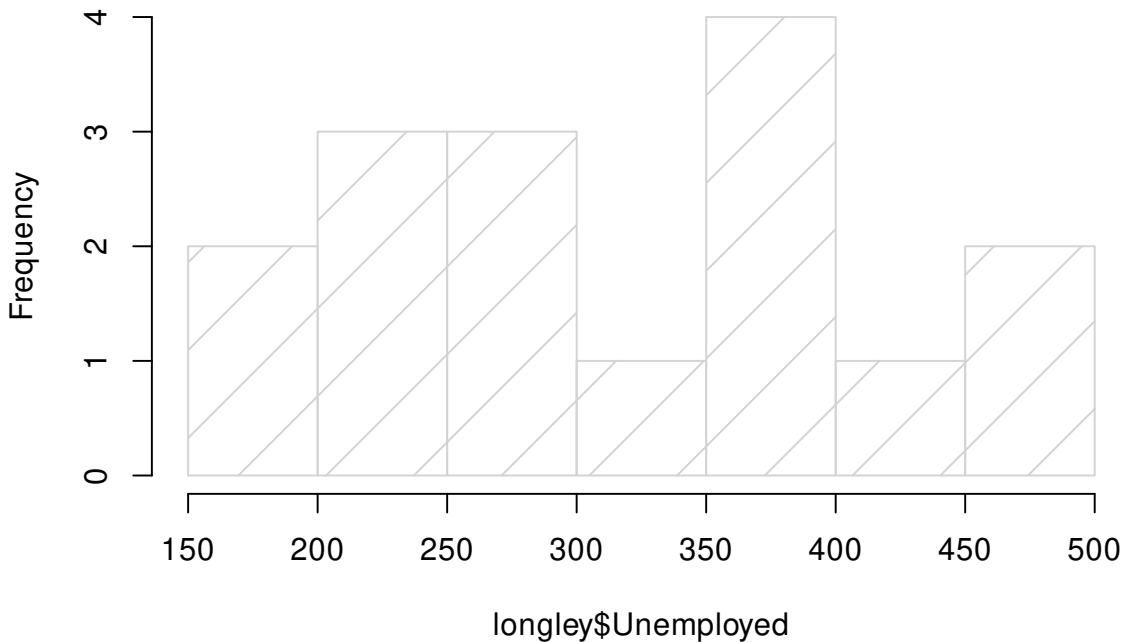
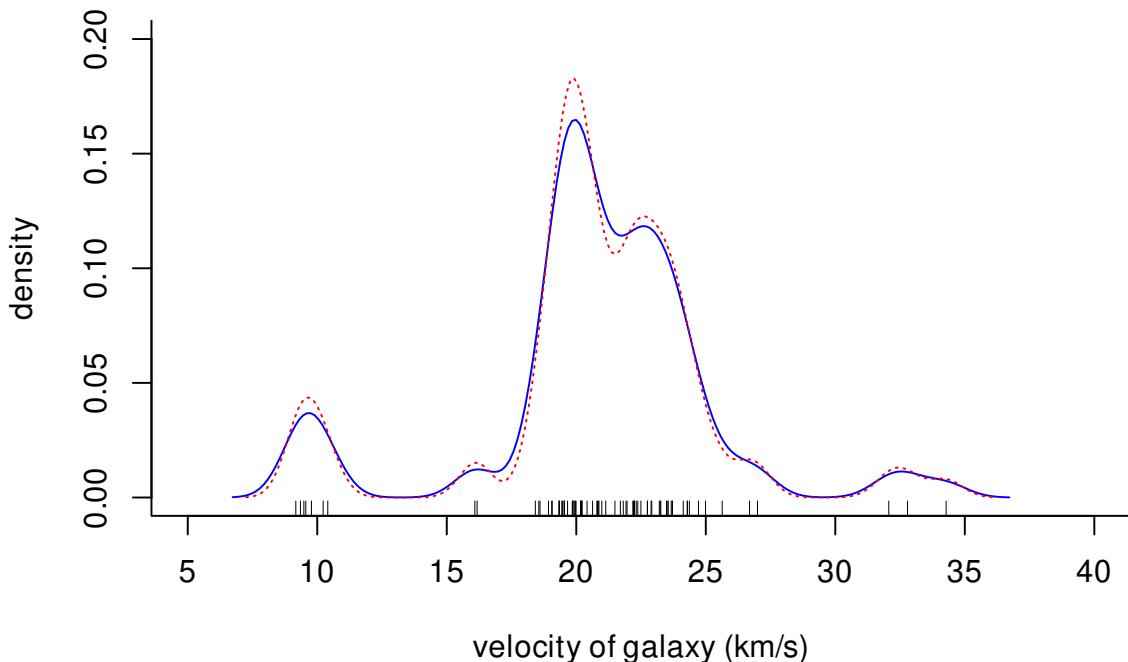


图 10.45: density 数值越大阴影线越密

### 10.2.3 密度图

```
data(galaxies, package = "MASS")
galaxies <- galaxies / 1000
# Bandwidth Selection by Pilot Estimation of Derivatives
c(MASS::width.SJ(galaxies, method = "dpi"), MASS::width.SJ(galaxies))
```

```
## [1] 3.256151 2.566423
plot(
  x = c(5, 40), y = c(0, 0.2), type = "n", bty = "l",
  xlab = "velocity of galaxy (km/s)", ylab = "density"
)
rug(galaxies)
lines(density(galaxies, width = 3.25, n = 200), col = "blue", lty = 1)
lines(density(galaxies, width = 2.56, n = 200), col = "red", lty = 3)
```



```
x <- seq(from = 100, to = 174, by = 0.5)
y1 <- dnorm(x, mean = 145, sd = 9)
y2 <- dnorm(x, mean = 128, sd = 8)
plot(x, y1,
  type = "l", lwd = 2, col = "firebrick3",
  main = "Systolic Blood Pressure Before and After Treatment",
  xlab = "Systolic Blood Pressure (mmHg)",
  ylab = "Frequency", yaxt = "n",
  xlim = c(100, 175), ylim = c(0, 0.05)
)

lines(x, y2, col = "dodgerblue4")
polygon(c(117, x, 175), c(0, y2, 0),
  col = "dodgerblue4",
  border = "white"
)

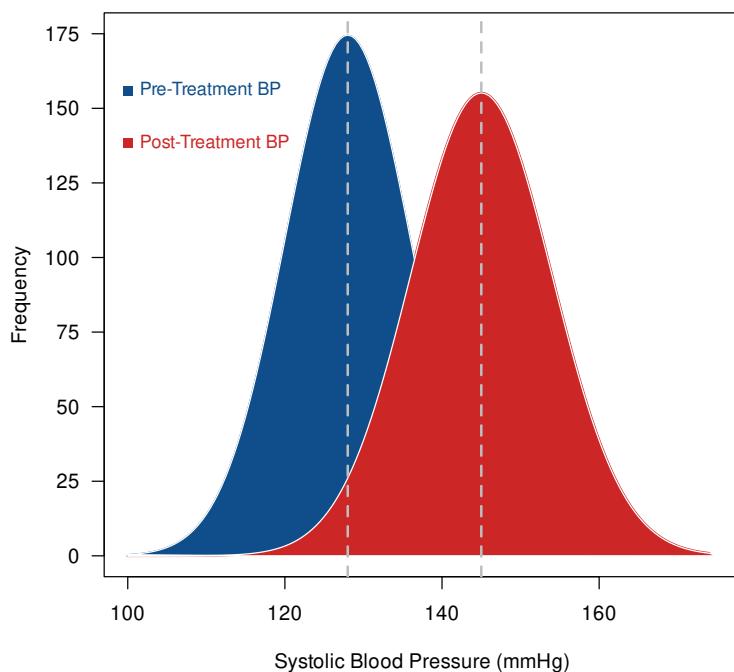
polygon(c(100, x, 175), c(0, y1, 0),
  col = "firebrick3",
  border = "white"
)

axis(2,
  at = seq(from = 0, to = 0.05, length.out = 8),
```

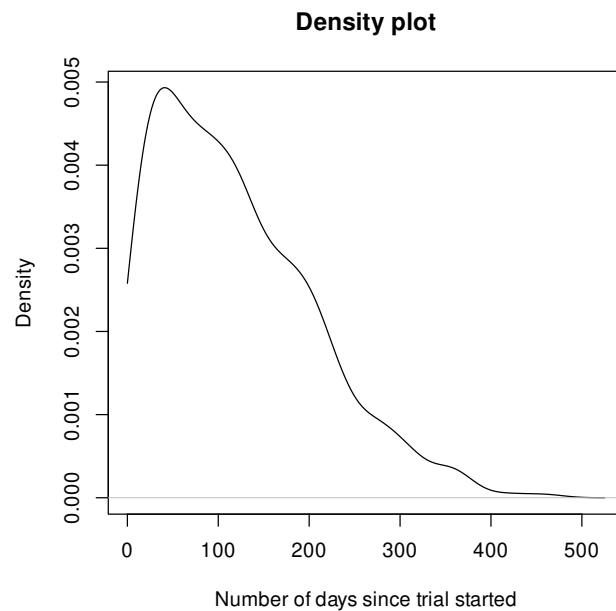
```
  labels = seq(from = 0, to = 175, by = 25), las = 1
)

text(x = 100, y = 0.0445, "Pre-Treatment BP", col = "dodgerblue4", cex = 0.9, pos = 4)
text(x = 100, y = 0.0395, "Post-Treatment BP", col = "firebrick3", cex = 0.9, pos = 4)
points(100, 0.0445, pch = 15, col = "dodgerblue4")
points(100, 0.0395, pch = 15, col = "firebrick3")
abline(v = c(145, 128), lwd = 2, lty = 2, col = 'gray')
```

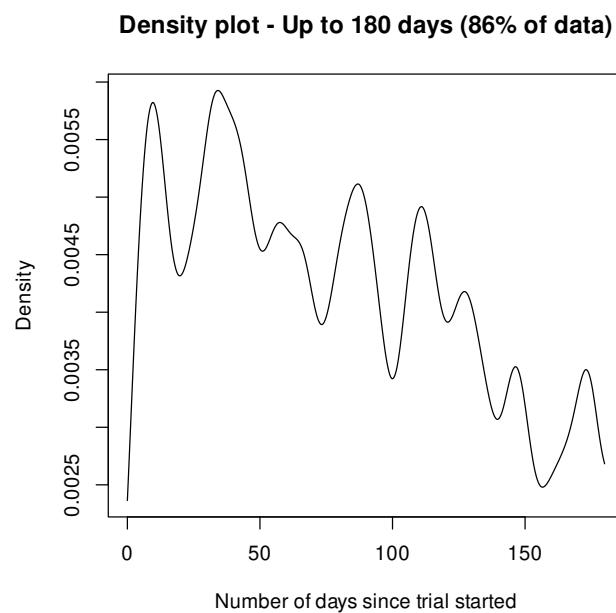
Systolic Blood Pressure Before and After Treatment



```
days <- abs(rnorm(1000, 80, 125))
plot(density(days, from = 0),
  main = "Density plot",
  xlab = "Number of days since trial started"
)
```

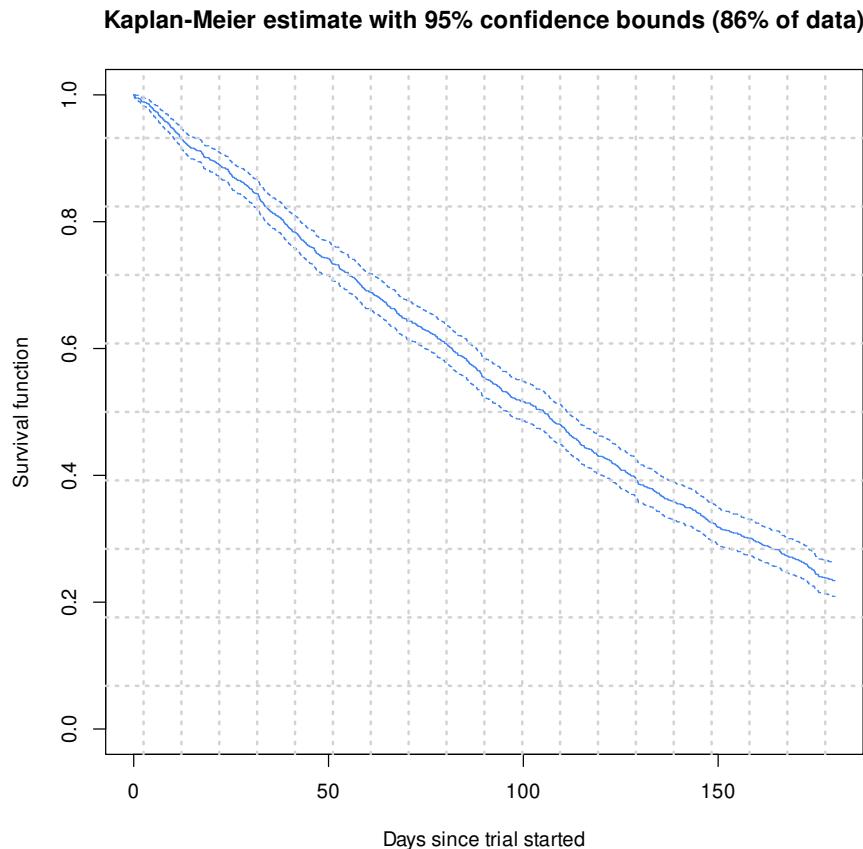


```
plot(density(days, from = 0, to = 180, adjust = 0.2),
  main = "Density plot - Up to 180 days (86% of data)",
  xlab = "Number of days since trial started"
)
```



```
library(survival)
surv.days <- Surv(days)
surv.fit <- survfit(surv.days ~ 1)
plot(surv.fit,
  main = "Kaplan-Meier estimate with 95% confidence bounds (86% of data)",
  xlab = "Days since trial started",
  xlim = c(0, 180),
```

```
  ylab = "Survival function"  
)  
grid(20, 10, lwd = 2)
```



#### 10.2.4 经验图

```
with(data = faithful, {  
  long <- eruptions[eruptions > 3]  
  plot(ecdf(long), do.points = FALSE, verticals = TRUE, main = "")  
  x <- seq(3, 5.4, 0.01)  
  lines(x, pnorm(x, mean = mean(long), sd = sqrt(var(long))), lty = 3)  
})
```

#### 10.2.5 QQ图

```
with(data = faithful, {  
  long <- eruptions[eruptions > 3]  
  par(pty = "s") # arrange for a square figure region  
  qqnorm(long, main = "")  
  qqline(long)
```

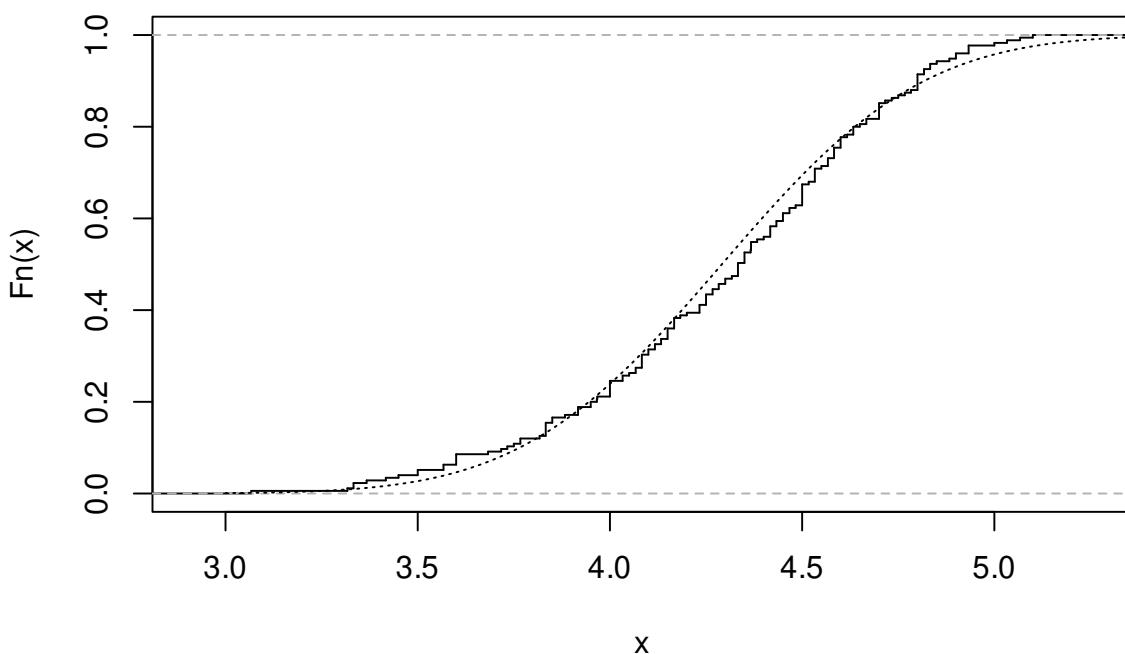
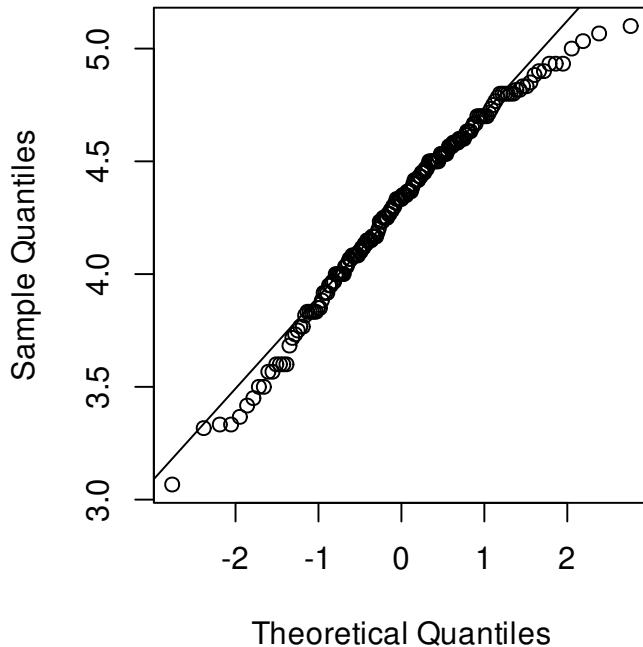


图 10.46: 累积经验分布图

})



### 10.2.6 时序图

时序图最适合用来描述股价走势

```
matplotlib.pyplot.time(EuStockMarkets), EuStockMarkets,
    main = "",
    xlab = "Date", ylab = "closing prices",
    pch = 17, type = "l", col = 1:4
)
legend("topleft", colnames(EuStockMarkets), pch = 17, lty = 1, col = 1:4)
```

### 10.2.7 饼图

clockwise 参数

```
pie.sales <- c(0.12, 0.3, 0.26, 0.16, 0.04, 0.12)
names(pie.sales) <- c(
  "Blueberry", "Cherry",
  "Apple", "Boston Cream", "Other", "Vanilla Cream"
)
pie(pie.sales, clockwise = TRUE, main = "")
segments(0, 0, 0, 1, col = "red", lwd = 2)
text(0, 1, "init.angle = 90", col = "red")
```

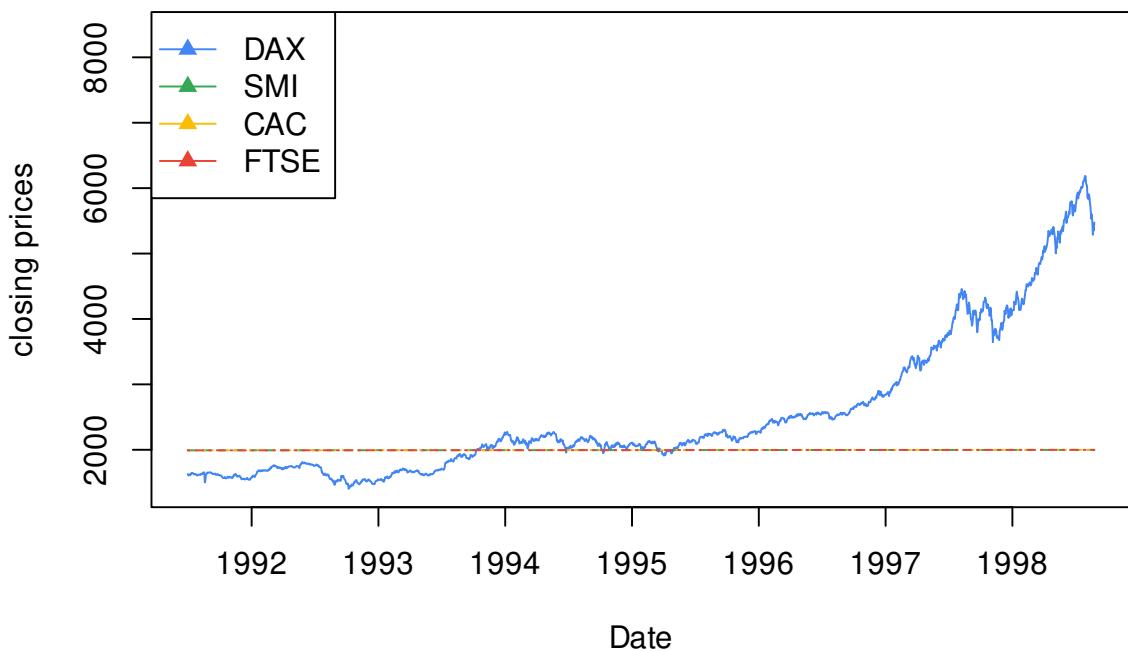
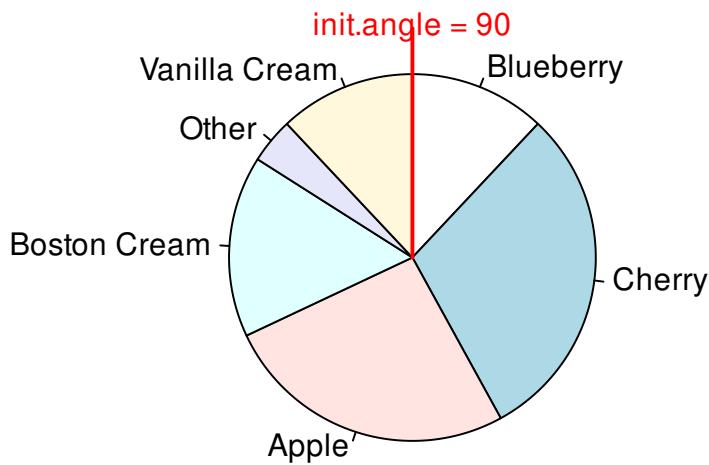


图 10.47: 1991–1998 年间主要欧洲股票市场日闭市价格指数图德国 DAX (Ibis), Switzerland SMI, 法国 CAC 和英国 FTSE



### 10.2.8 茎叶图

```
stem(longley$Unemployed)

##
## The decimal point is 2 digit(s) to the right of the |
##
## 1 | 99
## 2 | 134899
## 3 | 46789
## 4 | 078
```

### 10.2.9 散点图

在一维空间上，绘制散点图，其实是在看散点的疏密程度随坐标轴的变化

```
stripchart(longley$Unemployed,
  method = "jitter",
  jitter = 0.1, pch = 16, col = "lightblue"
)
stripchart(longley$Unemployed,
  method = "overplot",
```

```
  pch = 16, col = "lightblue"  
)
```



(a) 抖动图

图 10.48: 一维散点图

气泡图是二维散点图的一种变体，气泡的大小可以用来描述第三个变量，下面以数据集 topo 为例展示气泡图

```
# 加载数据集  
data(topo, package = "MASS")  
# 查看数据集  
str(topo)  
  
## 'data.frame': 52 obs. of 3 variables:  
## $ x: num 0.3 1.4 2.4 3.6 5.7 1.6 2.9 3.4 3.4 4.8 ...  
## $ y: num 6.1 6.2 6.1 6.2 6.2 5.2 5.1 5.3 5.7 5.6 ...  
## $ z: int 870 793 755 690 800 800 730 728 710 780 ...
```

topo 是空间地形数据集，包含有 52 行 3 列，数据点是 310 平方英尺范围内的海拔高度数据，x 坐标每单位 50 英尺，y 坐标单位同 x 坐标，海拔高度 z 单位是英尺

```
plot(y ~ x,  
  cex = (960 - z) / (960 - 690) * 3, data = topo,  
  xlab = "X Coordinates", ylab = "Y coordinates"  
)
```

散点图也适合分类数据的展示，在图中用不同颜色或符号标记数据点所属类别，即在普通散点图的基础上添加一分类变量的描述

```
plot(mpg ~ hp,  
  data = subset(mtcars, am == 1), pch = 16, col = "blue",  
  xlim = c(50, 350), ylim = c(10, 35)  
)  
points(mpg ~ hp,  
  col = "red", pch = 16,  
  data = subset(mtcars, am == 0)  
)
```

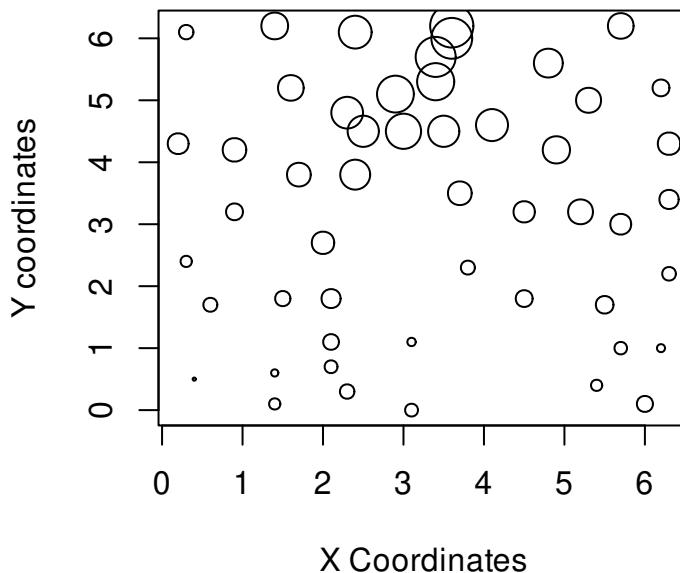


图 10.49: 地形图之海拔高度

```
legend(300, 35,
  c("1", "0"),
  title = "am",
  col = c("blue", "red"),
  pch = c(16, 16)
)
```

iris 数据

```
plot(Sepal.Length ~ Sepal.Width, data = iris, col = Species, pch = 16)
legend("topright",
  legend = unique(iris$Species), box.col = "gray",
  pch = 16, col = unique(iris$Species)
)
box(col = "gray")
```

分组散点图和平滑

```
library(carData)
library(car)
scatterplot(Sepal.Length ~ Sepal.Width,
  col = c("black", "red", "blue"), pch = c(16, 16, 16),
  smooth = TRUE, boxplots = "xy", groups = iris$Species,
  xlab = "Sepal.Width", ylab = "Sepal.Length", data = iris
)
```

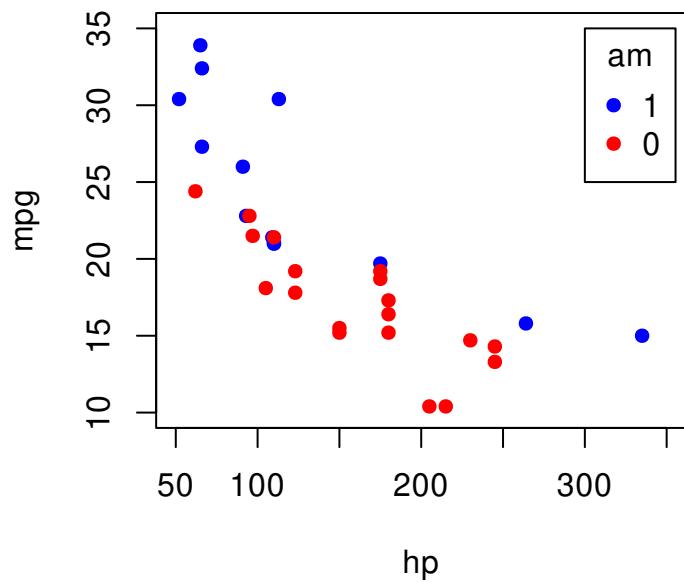


图 10.50: 分类散点图

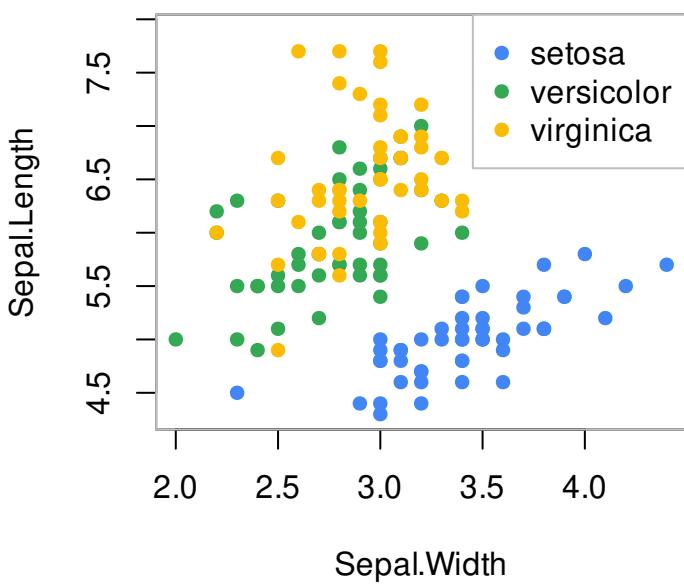


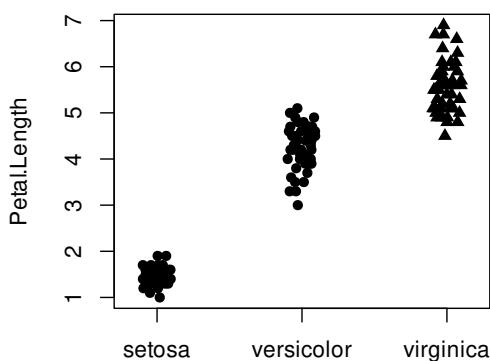
图 10.51: 分类散点图

有时为了实现特定的目的，需要高亮其中某些点，按类别或者因子变量分组绘制散点图，这里继续采用 `stripchart` 函数绘制二维散点图10.52，由左图可知，函数 `stripchart` 提供的参数 `pch` 不接受向量，实际只是取了前三个值 16 16 17 对应于 `Species` 的三类，关键是高亮的分界点是有区分意义的

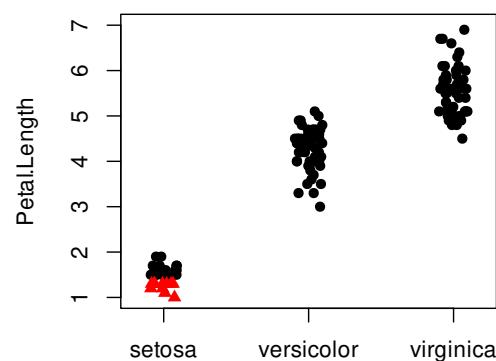
```

data("iris")
pch <- rep(16, length(iris$Petal.Length))
pch[which(iris$Petal.Length < 1.4)] <- 17
stripchart(Petal.Length ~ Species,
  data = iris,
  vertical = TRUE, method = "jitter",
  pch = pch
)
# 对比一下
stripchart(Petal.Length ~ Species,
  data = iris, subset = Petal.Length > 1.4,
  vertical = TRUE, method = "jitter", ylim = c(1, 7),
  pch = 16
)
stripchart(Petal.Length ~ Species,
  data = iris, subset = Petal.Length < 1.4,
  vertical = TRUE, method = "jitter", add = TRUE,
  pch = 17, col = "red"
)

```



(a) 原图



(b) 高亮

图 10.52: 高亮图中部分散点

如果存在大量散点

```

densCols(x,
  y = NULL, nbin = 128, bandwidth,
  colramp = colorRampPalette(blues9[-(1:3)])
)

```

)

`densCols` 函数根据点的局部密度生成颜色，密度估计采用核平滑法，由 **KernSmooth** 包的 `bkde2D` 函数实现。参数 `colramp` 传递一个函数，`colorRampPalette` 根据给定的几种颜色生成函数，参数 `bandwidth` 实际上是传给 `bkde2D` 函数

```
plot(faithful,
  col = densCols(faithful),
  pch = 20, panel.first = grid()
)
```

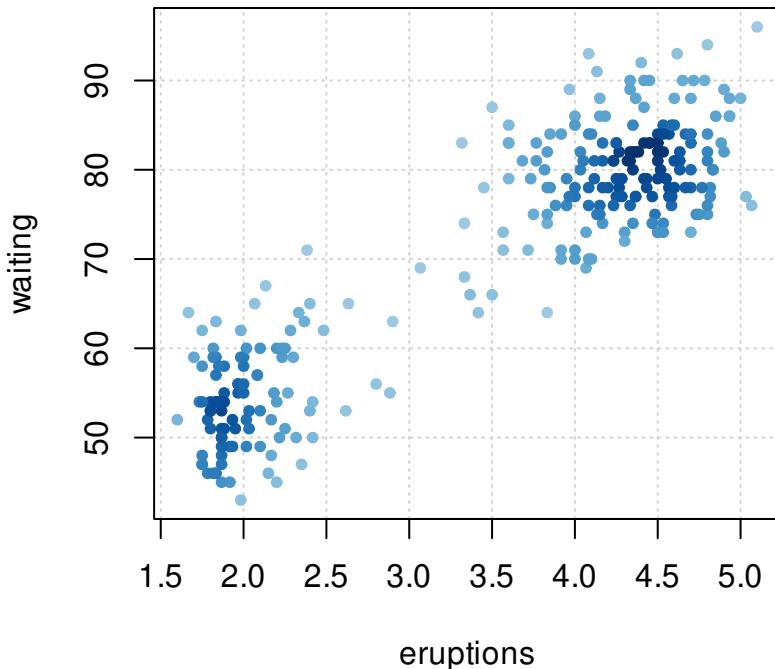


图 10.53: 根据点的密度生成颜色

气泡图也是散点图的一种

```
plot(Volume ~ Height,
  data = trees, pch = 16, cex = Girth / 8,
  col = rev(terrain.colors(nrow(trees), alpha = .5))
)
box(col = "gray")
```

气泡图

```
# 空白画布
plot(c(1, 5, 10), c(1, 5, 10), panel.first = grid(10, 10),
  type = "n", axes = FALSE, ann = FALSE)
```

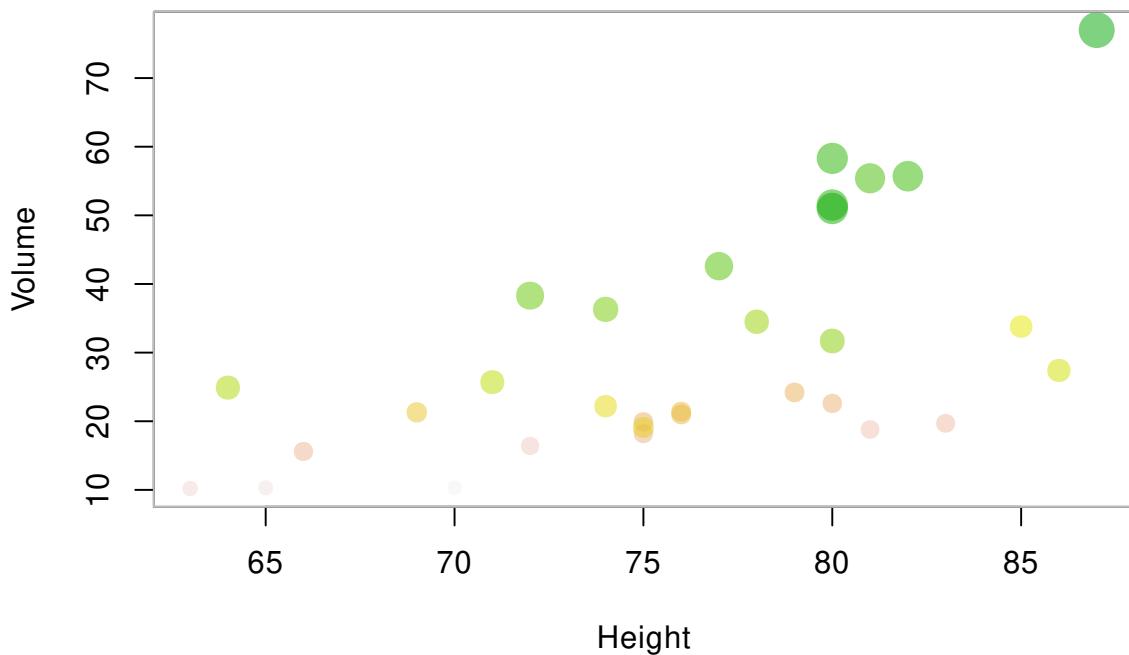
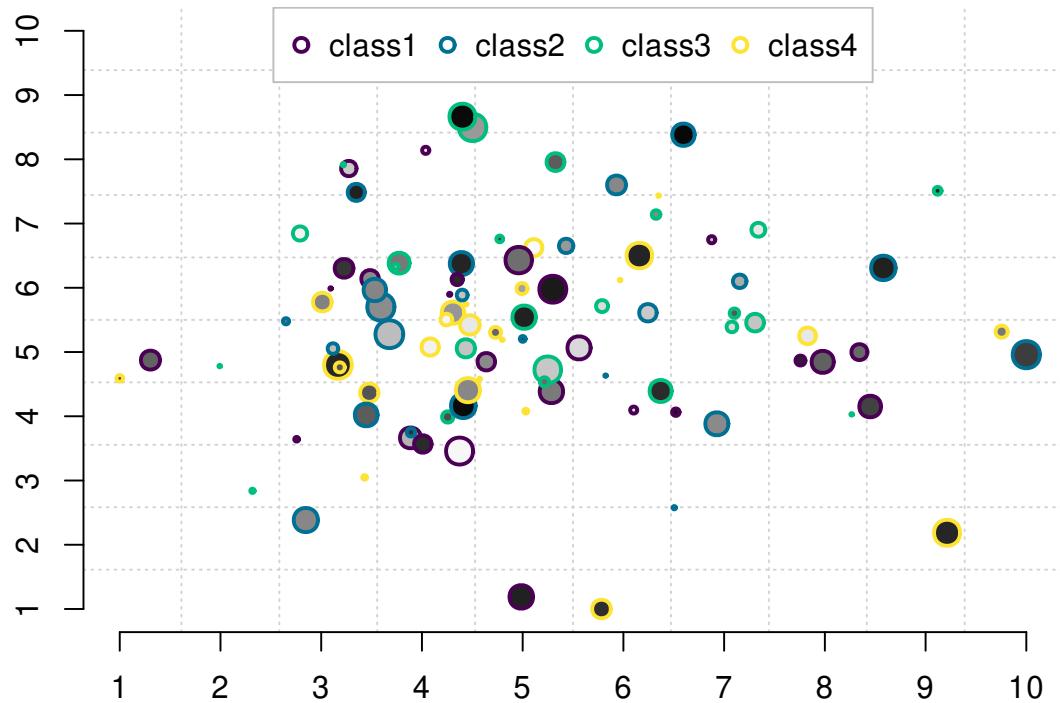


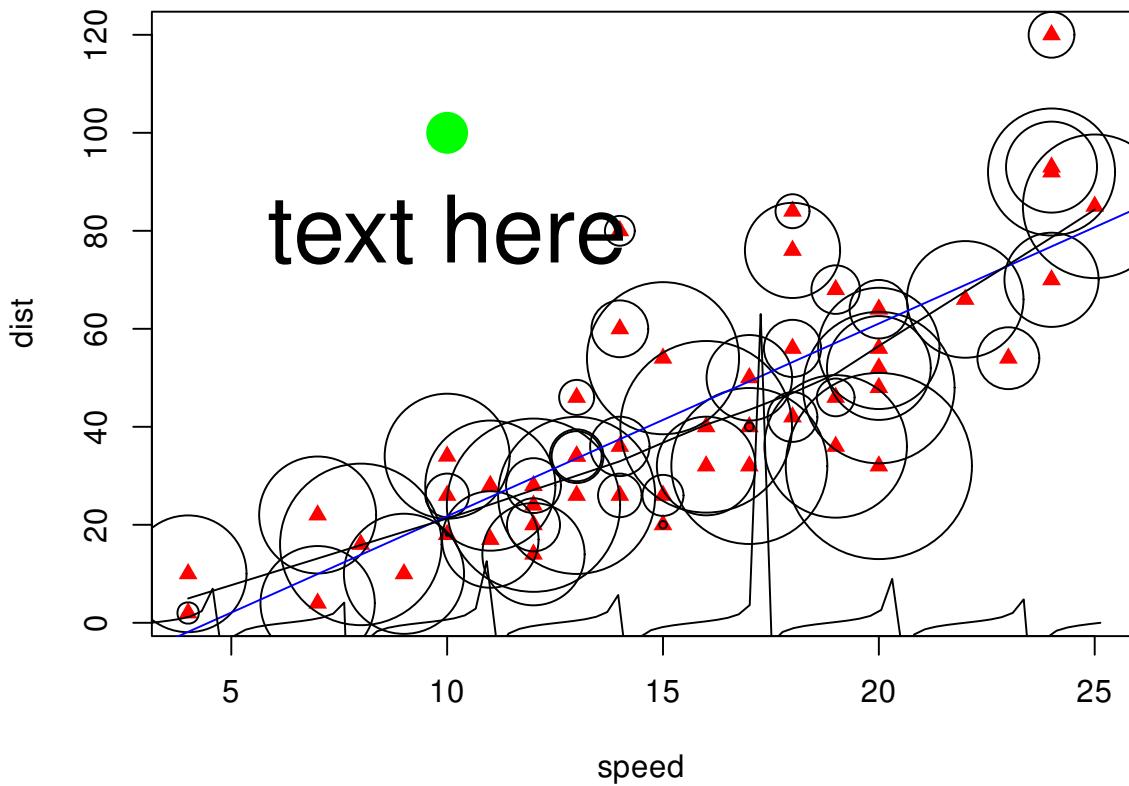
图 10.54: 气泡图

```
# 添加坐标轴
axis(1, at = seq(10), labels = TRUE)
axis(2, at = seq(10), labels = TRUE)
par(new = TRUE) # 在当前图形上添加图形
# axes 坐标轴上的刻度 "xaxt" or "yaxt" ann 坐标轴和标题的标签
set.seed(1234)
plot(rnorm(100, 5, 1), rnorm(100, 5, 1),
  cex = runif(100, 0, 2),
  col = hcl.colors(4)[rep(seq(4), 100)],
  bg = paste0("gray", replicate(100, sample(seq(100), 1, replace = TRUE))),
  axes = FALSE, ann = FALSE, pch = 21, lwd = 2
)
legend("top",
  legend = paste0("class", seq(4)), col = hcl.colors(4),
  pt.lwd = 2, pch = 21, box.col = "gray", horiz = TRUE
)
```



除了 `par(new=TRUE)` 设置外，有些函数本身就具有 `add` 选项

```
set.seed(1234)
plot(dist ~ speed, data = cars, pch = 17, col = "red", cex = 1)
with(cars, symbols(dist ~ speed,
  circles = runif(length(speed), 0, 1),
  pch = 16, inches = .5, add = TRUE
))
z <- lm(dist ~ speed, data = cars)
abline(z, col = "blue")
curve(tan, from = 0, to = 8 * pi, n = 100, add = TRUE)
lines(stats::lowess(cars))
points(10, 100, pch = 16, cex = 3, col = "green")
text(10, 80, "text here", cex = 3)
```



### 10.2.10 抖动图

抖动散点图

```
mat <- matrix(1:length(colors()), ncol = 9, byrow = TRUE)
df <- data.frame(
  col = colors(),
  x = as.integer(cut(1:length(colors()), 9)),
  y = rep(1:73, 9), stringsAsFactors = FALSE
)
par(mar = c(4, 4, 1, 0.1))
plot(y ~ jitter(x),
  data = df, col = df$col,
  pch = 16, main = "Visualizing colors() split in 9 groups",
  xlab = "Group",
  ylab = "Element of the group (min = 1, max = 73)",
  sub = "x = 3, y = 1 means that it's the 2 * 73 + 1 = 147th color"
)
```

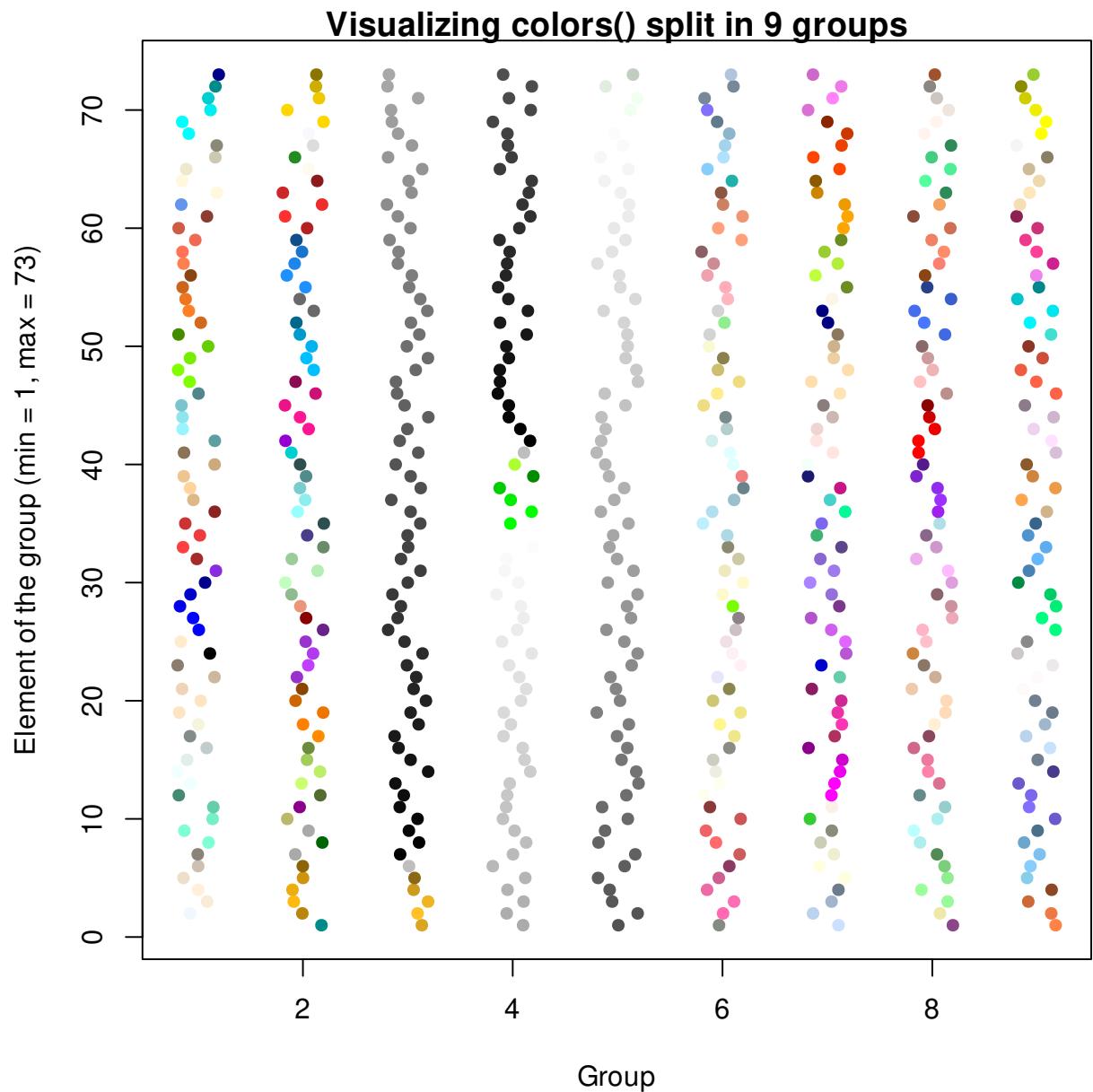


图 10.55: 抖动散点图



### 10.2.11 箱线图

boxplotdbl: Double Box Plot for Two-Axes Correlation. Correlation chart of two set (x and y) of data. Using Quartiles with boxplot style. Visualize the effect of factor.

#### 复合箱线图

```
with(data = iris, {  
  op <- par(mfrow = c(2, 2), mar = c(4, 4, 2, .5))  
  plot(Sepal.Length ~ Species)  
  plot(Sepal.Width ~ Species)  
  plot(Petal.Length ~ Species)  
  plot(Petal.Width ~ Species)  
  par(op)  
  mtext("Edgar Anderson's Iris Data", side = 3, line = 4)  
})
```

箱线图的花样也很多

```
data(InsectSprays)  
par(mar = c(4, 4, .5, .5))  
boxplot(  
  data = InsectSprays, count ~ spray,  
  col = "gray", xlab = "Spray", ylab = "Count"  
)  
  
boxplot(  
  data = InsectSprays, count ~ spray,  
  col = "gray", horizontal = TRUE,  
  las = 1, xlab = "Count", ylab = "Spray"  
)
```

#### Notched Boxplots

```
set.seed(1234)  
n <- 8  
g <- gl(n, 100, n * 100) # n水平个数 100是重复次数  
x <- rnorm(n * 100) + sqrt(as.numeric(g))  
boxplot(split(x, g), col = gray.colors(n), notch = TRUE)  
title(  
  main = "Notched Boxplots", xlab = "Group",  
  font.main = 4, font.lab = 1  
)
```

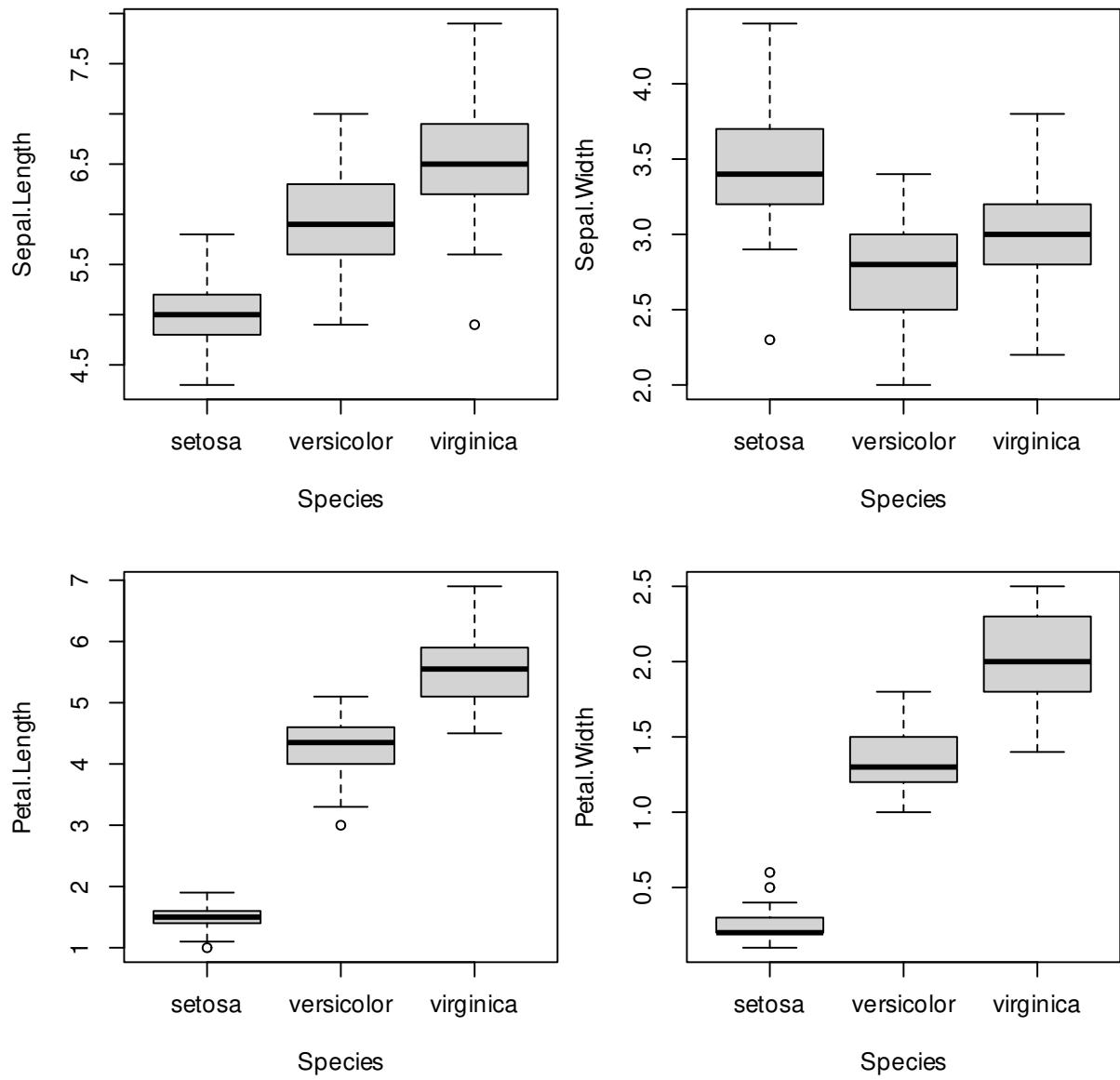
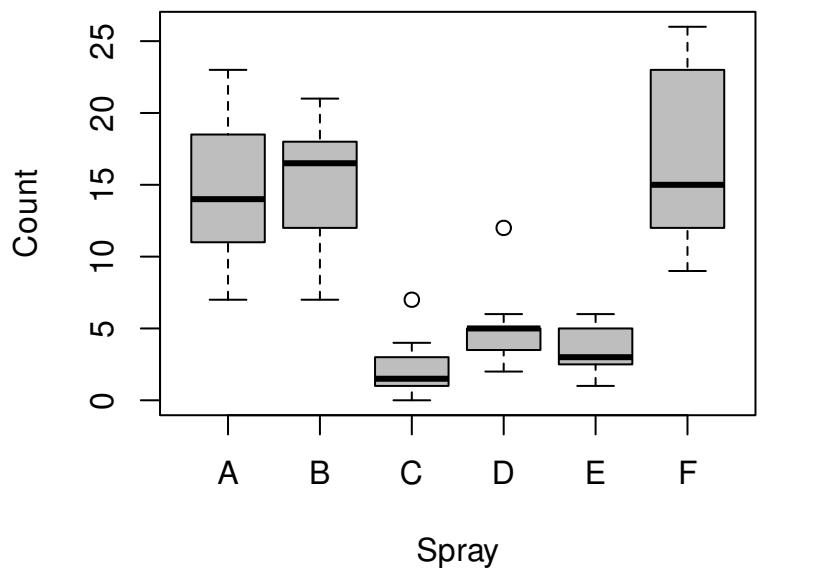
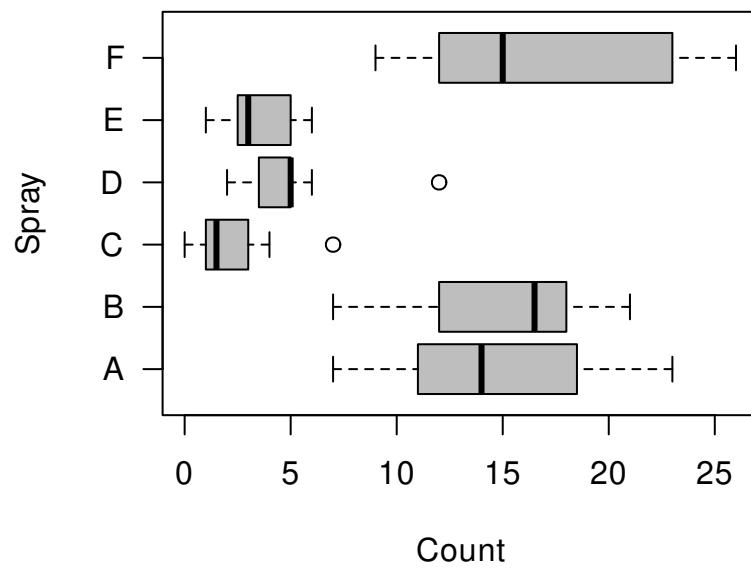


图 10.56: 安德森的鸢尾花数据



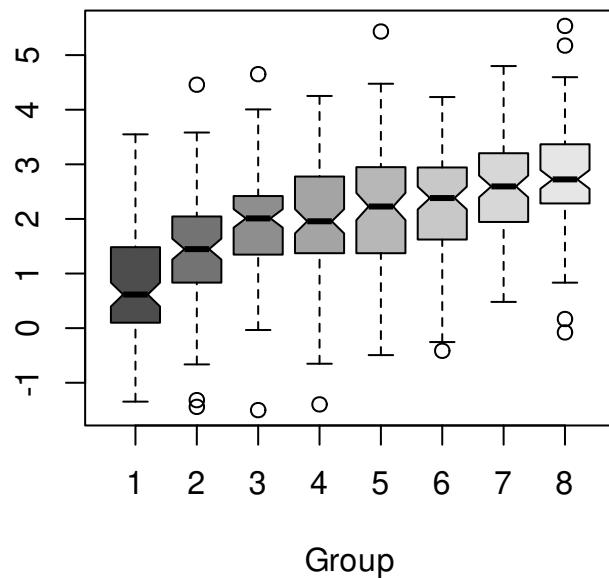
(a) 垂直放置



(b) 水平放置

图 10.57: 箱线图

### Notched Boxplots



真实的情况是这样的

```
cumcm2011A <- readRDS(file = "cumcm2011A.RDS")
par(mfrow = c(2, 4), mar = c(4, 3, 1, 1))
with(cumcm2011A, boxplot(As, xlab = "As"))
abline(h = c(1.8, 3.6, 5.4), col = c("green", "blue", "red"), lty = 2)

with(cumcm2011A, boxplot(Cd, xlab = "Cd"))
abline(h = c(70, 130, 190), col = c("green", "blue", "red"), lty = 2)

with(cumcm2011A, boxplot(Cr, xlab = "Cr"))
abline(h = c(13, 31, 49), col = c("green", "blue", "red"), lty = 2)

with(cumcm2011A, boxplot(Cu, xlab = "Cu"))
abline(h = c(6.0, 13.2, 20.4), col = c("green", "blue", "red"), lty = 2)

with(cumcm2011A, boxplot(Hg, xlab = "Hg"))
abline(h = c(19, 35, 51), col = c("green", "blue", "red"), lty = 2)

with(cumcm2011A, boxplot(Ni, xlab = "Ni"))
abline(h = c(4.7, 12.3, 19.9), col = c("green", "blue", "red"), lty = 2)

with(cumcm2011A, boxplot(Pb, xlab = "Pb"))
abline(h = c(19, 31, 43), col = c("green", "blue", "red"), lty = 2)

with(cumcm2011A, boxplot(Zn, xlab = "Zn"))
```



```
abline(h = c(41, 69, 97), col = c("green", "blue", "red"), lty = 2)

boxplot(As ~ area,
  data = cumcm2011A,
  col = hcl.colors(5)
)
abline(
  h = c(1.8, 3.6, 5.4), col = c("green", "blue", "red"),
  lty = 2, lwd = 2
)
```

### 10.2.12 残差图

iris 四个测量指标

```
vec_mean <- colMeans(iris[, -5])
vec_sd <- apply(iris[, -5], 2, sd)
plot(seq(4), vec_mean,
  ylim = range(c(vec_mean - vec_sd, vec_mean + vec_sd)),
  xlab = "Species", ylab = "Mean +/- SD", lwd = 1, pch = 19,
  axes = FALSE
)
axis(1, at = seq(4), labels = colnames(iris)[-5])
axis(2, at = seq(7), labels = seq(7))
arrows(seq(4), vec_mean - vec_sd, seq(4), vec_mean + vec_sd,
  length = 0.05, angle = 90, code = 3
)
box()
```

### 10.2.13 提琴图

Tom Kelly 维护的 vioplot 包 <https://github.com/TomKellyGenetics/vioplot>

### 10.2.14 轮廓图

topo 是地形数据

等高线图

### 10.2.15 折线图

函数曲线, 样条曲线, 核密度曲线, 平行坐标图

- 折线图
- 点线图 `plot(type="b")` 函数曲线图 `curve` `matplot` X 样条曲线 `xspline`

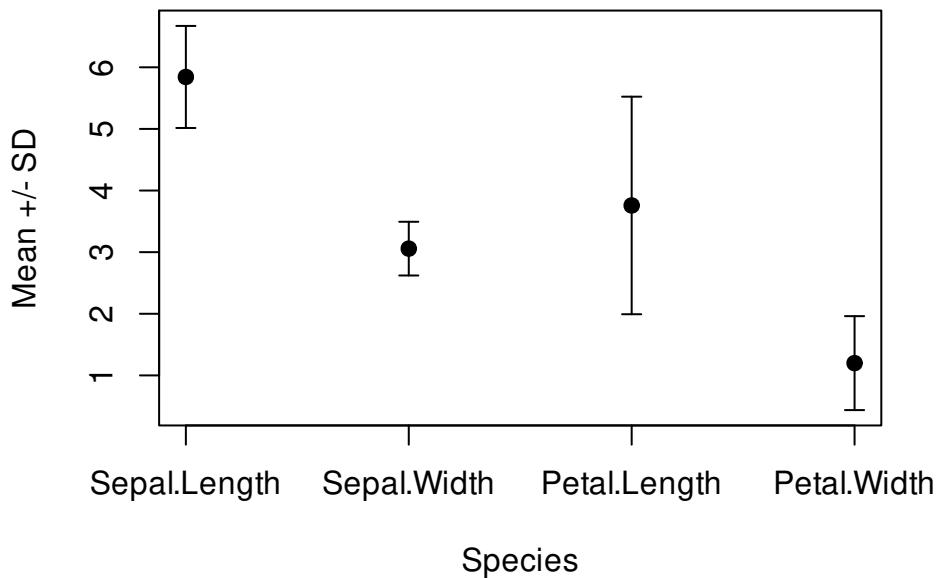


图 10.58: 带标准差的均值散点图

- 时序图

太阳黑子活动数据

sunspot.month Monthly Sunspot Data, from 1749 to “Present” sunspot.year Yearly Sunspot Data, 1700-1988 sunspots Monthly Sunspot Numbers, 1749-1983

```
plot(AirPassengers)
box(col = "gray")
```

### 10.2.16 函数图

```
x0 <- 2^(-20:10)
nus <- c(0:5, 10, 20)
x <- seq(0, 4, length.out = 501)

plot(x0, x0^-8,
      frame.plot = TRUE, # 添加绘图框
      log = "xy", # x 和 y 轴都取对数尺度
      axes = FALSE, # 去掉坐标轴
      xlab = "$u$",
      ylab = "$\mathcal{K}_{\kappa}(u)$", # 设置坐标轴标签
      type = "n", # 清除绘图区域的内容
      ann = TRUE, # 添加标题 x和y轴标签
      panel.first = grid() # 添加背景参考线
```

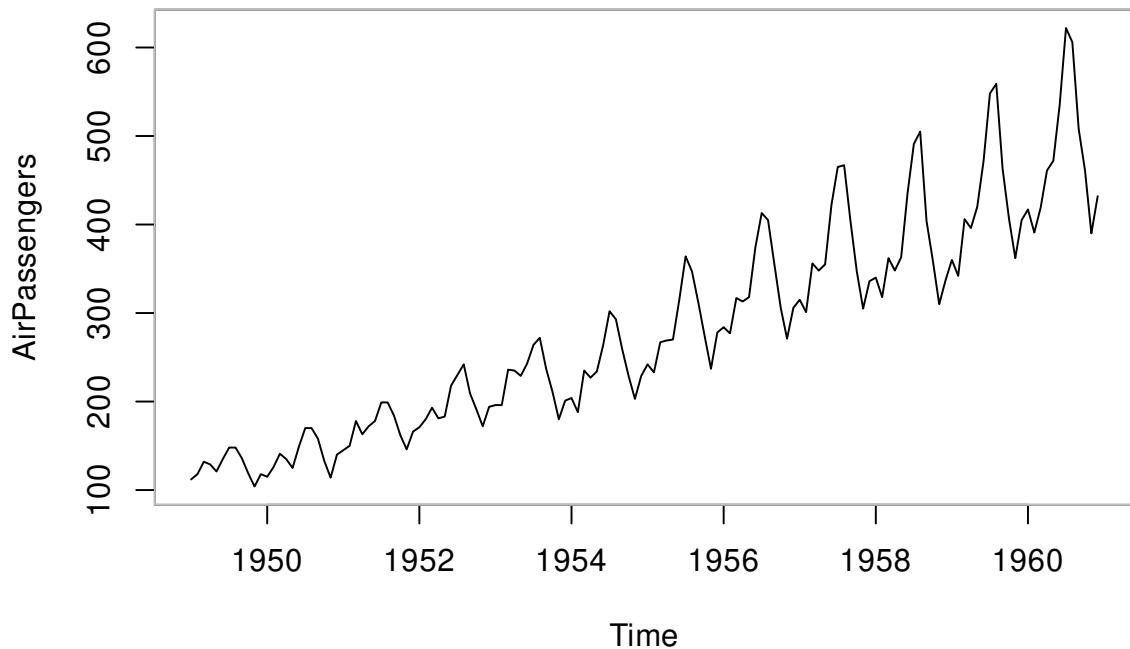


图 10.59: 折线图

```
)  
  
axis(1,  
  at = 10^seq(from = -8, to = 2, by = 2),  
  labels = paste0("$\\mathsf{10^{", seq(from = -8, to = 2, by = 2), "}}$")  
)  
axis(2,  
  at = 10^seq(from = -8, to = 56, by = 16),  
  labels = paste0("$\\mathsf{10^{", seq(from = -8, to = 56, by = 16), "}}$"), las = 1  
)  
  
for (i in seq(length(nus))) {  
  lines(x0, besselK(x0, nu = nus[i]), col = hcl.colors(9)[i], lwd = 2)  
}  
legend("topright",  
  legend = paste0("$\\kappa=", rev(nus), "$"),  
  col = hcl.colors(9, rev = T), lwd = 2, cex = 1  
)
```

还有 eta 函数和 gammaz 函数

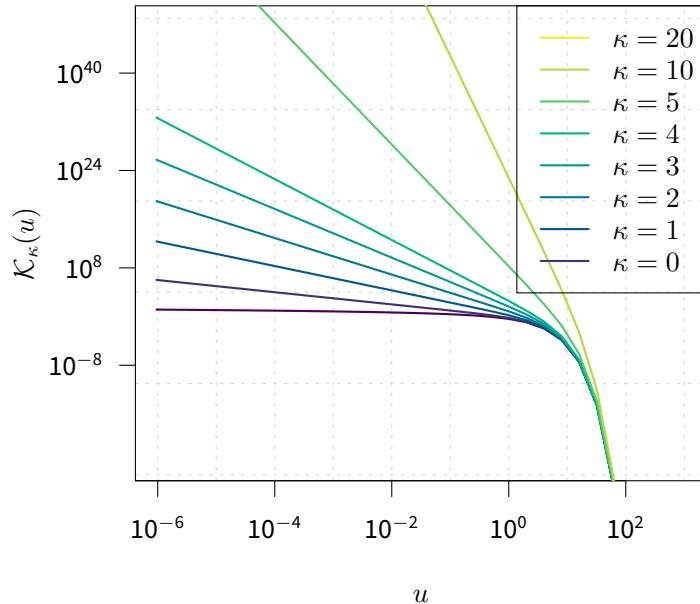


图 10.60: 贝塞尔函数

### 10.2.17 马赛克图

马赛克图 mosaicplot

```
plot(HairEyeColor, col = "lightblue", border = "white", main = "")
```

### 10.2.18 点图

dotchart 克利夫兰点图

条件图 coplot

### 10.2.19 矩阵图

在对角线上添加平滑曲线、密度曲线

```
pairs(longley,
  gap = 0,
  diag.panel = function(x, ...) {
    par(new = TRUE)
    hist(x,
      col = "light blue",
      probability = TRUE,
      axes = FALSE,
      main = "")
```

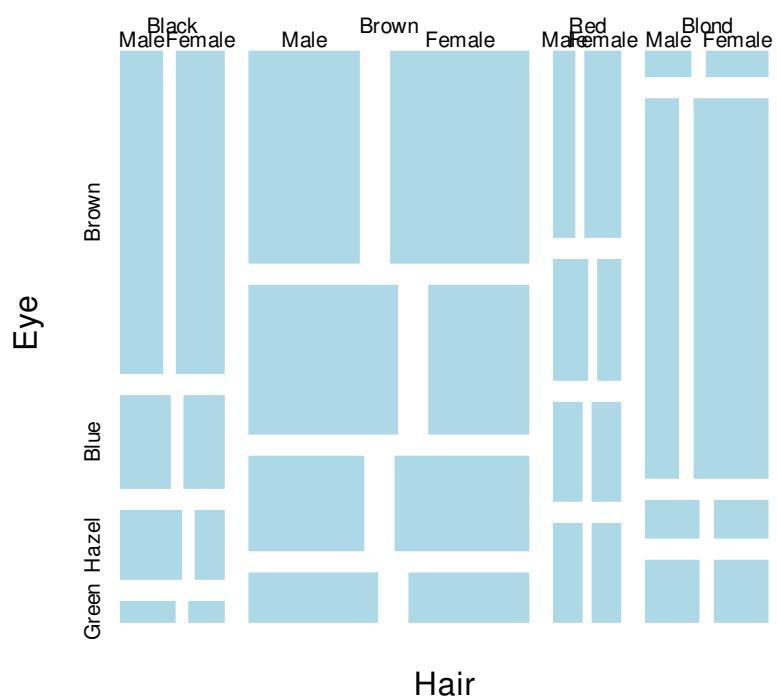


图 10.61: 马赛克图

```
    )
  lines(density(x),
    col = "red",
    lwd = 3
  )
  rug(x)
}
)
```

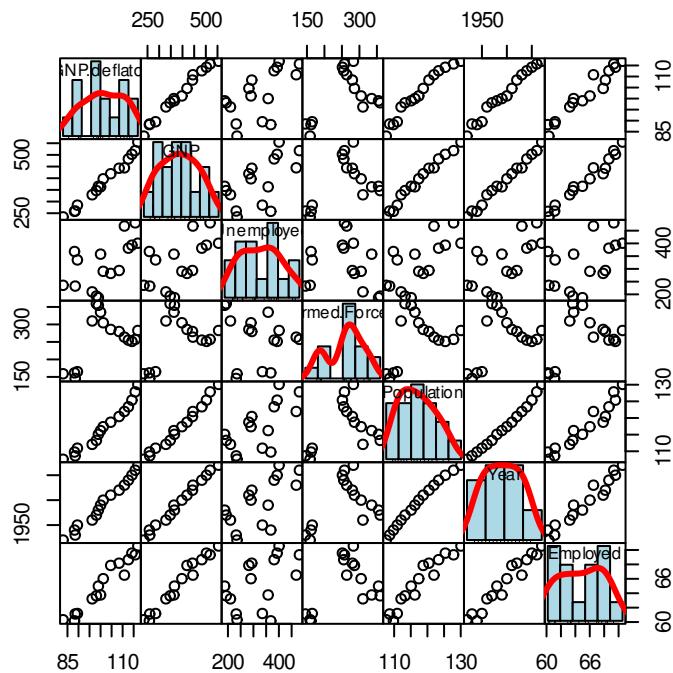


图 10.62: 变量关系

```
# 自带 layout
plot(iris[, -5], col = iris$Species)
```

### 10.2.20 雷达图

星图 stars 多元数据

### 10.2.21 玫瑰图

注意与 image 函数区别

```
x <- 10 * (1:nrow(volcano))
y <- 10 * (1:ncol(volcano))
image(x, y, volcano, col = terrain.colors(100), axes = FALSE)
```

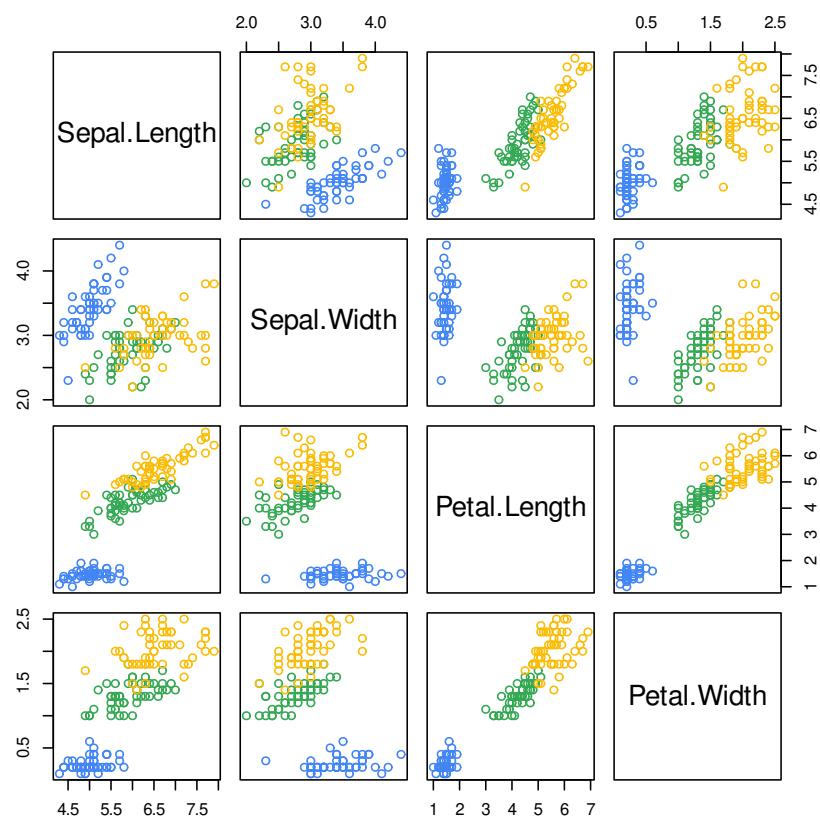


图 10.63: 矩阵图

```

contour(x, y, volcano,
  levels = seq(90, 200, by = 5),
  add = TRUE, col = "peru"
)
axis(1, at = seq(100, 800, by = 100))
axis(2, at = seq(100, 600, by = 100))
box()
title(main = "Maunga Whau Volcano", font.main = 4)

```

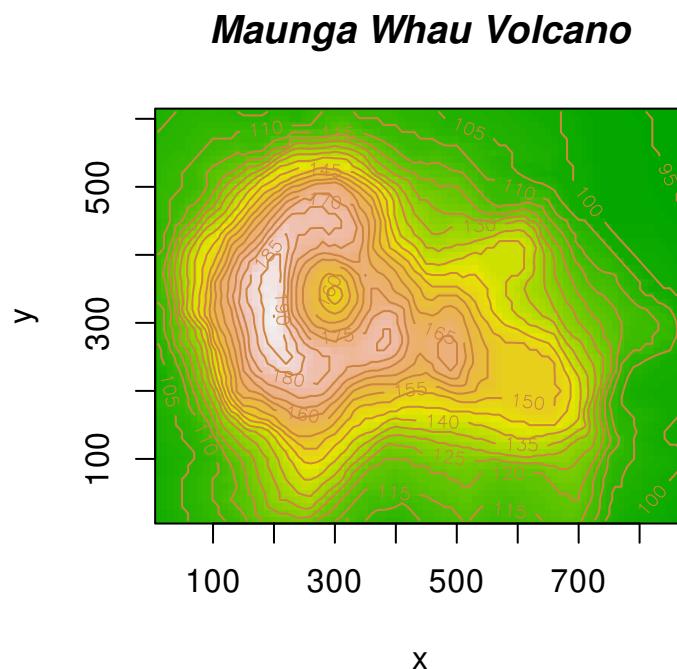


图 10.64: image 图形

## 10.2.22 透视图

```

par(mar = c(.5, 2.1, .5, .5))
x1 <- seq(-10, 10, length = 51)
x2 <- x1
f <- function(x1, x2, mu1 = 0, mu2 = 0, s11 = 10, s12 = 15, s22 = 10, rho = 0.5) {
  term1 <- 1 / (2 * pi * sqrt(s11 * s22 * (1 - rho^2)))
  term2 <- -1 / (2 * (1 - rho^2))
  term3 <- (x1 - mu1)^2 / s11
  term4 <- (x2 - mu2)^2 / s22
  term5 <- -2 * rho * ((x1 - mu1) * (x2 - mu2)) / (sqrt(s11) * sqrt(s22))
  term1 * exp(term2 * (term3 + term4 - term5))
}
z <- outer(x1, x2, f)

```

```
library(shape)
persp(x1, x2, z,
      xlab = "", ylab = "", zlab = "",
      col = drapecol(z, col = terrain.colors(20)),
      border = NA, shade = 0.1, r = 50, d = 0.1, expand = 0.5,
      theta = 120, phi = 15, ltheta = 90, lphi = 180,
      ticktype = "detailed", nticks = 5
)
```

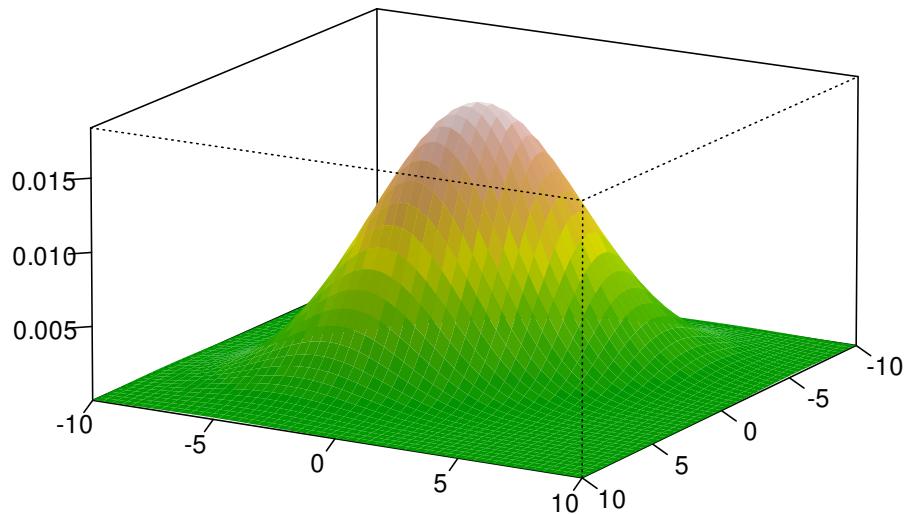


图 10.65: 统计学的世界

### 10.3 棚格统计图形

If you imagine that this pen is Trellis, then Lattice is not this pen.

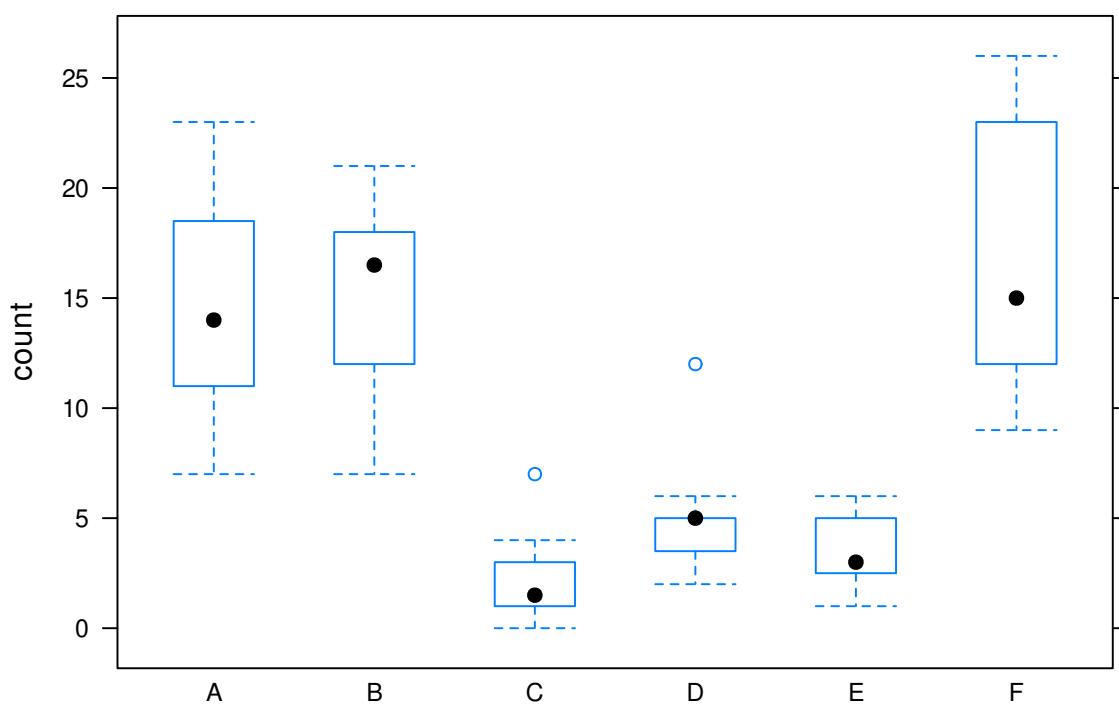
— Paul Murrell<sup>1</sup>

把网站搬出来, 汉化 <http://latticeextra.r-forge.r-project.org/>

#### 10.3.1 箱线图

```
library(lattice)
# plot(data = InsectSprays, count ~ spray)
bwplot(count ~ spray, data = InsectSprays)
```

<sup>1</sup>Paul 在 DSC 2001 大会上的幻灯片见<https://www.stat.auckland.ac.nz/~paul/Talks/dsc2001.pdf>

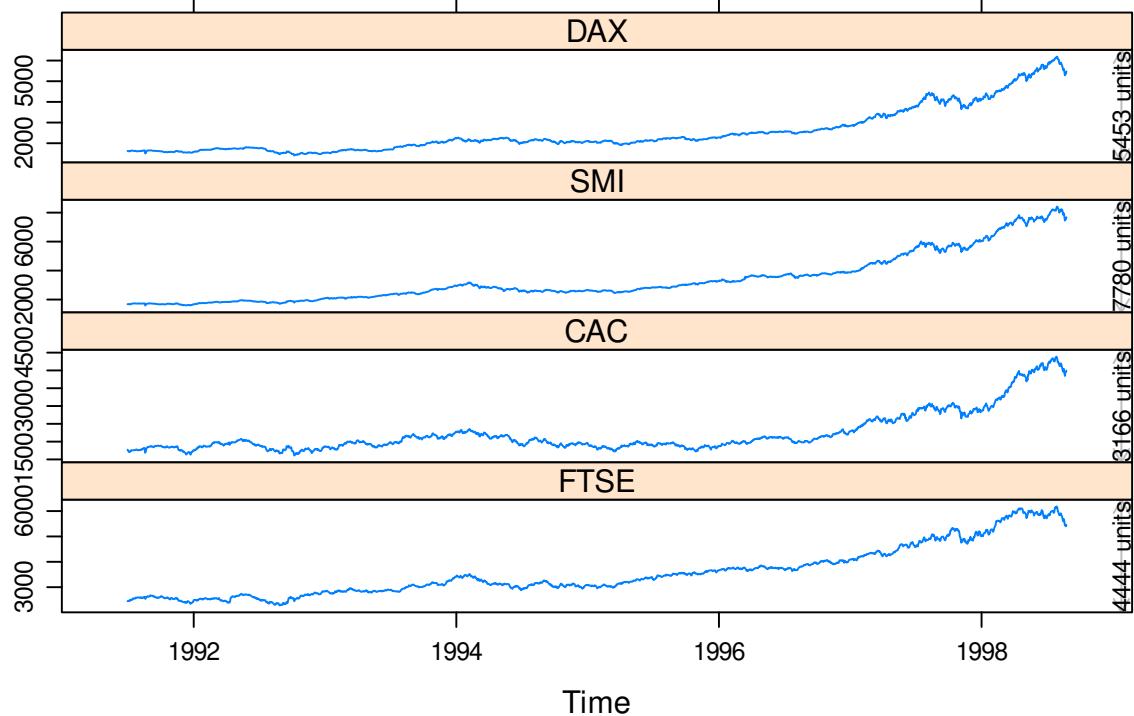


### 10.3.2 折线图

`latticeExtra` 包提供了强大的图层函数 `layer()`

多元时间序列

```
library(RColorBrewer)
library(latticeExtra)
xyplot(EuStockMarkets) +
  layer(panel.scaleArrow(
    x = 0.99, append = " units", col = "grey", srt = 90, cex = 0.8
  ))
```

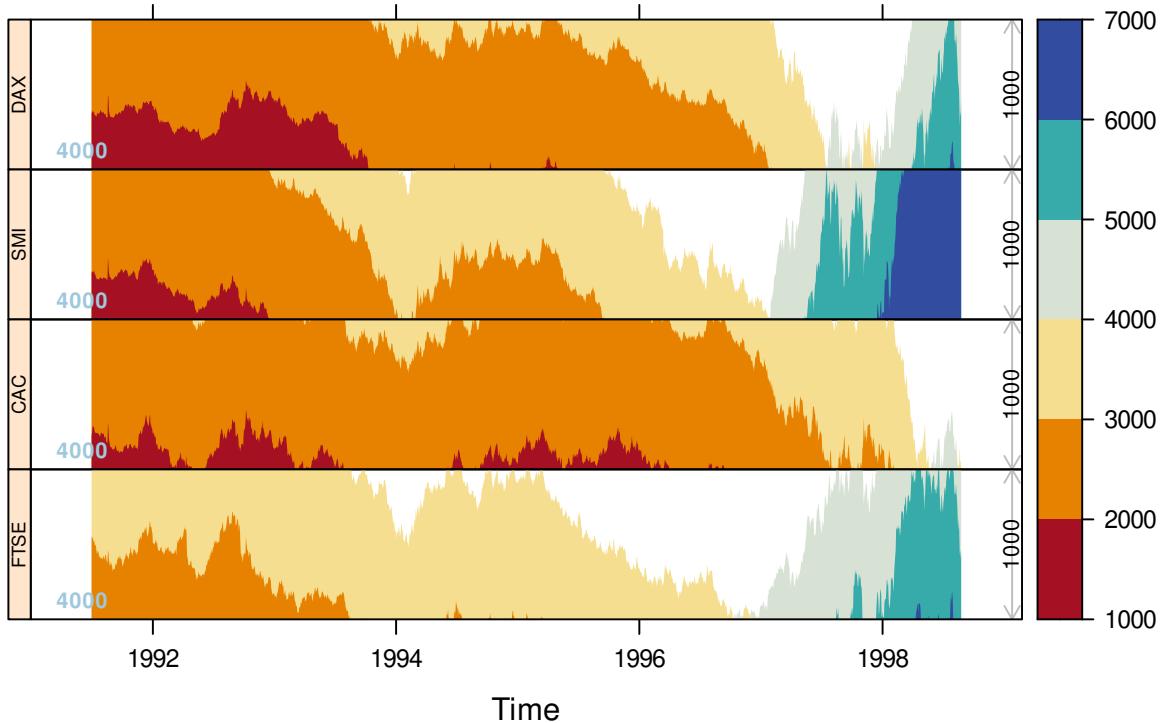


如何解释

时序图

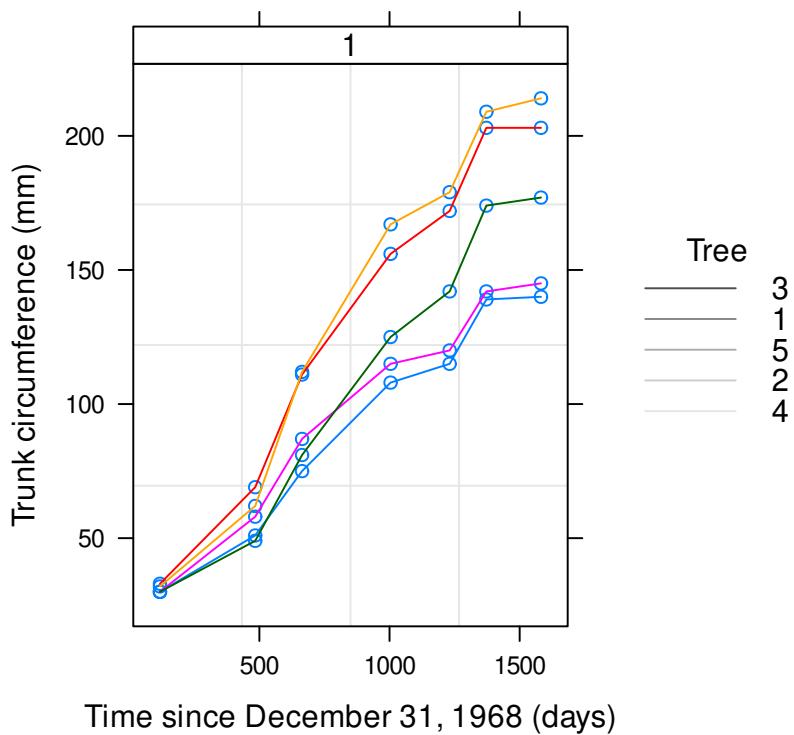
Plot many time series in parallel

```
horizonplot(EuStockMarkets,
  colorkey = TRUE,
  origin = 4000, horizonscale = 1000
) +
  layer(panel.scaleArrow(
    x = 0.99, digits = 1, col = "grey",
    srt = 90, cex = 0.7
)) +
  layer(
    lim <- current.panel.limits(),
    panel.text(lim$x[1], lim$y[1], round(lim$y[1], 1),
      font = 2,
      cex = 0.7, adj = c(-0.5, -0.5), col = "#9FC8DC"
    )
  )
```



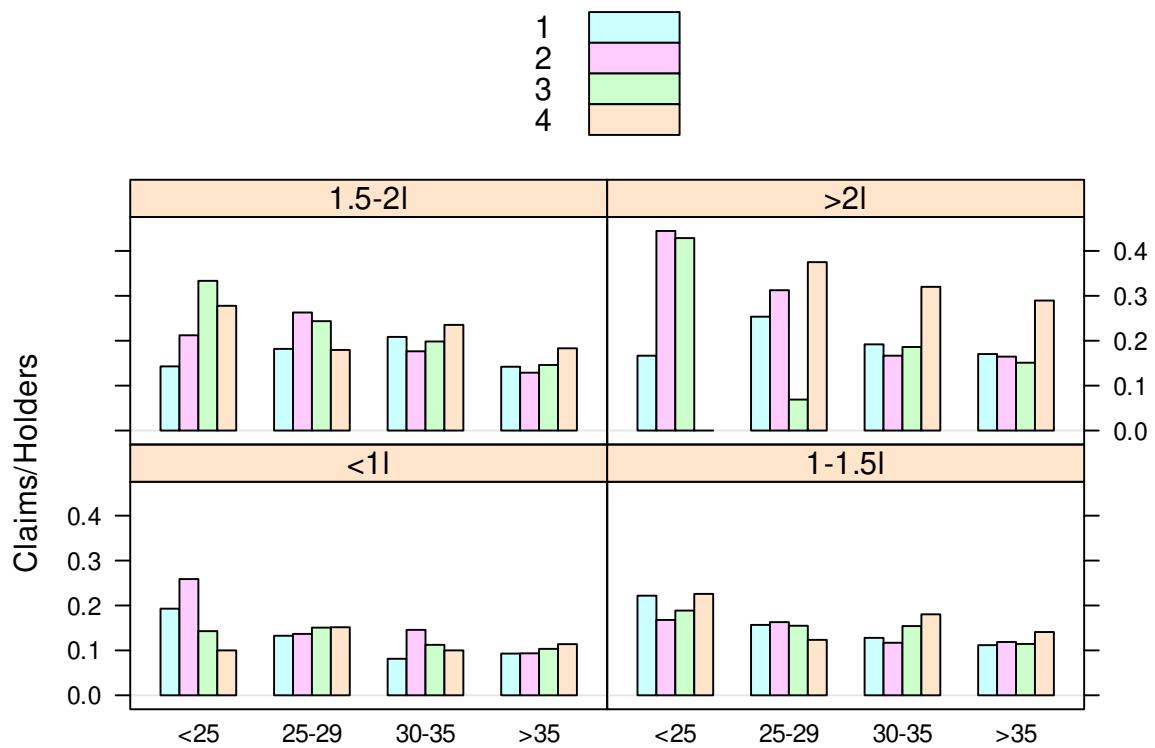
```
# # https://stackoverflow.com/questions/25109196/r-lattice-package-add-legend-to-a-figure
library(lattice)
library(nlme)

plot(Orange,
      outer = ~1,
      key = list(
        space = "right", title = "Tree", cex.title = 1,
        lines = list(lty = 1, col = gray.colors(5)),
        # points = list(pch = 1, col = gray.colors(5)),
        text = list(c("3", "1", "5", "2", "4")))
      ),
      par.settings = list(
        # plot.line = list(col = gray.colors(5), border = "transparent"),
        # plot.symbol = list(col = gray.colors(5), border = "transparent"),
        strip.background = list(col = "white"),
        strip.border = list(col = "black")
      )
    )
```

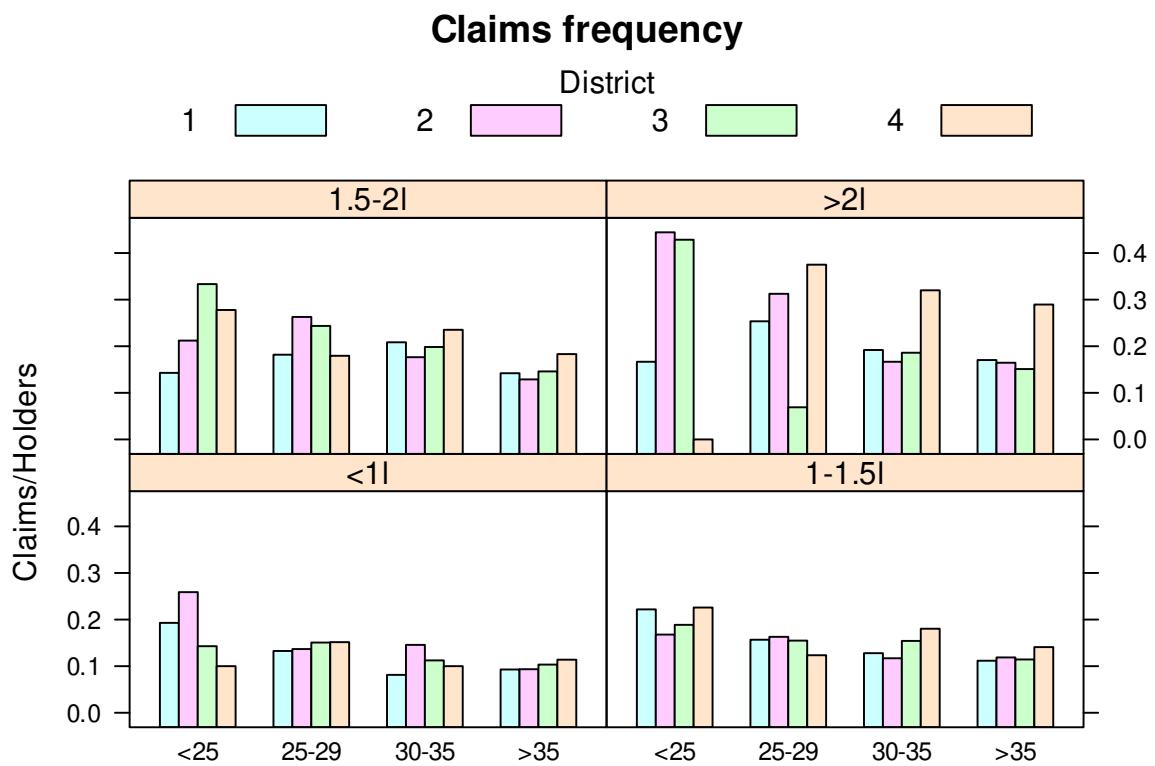


```
library(MASS)
library(lattice)
## Plot the claims frequency against age group by engine size and district

barchart(Claims / Holders ~ Age | Group,
  groups = District,
  data = Insurance, origin = 0, auto.key = TRUE
)
```

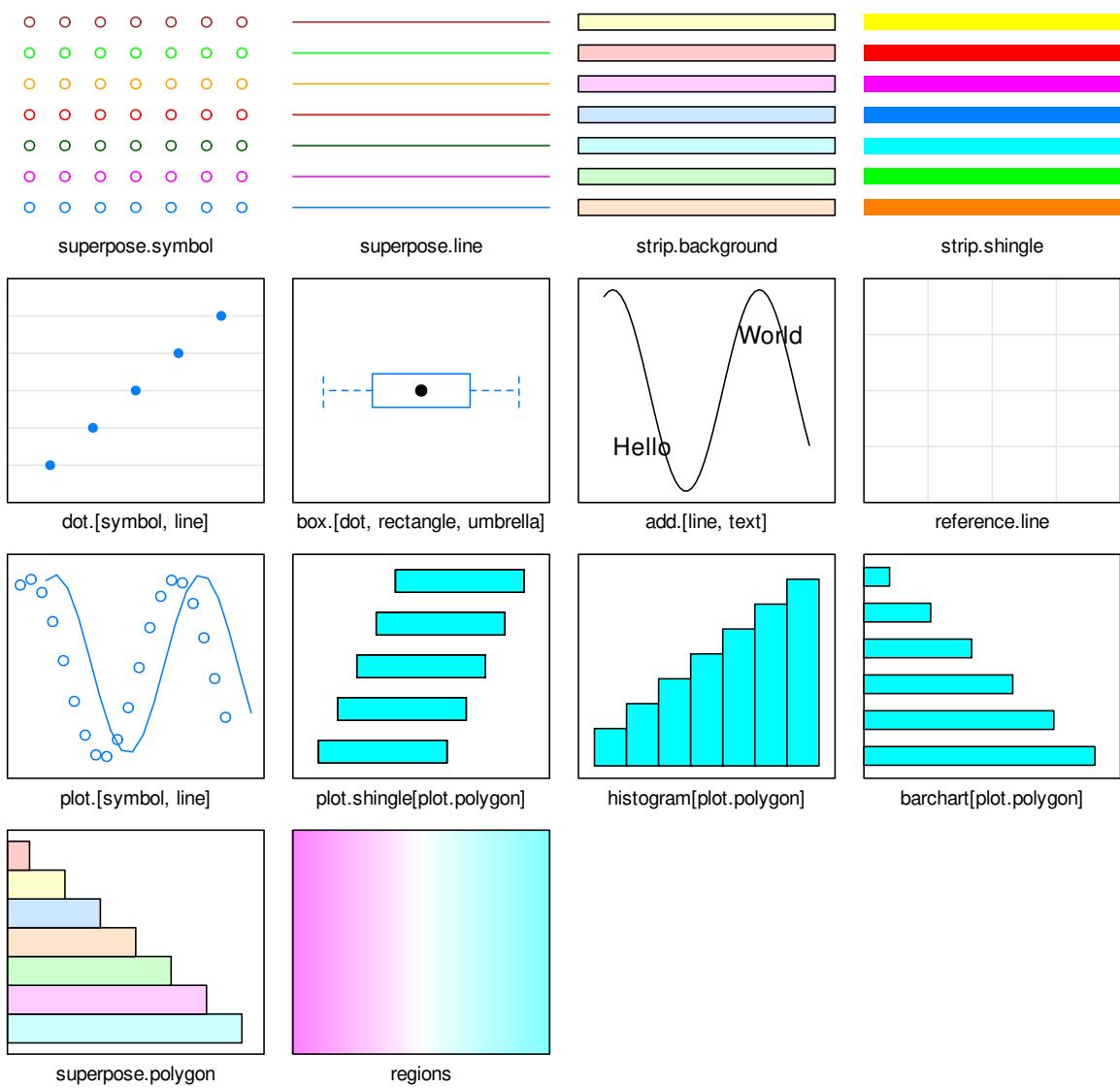


```
barchart(Claims / Holders ~ Age | Group,
  groups = District, data = Insurance,
  main = "Claims frequency",
  auto.key = list(
    space = "top", columns = 4,
    title = "District", cex.title = 1
  )
)
```



lattice 图形的参数设置

```
show.settings()
```



```

myColours <- brewer.pal(6, "Blues")
my.settings <- list(
  superpose.polygon = list(col = myColours[2:5], border = "transparent"),
  strip.background = list(col = myColours[6]),
  strip.border = list(col = "black")
)

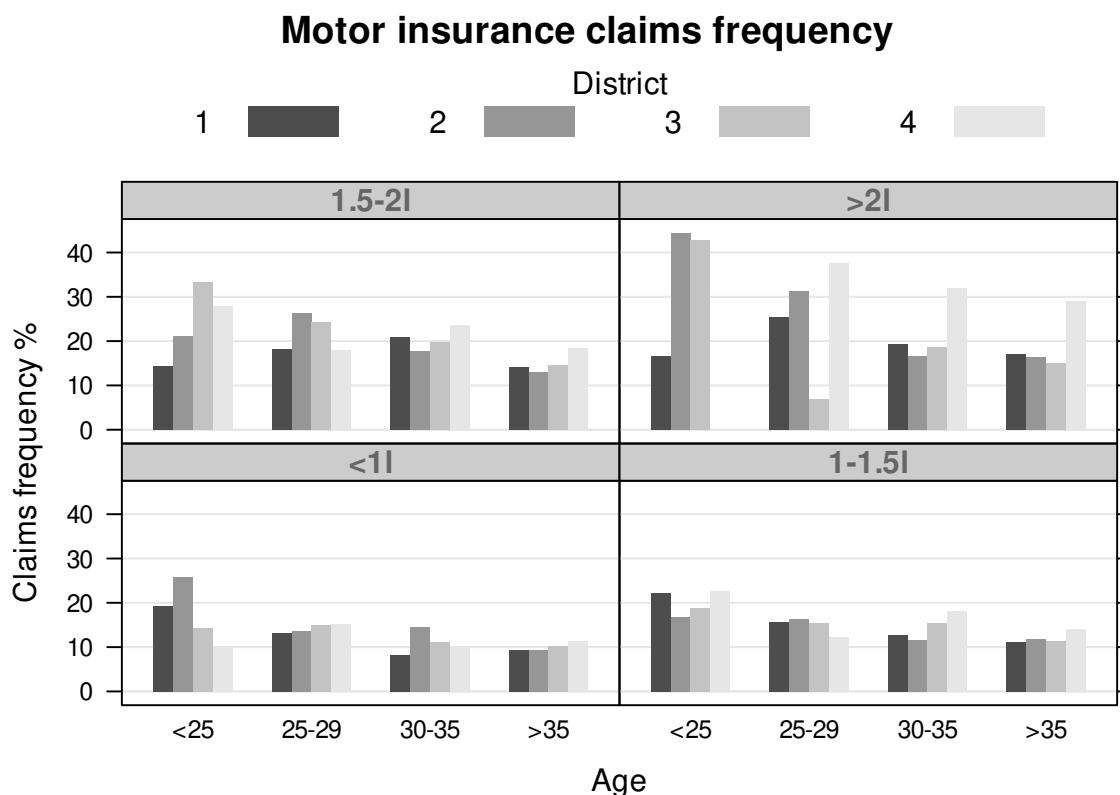
# 获取参数设置
trellis.par.get()

# 全局参数设置
trellis.par.set(my.settings)

library(MASS)
library(lattice)

```

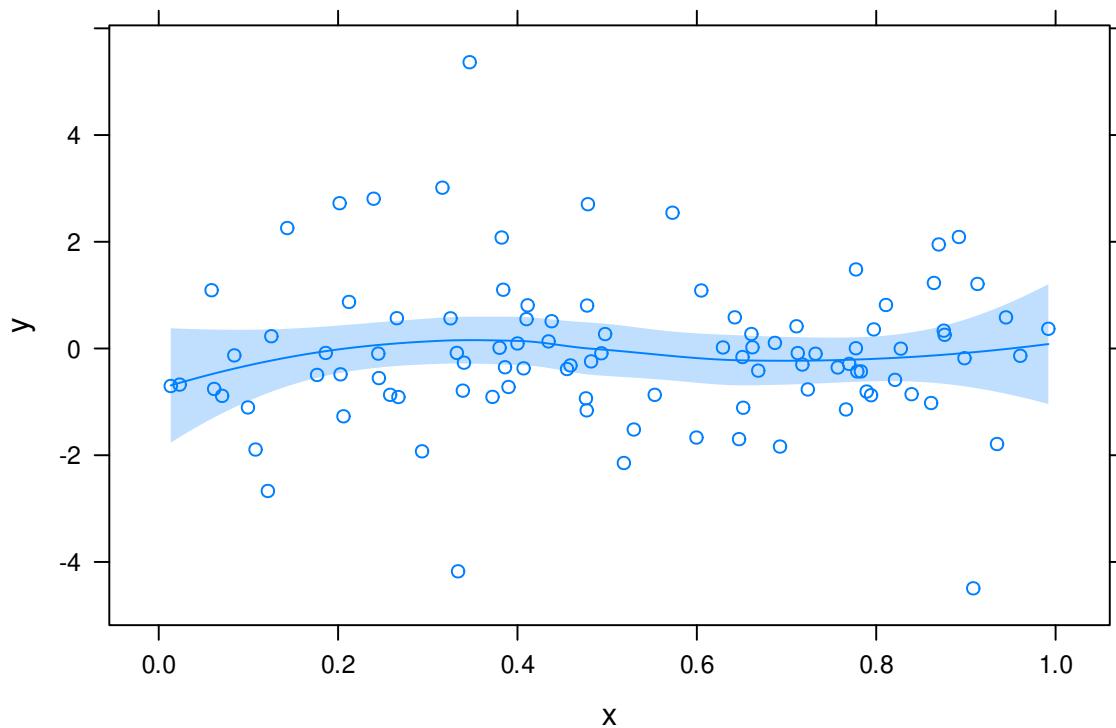
```
barchart(Claims / Holders * 100 ~ Age | Group,
  groups = District, data = Insurance,
  origin = 0, main = "Motor insurance claims frequency",
  xlab = "Age", ylab = "Claims frequency %",
  scales = list(alternating = 1),
  auto.key = list(
    space = "top", columns = 4,
    points = FALSE, rectangles = TRUE,
    title = "District", cex.title = 1
  ),
  par.settings = list(
    superpose.polygon = list(col = gray.colors(4), border = "transparent"),
    strip.background = list(col = "gray80"),
    strip.border = list(col = "black")
  ),
  par.strip.text = list(col = "gray40", font = 2),
  panel = function(x, y, ...) {
    panel.grid(h = -1, v = 0)
    panel.barchart(x, y, ...)
  }
)
```



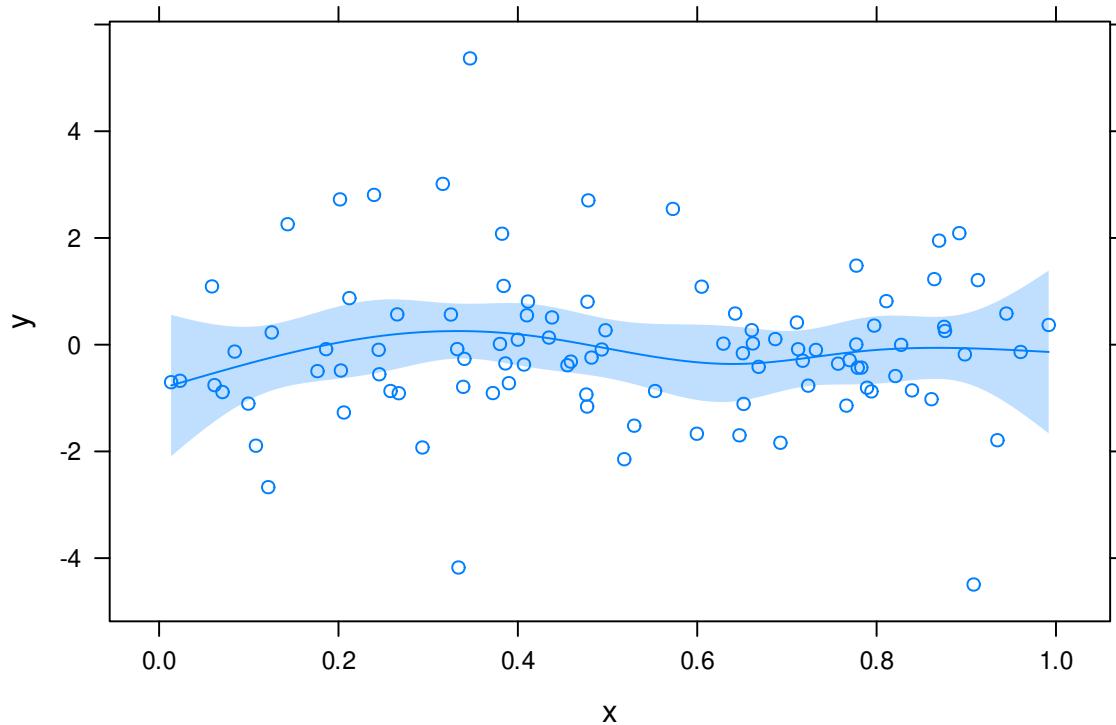
## 10.3.3 平滑图

```
set.seed(1)
xy <- data.frame(
  x = runif(100),
  y = rt(100, df = 5)
)

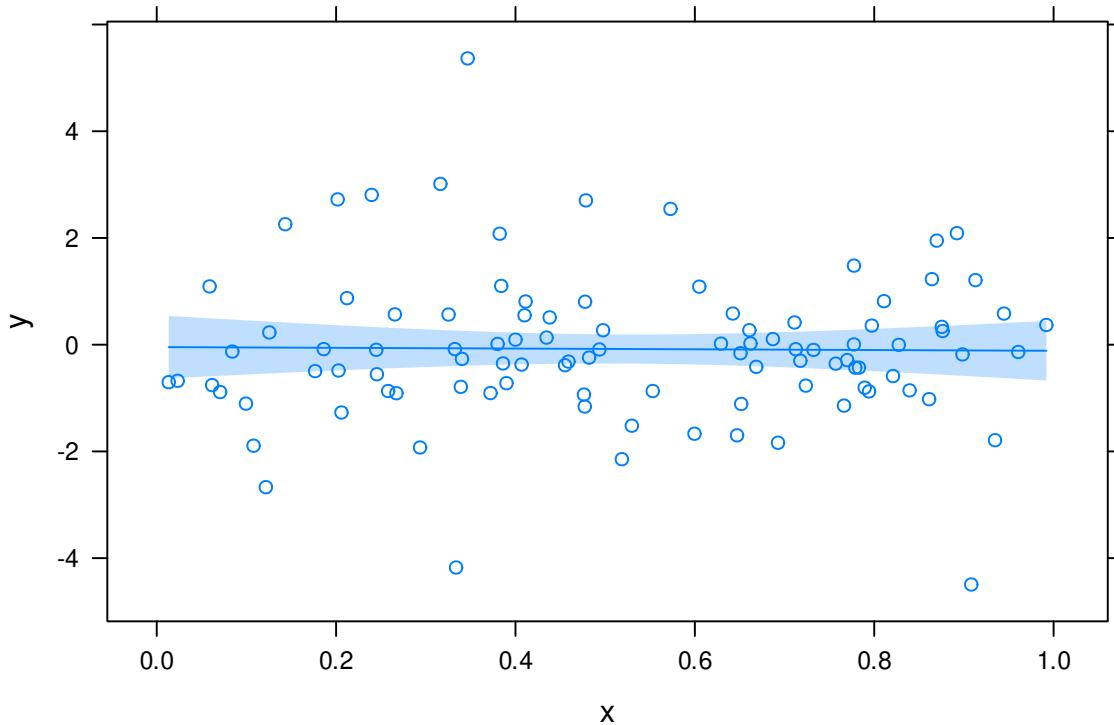
xyplot(y ~ x, xy, panel = function(...) {
  panel.xyplot(...)
  panel.smoother(..., span = 0.9)
})
```



```
library(splines)
xyplot(y ~ x, xy) +
  layer(panel.smoother(y ~ ns(x, 5), method = "lm"))
```

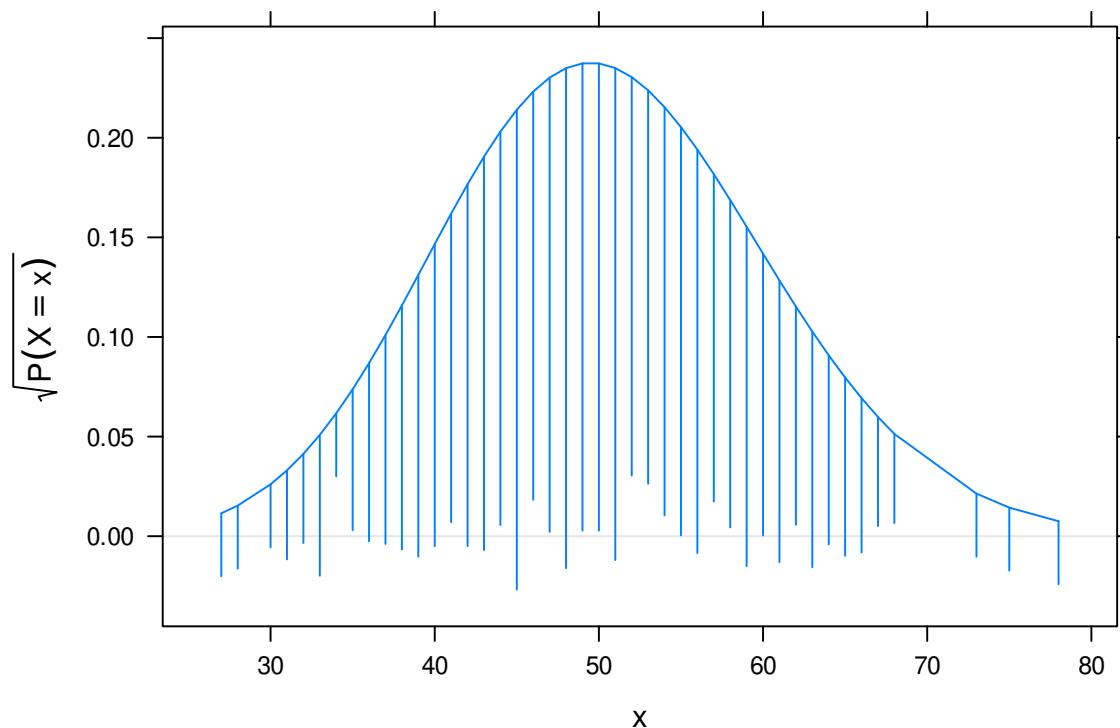


```
library(nlme)
library(mgcv)
xyplot(y ~ x, xy) +
  layer(panel.smoother(y ~ s(x), method = "gam"))
```



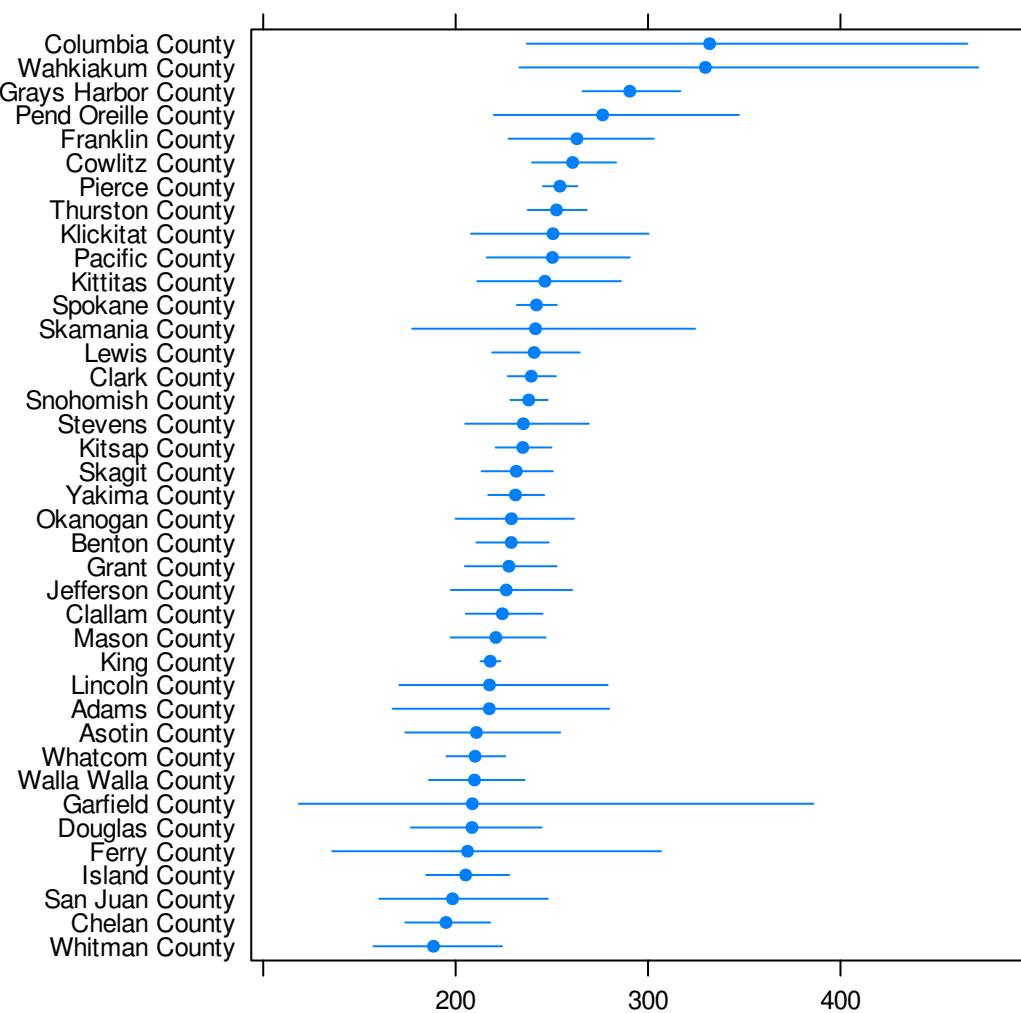
Trellis Displays of Tukey's Hanging Rootograms

```
x <- rpois(1000, lambda = 50)
rootogram(~x, dfun = function(x) dpois(x, lambda = 50))
```



#### 10.3.4 点图

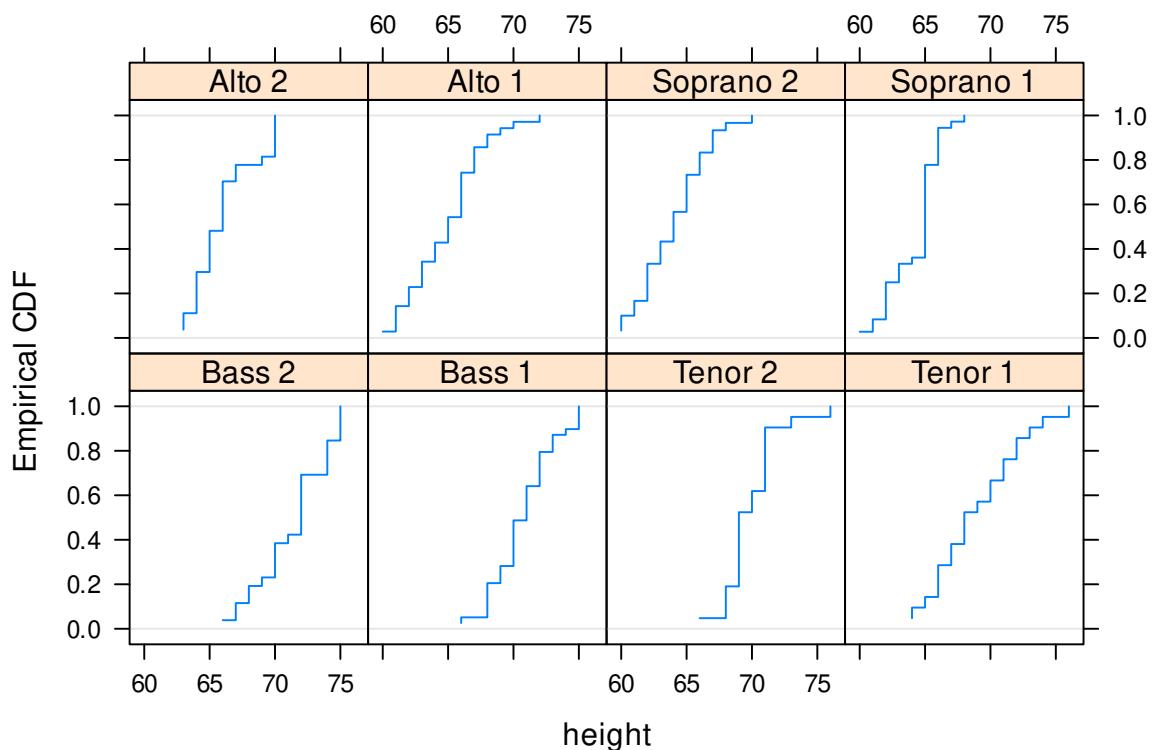
```
# 添加背景网格线作为参考线
segplot(reorder(factor(county), rate.male) ~ LCL95.male + UCL95.male,
  data = subset(USCancerRates, state == "Washington"),
  draw.bands = FALSE, centers = rate.male
)
```



### 10.3.5 阶梯图

经验累积分布图

```
ecdfplot(~height | voice.part, data = singer)
```

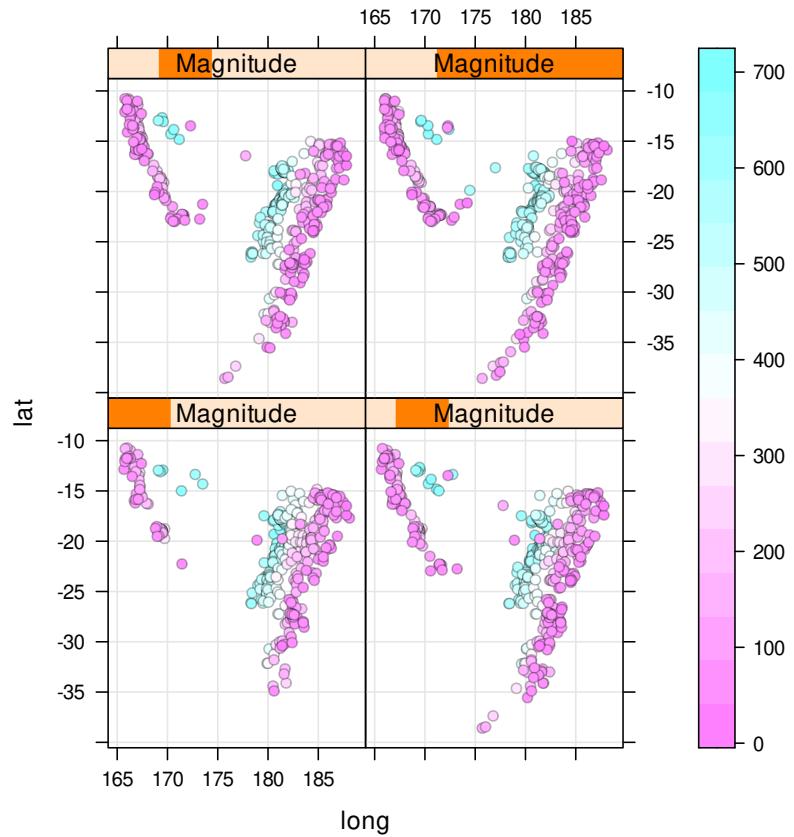


### 10.3.6 分面图

```
## a variant of Figure 5.6 from Sarkar (2008)
## http://lmdvr.r-forge.r-project.org/figures/figures.html?chapter=05;figure=05\_06

depth.ord <- rev(order(quakes$depth))
quakes$Magnitude <- equal.count(quakes$mag, 4)
quakes.ordered <- quakes[depth.ord, ]

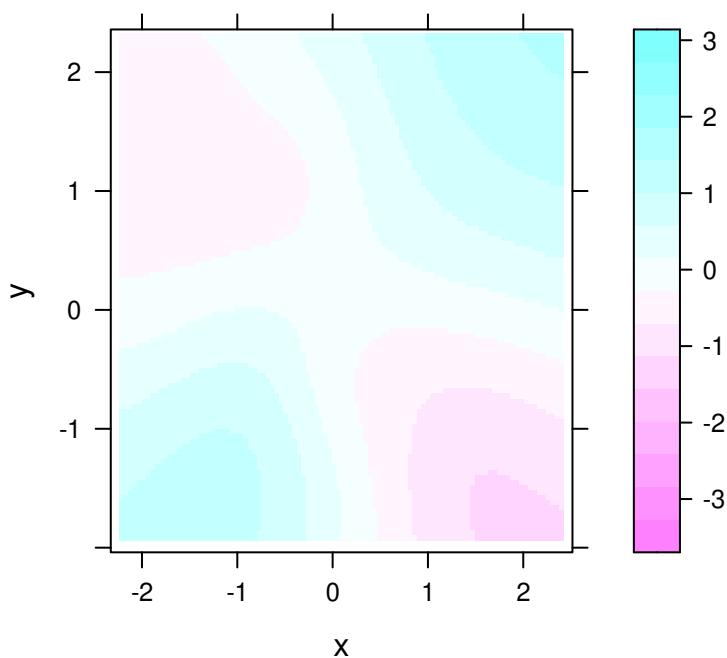
levelplot(depth ~ long + lat | Magnitude,
  data = quakes.ordered,
  panel = panel.levelplot.points, type = c("p", "g"),
  aspect = "iso", prepanel = prepanel.default.xyplot
)
```



### 10.3.7 等高线图

```
set.seed(1)
xyz <- data.frame(x = rnorm(100), y = rnorm(100))
xyz$z <- with(xyz, x * y + rnorm(100, sd = 1))

## GAM smoother with smoothness by cross validation
library(mgcv)
levelplot(z ~ x * y, xyz,
          panel = panel.2dsmoother,
          form = z ~ s(x, y), method = "gam")
```



### 10.3.8 透視图

```
library(shape)
persp(volcano,
  theta = 30, phi = 20,
  r = 50, d = 0.1, expand = 0.5, ltheta = 90, lphi = 180,
  shade = 0.1, ticktype = "detailed", nticks = 5, box = TRUE,
  col = drapecol(volcano, col = terrain.colors(100)),
  xlab = "X", ylab = "Y", zlab = "Z", border = "transparent",
  main = "Topographic Information \n on Auckland's Maunga Whau Volcano"
)
```

### 10.3.9 聚类图

```
xyplot(Sepal.Length ~ Petal.Length,
  groups = Species,
  data = iris, scales = "free",
  par.settings = list(
    superpose.symbol = list(pch = c(15:17)),
    superpose.line = list(lwd = 2, lty = 1:3)
  ),
  panel = function(x, y, ...) {
    panel.xyplot(x, y, ...)
```

### Topographic Information on Auckland's Maunga Whau Volcano

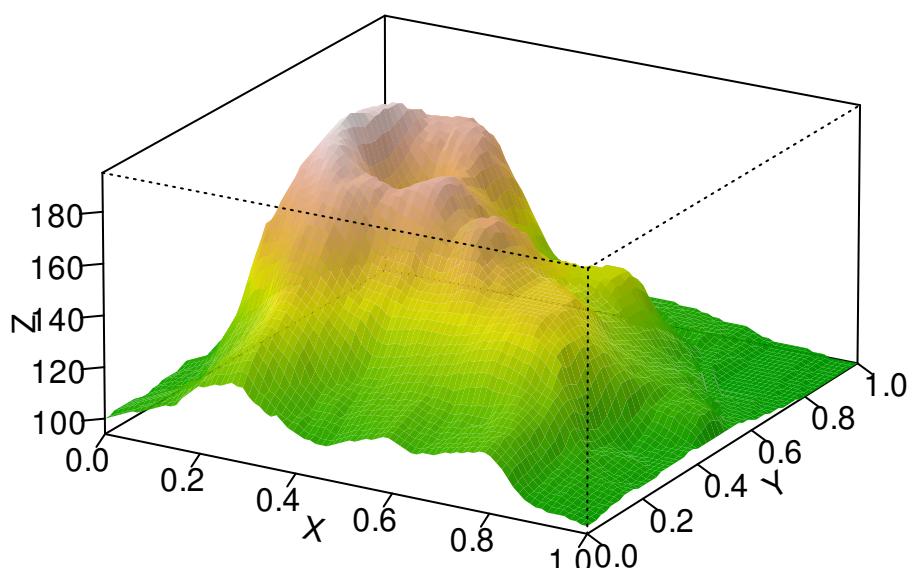
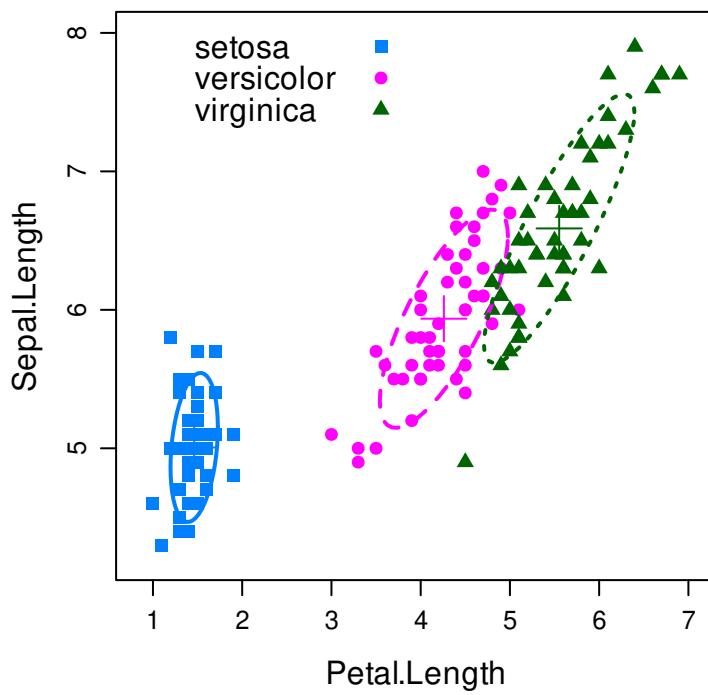


图 10.66: (ref:volcano-topo)

C



```
# lattice 书 6.3.1 节 参数曲面
```

```
kx <- function(u, v) cos(u) * (r + cos(u / 2))
ky <- function(u, v) {
  sin(u) * (r + cos(u / 2) * sin(t * v) -
    sin(u / 2) * sin(2 * t * v)) * sin(t * v) -
    sin(u / 2) * sin(2 * t * v)
}

kz <- function(u, v) sin(u / 2) * sin(t * v) + cos(u / 2) * sin(t * v)
n <- 50
u <- seq(0.3, 1.25, length = n) * 2 * pi
v <- seq(0, 1, length = n) * 2 * pi
um <- matrix(u, length(u), length(u))
vm <- matrix(v, length(v), length(v), byrow = TRUE)
r <- 2
t <- 1

wireframe(kz(um, vm) ~ kx(um, vm) + ky(um, vm),
  shade = TRUE, xlab = expression(x[1]),
```

```
ylab = expression(x[2]),
zlab = list(expression(italic(f) ~ group("(", list(x[1], x[2]), ")")), rot = 90),
screen = list(z = 170, x = -60),
alpha = 0.75, panel.aspect = 0.6, aspect = c(1, 0.4),
scales = list(arrows = FALSE, col = "black"),
lattice.options = list(
  layout.widths = list(
    left.padding = list(x = -.6, units = "inches"),
    right.padding = list(x = -1.0, units = "inches")
  ),
  layout.heights = list(
    bottom.padding = list(x = -.8, units = "inches"),
    top.padding = list(x = -1.0, units = "inches")
  )
),
par.settings = list(
  axis.line = list(col = "transparent")
)
)
```

## 10.4 运行环境

```
xfun::session_info()

## R version 4.2.0 (2022-04-22)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.4 LTS
##
## Locale:
##   LC_CTYPE=en_US.UTF-8          LC_NUMERIC=C
##   LC_TIME=en_US.UTF-8           LC_COLLATE=en_US.UTF-8
##   LC_MONETARY=en_US.UTF-8       LC_MESSAGES=en_US.UTF-8
##   LC_PAPER=en_US.UTF-8          LC_NAME=C
##   LC_ADDRESS=C                  LC_TELEPHONE=C
##   LC_MEASUREMENT=en_US.UTF-8   LC_IDENTIFICATION=C
##
## Package version:
##   base64enc_0.1.3    bookdown_0.26    bslib_0.3.1
##   cli_3.3.0          compiler_4.2.0    curl_4.3.2
##   digest_0.6.29      evaluate_0.15    fastmap_1.1.0
##   fs_1.5.2           glue_1.6.2      graphics_4.2.0
##   grDevices_4.2.0    grid_4.2.0      highr_0.9
##   htmltools_0.5.2    jpeg_0.1-9     jquerylib_0.1.4
##   jsonlite_1.8.0     KernSmooth_2.23-20  knitr_1.39
```



```
## lattice_0.20-45      latticeExtra_0.6-29 magrittr_2.0.3
## mapproj_1.2.8        maps_3.4.0       MASS_7.3-57
## Matrix_1.4-1         methods_4.2.0    mgcv_1.8-40
## nlme_3.1-157         png_0.1-7       R6_2.5.1
## rappdirs_0.3.3       RColorBrewer_1.1-3 rlang_1.0.2
## rmarkdown_2.14         sass_0.4.1      shape_1.4.6
## splines_4.2.0         stats_4.2.0     stringi_1.7.6
## stringr_1.4.0         survival_3.3-1  sysfonts_0.8.8
## tinytex_0.39          tools_4.2.0     utils_4.2.0
## xfun_0.31             yaml_2.3.5
```

## 第十一章 数据可视化

```
library(ggplot2)          # ggplot2 图形
library(patchwork)        # 图形布局
library(magrittr)         # 管道操作
library(ggrepel)          # 文本注释
library(extrafont)         # 加载外部字体 TTF
library(hrbrthemes)        # 主题
library(maps)              # 地图数据
library(mapdata)           # 地图数据
library(xkcd)              # 漫画字体
library(RgoogleMaps)       # 静态地图
library(data.table)         # 数据操作
library(KernSmooth)        # 核平滑
library(ggnormalviolin)     # 提琴图
library(ggbeeswarm)         # 蜂群图
library(gert)                # Git 数据操作
library(ggridges)           # 岭线图
library(ggpubr)              # 组合图
library(treemap)             # 树状图
library(treemapify)          # 树状图
library(ggalluvial)          # 桑基图
library(ggquiver)             # 向量场图
library(ggmosaic)            # 马赛克图
library(ggbump)                # 凹凸图
library(ggstream)             # 水流图
library(timelines)            # 时间线
library(ggdendro)             # 聚类图
library(ggfortify)            # 统计分析结果可视化：主成分图
library(gganimate)             # 动态图
```

David Robinson 给出为何使用 ggplot2<sup>1</sup> 当然也有 Jeff Leek 指出在某些重要场合不适合 ggplot2<sup>2</sup> 并且给出强有力的证据，其实不管怎么样，适合自己的才是好的。也不枉费 Garrick Aden-Buie 花费 160 页幻灯片逐步分解介绍 优雅的 ggplot2，Malcolm Barrett 也介绍了 ggplot2 基础用法，还有 Selva Prabhakaran 精心总结给出了 50 个 ggplot2 数据可视化的 例子 以及 Victor Perrier 为小白用 ggplot2 操碎了心地开发

<sup>1</sup><http://varianceexplained.org/r/why-I-use-ggplot2/>

<sup>2</sup><https://simplystatistics.org/2016/02/11/why-i-dont-use-ggplot2/>

RStudio 插件 [esquisse](#) 包，Claus O. Wilke 教你一步步创建出版级的图形 [https://github.com/clauswilke/practical\\_ggplot2](https://github.com/clauswilke/practical_ggplot2)。

ggplot2 是十分方便的统计作图工具，相比 Base R，为了一张出版级的图形，不需要去调整每个参数，实现快速出图。集成了很多其它统计计算的 R 包，支持丰富的统计分析和计算功能，如回归、平滑等，实现了作图和模型的无缝连接。比如图11.1，使用 loess 局部多项式平滑得到数据的趋势，不仅仅是散点图，代码量也非常少。

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(color = class)) +
  geom_smooth(se = TRUE, method = "loess") +
  labs(
    title = "Fuel efficiency generally decreases with engine size",
    subtitle = "Two seaters (sports cars) are an exception because of their light weight",
    caption = "Data from fueleconomy.gov"
  )
```

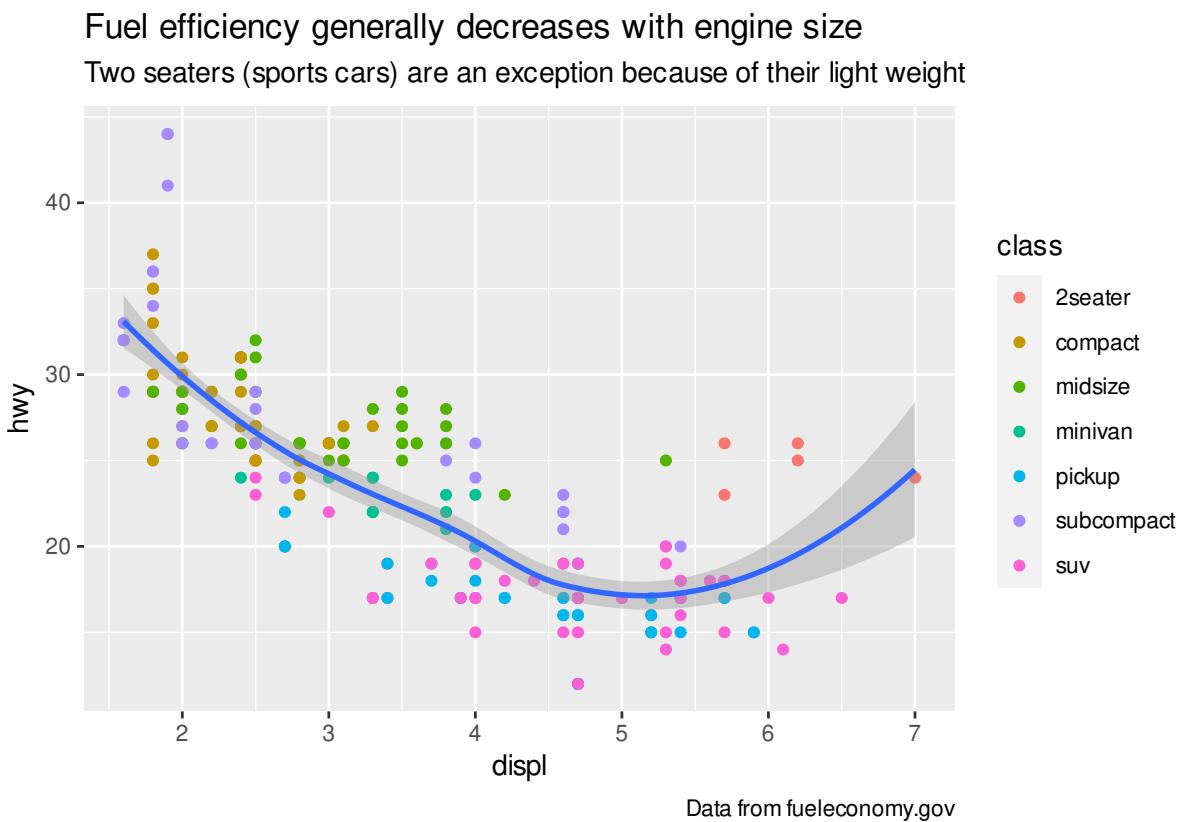


图 11.1: 简洁美观

故事源于一幅图片，我不记得第一次见到这幅图是什么时候了，只因多次在多个场合中见过，所以留下了深刻的印象，后来才知道它出自于一篇博文 — [Using R packages and education to scale Data Science at Airbnb](#)，作者 Ricardo Bion 还在其 [Github](#) 上传了相关代码<sup>3</sup>。除此之外还有几篇重要的参考资料：

- ## 1. Pablo Barberá 的 Data Visualization with R and ggplot2

<sup>3</sup>[https://github.com/ricardo-bion/medium\\_visualization](https://github.com/ricardo-bion/medium_visualization)

2. Matt Leonawicz 的新作 [mapmate](#), 可以去其主页欣赏系列作品<sup>4</sup>
3. [tidytuesday](#) 可视化挑战官方项目 还有 [tidytuesday](#)
4. [ggstatsplot](#) 可视化统计检验、模型的结果
5. [ggpubr](#) 制作出出版级统计图形
6. Thomas Lin Pedersen [Drawing Anything with ggplot2](#)
7. [Designing ggplots: making clear figures that communicate](#)
8. [ggh4x](#) 提供 ggplot2 的额外定制功能
9. [ggdist](#) Visualizations of distributions and uncertainty
10. [gghighlight](#)
11. [ggnetwork](#)
12. [ggPMX](#) 'ggplot2' Based Tool to Facilitate Diagnostic Plots for NLME Models
13. [ggpp](#) ggpp: Grammar Extensions to 'ggplot2'

如 Berton Gunter 所说, 数据可视化只是一种手段, 根据数据实际情况作展示才是重要的, 并不是要追求酷炫。

3-D bar plots are an abomination. Just because Excel can do them doesn't mean you should.  
(Dismount pulpit).

— Berton Gunter<sup>5</sup>

[grid](#) 是 [lattice](#) 和 [ggplot2](#) 的基础, [gganimate](#) 是 [ggplot2](#) 一个扩展, 它将静态图形视为帧, 调用第三方工具合成 GIF 动图或 MP4 视频等, 要想深入了解 [ggplot2](#), 可以去看 [Hadley Wickham](#), [Danielle Navarro](#), and [Thomas Lin Pedersen](#) 合著的《[ggplot2: elegant graphics for data analysis](#)》第三版 <https://ggplot2-book.org/>。

## 11.1 元素

以数据集 [airquality](#) 为例介绍 GGplot2 图层、主题、配色、坐标、尺度、注释和组合等

### 11.1.1 图层

```
ls("package:ggplot2", pattern = "geom_")

## [1] "geom_abline"           "geom_area"           "geom_bar"
## [4] "geom_bin_2d"           "geom_bin2d"          "geom_blank"
## [7] "geom_boxplot"          "geom_col"            "geom_contour"
## [10] "geom_contour_filled"   "geom_count"          "geom_crossbar"
## [13] "geom_curve"            "geom_density"        "geom_density_2d"
## [16] "geom_density_2d_filled" "geom_density2d"      "geom_density2d_filled"
## [19] "geom_dotplot"          "geom_errorbar"       "geom_errorbarh"
## [22] "geom_freqpoly"         "geom_function"       "geom_hex"
## [25] "geom_histogram"        "geom_hline"          "geom_jitter"
## [28] "geom_label"            "geom_line"           "geom_linerange"
```

<sup>4</sup><https://leonawicz.github.io/>

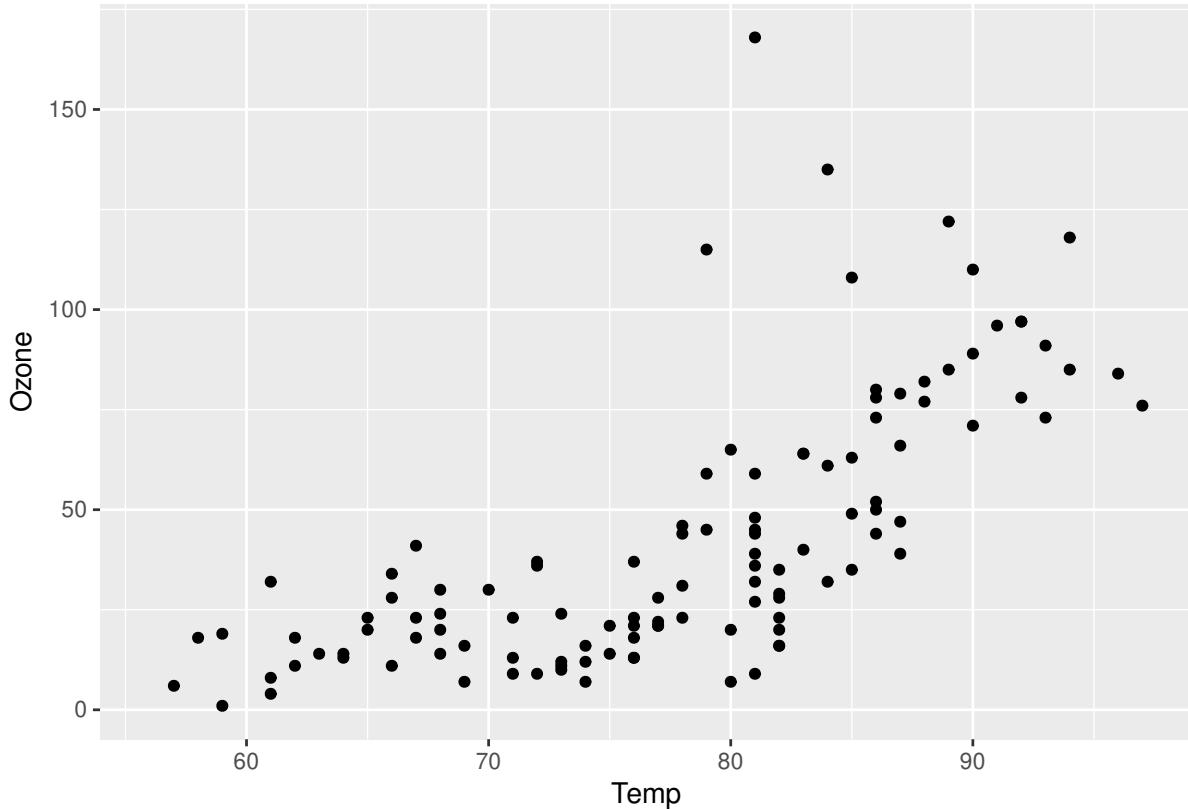
<sup>5</sup><https://stat.ethz.ch/pipermail/r-help/2007-October/142420.html>



```
## [31] "geom_map"  
## [34] "geom_pointrange"  
## [37] "geom_qq_line"  
## [40] "geom_rect"  
## [43] "geom_segment"  
## [46] "geom_sf_text"  
## [49] "geom_step"  
## [52] "geom_violin"  
"geom_path"  
"geom_polygon"  
"geom_quantile"  
"geom_ribbon"  
"geom_sf"  
"geom_smooth"  
"geom_text"  
"geom_vline"  
"geom_point"  
"geom_qq"  
"geom_raster"  
"geom_rug"  
"geom_sf_label"  
"geom_spoke"  
"geom_tile"
```

生成一个散点图

```
ggplot(airquality, aes(x = Temp, y = Ozone)) + geom_point()  
  
## Warning: Removed 37 rows containing missing values (geom_point).
```



### 11.1.2 标签

图形的标签分为横纵轴标签、刻度标签、主标题、副标题等

```
data.frame(  
  dates = seq.Date(  
    from = as.Date("1945-01-01"),  
    to = as.Date("1974-12-31"),  
    by = "quarter"  
)
```

```
presidents = as.vector(presidents)
) |>
  ggplot(aes(x = dates, y = presidents)) +
  geom_line(color = "slategray", na.rm = TRUE) +
  geom_point(size = 1.5, color = "darkslategray", na.rm = TRUE) +
  scale_x_date(date_breaks = "4 year", date_labels = "%Y") +
  labs(
    title = "1945年至1974年美国总统每季度支持率",
    x = "年份", y = "支持率 (%)",
    caption = "数据源: R 包 datasets"
  ) +
  theme_minimal(base_size = 10.54, base_family = "Noto Serif CJK SC")
```

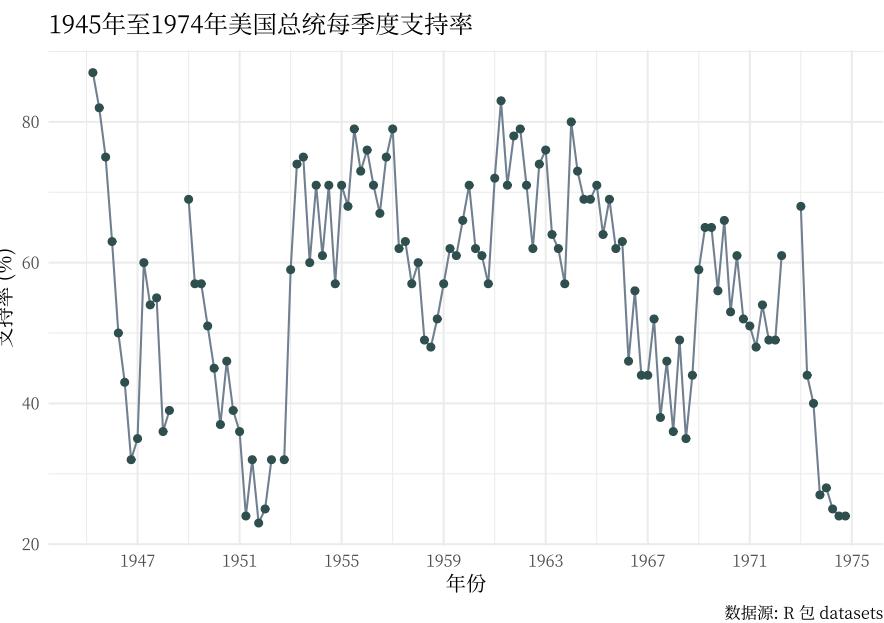


图 11.2: 自 1945 年第一季度至 1974 年第四季度美国总统的支持

### 11.1.3 注释

图中注释的作用在于高亮指出关键点，提请读者注意。文本注释可由 `ggrepel` 包提供的标签图层 `geom_label_repel()` 添加，标签数据可独立于之前的数据层，标签所在的位置可以通过参数 `direction` 和 `nudge_y` 精调，图 11.3 模拟了一组数据。

```
set.seed(2020)
library(ggrepel)
dat <- data.frame(
  x = seq(100),
  y = cumsum(rnorm(100))
)
anno_data <- dat |>
  subset(x %% 25 == 10) |>
```



```
transform(text = "text")

ggplot(data = dat, aes(x, y)) +
  geom_line() +
  geom_label_repel(aes(label = text),
    data = anno_data,
    direction = "y",
    nudge_y = c(-5, 5, 5, 5)
  ) +
  theme_minimal()
```

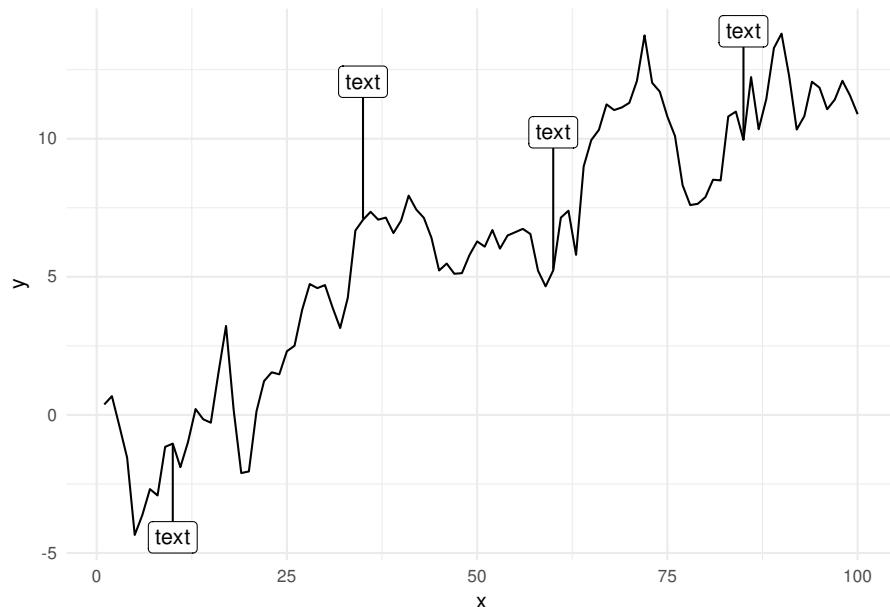


图 11.3: 文本注释

**ggrepel** 包的图层 `geom_text_repel()` 支持所有数据点的注释，并且自动调整文本的位置，防止重叠，增加辨识度，如图 11.4。当然，数据点如果过于密集也不适合全部注释，高亮其中的关键点即可。

```
mtcars |>
  transform(cyl = as.factor(cyl)) |>
  ggplot(aes(wt, mpg, label = rownames(mtcars), color = cyl)) +
  geom_point() +
  geom_text_repel(max.overlaps = 12) +
  theme_minimal()
```

Claus Wilke 开发的 `ggtext` 包支持更加丰富的注释样式，详见网站 <https://wilkelab.org/ggtext/>

```
ls("package:ggplot2", pattern = "annotation_")
```

```
## [1] "annotation_custom"    "annotation_logticks" "annotation_map"
## [4] "annotation_raster"
ggplot(airquality, aes(x = Temp, y = Ozone)) +
  geom_point(na.rm = TRUE)
```

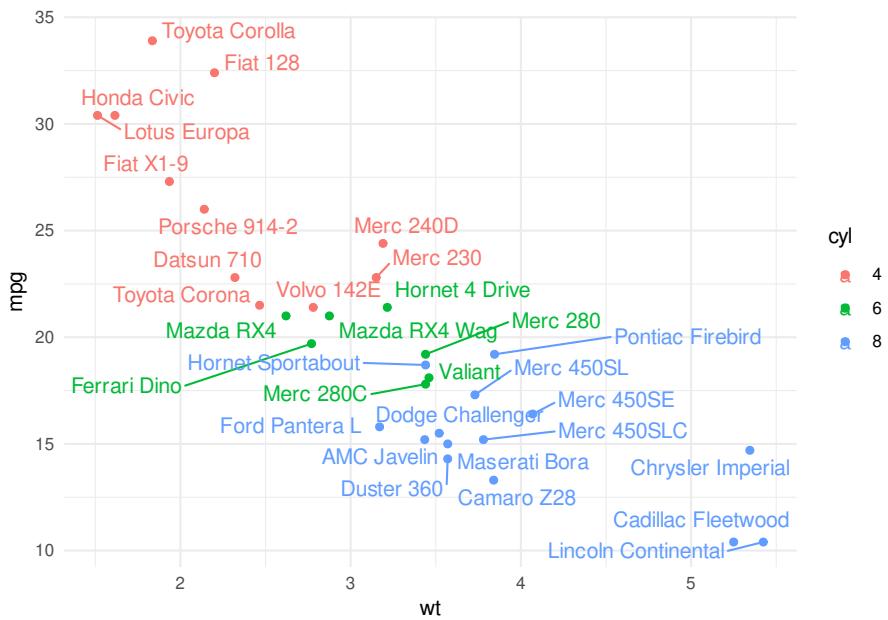
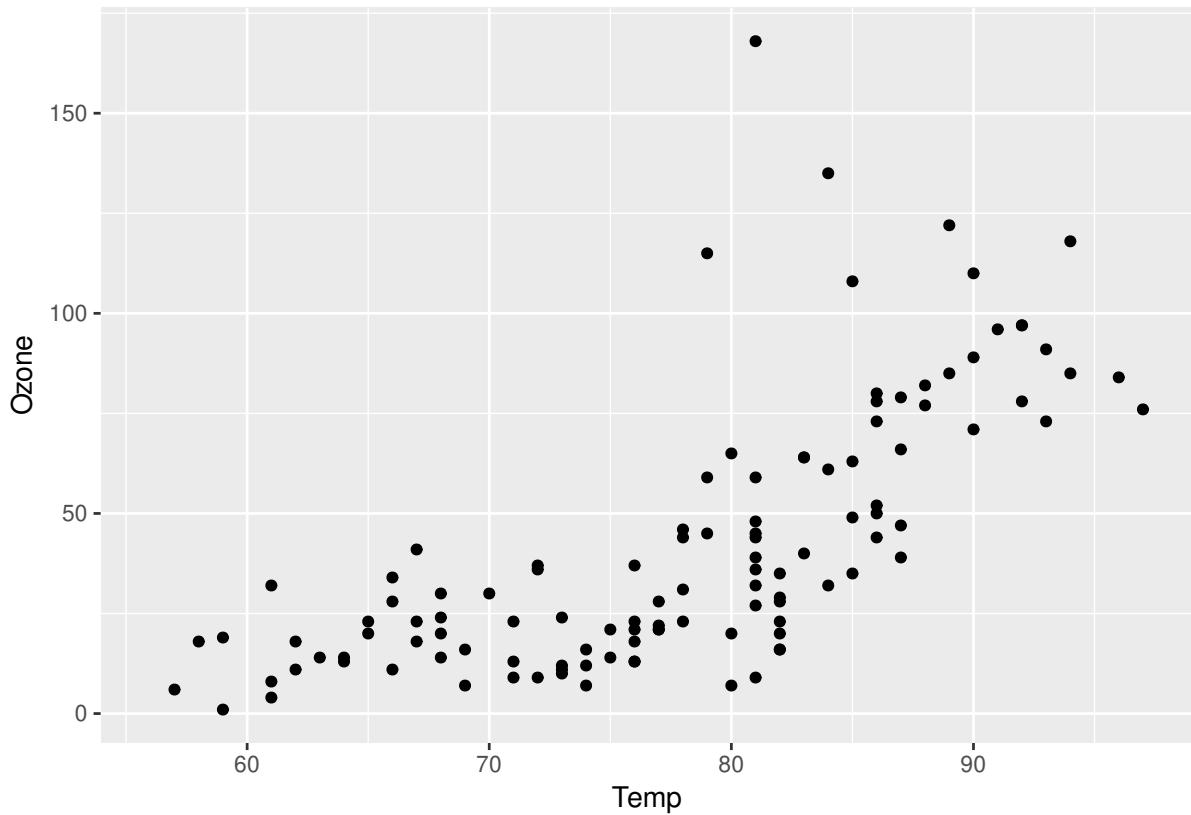
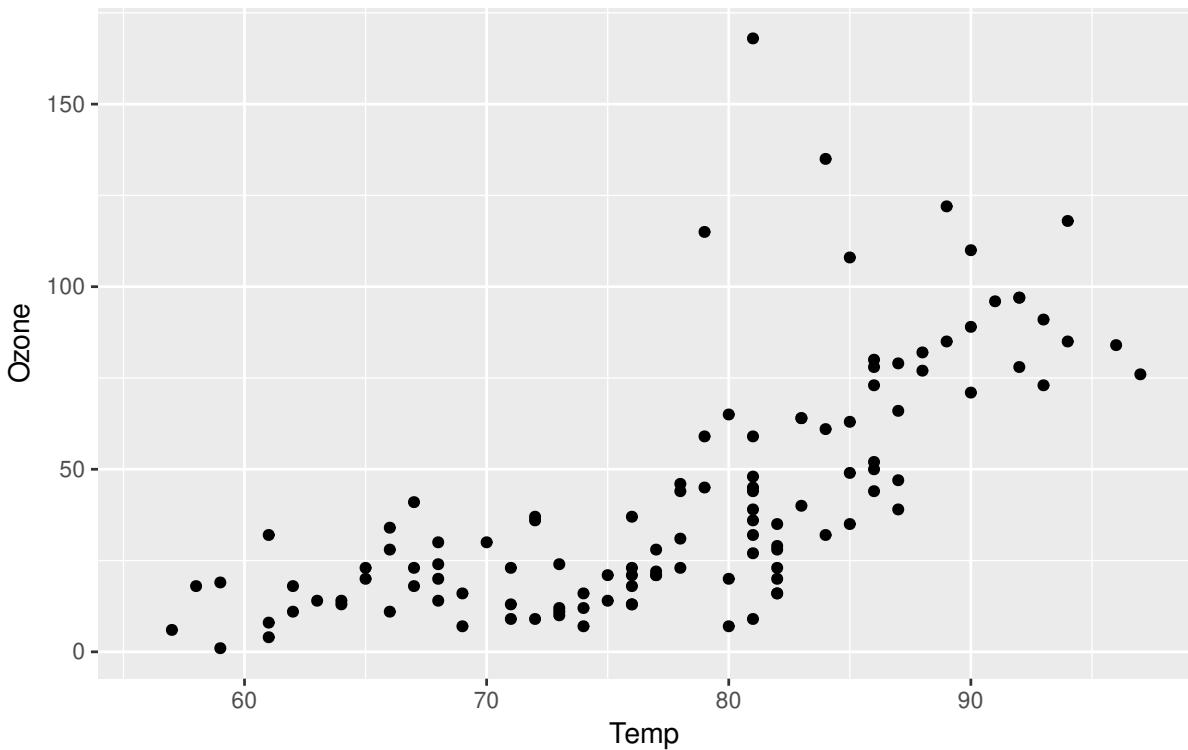


图 11.4: 少量点的情况下可以全部注释, 且可以解决注释重叠的问题



```
ggplot(airquality, aes(x = Temp, y = Ozone)) +
  geom_point(na.rm = TRUE) +
  labs(title = substitute(paste(d *
    bolditalic(x)[italic(t)] == alpha * (theta - bolditalic(x)[italic(t)]) *
    d * italic(t) + lambda * d * italic(B)[italic(t)]), list(lambda = 4)))
```

$$d\mathbf{x}_t = \alpha(\theta - \mathbf{x}_t)dt + 4dB_t$$



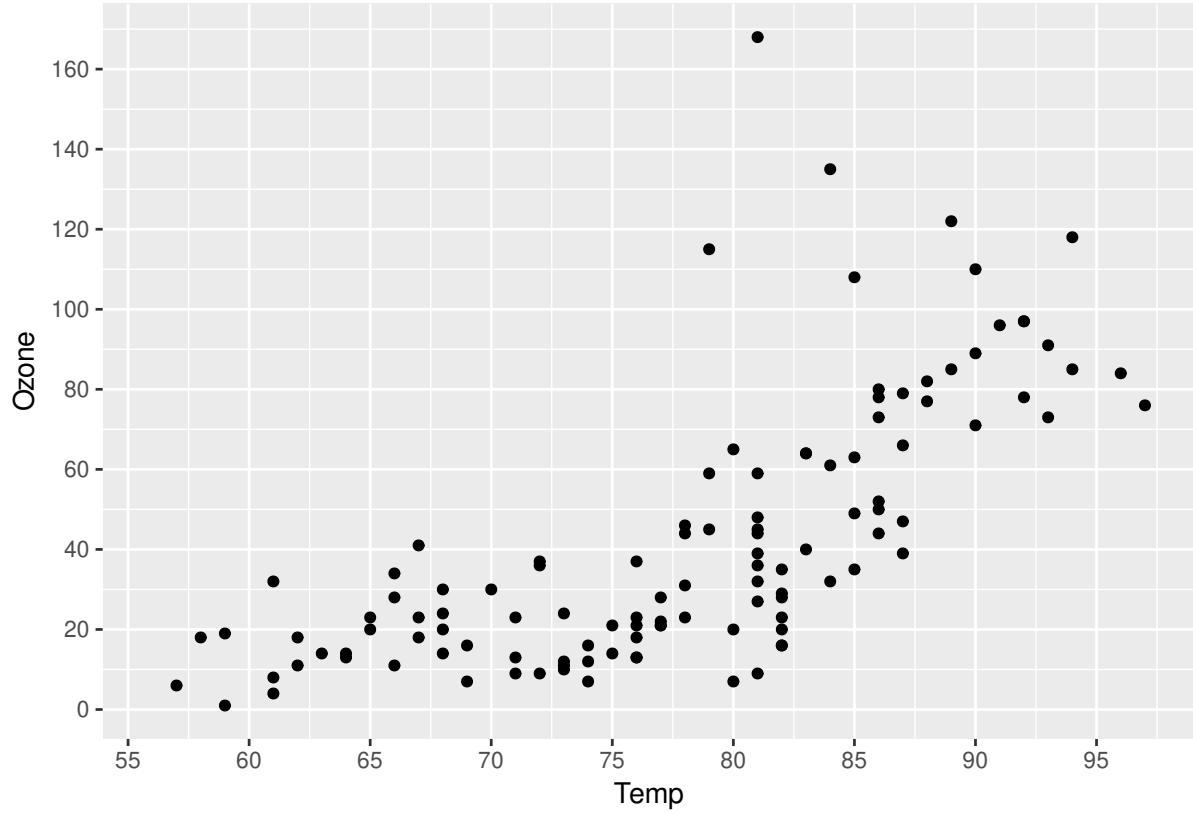
[geomtextpath](#) 曲线上的文字随曲线弯曲变化

[ggsave](#) 曲线上散点以图片、彩色图标表示

#### 11.1.4 刻度

```
ls("package:ggplot2", pattern = "^scale_(x|y)_")  
  
## [1] "scale_x_binned"      "scale_x_continuous" "scale_x_date"  
## [4] "scale_x_datetime"    "scale_x_discrete"   "scale_x_log10"  
## [7] "scale_x_reverse"     "scale_x_sqrt"      "scale_x_time"  
## [10] "scale_y_binned"      "scale_y_continuous" "scale_y_date"  
## [13] "scale_y_datetime"    "scale_y_discrete"   "scale_y_log10"  
## [16] "scale_y_reverse"     "scale_y_sqrt"      "scale_y_time"  
  
range(airquality$Temp, na.rm = TRUE)  
  
## [1] 56 97  
range(airquality$Ozone, na.rm = TRUE)  
  
## [1] 1 168  
ggplot(airquality, aes(x = Temp, y = Ozone)) +  
  geom_point(na.rm = TRUE) +  
  scale_x_continuous(breaks = seq(50, 100, 5)) +
```

```
scale_y_continuous(breaks = seq(0, 200, 20))
```



### 11.1.5 图例

二维的图例 `biscale` 和 `multiscales` 和 `ggnewscale`

### 11.1.6 坐标系

极坐标, 直角坐标

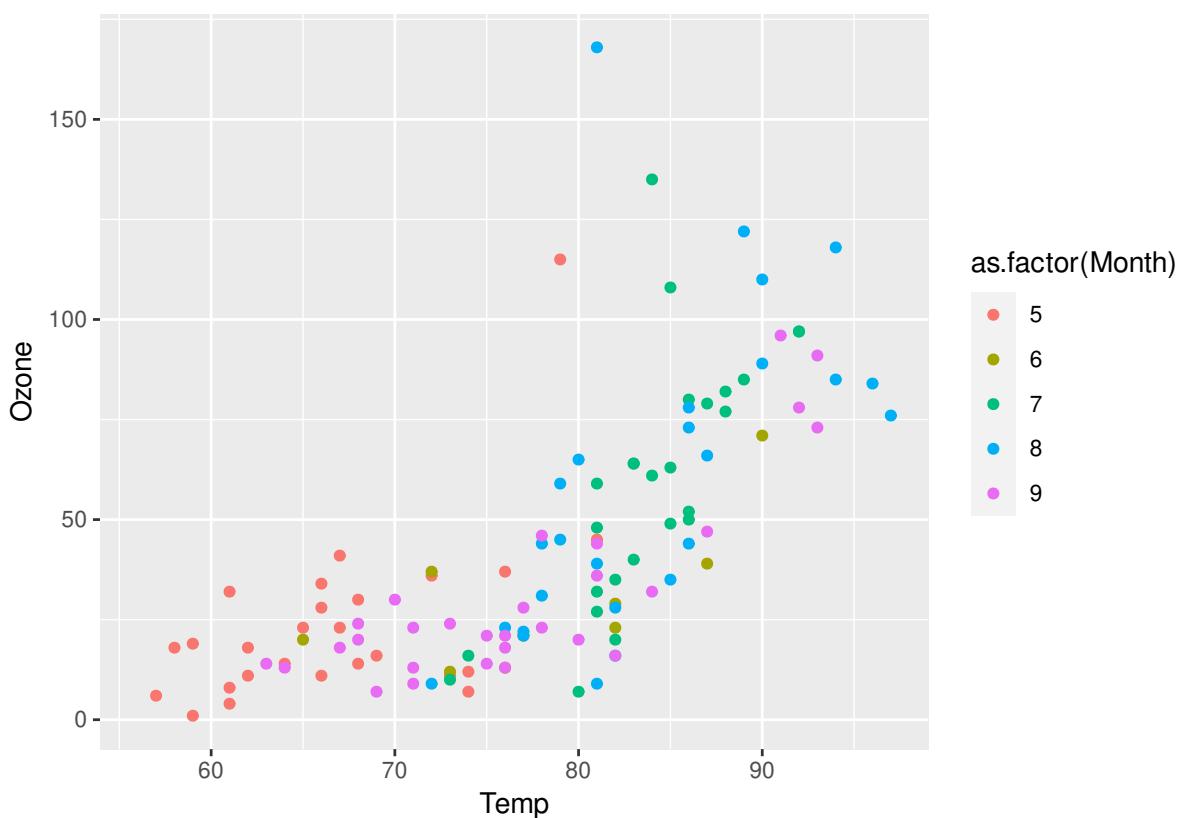
```
ls("package:ggplot2", pattern = "coord_")  
  
## [1] "coord_cartesian" "coord_equal"      "coord_fixed"      "coord_flip"  
## [5] "coord_map"        "coord_munch"      "coord_polar"      "coord_quickmap"  
## [9] "coord_sf"         "coord_trans"
```

### 11.1.7 坐标轴

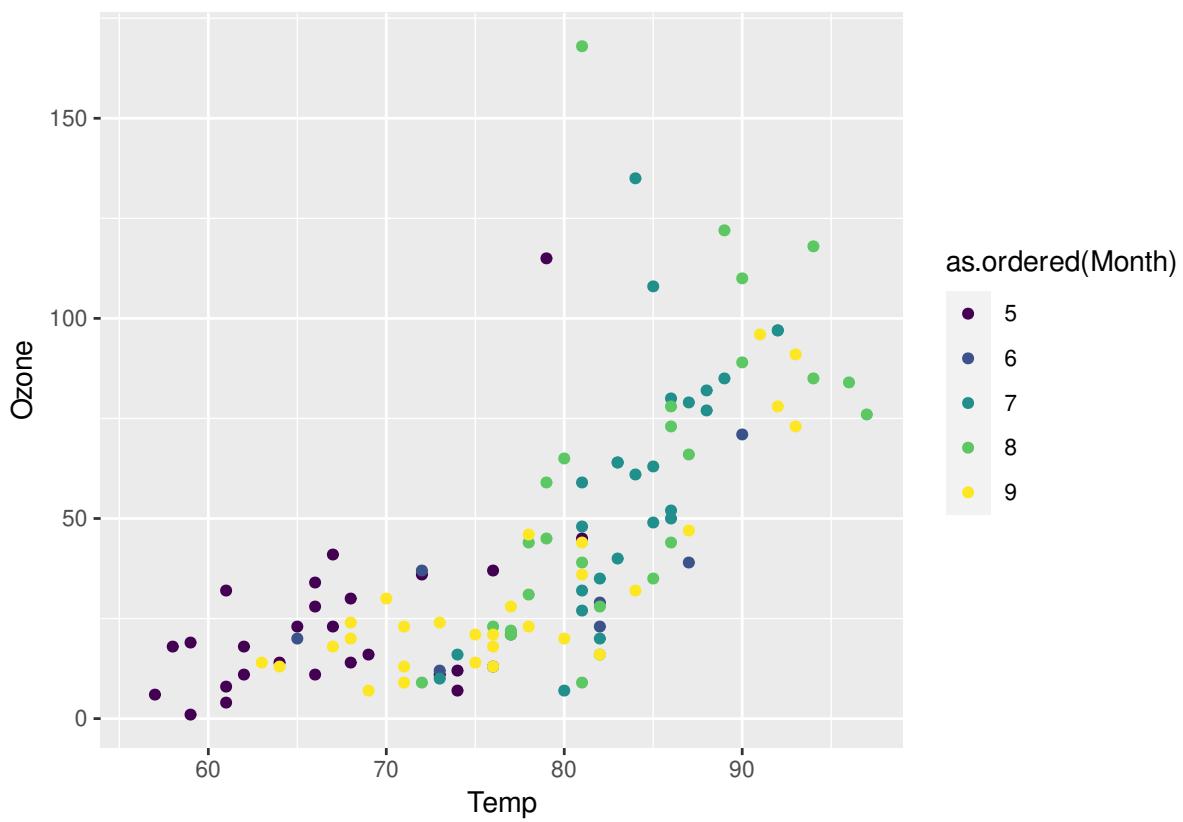
坐标轴标签位置、大小、字体

## 11.1.8 配色

```
ls("package:ggplot2", pattern = "^\$scale_(color|fill)_\$")  
  
## [1] "scale_color_binned"      "scale_color_brewer"      "scale_color_continuous"  
## [4] "scale_color_date"        "scale_color_datetime"  "scale_color_discrete"  
## [7] "scale_color_distiller"    "scale_color_fermenter"  "scale_color_gradient"  
## [10] "scale_color_gradient2"   "scale_color_gradientn" "scale_color_grey"  
## [13] "scale_color_hue"        "scale_color_identity"  "scale_color_manual"  
## [16] "scale_color_ordinal"    "scale_color_steps"     "scale_color_steps2"  
## [19] "scale_color_stepsn"    "scale_color_viridis_b" "scale_color_viridis_c"  
## [22] "scale_color_viridis_d"  "scale_fill_binned"     "scale_fill_brewer"  
## [25] "scale_fill_continuous" "scale_fill_date"       "scale_fill_datetime"  
## [28] "scale_fill_discrete"    "scale_fill_distiller"  "scale_fill_fermenter"  
## [31] "scale_fill_gradient"    "scale_fill_gradient2" "scale_fill_gradientn"  
## [34] "scale_fill_grey"        "scale_fill_hue"       "scale_fill_identity"  
## [37] "scale_fill_manual"      "scale_fill_ordinal"   "scale_fill_steps"  
## [40] "scale_fill_steps2"      "scale_fill_stepsn"   "scale_fill_viridis_b"  
## [43] "scale_fill_viridis_c"   "scale_fill_viridis_d"  
  
ggplot(airquality, aes(x = Temp, y = Ozone, color = as.factor(Month))) +  
  geom_point(na.rm = TRUE)
```



```
ggplot(airquality, aes(x = Temp, y = Ozone, color = as.ordered(Month))) +
  geom_point(na.rm = TRUE)
```



### 11.1.9 主題

ggcharts 和 bbplot prettyB 美化 Base R 图形 ggprism

```
ls("package:ggplot2", pattern = "^.theme_")
```

```
## [1] "theme_bw"         "theme_classic"    "theme_dark"       "theme_get"
## [5] "theme_gray"       "theme_grey"       "theme_light"      "theme_linedraw"
## [9] "theme_minimal"    "theme_replace"    "theme_set"        "theme_test"
## [13] "theme_update"     "theme_void"
```

这里只展示 theme\_bw() theme\_void() theme\_minimal() 和 theme\_void() 等四个常见主题，更多主题参考 [ggsci](#)、[ggthemes](#)、[ggtech](#)、[hrbrthemes](#)、[clcharts](#) 和 [ggthemr](#) 包

```
ggplot(airquality, aes(x = Temp, y = Ozone), na.rm = TRUE) +
  geom_point() +
  theme_bw()
```

```
## Warning: Removed 37 rows containing missing values (geom_point).
```

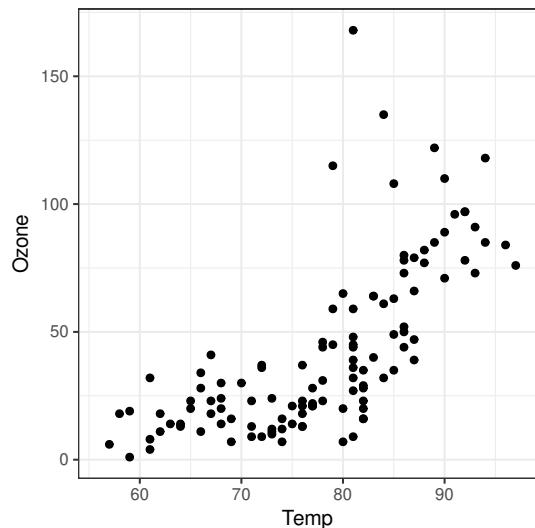
```
ggplot(airquality, aes(x = Temp, y = Ozone), na.rm = TRUE) +
  geom_point() +
  theme_void()
```

```
## Warning: Removed 37 rows containing missing values (geom_point).  
ggplot(airquality, aes(x = Temp, y = Ozone), na.rm = TRUE) +  
  geom_point() +  
  theme_minimal()
```

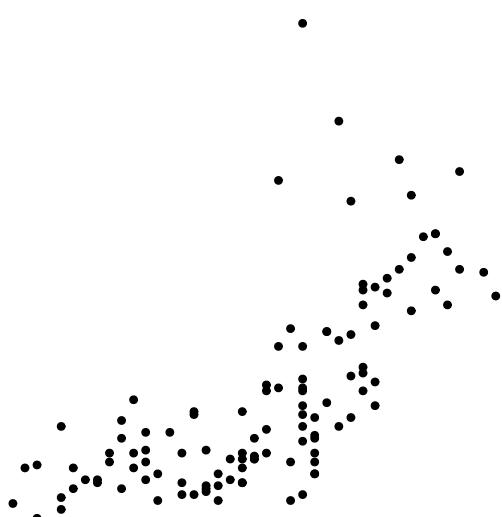
②

```
## Warning: Removed 37 rows containing missing values (geom_point).  
ggplot(airquality, aes(x = Temp, y = Ozone), na.rm = TRUE) +  
  geom_point() +  
  theme_classic()
```

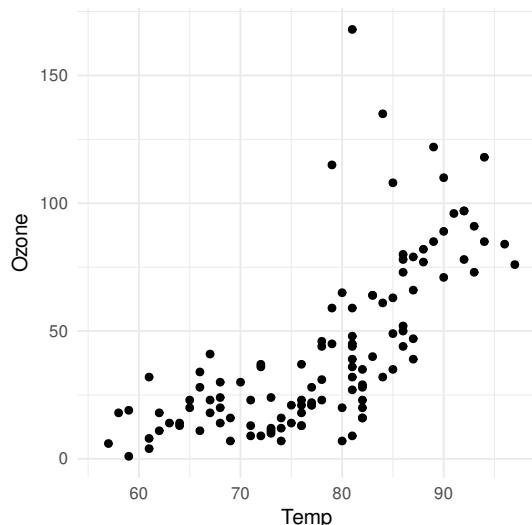
```
## Warning: Removed 37 rows containing missing values (geom_point).
```



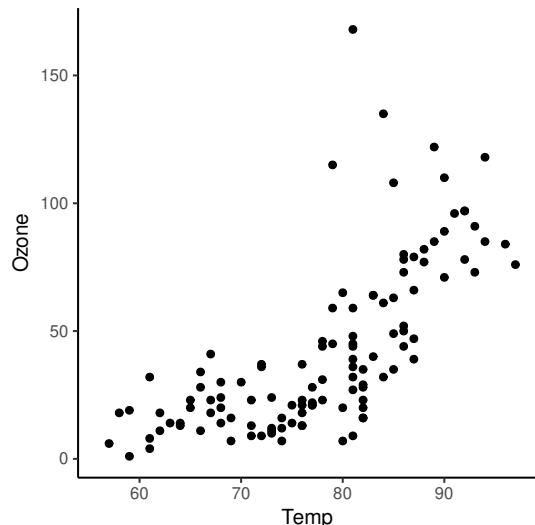
(a) 黑白主题



(b) 无主题



(c) 极少配置的主题



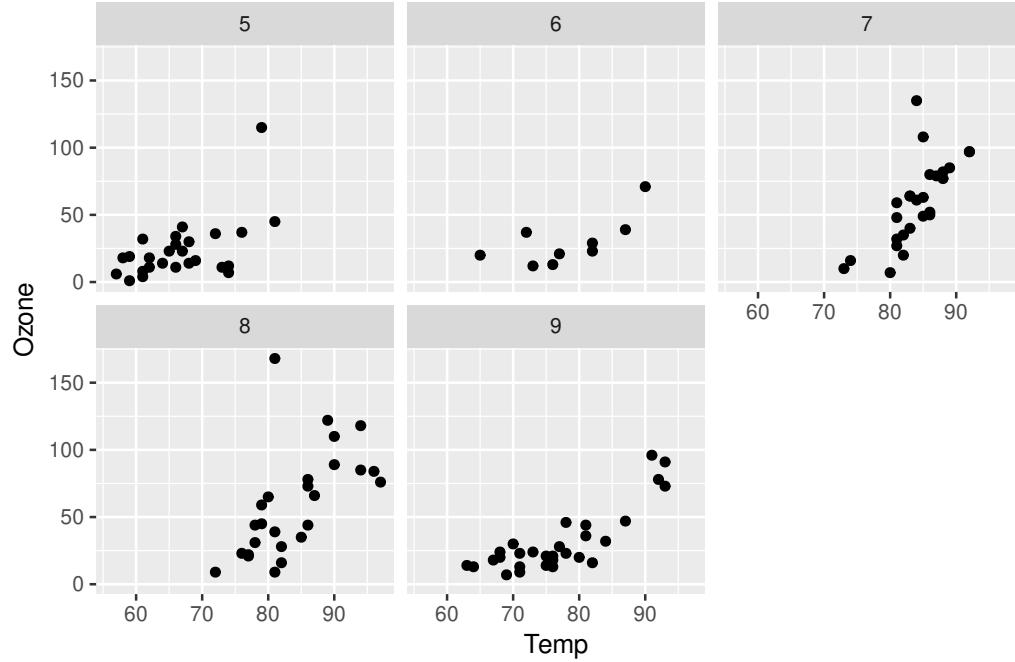
(d) 经典主题

图 11.5: ggplot2 内置的主题

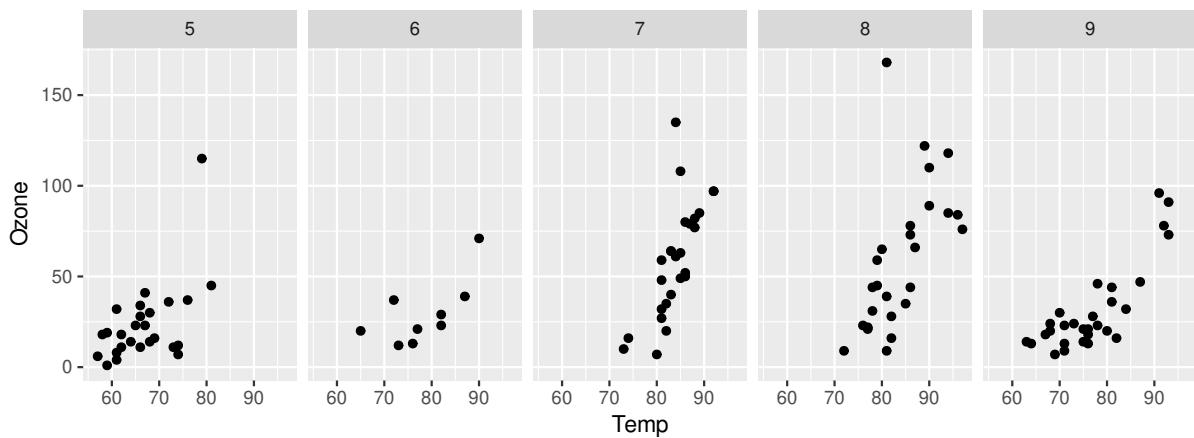
除主题之外，还有一类提供一整套统一的风格样式来绘制各种统计图形，如 [ggpubr](#) 和 [biplot](#)

## 11.1.10 布局

```
ggplot(airquality) +  
  geom_point(aes(x = Temp, y = Ozone), na.rm = TRUE) +  
  facet_wrap(~ as.ordered(Month))
```



```
ggplot(airquality) +  
  geom_point(aes(x = Temp, y = Ozone), na.rm = TRUE) +  
  facet_wrap(~ as.ordered(Month), nrow = 1)
```



cowplot 是以作者 Claus O. Wilke 命名的，用来组合 ggplot 对象画图，类似的组合图形的功能包还有 baptiste auguié 开发的 gridExtra 和 egg，Thomas Lin Pedersen 开发的 patchwork

Dean Attali 开发的 ggExtra 可以在图的边界添加密度估计曲线，直方图等



## 11.2 字体

`firatheme` 包提供基于 fira sans 字体的 `ggplot2` 主题，类似的字体主题包还有 `trekfont`、`fontHind`，`fontquiver` 包与 `fontBitstreamVera` (Bitstream Vera 字体)、`fontLiberation` (Liberation 字体) 包和 `fontDejaVu` (DejaVu 字体) 包一道提供了一些可允许使用的字体文件，这样，我们可以不依赖系统制作可重复的图形。Thomas Lin Pedersen 开发的 `systemfonts` 可直接使用系统自带的字体。

### 11.2.1 系统字体

以 CentOS 系统为例，软件仓库中包含 `Noto`、`DejaVu`、`liberation` 等字体。可以安装自己喜欢的字体类型，比如：

```
sudo dnf install -y \
  google-noto-mono-fonts \
  google-noto-sans-fonts \
  google-noto-serif-fonts \
  dejavu-sans-mono-fonts \
  dejavu-sans-fonts \
  dejavu-serif-fonts
# 或者
sudo dnf install -y dejavu-fonts liberation-fonts
```

`liberation` 系列的四款字体可以用来替换 Windows 系统上对应的四款字体，对应关系见表 11.1

表 11.1: Windows 系统上四款字体的替代品

	CentOS 系统	Windows 系统
衬线体/宋体	<code>liberation-serif-fonts</code>	Times New Roman
无衬线体/黑体	<code>liberation-sans-fonts</code>	Arial
Arial 的细瘦版	<code>liberation-narrow-fonts</code>	Arial Narrow
等宽体/微软雅黑	<code>liberation-mono-fonts</code>	Courier New

Lionel Henry 将 `Liberation` 系列字体打包到 R 包 `fontLiberation`，非常便携，不需要操心跨平台的字体安装了。那如何使用呢？

```
# install.packages("fontLiberation")
system.file(package = "fontLiberation", "fonts", "liberation-fonts")
## [1] ""
```

此外，我们还可以从网上获取各种各样的字体，特别地，Boryslav Larin 收录的 `awesome-fonts` 列表是一个不错的开始，比如图标字体 `Font-Awesome`，

```
sudo dnf install -y fontawesome-fonts
```

再安装宏包 `fontawesome` 后，即可在 LaTeX 文档中使用，下面这个示例推荐用 XeLaTeX 引擎编译。

```
\documentclass[border=10pt]{standalone}
\usepackage{fontawesome}
\begin{document}
Hello, \faGithub
\end{document}
```

而在 R 绘制的图形中，通过指定 `par()`、`plot()`、`title()` 等函数的 `family` 参数值，比如 `family = "Liberation Sans"` 来调用系统无衬线 Liberation 字体，效果见图 11.6。

```
library(extrafont)
plot(data = pressure, pressure ~ temperature,
      xlab = "Temperature (deg C)", ylab = "Pressure (mm of Hg)",
      col.lab = "red", col.axis = "blue",
      font.lab = 3, font.axis = 2, family = "Liberation Sans")
title(main = "Vapor Pressure of Mercury as a Function of Temperature",
      family = "Liberation Serif", font.main = 3)
title(sub = "Data Source: Weast, R. C",
      family = "Liberation Mono", font.sub = 1)
```

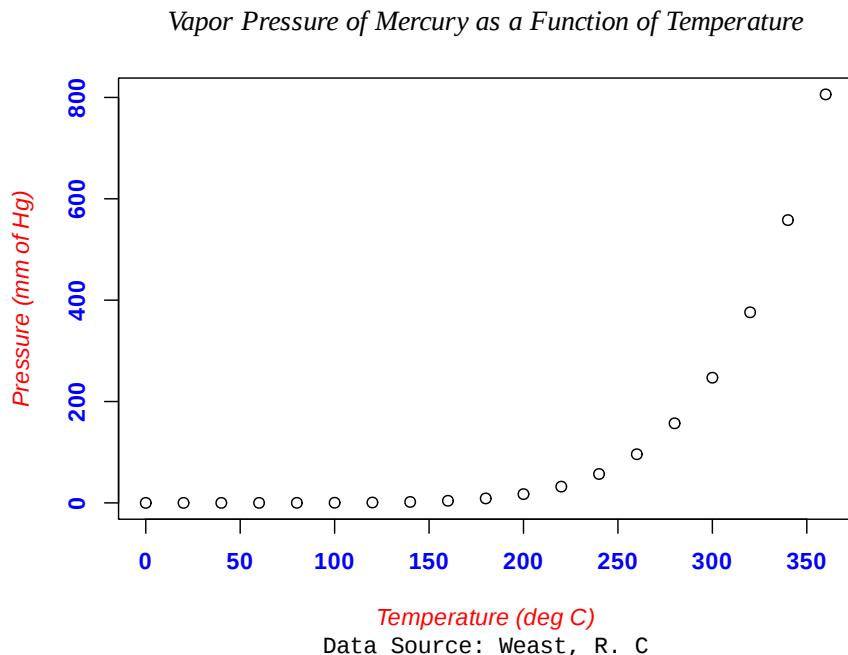


图 11.6: 调用系统字体绘图

为了符合出版的要求，需要在 11.6 中嵌入字体，

```
# embed fonts to pdf
embed_fonts <- function(fig_path) {
  if(knitr:::is_latex_output()){
    embedFonts(
      file = fig_path, outfile = fig_path,
      fontpaths = "~/Library/Fonts"
```



```
    )
}
return(fig_path)
}
```

设置代码块选项 `fig.process=embed_fonts`, 这样生成 PDF 格式图形的时候, 会调用此函数处理 PDF 图形。在 `ggplot2` 绘图中的调用方式是类似的, 便不再赘述了。值得注意的是, `extrafont` 和 `showtext` 有些不一样, 前者只能处理系统字体, 后者还能获取网络字体和使用 OTF 字体, 下面从 Google 开源的字体库获取 Noto 系列的四款字体, 如图 11.7。

```
sysfonts::font_add_google(name = "Noto Sans", family = "Noto Sans")
sysfonts::font_add_google(name = "Noto Serif", family = "Noto Serif")
sysfonts::font_add_google(name = "Noto Serif SC", family = "Noto Serif SC")
sysfonts::font_add_google(name = "Noto Sans SC", family = "Noto Sans SC")
```

警告

在本书中, 不要全局加载 `showtext` 包或调用 `showtext::showtext_auto()`, 会和 `extrafont` 冲突, 使得绘图时默认就只能使用 `showtext` 提供的字体。`extrafont` 包提供的函数 `font_import()` 仅支持系统安装的 TrueType/Type1 字体

```
p1 <- ggplot(pressure, aes(x = temperature, y = pressure)) +
  geom_point() +
  ggtitle(label = "默认字体设置")

p2 <- p1 + theme(
  axis.title = element_text(family = "Noto Sans"),
  axis.text = element_text(family = "Noto Serif"))
) +
  theme(
    title = element_text(family = "Noto Serif SC"))
) +
  ggtitle(label = "英文字体设置")

p3 <- p1 + labs(x = "温度", y = "压力") +
  theme(
    axis.title = element_text(family = "Noto Serif SC"),
    axis.text = element_text(family = "Noto Serif"))
) +
  ggtitle(label = "中文字体设置")

p4 <- p1 + labs(
  x = "温度", y = "压力", title = "散点图",
  subtitle = "Vapor Pressure of Mercury as a Function of Temperature",
  caption = paste("Data on the relation
                  between temperature in degrees Celsius and",
  "vapor pressure of mercury in millimeters (of mercury).",
```

```

sep = "\n"
)
) +
theme(
  axis.title = element_text(family = "Noto Serif SC"),
  axis.text.x = element_text(family = "Noto Serif"),
  axis.text.y = element_text(family = "Noto Sans"),
  title = element_text(family = "Noto Serif SC"),
  plot.subtitle = element_text(family = "Noto Sans", size = rel(0.7)),
  plot.caption = element_text(family = "Noto Sans", size = rel(0.6))
) +
ggttitle(label = "任意字体设置")

(p1 + p2) / (p3 + p4)

```

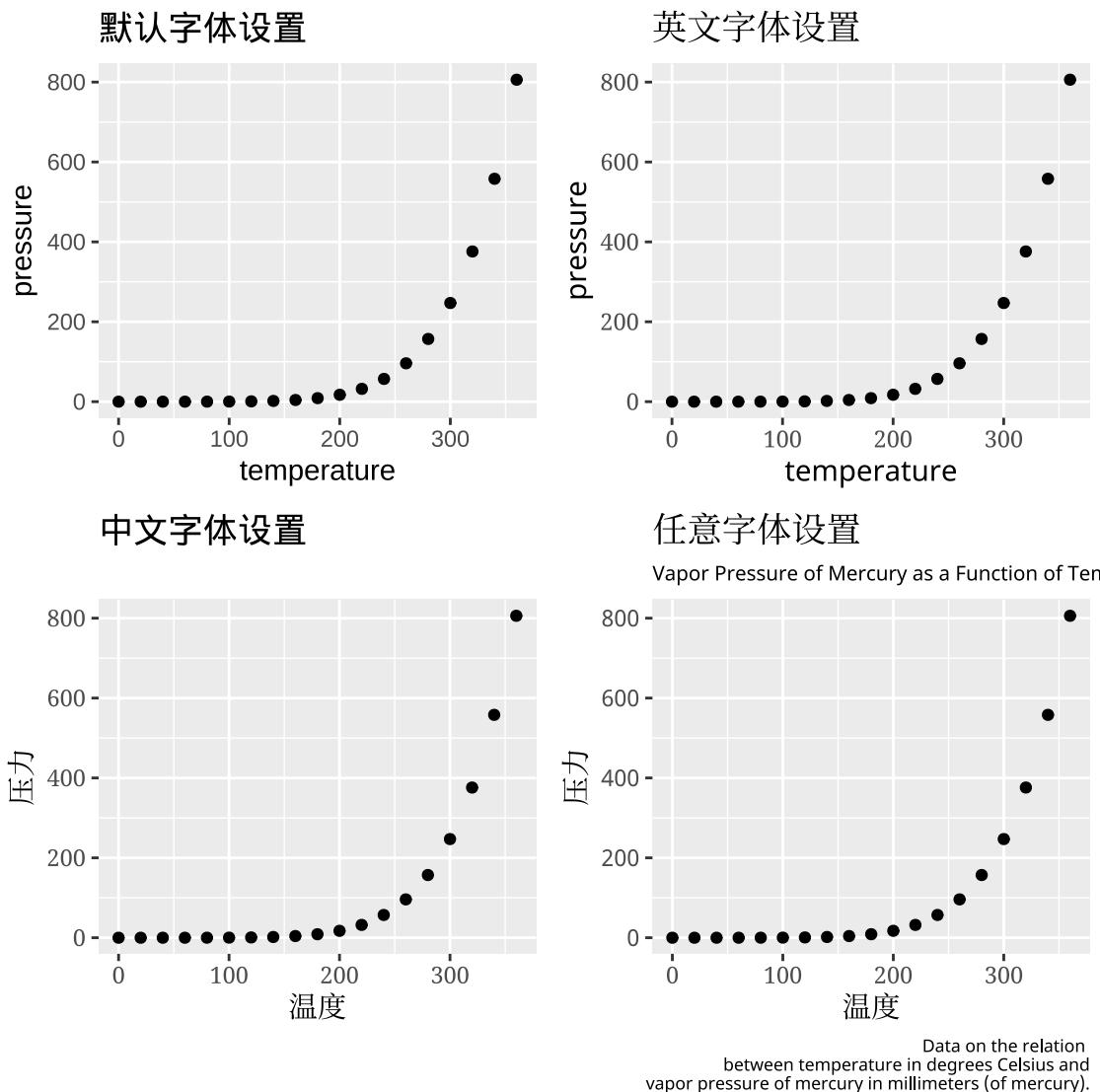


图 11.7: 在 ggplot2 绘图系统中设置中英文字体



另外值得一提的是 `hrbrthemes` 包，除了定制了很多 `ggplot2` 主题，它还打包了很多的字体主题。比如默认主题 `theme_ipsum()` 使用 Arial Narrow 字体，如果没有该字体就自动寻找系统中的替代品，如图 11.8 实际使用的是 Nimbus Sans Narrow 字体，因为在 GitHub Action 中，我实际使用的测试环境是 Ubuntu 20.04，该系统自带 Nimbus Sans Narrow 字体，Arial Narrow 毕竟是 Windows 上的闭源字体。

```
# brew install font-roboto
# 导入字体
# hrbrthemes::import_roboto_condensed()
sysfonts::font_add_google(name = "Roboto Condensed", family = "Roboto Condensed")

library(hrbrthemes)
ggplot(mtcars, aes(mpg, wt)) +
  geom_point() +
  labs(
    x = "Fuel efficiency (mpg)", y = "Weight (tons)",
    title = "Seminal ggplot2 scatterplot example",
    subtitle = "A plot that is only useful for demonstration purposes",
    caption = "Brought to you by the letter 'g'"
  ) +
  theme_ipsum(base_family = "Roboto Condensed")
```

## Seminal ggplot2 scatterplot example

A plot that is only useful for demonstration purposes

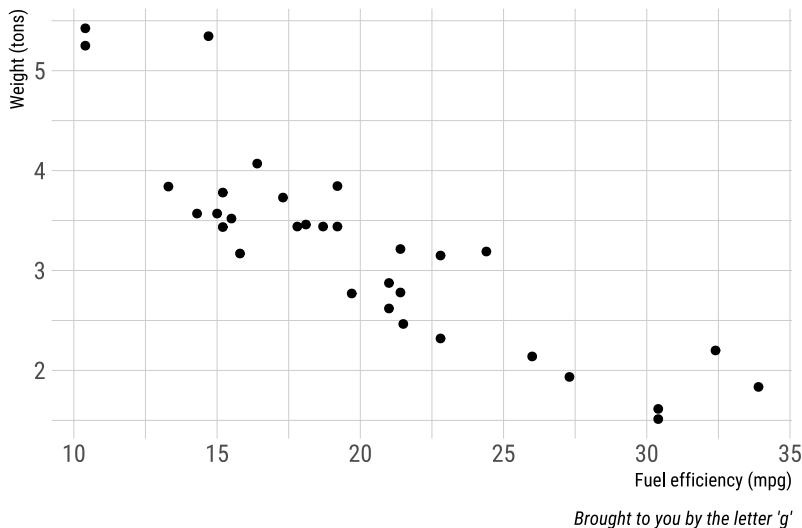


图 11.8: 调用 `hrbrthemes` 包设置字体主题

如果系统没有安装 Arial Narrow 字体，可以导入 `hrbrthemes` 包自带的一些字体，比如 `hrbrthemes::import_roboto_condensed()`，然后调用字体主题 `theme_ipsum_rc()`。如果不想使用这个包自带的字体，可以用系统中安装的字体去修改主题 `theme_ipsum()` 和 `theme_ipsum_rc()` 中的字体设置。如图 11.9 使用了 `theme_ipsum()` 中的 Arial Narrow 字体。

```
ggplot(mtcars, aes(mpg, wt)) +  
  geom_point() +  
  labs(  
    x = "Fuel efficiency (mpg)", y = "Weight (tons)",  
    title = "Seminal ggplot2 scatterplot example",  
    subtitle = "A plot that is only useful for demonstration purposes",  
    caption = "Brought to you by the letter 'g'"  
) +  
  theme_ipsum(base_family = "Noto Sans")
```

## Seminal ggplot2 scatterplot example

A plot that is only useful for demonstration purposes

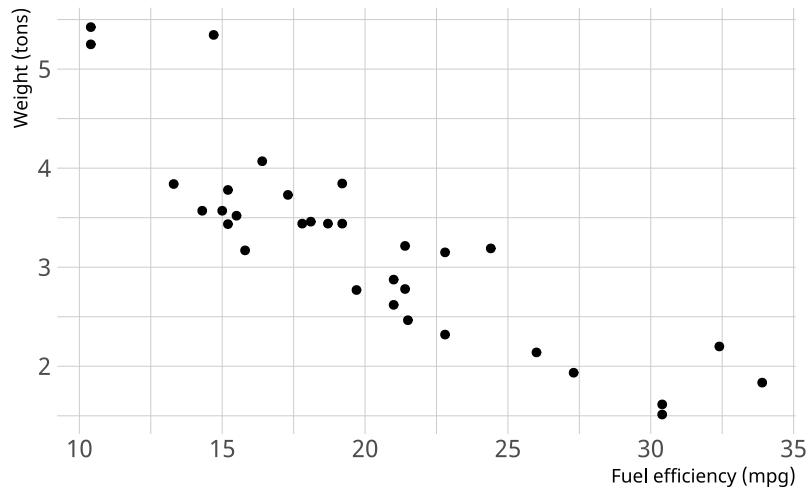


图 11.9: 默认字体 Arial Narrow

提示

**hrbrthemes** 包提供了一个全局字体加载选项 `hrbrthemes.loadfonts`，如果设置为 TRUE，即 `options(hrbrthemes.loadfonts = TRUE)` 会先调用函数 `extrafont::loadfonts()` 预加载系统字体，就不用一次次手动加载字体了。后续在第 11.2.3 节还会提及 `extrafont` 包的其它功能。

### 11.2.2 思源字体

邱怡轩开发的 `showtext` 包支持丰富的外部字体，支持 Base R 和 ggplot2 图形，图 11.10 嵌入了 5 号思源宋体，图例和坐标轴文本使用 serif 字体，更多详细的使用文档见 [Qiu, 2015]。

```
# 安装 showtext 包  
install.packages('showtext')  
# 思源宋体  
showtextdb::font_install(showtextdb::source_han_serif())
```



```
# 思源黑体
showtextdb::font_install(showtextdb::source_han_sans())

# ggplot(iris, aes(Sepal.Length, Sepal.Width)) +
#   geom_point(aes(colour = Species)) +
#   scale_colour_brewer(palette = "Set1") +
#   labs(
#     title = "鸢尾花数据的散点图",
#     x = "萼片长度", y = "萼片宽度", colour = "鸢尾花类别",
#     caption = "鸢尾花数据集最早见于 Edgar Anderson (1935) "
#   ) +
#   theme(
#     title = element_text(family = "source-han-sans-cn"),
#     axis.title = element_text(family = "source-han-serif-cn"),
#     legend.title = element_text(family = "source-han-serif-cn")
#   )

ggplot(iris, aes(Sepal.Length, Sepal.Width)) +
  geom_point(aes(colour = Species)) +
  scale_colour_brewer(palette = "Set1") +
  labs(
    title = "鸢尾花数据的散点图",
    x = "萼片长度", y = "萼片宽度", colour = "鸢尾花类别",
    caption = "鸢尾花数据集最早见于 Edgar Anderson (1935) "
  ) +
  theme(
    title = element_text(family = "Noto Sans SC"),
    axis.title = element_text(family = "Noto Serif SC"),
    legend.title = element_text(family = "Noto Serif SC")
  )
```

### 11.2.3 数学字体

Winston Chang 将 Paul Murrell 的 Computer Modern 字体文件打包成 `fontcm` 包 [Chang et al., 2014], `fontcm` 包可以在 Base R 图形中嵌入数学字体<sup>6</sup>，图形中嵌入重音字符<sup>7</sup>。下面先下载、安装、加载字体，

```
library(extrafont)
if (!"fontcm" %in% .packages(T)) {
  install.packages("fontcm")
}
```

查看可被 `pdf()` 图形设备使用的字体列表

```
# 可用的字体
fonts()
```

<sup>6</sup><https://www.stat.auckland.ac.nz/~paul/R/CM/CMR.html>

<sup>7</sup><https://www.stat.auckland.ac.nz/~paul/Reports/maori/maori.html>

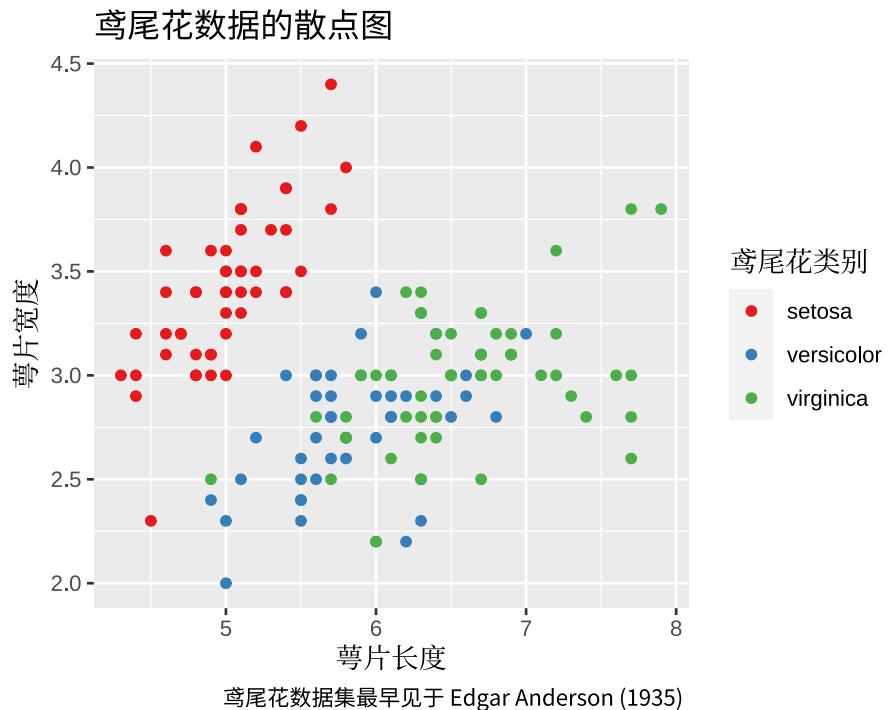


图 11.10: showtext 包处理图里的中文

fontcm 包提供数学字体, grDevices::embedFonts() 函数调用 Ghostscript 软件将数学字体嵌入 ggplot2 图形中, 达到正确显示数学公式的目的, 此方法适用于 pdf 设备保存的图形, 对 cairo\_pdf() 保存的 PDF 格式图形无效。

```
library(fontcm)
library(ggplot2)
library(extrafont)
library(patchwork)
p <- ggplot(
  data = data.frame(x = c(1, 5), y = c(1, 5)),
  aes(x = x, y = y)
) +
  geom_point() +
  labs(
    x = "Made with CM fonts", y = "Made with CM fonts",
    title = "Made with CM fonts"
  )
# 公式
eq <- "italic(sum(frac(1, n*!!), n==0, infinity) ==
  lim(bgroup('(', 1 + frac(1, n), ')')^n, n %-%>% infinity))"
# 默认字体
p1 <- p + annotate("text",
  x = 3, y = 3,
  parse = TRUE, label = eq # , family = "CM Roman"
)
```



```
# 使用 CM Roman 字体
p2 <- p + annotate("text",
  x = 3, y = 3,
  parse = TRUE, label = eq, family = "CM Roman"
) +
  theme(
    text = element_text(size = 10, family = "CM Roman"),
    axis.title.x = element_text(face = "italic"),
    axis.title.y = element_text(face = "bold")
)
p1 + p2
```

为实现图 ?? 的最终效果，需要启用一个有超级牛力的 `fig.process` 选项，主要是传递一个函数给它，对用 R 语言生成的图形再操作。

```
# embed math fonts to pdf
embed_math_fonts <- function(fig_path) {
  if(knitr:::is_latex_output()){
    embedFonts(
      file = fig_path, outfile = fig_path,
      fontpaths = system.file("fonts", package = "fontcm")
    )
  }
  return(fig_path)
}
```

代码块选项中设置 `fig.process=embed_math_fonts` 可在绘图后，立即插入字体，此操作仅限于以 `pdf` 格式保存的图形设备，也适用于 Base R 绘制的图形，见图 ??。

```
par(mar = c(4.1, 4.1, 1.5, 0.5), family = "CM Roman")
x <- seq(-4, 4, len = 101)
y <- cbind(sin(x), cos(x))
matplot(x, y,
  type = "l", xaxt = "n",
  main = expression(paste(
    plain(sin) * phi, " and ",
    plain(cos) * phi
)),
  ylab = expression("sin" * phi, "cos" * phi),
  xlab = expression(paste("Phase Angle ", phi)),
  col.main = "blue"
)
axis(1,
  at = c(-pi, -pi / 2, 0, pi / 2, pi),
  labels = expression(-pi, -pi / 2, 0, pi / 2, pi)
)
```



### 11.2.4 TikZ 设备

与 11.2.3 小节不同, Ralf Stubner 维护的 `tikzDevice` 包提供了另一种嵌入数学字体的方式, 其提供的 `tikzDevice::tikz()` 绘图设备将图形对象转化为 TikZ 代码, 调用 LaTeX 引擎编译成 PDF 文档。安装后, 先测试一下 LaTeX 编译环境是否正常。

```
tikzDevice::tikzTest()

## 
## Active compiler:
## /home/runner/.TinyTeX/bin/x86_64-linux/xelatex
## XeTeX 3.141592653-2.6-0.999994 (TeX Live 2022)
## kpsewhich version 6.3.4

## [1] 7.90259
```

确认没有问题后, 下面图 11.11 的坐标轴标签, 标题, 图例等位置都支持数学公式, 使用 `tikzDevice` 打造出版级的效果图。更多功能的介绍见 <https://www.daqana.org/tikzDevice/>。

```
x <- rnorm(10)
y <- x + rnorm(5, sd = 0.25)
model <- lm(y ~ x)
rsq <- summary(model)$r.squared
rsq <- signif(rsq, 4)
plot(x, y,
  main = "Hello \\LaTeX!", xlab = "$x$", ylab = "$y$",
  sub = "$\\mathcal{N}(x; \\mu, \\Sigma)$"
)
abline(model, col = "red")
mtext(paste0("Linear model: $R^2=", rsq, "$"), line = 0.5)
legend("bottomright",
  legend = paste0(
    "$y = $",
    round(coef(model)[2], 3),
    "x +",
    round(coef(model)[1], 3),
    "$"
  ),
  bty = "n"
)
```

推荐的全局 LaTeX 环境配置如下:

```
options(
  tinytex.engine = "xelatex",
  tikzDefaultEngine = "xetex",
  tikzDocumentDeclaration = "\\documentclass[tikz]{standalone}\n",
  tikzXelatexPackages = c(
    "\\usepackage[fontset=adobe]{ctex}",
```

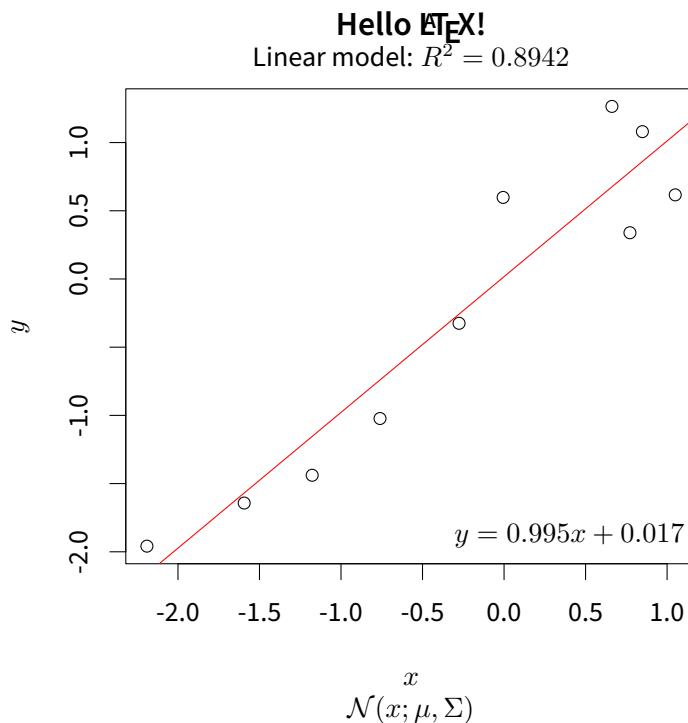


图 11.11: 线性回归模型

```

"\\usepackage[default,semibold]{sourcesanspro}",
"\\usepackage{amsfonts,mathrsfs,amssymb}\\n"
)
)

```

设置默认的 LaTeX 编译引擎为 XeLaTeX，相比于 PDFLaTeX，它对中文的兼容性更好，支持多平台下的中文环境，中文字体这里采用了 Adobe 的字体，默认加载了 `mathrsfs` 宏包支持 `\mathcal`、`\mathscr` 等命令，此外，TeX 发行版采用谢益辉自定义的 `TinyTeX`。绘制独立的 PDF 图形的过程如下：

```

library(tikzDevice)
tf <- file.path(getwd(), "tikz-regression.tex")
tikz(tf, width = 6, height = 5.5, pointsize = 30, standAlone = TRUE)
# 绘图代码
dev.off()
# 编译成 PDF 图形
tinytex::latexmk(file = "tikz-regression.tex")

```

## 11.2.5 漫画字体

下载 XKCD 字体，并刷新系统字体缓存

```

mkdir -p ~/.fonts
curl -fLo ~/.fonts/xkcd.ttf http://simonsoftware.se/other/xkcd.ttf
fc-cache -fsv

```



将 XKCD 字体导入到 R 环境，以便后续被 `ggplot2` 图形设备调用。

```
R -e 'library(extrafont);font_import(pattern="[X/x]kcd.ttf", prompt = FALSE)'
```

下图是一个使用 `xkcd` 字体的简单例子，更多高级特性请看 `xkcd` 包文档 [Torres-Manzanera, 2018]

```
library(xkcd)
ggplot(aes(mpg, wt), data = mtcars) +
  geom_point() +
  theme_xkcd()
```

### 11.2.6 表情字体

余光创开发的 `emojifont` 包和 Hadley 开发的 `emo` 包，下面使用 Noto Emoji 字体，支持的表情图见 <https://www.google.com/get/noto/help/emoji/food-drink/>，下面给出一个示例。先从 GitHub 安装 `emo` 包，目前它还未正式发布到 CRAN 上。

```
remotes::install_github("hadley/emo")
```

除了安装 `emo` 包，系统需要先安装好 emoji 字体，图形才会正确地渲染出来，想调用更多 emoji 图标请参考 [Emoji 速查手册](#)，给出 emoji 对应的名字。

```
# CentOS
sudo dnf install -y google-noto-emoji-color-fonts \
  google-noto-emoji-fonts
# MacOS
brew cask install font-noto-color-emoji font-noto-emoji
```

```
data.frame(
  category = c("pineapple", "apple", "watermelon", "mango", "pear"),
  value = c(5, 4, 3, 6, 2)
) |>
  transform(category = sapply(category, emo::ji)) |>
  ggplot(aes(x = category, y = value)) +
  scale_y_continuous(limits = c(2, 7)) +
  geom_text(aes(label = category), size = 12, vjust = -0.5) +
  theme_minimal()
```

Noto Color Emoji 字体在 MacOS 上有问题，为了跨平台的便携性，提供 `emojifont` 包的例子，要引入更多的依赖。

```
library(ggplot2)
library(emojifont)

names <- c("smile", "school", "office", "blush", "smirk", "heart_eyes")
n <- length(names):1
e <- sapply(names, emojifont::emoji)
dat <- data.frame(emoji_name = names, n = n, emoji = e, stringsAsFactors = F)

ggplot(data = dat, aes(emoji_name, n)) +
```

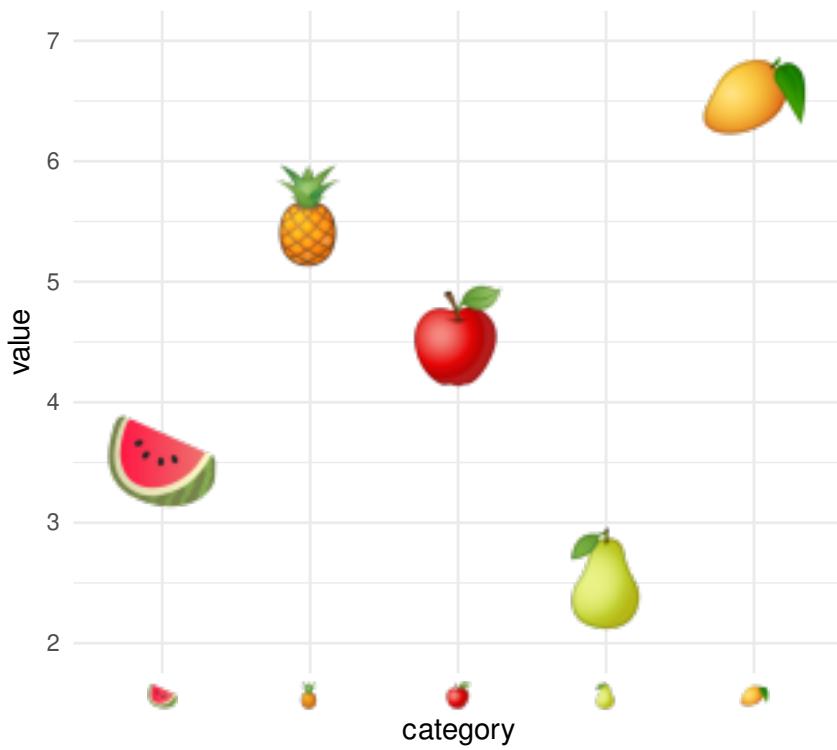


图 11.12: 表情字体

```
geom_bar(stat = "identity") +  
  scale_x_discrete(breaks = dat$emoji_name, labels = dat$emoji) +  
  theme(axis.text.y = element_text(size = 20, family = "EmojiOne")) +  
  coord_flip()
```

### 11.3 配色

配色真的是一门学问，有的人功力非常深厚，仅用黑白灰就可以创造出一个世界，如中国的水墨画，科波拉执导的《教父》，沃卓斯基姐妹执导的《黑客帝国》等。黑西装、白衬衫和黑领带是《黑客帝国》的经典元素，《教父》开场的黑西装、黑领结和白衬衫，尤其胸前的红玫瑰更是点睛之笔。导演将黑白灰和光影混合形成了层次丰富立体的画面，打造了一场视觉盛宴，无论是呈现在纸上还是银幕上都可以给人留下深刻的印象。正所谓食色性也，花花世界，岂能都是法印眼中的白骨！再说《红楼梦》里，芍药丛中，桃花树下，滴翠亭边，栊翠庵里，处处都是湘云、黛玉、宝钗、妙玉留下的四季诗歌。

为什么需要这么多颜色模式呢？主要取决于颜色输出的通道，比如印刷机，照相机，自然界，网页，人眼等，显示器因屏幕和分辨率的不同呈现的色彩数量是不一样的。读者大概都听说过 RGB、CMYK、AdobeRGB、sRGB、P3 广色域等名词，我想这主要归功于各大电子设备厂商的宣传。普清、高清、超高清、全高清、2K、4K、5K、视网膜屏，而 HSV、HCL 估计听说的人就少很多了。本节的目的是简单阐述背后的色彩原理，颜色模式及其之间的转化，在应对天花乱坠的销售时少交一些智商税，同时，告诉读者如何在 R 环境中使用色彩。早些时候我在统计之都论坛上发帖 - R 语言绘图用调色板大全 <https://d.cosx.org/d/419378>，如果读者希望拿来即用，不妨去看看。

```
filled.contour(volcano, nlevels = 10, color.palette = terrain.colors)
filled.contour(volcano, nlevels = 10, color.palette = heat.colors)
filled.contour(volcano, nlevels = 10, color.palette = topo.colors)
filled.contour(volcano, nlevels = 10, color.palette = cm.colors)
```

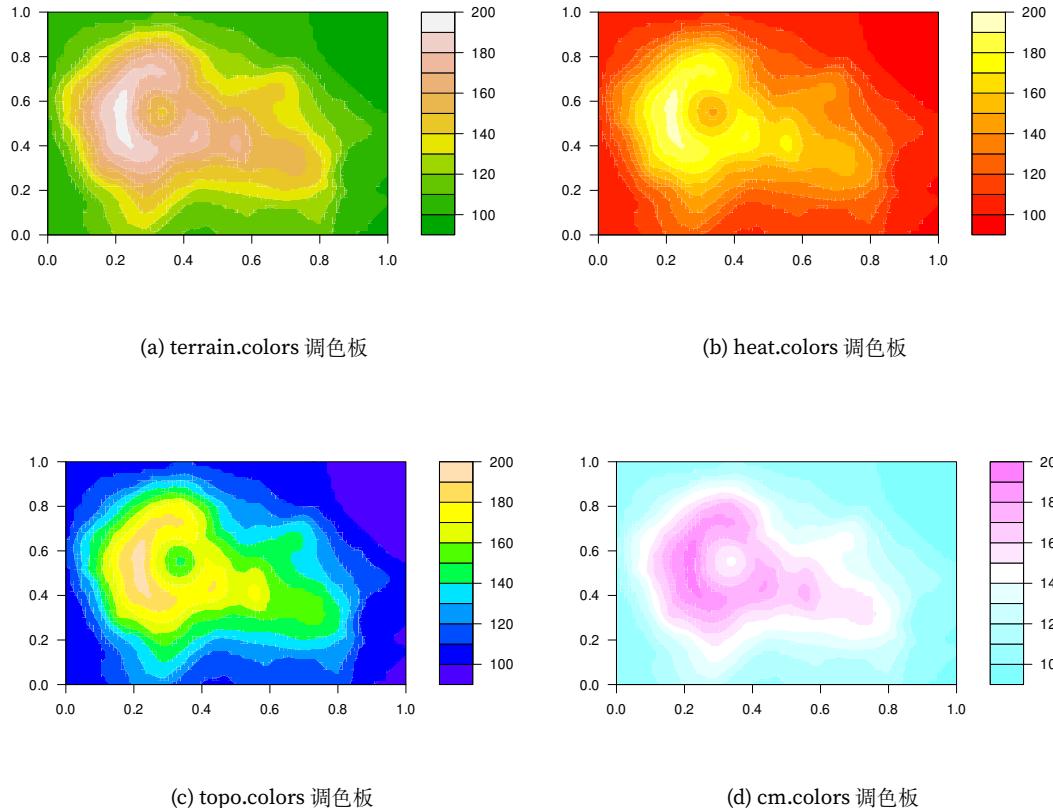


图 11.13: R 3.6.0 以前的调色板

```
filled.contour(volcano,
  nlevels = 10,
  color.palette = function(n, ...) hcl.colors(n, "Grays", rev = TRUE, ...))
)
filled.contour(volcano,
  nlevels = 10,
  color.palette = function(n, ...) hcl.colors(n, "YlOrRd", rev = TRUE, ...))
)
filled.contour(volcano,
  nlevels = 10,
  color.palette = function(n, ...) hcl.colors(n, "purples", rev = TRUE, ...))
)
filled.contour(volcano,
  nlevels = 10,
  color.palette = function(n, ...) hcl.colors(n, "viridis", rev = FALSE, ...))
)
```

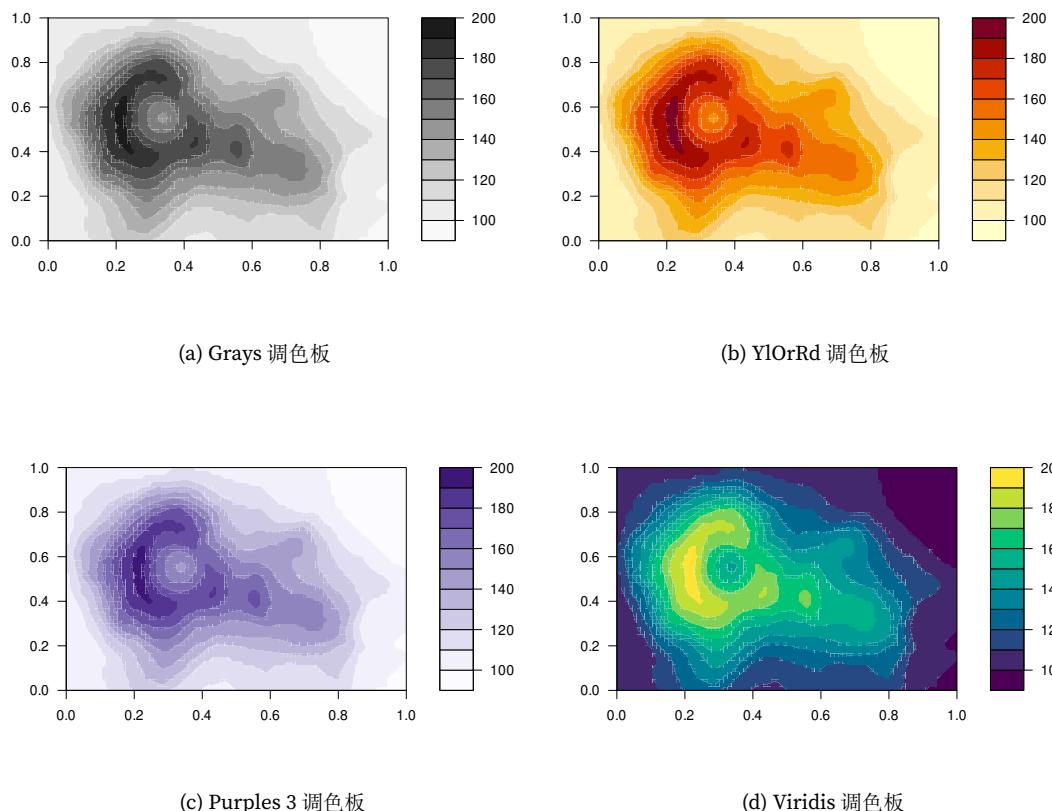


图 11.14: R 3.6.0 以后的调色板

注意

`hcl.colors()` 函数是在 R 3.6.0 引入的，之前的 R 软件版本中没有，同时内置了 110 个调色板，详见 `hcl.pals()`。

### 11.3.1 调色板

R 预置的灰色有 224 种，挑出其中的调色板

```
grep("gr(a|e)y", grep("gr(a|e)y", colors(), value = TRUE),
     value = TRUE, invert = TRUE)

## [1] "darkgray"        "darkgrey"        "darkslategray"   "darkslategray1"
## [5] "darkslategray2"  "darkslategray3"  "darkslategray4"  "darkslategrey"
## [9] "dimgray"         "dimgrey"        "lightgray"       "lightgrey"
## [13] "lightslategray"  "lightslategrey"  "slategray"       "slategray1"
## [17] "slategray2"      "slategray3"     "slategray4"      "slategrey"

gray_colors <- paste0(rep(c("slategray", "darkslategray"), each = 4), seq(4))
barplot(1:8, col = gray_colors, border = NA)
```

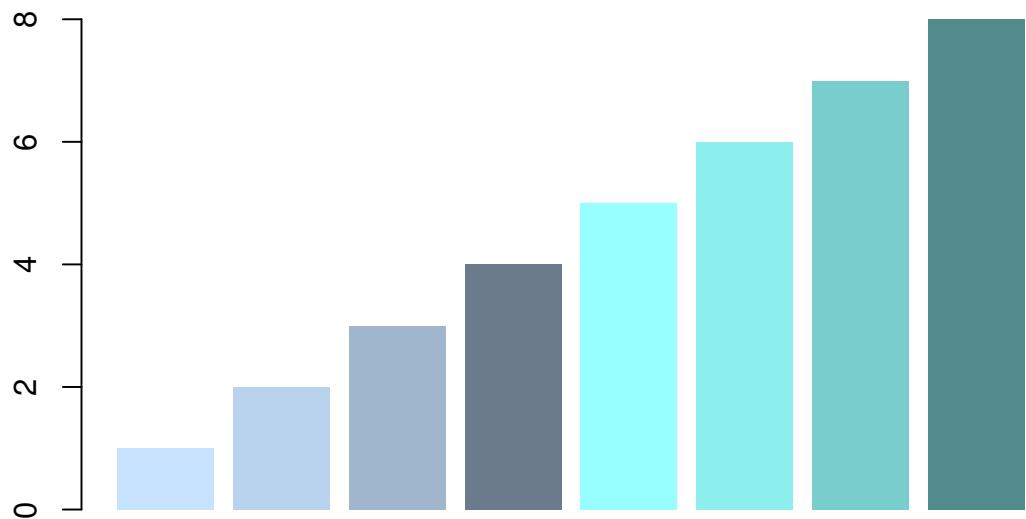


图 11.15: 灰度调色板

gray 与 grey 是一样的，类似 color 和 colour 的关系，可能是美式和英式英语的差别，且看

```
all.equal(
  col2rgb(paste0("gray", seq(100))),
  col2rgb(paste0("grey", seq(100)))
)
```

## [1] TRUE

gray100 代表白色, gray0 代表黑色, 提取灰色调色板, 去掉首尾部分是必要的

```
barplot(1:8,
  col = gray.colors(8, start = .3, end = .9),
  main = "gray.colors function", border = NA
)
```

## gray.colors function

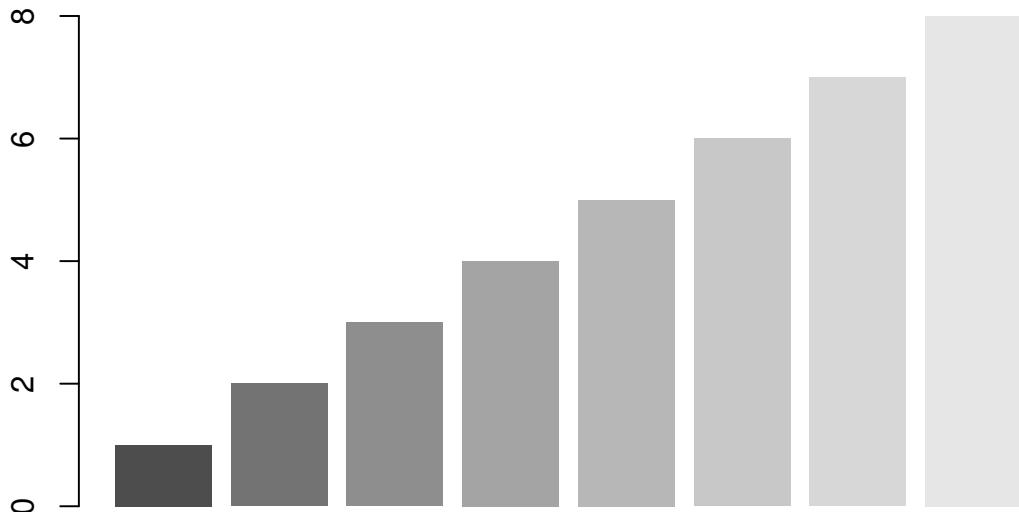


图 11.16: 提取 10 种灰色做调色板

首先选择一组合适的颜色, 比如从桃色到梨色, 选择 6 种颜色, 以此为基础, 可以借助 `grDevices::colorRampPalette()` 函数扩充至想要的数目, 用 `graphics::rect()` 函数预览这组颜色配制的调色板

```
# Colors from https://github.com/johannesbjork/LaCroixColor
colors_vec <- c("#FF3200", "#E9A17C", "#E9E4A6",
  "#1BB6AF", "#0076BB", "#172869")
# 代码来自 ?colorspace::rainbow_hcl
pal <- function(n = 20, colors = colors, border = "light gray", ...) {
  colorname <- (grDevices::colorRampPalette(colors))(n)
  plot(0, 0,
    type = "n", xlim = c(0, 1), ylim = c(0, 1),
```

```
    axes = FALSE, ...  
)  
rect(0:(n - 1) / n, 0, 1:n / n, 1, col = colordname, border = border)  
}  
par(mar = rep(0, 4))  
pal(n = 20, colors = colors_vec, xlab = "Colors from Peach to Pear", ylab = "")
```

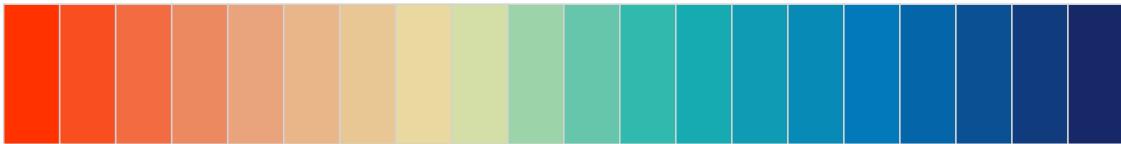


图 11.17: 桃色至梨色的渐变

colorRampPalette() 自制调色板

```
create_palette <- function(n = 1000, colors = c("blue", "orangeRed")) {  
  color_palette <- colorRampPalette(colors)(n)  
  barplot(rep(1, times = n), col = color_palette,  
         border = color_palette, axes = FALSE)  
}  
par(mfrow = c(3, 1), mar = c(0.1, 0.1, 0.5, 0.1), xaxs = "i", yaxs = "i")  
create_palette(n = 1000, colors = c("blue", "orangeRed"))  
create_palette(n = 1000, colors = c("darkgreen", "yellow", "orangered"))  
create_palette(n = 1000, colors = c("blue", "white", "orangered"))
```

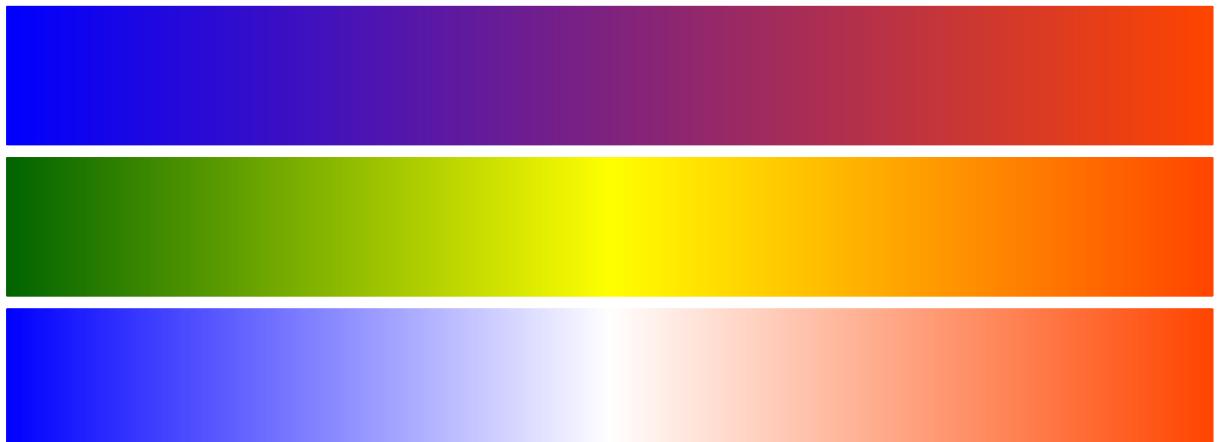


图 11.18: colorRampPalette 自制调色板

```
par(mar = c(0, 4, 0, 0))  
RColorBrewer::display.brewer.all()  
  
# 代码来自 ?palettes  
demo.pal <- function(n, border = if (n < 32) "light gray" else NA,  
                      main = paste("color palettes: alpha = 1, n=", n),  
                      ch.col = c(  
                        "rainbow(n, start=.7, end=.1)", "heat.colors(n)",
```

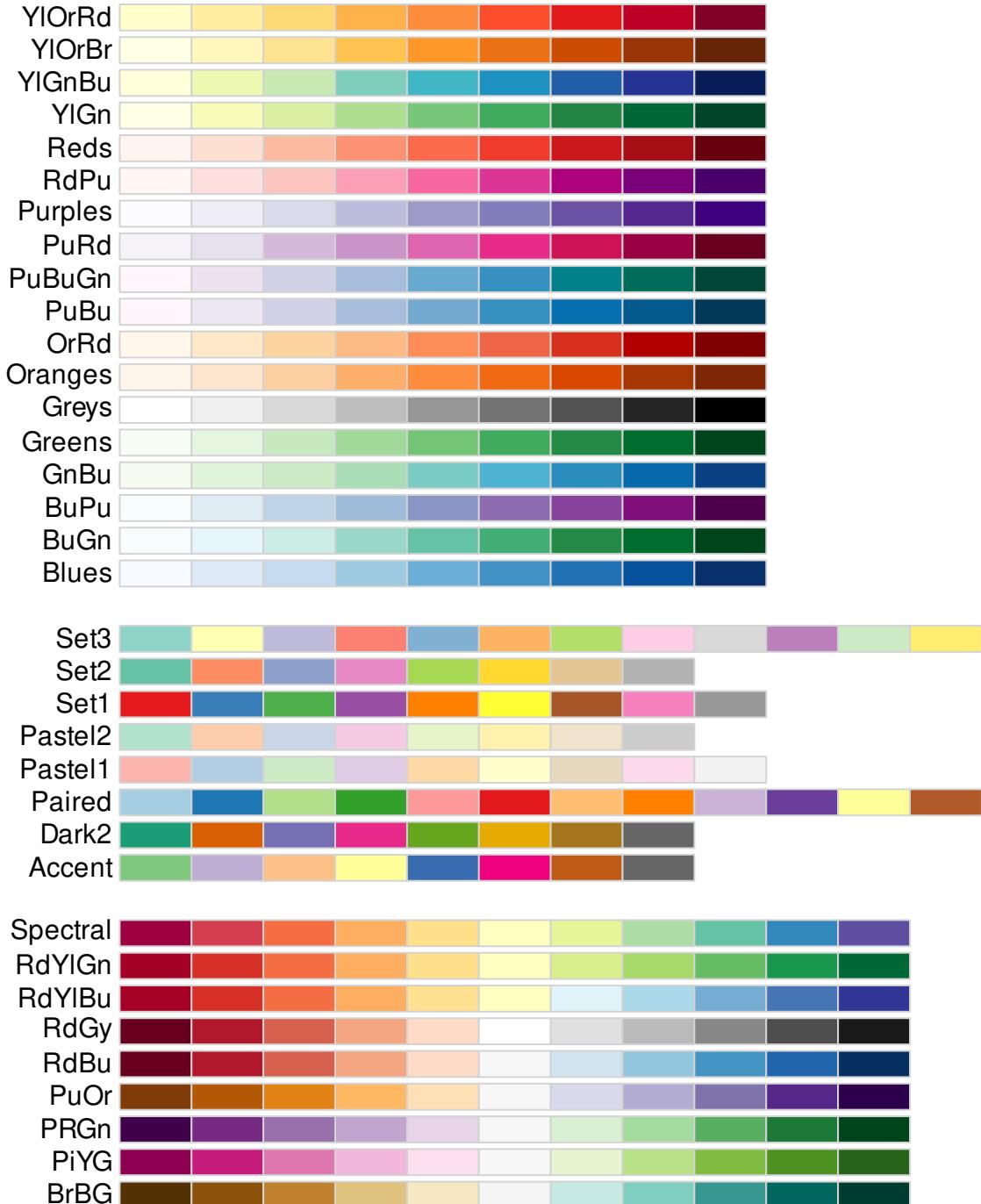


图 11.19: RColorBrewer 调色板



```
    "terrain.colors(n)", "topo.colors(n)",
    "cm.colors(n)", "gray.colors(n, start = 0.3, end = 0.9)"
  )) {
  nt <- length(ch.col)
  i <- 1:n
  j <- n / nt
  d <- j / 6
  dy <- 2 * d
  plot(i, i + d, type = "n", axes = FALSE, ylab = "", xlab = "", main = main)
  for (k in 1:nt) {
    rect(i - .5, (k - 1) * j + dy, i + .4, k * j,
         col = eval(parse(text = ch.col[k])), border = border
    )
    text(2 * j, k * j + dy / 4, ch.col[k])
  }
}
n <- if (.Device == "postscript") 64 else 16
# Since for screen, larger n may give color allocation problem
par(mar = c(0, 0, 2, 0))
demo.pal(n)

par(mfrow = c(33, 1), mar = c(0, 0, .8, 0))
for (i in seq(32)) {
  pal(
    n = length((1 + 20 * (i - 1)):(20 * i)),
    colors()[(1 + 20 * (i - 1)):(20 * i)],
    main = paste(1 + 20 * (i - 1), "to", 20 * i)
  )
}
pal(n = 17, colors()[641:657], main = "641 to 657")

library(colorspace)
## a few useful diverging HCL palettes
par(mar = c(0, 0, 2, 0), mfrow = c(16, 2))

pal(n = 16, diverge_hcl(16), main = "diverging HCL palettes")
pal(n = 16, diverge_hcl(16, h = c(246, 40), c = 96, l = c(65, 90)))
pal(n = 16, diverge_hcl(16, h = c(130, 43), c = 100, l = c(70, 90)))
pal(n = 16, diverge_hcl(16, h = c(180, 70), c = 70, l = c(90, 95)))

pal(n = 16, diverge_hcl(16, h = c(180, 330), c = 59, l = c(75, 95)))
pal(n = 16, diverge_hcl(16, h = c(128, 330), c = 98, l = c(65, 90)))
pal(n = 16, diverge_hcl(16, h = c(255, 330), l = c(40, 90)))
pal(n = 16, diverge_hcl(16, c = 100, l = c(50, 90), power = 1))
```

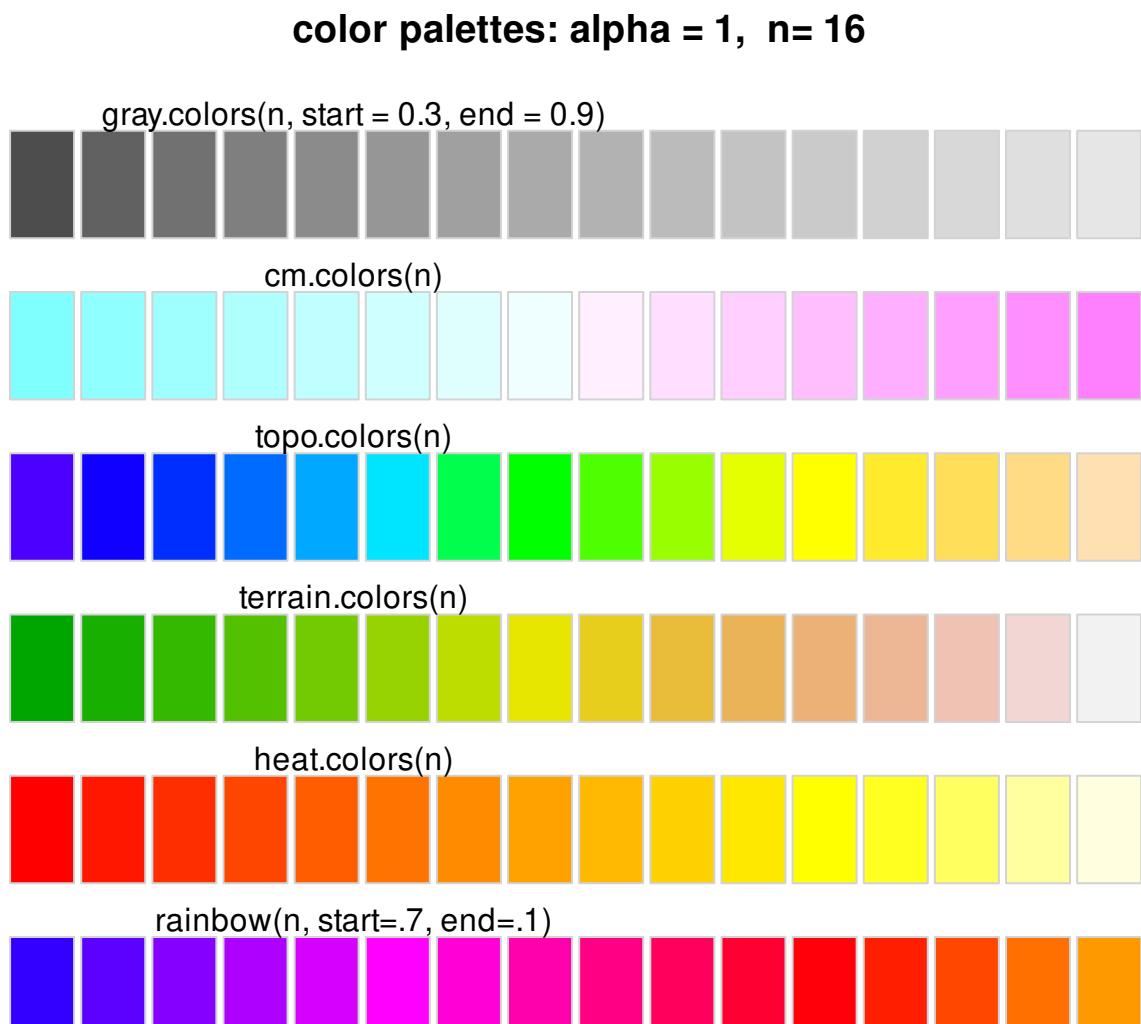


图 11.20: grDevices 调色板

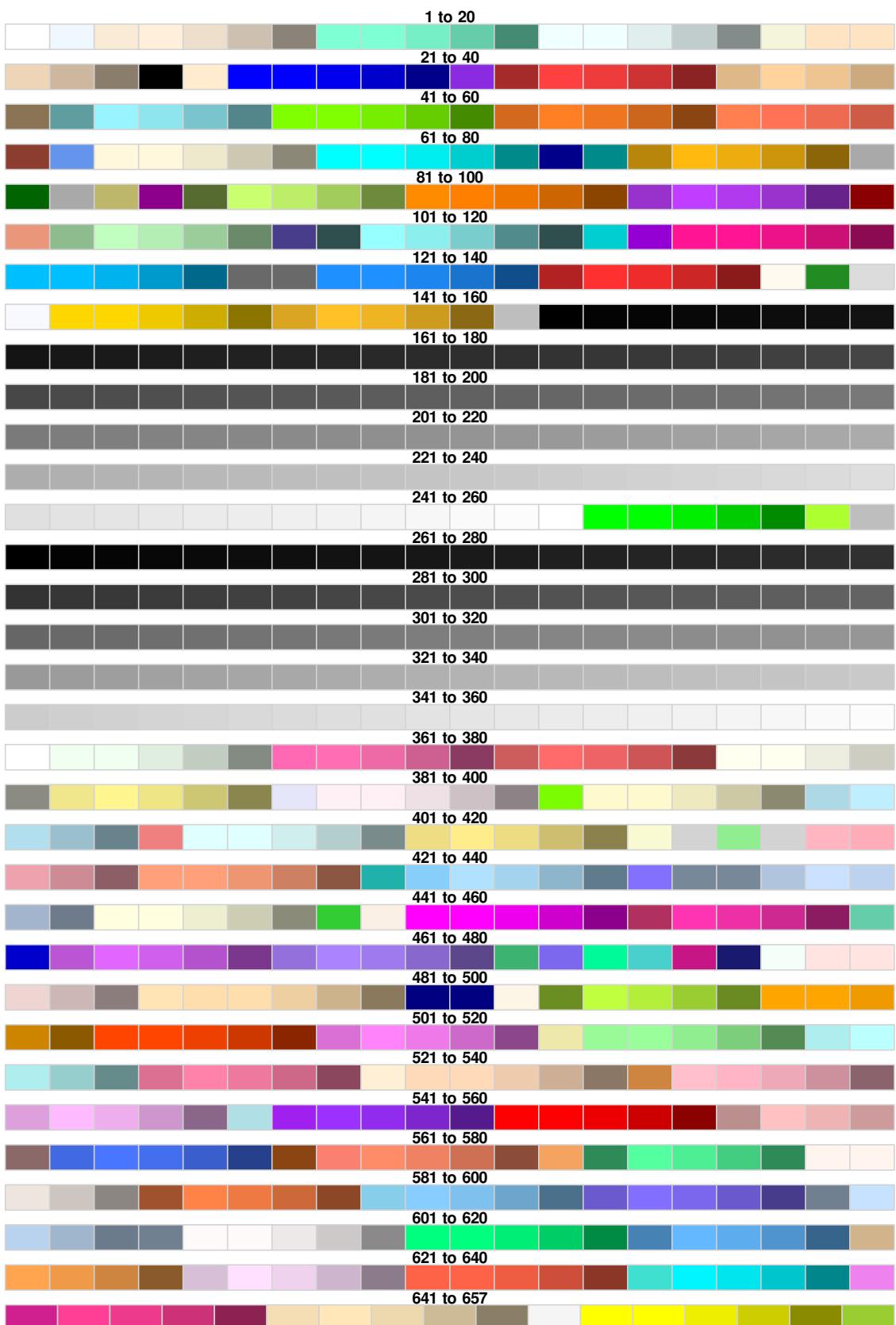


图 11.21: grDevices 调色板



```
## sequential palettes
pal(n = 16, sequential_hcl(16), main= "sequential palettes")
pal(n = 16, heat_hcl(16, h = c(0, -100),
                     l = c(75, 40), c = c(40, 80), power = 1))
pal(n = 16, terrain_hcl(16, c = c(65, 0), l = c(45, 95), power = c(1/3, 1.5)))
pal(n = 16, heat_hcl(16, c = c(80, 30), l = c(30, 90), power = c(1/5, 1.5)))

## compare base and colorspace palettes
## (in color and desaturated)
## diverging red-blue colors
pal(n = 16, diverge_hsv(16), main = "diverging red-blue colors")
pal(n = 16, diverge_hcl(16, c = 100, l = c(50, 90)))
pal(n = 16, desaturate(diverge_hsv(16)))
pal(n = 16, desaturate(diverge_hcl(16, c = 100, l = c(50, 90)))) 

## diverging cyan-magenta colors
pal(n = 16, cm.colors(16), main = "diverging cyan-magenta colors")
pal(n = 16, diverge_hcl(16, h = c(180, 330), c = 59, l = c(75, 95)))
pal(n = 16, desaturate(cm.colors(16)))
pal(n = 16, desaturate(diverge_hcl(16, h = c(180, 330), c = 59, l = c(75, 95)))) 

## heat colors
pal(n = 16, heat.colors(16), main = "heat colors")
pal(n = 16, heat_hcl(16))
pal(n = 16, desaturate(heat.colors(16)))
pal(n = 16, desaturate(heat_hcl(16)))

## terrain colors
pal(n = 16, terrain.colors(16), main = "terrain colors")
pal(n = 16, terrain_hcl(16))
pal(n = 16, desaturate(terrain.colors(16)))
pal(n = 16, desaturate(terrain_hcl(16)))

pal(n = 16, rainbow_hcl(16, start = 30, end = 300), main = "dynamic")
pal(n = 16, rainbow_hcl(16, start = 60, end = 240), main = "harmonic")
pal(n = 16, rainbow_hcl(16, start = 270, end = 150), main = "cold")
pal(n = 16, rainbow_hcl(16, start = 90, end = -30), main = "warm")
```

除之前提到的 `grDevices` 包, `colorspace` (<https://hclwizard.org/>) 包 [Stauffer et al., 2009, Zeileis et al., 2009, 2019], `RColorBrewer` 包 [Neuwirth, 2014] <https://colorbrewer2.org/>, `viridis` 包、`colourvalues`、`westanderson`、`dichromat` 包、`pals` 包, `palr` 包, `colorRamps` 包、`ColorPalette` 包、`colortools` 包就不一一详细介绍 了。

`colormap` 包基于 `node.js` 的 `colormap` 模块提供 44 个预定义的调色板 `paletteer` 包收集了很多 R 包提供的调色板, 同时也引入了很多依赖。根据电影 Harry Potter 制作的调色板 `harrypotter`, 根据网站 `CARTO` 设计的 `rcartocolor` 包, `colorblindr` 模拟色盲环境下的配色方案。

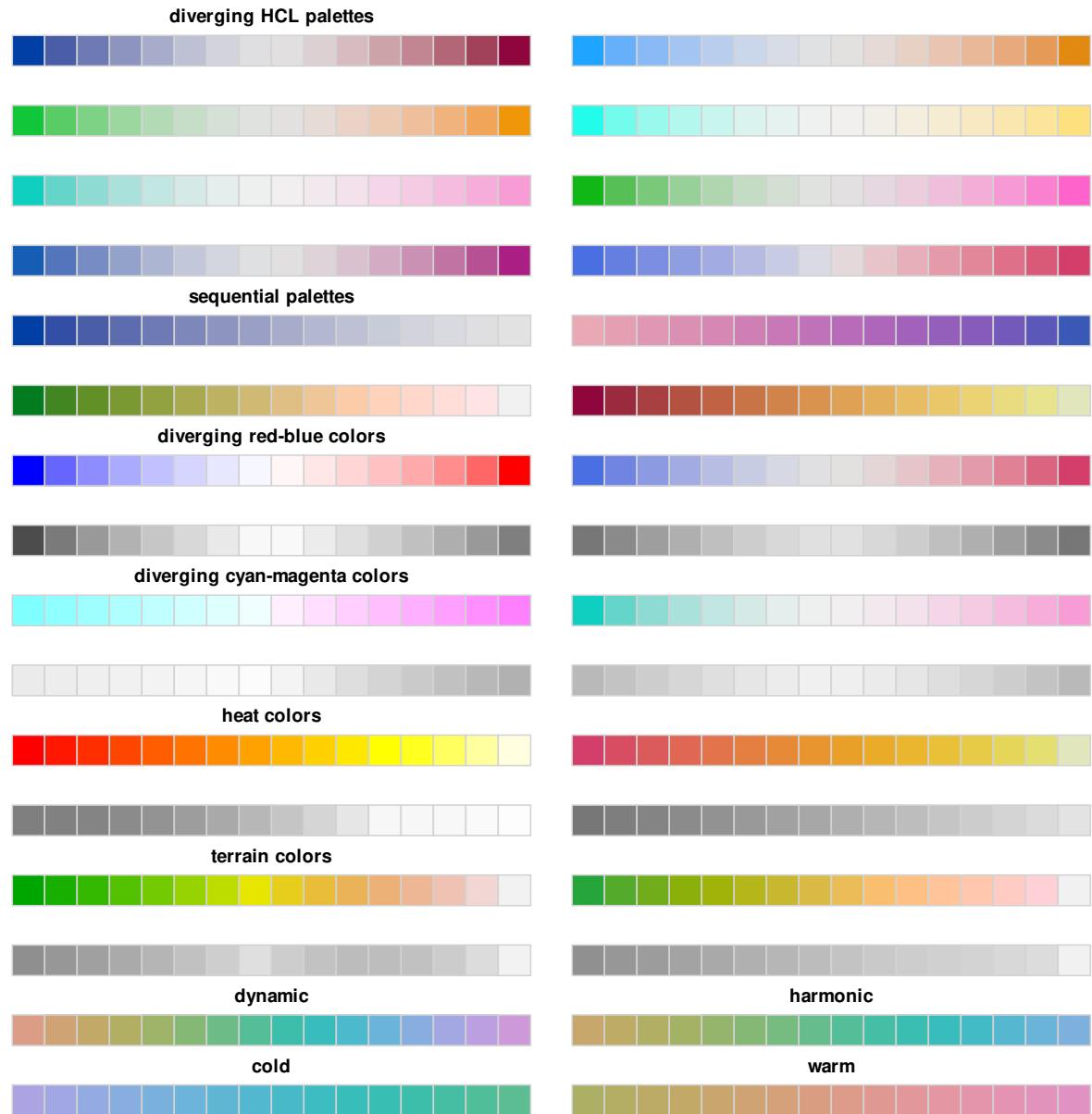


图 11.22: colorspace 调色板



`yarrr` 包主要是为书籍《YaRrr! The Pirate's Guide to R》<https://github.com/ndphillips/ThePiratesGuideToR> 提供配套资源，兼顾收集了一组调色板。

注意

RColorBrewer 调色板数量必须至少 3 个，这是上游 colorbrewer 的 问题，具体体现在调用 `RColorBrewer::brewer.pal(n = 2, name = "Set2")` 时会有警告。plotly 调用

```
[1] "#66C2A5" "#FC8D62" "#8DA0CB"  
Warning message:  
In RColorBrewer::brewer.pal(n = 2, name = "Set2") :  
  minimal value for n is 3, returning requested palette with 3 different levels
```

```
par(mar = c(1, 2, 1, 0), mflow = c(3, 2))  
set.seed(1234)  
x <- sample(seq(8), 8, replace = FALSE)  
barplot(x, col = palette(), border = "white")  
barplot(x, col = heat.colors(8), border = "white")  
barplot(x, col = gray.colors(8), border = "white")  
barplot(x, col = "lightblue", border = "white")  
barplot(x, col = colorspace::sequential_hcl(8), border = "white")  
barplot(x, col = colorspace::diverge_hcl(8,  
  h = c(130, 43),  
  c = 100, l = c(70, 90)  
, border = "white")
```

与图 11.87 对比，图11.24 的层次更加丰富，识别性更高

```
expand.grid(months = month.abb, years = 1949:1960) |>  
  transform(num = as.vector(AirPassengers)) |>  
  ggplot(aes(x = years, y = months, fill = num)) +  
  scale_fill_distiller(palette = "Spectral") +  
  geom_tile(color = "white", size = 0.4) +  
  scale_x_continuous(  
    expand = c(0.01, 0.01),  
    breaks = seq(1949, 1960, by = 1),  
    labels = 1949:1960  
) +  
  theme_minimal(  
    base_size = 10.54,  
    base_family = "Noto Serif SC"  
) +  
  labs(x = "年", y = "月", fill = "人数")
```

再举例子，图 11.25 是正负例对比，其中好在哪里呢？这张图要表达美国黄石国家公园的老忠实泉间歇喷发的时间规律，那么好的标准就是层次分明，以突出不同颜色之间的时间差异。这个差异，还要看起来不那么费眼睛，一目了然最好。

```
erupt <- ggplot(faithful, aes(waiting, eruptions, fill = density)) +  
  geom_raster() +
```

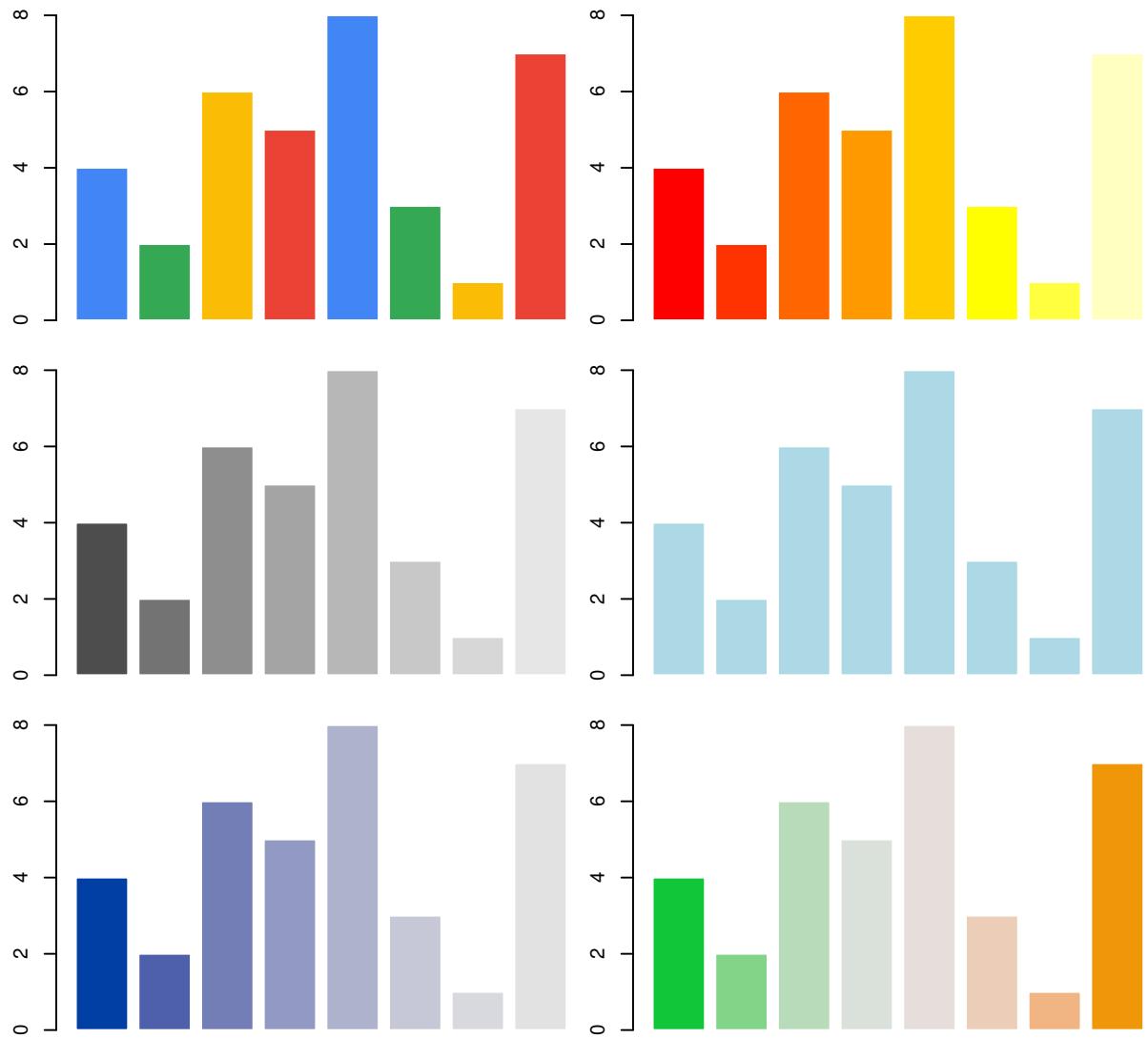


图 11.23: 源起

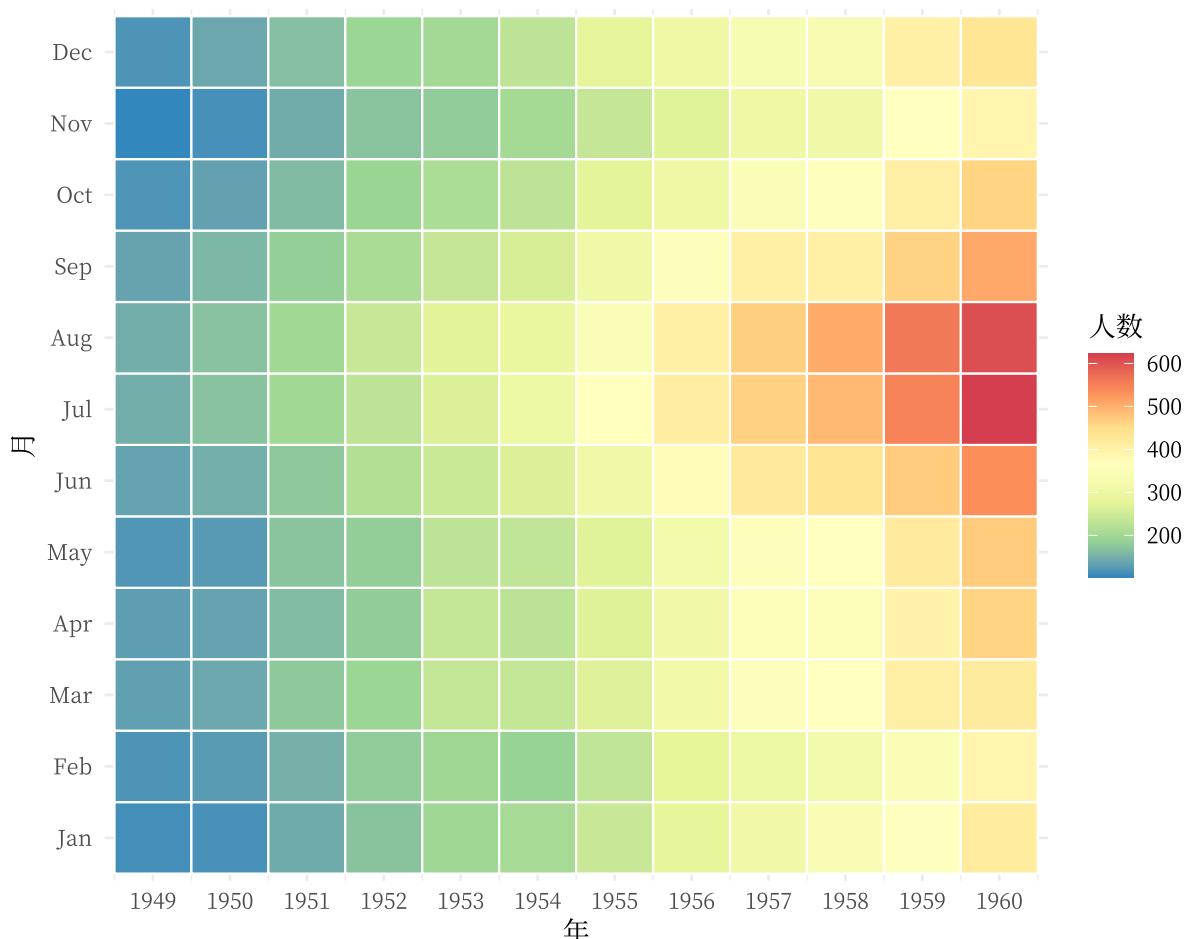


图 11.24: Spectral 调色板

```

scale_x_continuous(NULL, expand = c(0, 0)) +
scale_y_continuous(NULL, expand = c(0, 0)) +
theme(legend.position = "none")
p1 <- erupt + scale_fill_gradientn(colours = gray.colors(7))
p2 <- erupt + scale_fill_distiller(palette = "Spectral")
p3 <- erupt + scale_fill_gradientn(colours = terrain.colors(7))
p4 <- erupt + scale_fill_continuous(type = 'viridis')
(p1 + p2) / (p3 + p4)

```

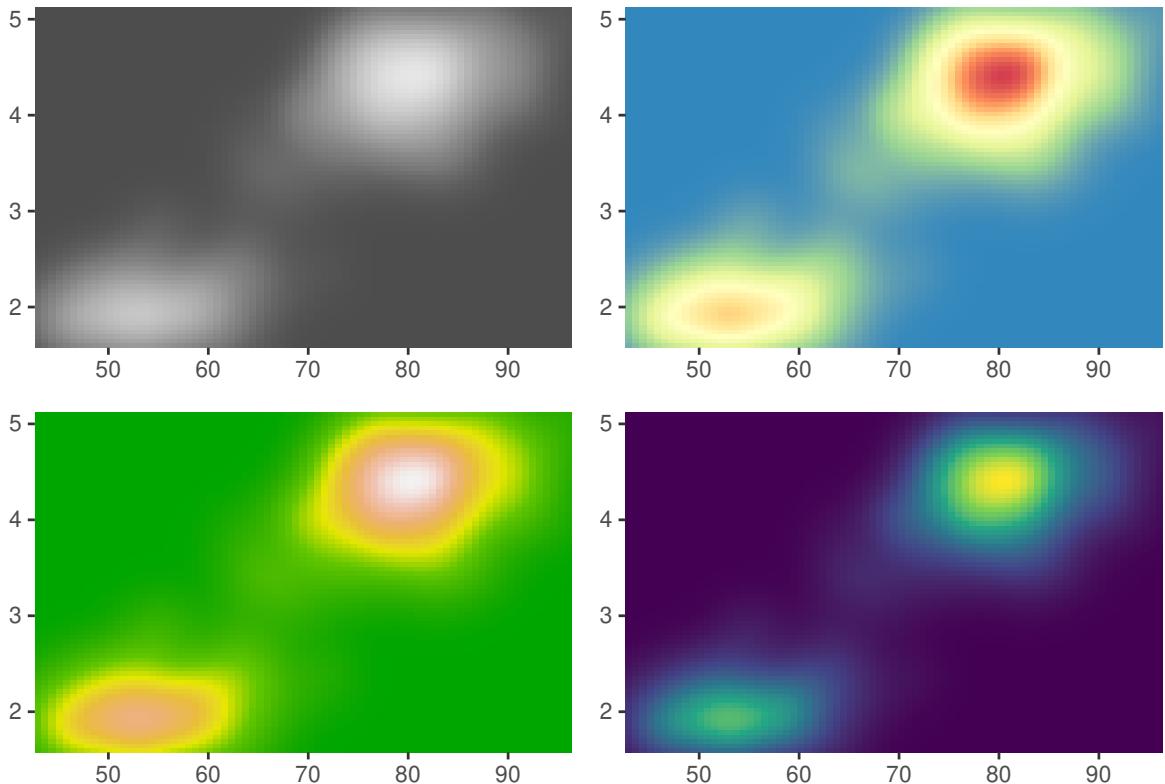


图 11.25: 美国黄石国家公园的老忠实泉

RColorBrewer 包提供了有序 (Sequential)、定性 (Qualitative) 和发散 (Diverging) 三类调色板，一般来讲，分别适用于连续或有序分类变量、无序分类变量、两类分层对比变量的绘图。再加上强大的 ggplot2 包内置的对颜色处理的函数，如 `scale_alpha_*`、`scale_colour_*` 和 `scale_fill_*` 等，详见：

```
ls("package:ggplot2", pattern = "scale_col(ou|o)r_")
```

```

## [1] "scale_color_binned"      "scale_color_brewer"
## [3] "scale_color_continuous"  "scale_color_date"
## [5] "scale_color_datetime"    "scale_color_discrete"
## [7] "scale_color_distiller"   "scale_color_fermenter"
## [9] "scale_color_gradient"    "scale_color_gradient2"
## [11] "scale_color_gradientn"   "scale_color_grey"
## [13] "scale_color_hue"        "scale_color_identity"
## [15] "scale_color_manual"     "scale_color_ordinal"

```



```
## [17] "scale_color_steps"      "scale_color_steps2"
## [19] "scale_color_stepsn"     "scale_color_viridis_b"
## [21] "scale_color_viridis_c"   "scale_color_viridis_d"
## [23] "scale_colour_binned"    "scale_colour_brewer"
## [25] "scale_colour_continuous" "scale_colour_date"
## [27] "scale_colour_datetime"   "scale_colour_discrete"
## [29] "scale_colour_distiller"   "scale_colour_fermenter"
## [31] "scale_colour_gradient"    "scale_colour_gradient2"
## [33] "scale_colour_gradientn"   "scale_colour_grey"
## [35] "scale_colour_hue"        "scale_colour_identity"
## [37] "scale_colour_manual"     "scale_colour_ordinal"
## [39] "scale_colour_steps"      "scale_colour_steps2"
## [41] "scale_colour_stepsn"     "scale_colour_viridis_b"
## [43] "scale_colour_viridis_c"   "scale_colour_viridis_d"

ls("package:ggplot2", pattern = "scale_fill_")

## [1] "scale_fill_binned"      "scale_fill_brewer"      "scale_fill_continuous"
## [4] "scale_fill_date"         "scale_fill_datetime"   "scale_fill_discrete"
## [7] "scale_fill_distiller"    "scale_fill_fermenter"  "scale_fill_gradient"
## [10] "scale_fill_gradient2"    "scale_fill_gradientn"  "scale_fill_grey"
## [13] "scale_fill_hue"          "scale_fill_identity"   "scale_fill_manual"
## [16] "scale_fill_ordinal"      "scale_fill_steps"      "scale_fill_steps2"
## [19] "scale_fill_stepsn"       "scale_fill_viridis_b"   "scale_fill_viridis_c"
## [22] "scale_fill_viridis_d"
```

colourlovers 包借助 XML, jsonlite 和 httr 包可以在线获取网站 [COLOURlovers](#) 的调色板

```
library(colourlovers)
palette1 <- clpalette('113451')
palette2 <- clpalette('92095')
palette3 <- clpalette('629637')
palette4 <- clpalette('694737')
```

使用调色板

```
layout(matrix(1:4, nrow = 2))
par(mar = c(2, 2, 2, 2))

barplot(VADeaths, col = swatch(palette1)[[1]], border = NA)
barplot(VADeaths, col = swatch(palette2)[[1]], border = NA)
barplot(VADeaths, col = swatch(palette3)[[1]], border = NA)
barplot(VADeaths, col = swatch(palette4)[[1]], border = NA)
```

调色板的描述信息

```
palette1
```

获取调色板中的颜色向量



```
swatch(palette1)[[1]]
```

### 11.3.2 颜色模式

不同的颜色模式，从 RGB 到 HCL 的基本操作 [https://stat545.com/block018\\_colors.html](https://stat545.com/block018_colors.html)

```
# https://github.com/hadley/ggplot2-book
hcl <- expand.grid(x = seq(-1, 1, length = 100), y = seq(-1, 1, length = 100)) |>
  subset(subset = x^2 + y^2 < 1) |>
  transform(
    r = sqrt(x^2 + y^2)
  ) |>
  transform(
    h = 180 / pi * atan2(y, x),
    c = 100 * r,
    l = 65
  ) |>
  transform(
    colour = hcl(h, c, l)
  )

# sin(h) = y / (c / 100)
# y = sin(h) * c / 100

cols <- scales::hue_pal()(5)
selected <- colorspace::RGB(t(col2rgb(cols)) / 255) %>%
  as("polarLUV") %>%
  colorspace::coords() %>%
  as.data.frame() %>%
  transform(
    x = cos(H / 180 * pi) * C / 100,
    y = sin(H / 180 * pi) * C / 100,
    colour = cols
  )

ggplot(hcl, aes(x, y)) +
  geom_raster(aes(fill = colour)) +
  scale_fill_identity() +
  scale_colour_identity() +
  coord_equal() +
  scale_x_continuous("", breaks = NULL) +
  scale_y_continuous("", breaks = NULL) +
  geom_point(data = selected, size = 10, color = "white") +
  geom_point(data = selected, size = 5, aes(colour = colour))
```

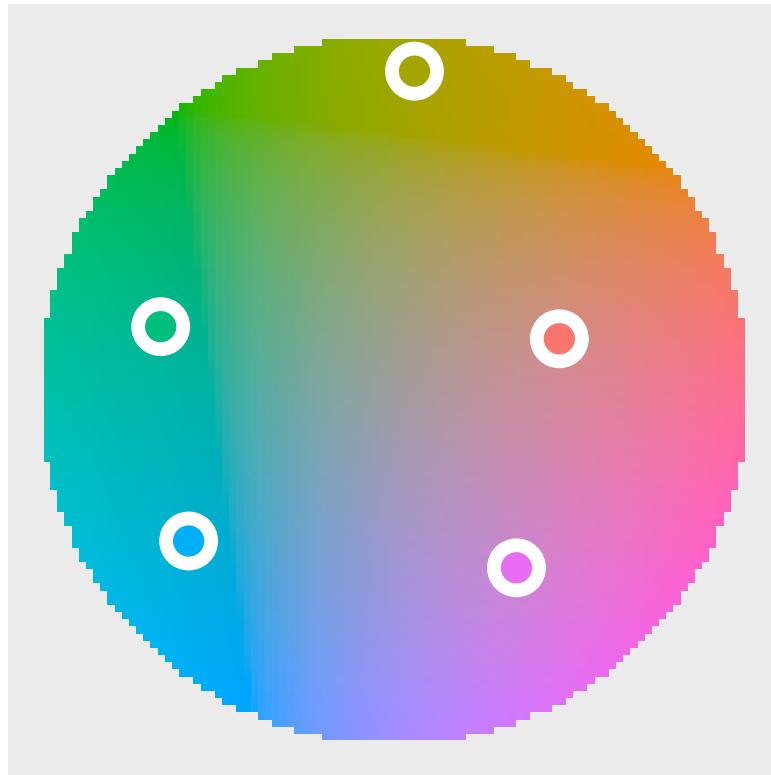


图 11.26: HCL 调色

R 内置了 502 种不同颜色的名称，下面随机地选取 20 种颜色

```
sample(colors(TRUE), 20)

## [1] "royalblue4"      "plum1"        "papayawhip"    "darkslategray"
## [5] "darkturquoise"  "gray79"        "darkred"       "maroon4"
## [9] "darkolivegreen4" "springgreen2"  "orchid4"       "lemonchiffon2"
## [13] "paleturquoise4" "gray49"        "cyan"          "antiquewhite1"
## [17] "yellow2"        "gray13"        "cadetblue2"   "gray77"
```

R 包 grDevices 提供 hcl 调色板<sup>8</sup> 调制两个色板

```
# Colors from https://github.com/johannesbjork/LaCroixColor
color_pal <- c("#FF3200", "#E9A17C", "#E9E4A6", "#1BB6AF", "#0076BB", "#172869")
n <- 16
more_colors <- (grDevices::colorRampPalette(color_pal))(n)
scales::show_col(colours = more_colors)

# colors in colortools from http://www.gastonsanchez.com/
fish_pal <- c(
  "#69D2E7", "#6993E7", "#7E69E7", "#BD69E7",
  "#E769D2", "#E76993", "#E77E69", "#E7BD69",
  "#D2E769", "#93E769", "#69E77E", "#69E7BD"
```

<sup>8</sup><https://developer.r-project.org/Blog/public/2019/04/01/hcl-based-color-palettes-in-grdevices/index.html>

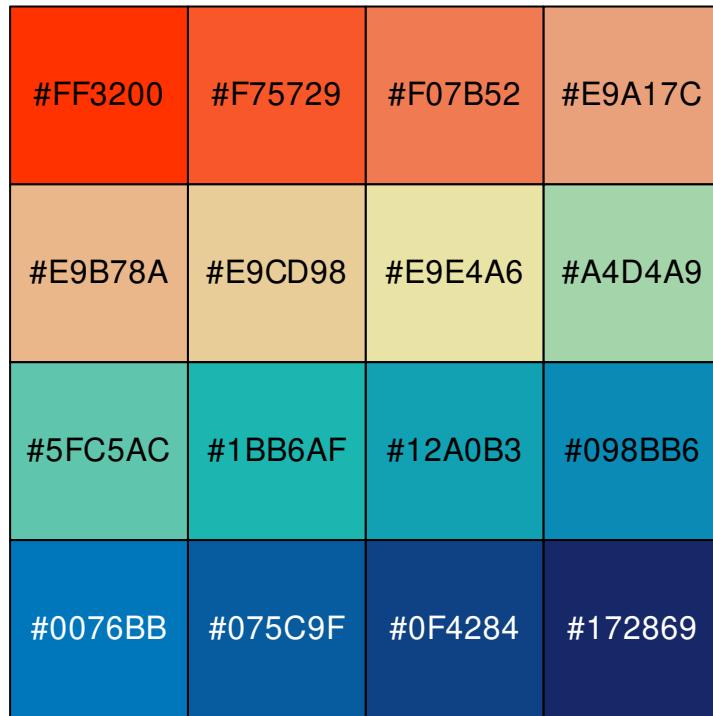


图 11.27: 桃色至梨色的渐变

```

)
more_colors <- (grDevices::colorRampPalette(fish_pal))(n)
scales::show_col(colours = more_colors)

rgb(red = 86, green = 180, blue = 233, maxColorValue = 255) # "#56B4E9"

## [1] "#56B4E9"

rgb(red = 0, green = 158, blue = 115, maxColorValue = 255) # "#009E73"

## [1] "#009E73"

rgb(red = 240, green = 228, blue = 66, maxColorValue = 255) # "#F0E442"

## [1] "#F0E442"

rgb(red = 0, green = 114, blue = 178, maxColorValue = 255) # "#0072B2"

## [1] "#0072B2"

```

举例子，直方图配色与不配色

```

# library(pander)
# evalsOptions('graph.unify', TRUE)
# panderOptions('graph.colors') 获取调色板
# https://www.fontke.com/tool/rgbschemes/ 在线配色
cols <- c(
  "#56B4E9", "#009E73", "#F0E442", "#0072B2",
  "#D55E00", "#CC79A7", "#999999", "#E69F00"
)

```



图 11.28: Hue-Saturation-Value (HSV) 颜色模型

```
)  
hist(mtcars$hp, col = "#56B4E9", border = "white", grid = grid())  
  
ggplot(mtcars) +  
  geom_histogram(aes(x = hp, fill = as.factor(..count..)),  
    color = "white", bins = 6  
) +  
  scale_fill_manual(values = rep("#56B4E9", 10)) +  
  ggtitle("Histogram with ggplot2") +  
  theme_minimal() +  
  theme(legend.position = "none")
```

### 11.3.2.1 RGB

红 (red)、绿 (green)、蓝 (blue) 是三原色

```
rgb(red, green, blue, alpha, names = NULL, maxColorValue = 1)
```

函数参数说明：

- red, blue, green, alpha 取值范围  $[0, M]$ ,  $M$  是 maxColorValue
- names 字符向量, 给这组颜色值取名
- maxColorValue 红, 绿, 蓝三色范围的最大值

The colour specification refers to the standard sRGB colorspace (IEC standard 61966).

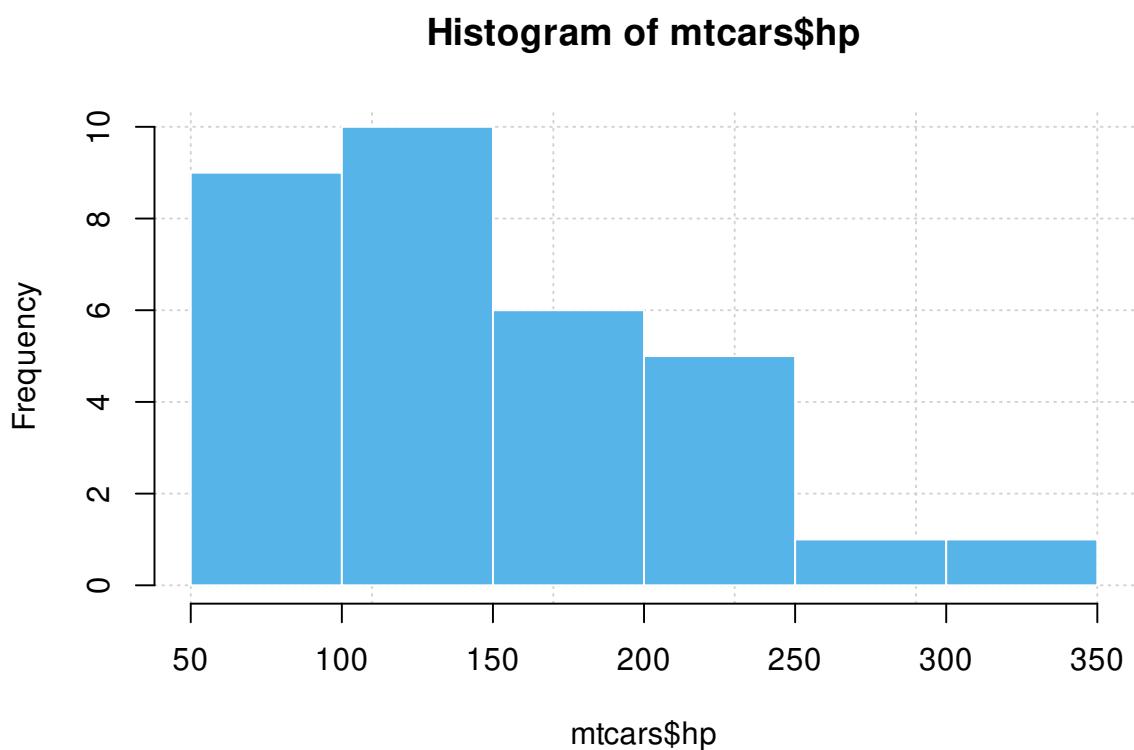


图 11.29: 直方图

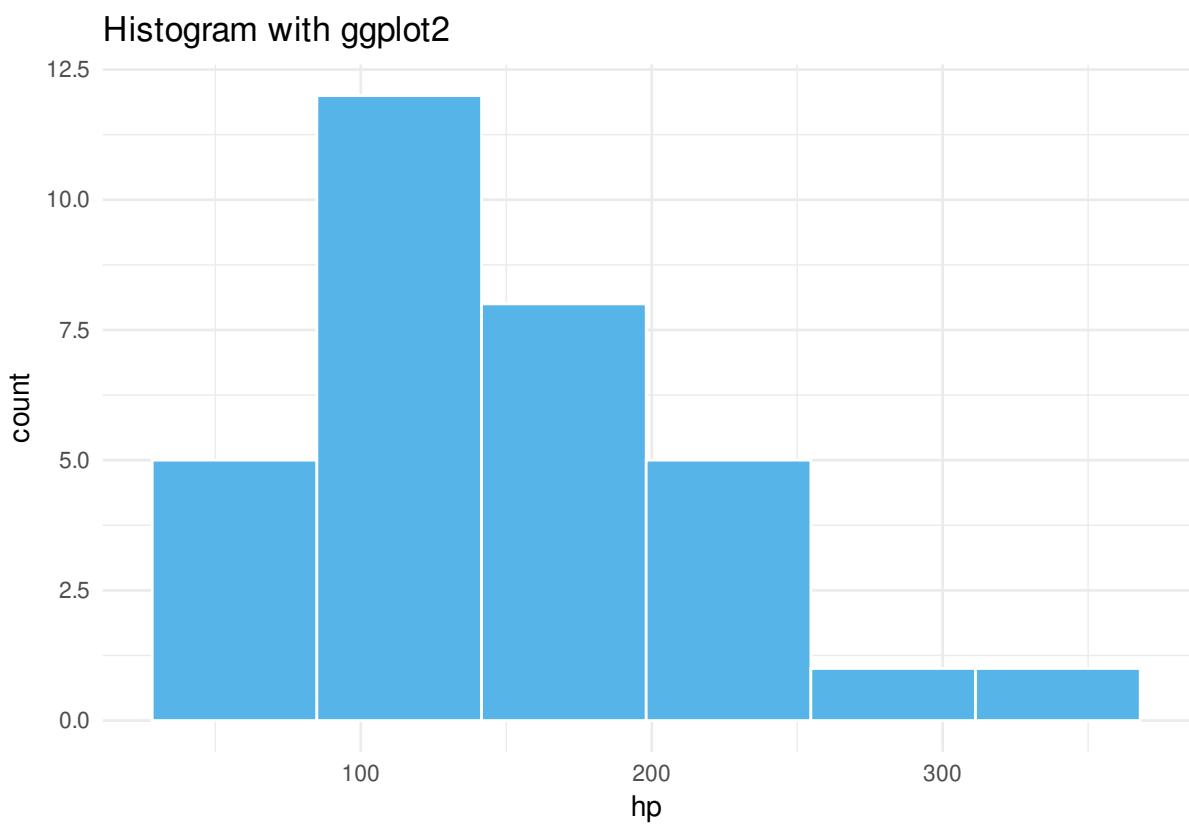


图 11.30: 直方图



rgb 产生一种颜色, 如 `rgb(255, 0, 0, maxColorValue = 255)` 的颜色是 "#FF0000", 这是一串 16 进制数, 每两个一组, 那么一组有  $16^2 = 256$  种组合, 整个一串有  $256^3 = 16777216$  种组合, 这就是 RGB 表达的所有颜色。

### 11.3.2.2 HSL

色相饱和度亮度 hue-saturation-luminance (HSL)

### 11.3.2.3 HSV

Create a vector of colors from vectors specifying hue, saturation and value. 色相饱和度值

```
hsv(h = 1, s = 1, v = 1, alpha)
```

This function creates a vector of colors corresponding to the given values in HSV space. `rgb` and `rgb2hsv` for RGB to HSV conversion;

`hsv` 函数通过设置色调、饱和度和亮度获得颜色, 三个值都是 0-1 的相对量

RGB HSV HSL 都是不连续的颜色空间, 缺点

### 11.3.2.4 HCL

基于感知的颜色空间替代 RGB 颜色空间

通过指定色相 (hue), 色度 (chroma) 和亮度 (luminance/lightness), 创建一组 (种) 颜色

```
hcl(h = 0, c = 35, l = 85, alpha, fixup = TRUE)
```

函数参数说明:

- **h** 颜色的色调, 取值范围 [0,360], 0、120、240 分别对应红色、绿色、蓝色
- **c** 颜色的色度, 其上界取决于色调和亮度
- **l** 颜色的亮度, 取值范围 [0,100], 给定色调和色度, 只有一部分子集可用
- **alpha** 透明度, 取值范围 [0,1], 0 和 1 分别表示透明和不透明

This function corresponds to polar coordinates in the CIE-LUV color space

选色为什么这么难

色相与阴影相比是无关紧要的, 色相对于标记和分类很有用, 但表示 (精细的) 空间数据或形状的效果较差。颜色是改善图形的好工具, 但糟糕的配色方案 (color schemes) 可能会导致比灰度调色板更差的效果。  
[Stauffer et al., 2009]

黑、白、灰, 看似有三种颜色, 其实只有一种颜色, 黑和白只是灰色的两极, 那么如何设置灰色梯度, 使得人眼比较好区分它们呢? 这样获得的调色板适用于什么样的绘图环境呢?



### 11.3.2.5 CMYK

印刷三原色：青 (cyan)、品红 (magenta)、黄 (yellow)

- 颜色模式转化

col2rgb()、rgb2hsv() 和 rgb() 函数 hex2RGB() 函数 colorspace col2hcl() 函数 scales col2HSV() colortools col2hex()

```
col2rgb("lightblue") # color to RGB
```

```
## [1] [,1]
## red    173
## green   216
## blue    230

scales::col2hcl("lightblue") # color to HCL

## [1] "#ADD8E6"

# palr::col2hex("lightblue") # color to HEX
# colortools::col2HSV("lightblue") # color to HSV

rgb(173, 216, 230, maxColorValue = 255) # RGB to HEX

## [1] "#ADD8E6"

colorspace::hex2RGB("#ADD8E6") # HEX to RGB
```

```
##          R          G          B
## [1,] 0.6784314 0.8470588 0.9019608

rgb(.678, .847, .902, maxColorValue = 1) # RGB to HEX
```

```
## [1] "#ADD8E6"

rgb2hsv(173, 216, 230, maxColorValue = 255) # RGB to HSV
```

```
## [1]
## h 0.5409357
## s 0.2478261
## v 0.9019608
```

### 11.3.3 LaTeX 配色

LaTeX 宏包 `xcolor` 中定义颜色的常用方式有两种，其一，`\textcolor{green!40!yellow}` 表示 40% 的绿色和 60% 的黄色混合色彩，其二，`\textcolor[HTML]{34A853}` HEX 表示的色彩直接在 LaTeX 文档中使用的方式，类似地 `\textcolor[RGB]{52,168,83}` 也表示 Google 图标中的绿色。

```
\documentclass[tikz, border=10pt]{standalone}
\begin{document}
\begin{tikzpicture}
\draw (0,0) rectangle (2,1) node [midway] {\textcolor[RGB]{52,168,83}{Hello} \textcolor[HTML]{34A853}
```



```
\end{tikzpicture}
\end{document}
```

对应于 R 中的调用方式:

```
rgb(52, 168, 83, maxColorValue = 255)
```



```
## [1] "#34A853"
```

### 11.3.4 ggplot2 配色

```
boxplot(weight ~ group,
  data = PlantGrowth, col = "lightgray",
  notch = FALSE, varwidth = TRUE
)
# 类似 boxplot
ggplot(data = PlantGrowth, aes(x = group, y = weight)) +
  geom_boxplot(notch = FALSE, varwidth = TRUE, fill = "lightgray")

# 默认调色板
ggplot(data = PlantGrowth, aes(x = group, y = weight, fill = group)) +
  geom_boxplot(notch = FALSE, varwidth = TRUE)

# Google 调色板
ggplot(data = PlantGrowth, aes(x = group, y = weight, fill = group)) +
  geom_boxplot(notch = FALSE, varwidth = TRUE) +
  scale_fill_manual(values = c("#4285f4", "#34A853", "#FBBC05", "#EA4335"))
```

## 11.4 图库

### 11.4.1 饼图

我对饼图是又爱又恨，爱的是它表示百分比的时候，往往让读者联想到蛋糕，份额这类根深蒂固的情景，从而让数字通俗易懂、深入人心，是一种很好的表达方式，恨的也是这一点，我用柱状图表达不香吗？人眼对角度的区分度远不如柱状图呢，特别是当两个类所占的份额比较接近的时候，所以很多时候，除了用饼图表达份额，还会在旁边标上百分比，从数据可视化的角度来说，如图 11.32 所示，这是信息冗余！

```
BOD %>% transform(., ratio = demand / sum(demand)) %>%
  ggplot(., aes(x = "", y = demand, fill = reorder(Time, demand))) +
  geom_bar(stat = "identity", show.legend = FALSE, color = "white") +
  coord_polar(theta = "y") +
  geom_text(aes(x = 1.6, label = paste0(round(ratio, digits = 4) * 100, "%")),
            position = position_stack(vjust = 0.5), color = "black")
  ) +
  geom_text(aes(x = 1.2, label = Time),
```

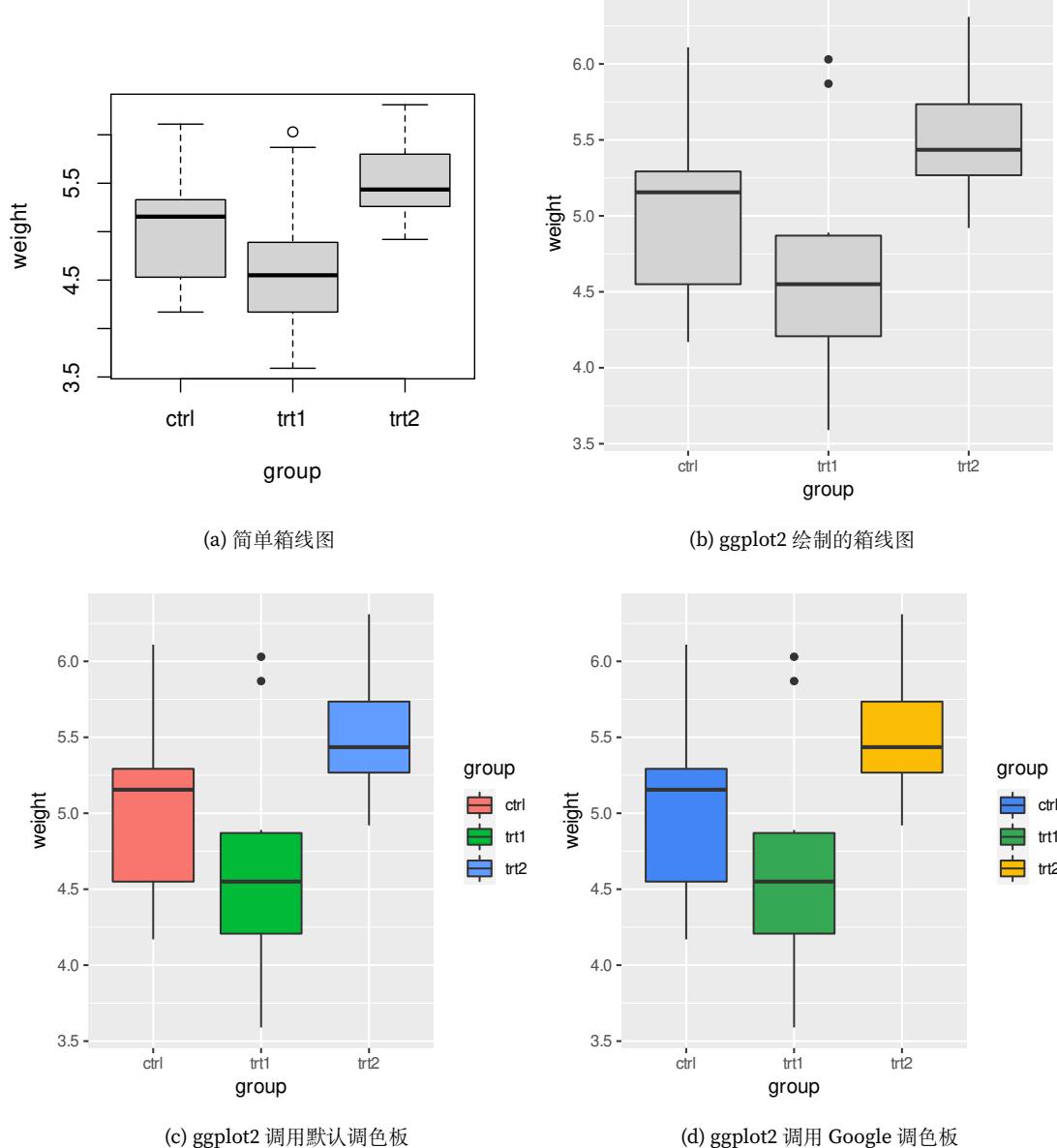


图 11.31: 几种不同的箱线图

```
position = position_stack(vjust = 0.5), color = "black"
) +
theme_void(base_size = 14)
```

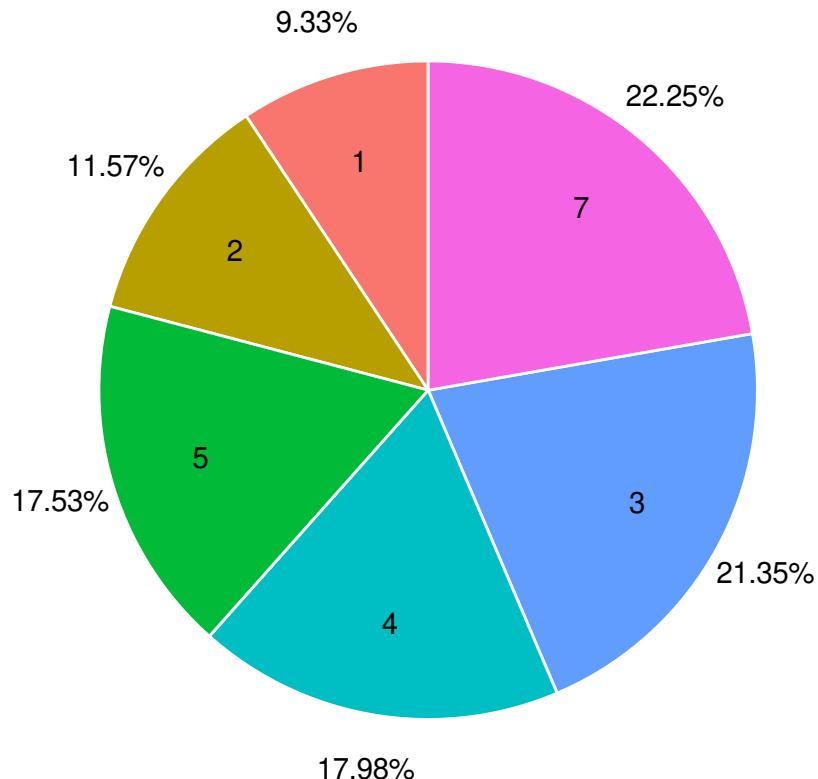


图 11.32: 饼图

`plot_ly(type = "pie", ... )` 和添加图层 `add_pie()` 的效果是一样的

```
dat = aggregate(carat ~ cut, data = diamonds, FUN = length)
plotly::plot_ly() %>%
  plotly::add_pie(
    data = dat, labels = ~cut, values = ~carat,
    name = "简单饼图1", domain = list(row = 0, column = 0)
  ) %>%
  plotly::add_pie(
    data = dat, labels = ~cut, values = ~carat, hole = 0.6,
    textposition = "inside", textinfo = "label+percent",
    name = "简单饼图2", domain = list(row = 0, column = 1)
  ) %>%
  plotly::layout(
    title = "多图布局", showlegend = F,
    grid = list(rows = 1, columns = 2),
```

```
xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE)
) %>%
plotly::config(displayModeBar = FALSE)
```

设置参数 `hole` 可以绘制环形饼图，比如 `hole = 0.6`

### 11.4.2 地图

USArrests 数据集描述了 1973 年美国 50 个州每 10 万居民中因袭击、抢劫和强奸而逮捕的人，以及城市人口占比。这里的地图是指按照行政区划为边界的示意图，比如图 11.33

```
library(maps)
crimes <- data.frame(state = tolower(rownames(USArrests)), USArrests)
# 等价于 crimes %>% tidyr::pivot_longer(Murder:Rape)
vars <- lapply(names(crimes)[-1], function(j) {
  data.frame(state = crimes$state, variable = j, value = crimes[[j]])
})
crimes_long <- do.call("rbind", vars)
states_map <- map_data("state")
ggplot(crimes, aes(map_id = state)) +
  geom_map(aes(fill = Murder), map = states_map) +
  expand_limits(x = states_map$long, y = states_map$lat) +
  scale_fill_binned(type = "viridis") +
  coord_map() +
  theme_minimal()
```

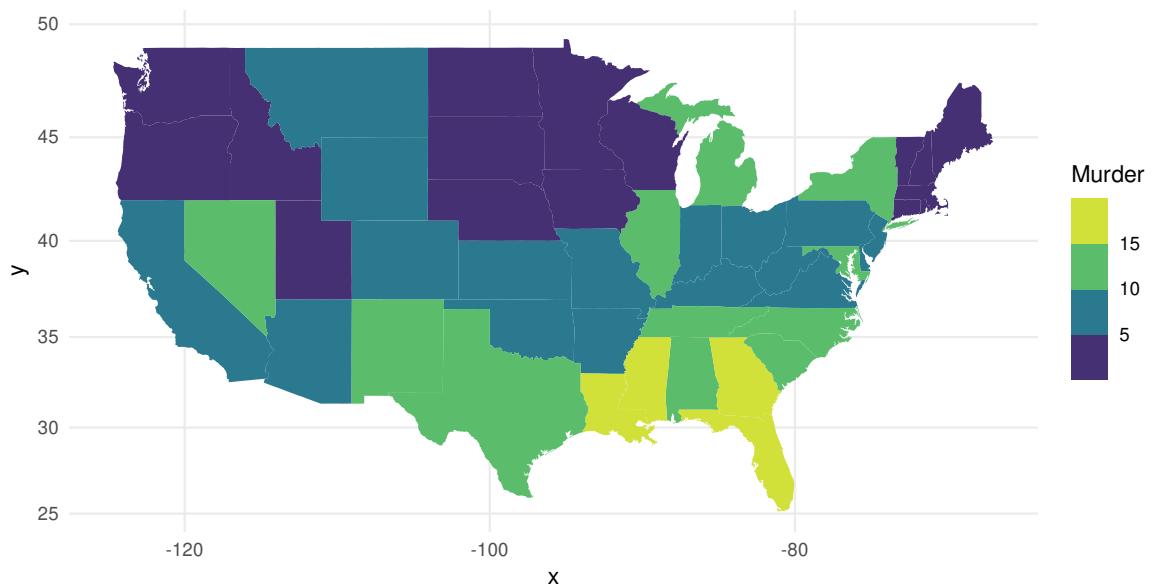


图 11.33: 1975 年美国各州犯罪事件

先来看看中国及其周边，见图 11.34，这个地图的缺陷就是中国南海及九段线没有标记，台湾和中国大陆不是一种颜色标记，这里的地图数据来自 R 包 `maps` 和 `mapdata`，像这样的地图就不宜在国内正式刊物

```
library(maps)
library(mapdata)
east_asia <- map_data("worldHires",
  region = c(
    "Japan", "Taiwan", "China",
    "North Korea", "South Korea"
  )
)
ggplot(east_asia, aes(x = long, y = lat, group = group, fill = region)) +
  geom_polygon(colour = "black") +
  scale_fill_brewer(palette = "Set2") +
  coord_map() +
  theme_minimal()
```

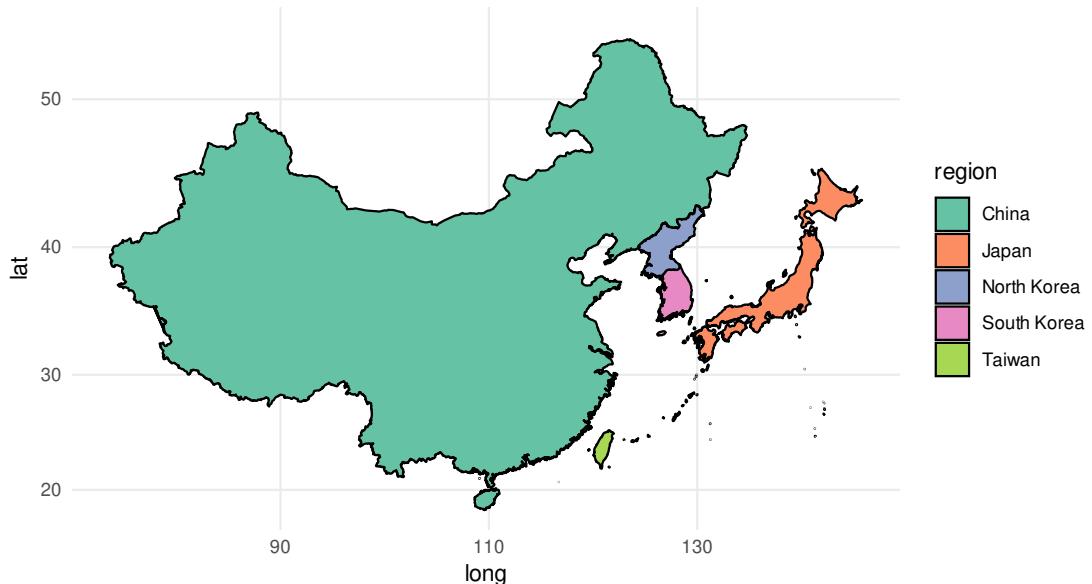


图 11.34: 中国及其周边

绘制真正的地图需要考虑投影坐标系、观察角度、分辨率、政策法规等一系列因素，它是一种复杂的图形，如图 11.35 所示。

```
worldmap <- map_data("world")

# 默认 mercator 投影下的默认视角 c(90, 0, mean(range(x)))
ggplot(worldmap, aes(long, lat, group = group)) +
  geom_polygon(aes(fill = region), show.legend = FALSE) +
  coord_map(
    xlim = c(-120, 40), ylim = c(30, 90)
  )

# 换观察角度
```



```
ggplot(worldmap, aes(long, lat, group = group)) +
  geom_polygon(aes(fill = region), show.legend = FALSE) +
  coord_map(
    xlim = c(-120, 40), ylim = c(30, 90),
    orientation = c(90, 0, 0)
  )

# 换投影坐标系
ggplot(worldmap, aes(long, lat, group = group)) +
  geom_polygon(aes(fill = region), show.legend = FALSE) +
  coord_map("ortho",
    xlim = c(-120, 40), ylim = c(30, 90)
  )

# 二者皆换
ggplot(worldmap, aes(long, lat, group = group)) +
  geom_polygon(aes(fill = region), show.legend = FALSE) +
  coord_map("ortho",
    xlim = c(-120, 40), ylim = c(30, 90),
    orientation = c(90, 0, 0)
  )
```

### Google 地图

```
library(RgoogleMaps)

# 一组坐标的中心位置
lat <- c(40.702147, 40.718217, 40.711614)
lon <- c(-74.012318, -74.015794, -73.998284)
center <- c(mean(lat), mean(lon))
zoom <- min(MaxZoom(range(lat), range(lon)))

# 矩形对角线的两个顶点
bb <- qbbox(lat, lon)

# 获取地图数据
myMap <- GetMap(center, size = c(640, 640), zoom = zoom, type = "osm")
# 在地图上添加红、蓝、绿三个点
PlotOnStaticMap(myMap,
  lat = lat, lon = lon, pch = 20, cex = 10,
  col = c("red", "blue", "green"))
)
```

### 11.4.3 热图

Zuguang Gu 开发的 **ComplexHeatmap** 包实现复杂数据的可视化，用以发现关联数据集之间的模式。特别地，比如基因数据、生存数据等，更多应用见开发者的书籍 **ComplexHeatmap 完全手册**。R 包发布在 Bioconductor 上 <https://www.bioconductor.org/packages/ComplexHeatmap>。使用之前我要确保已经安装

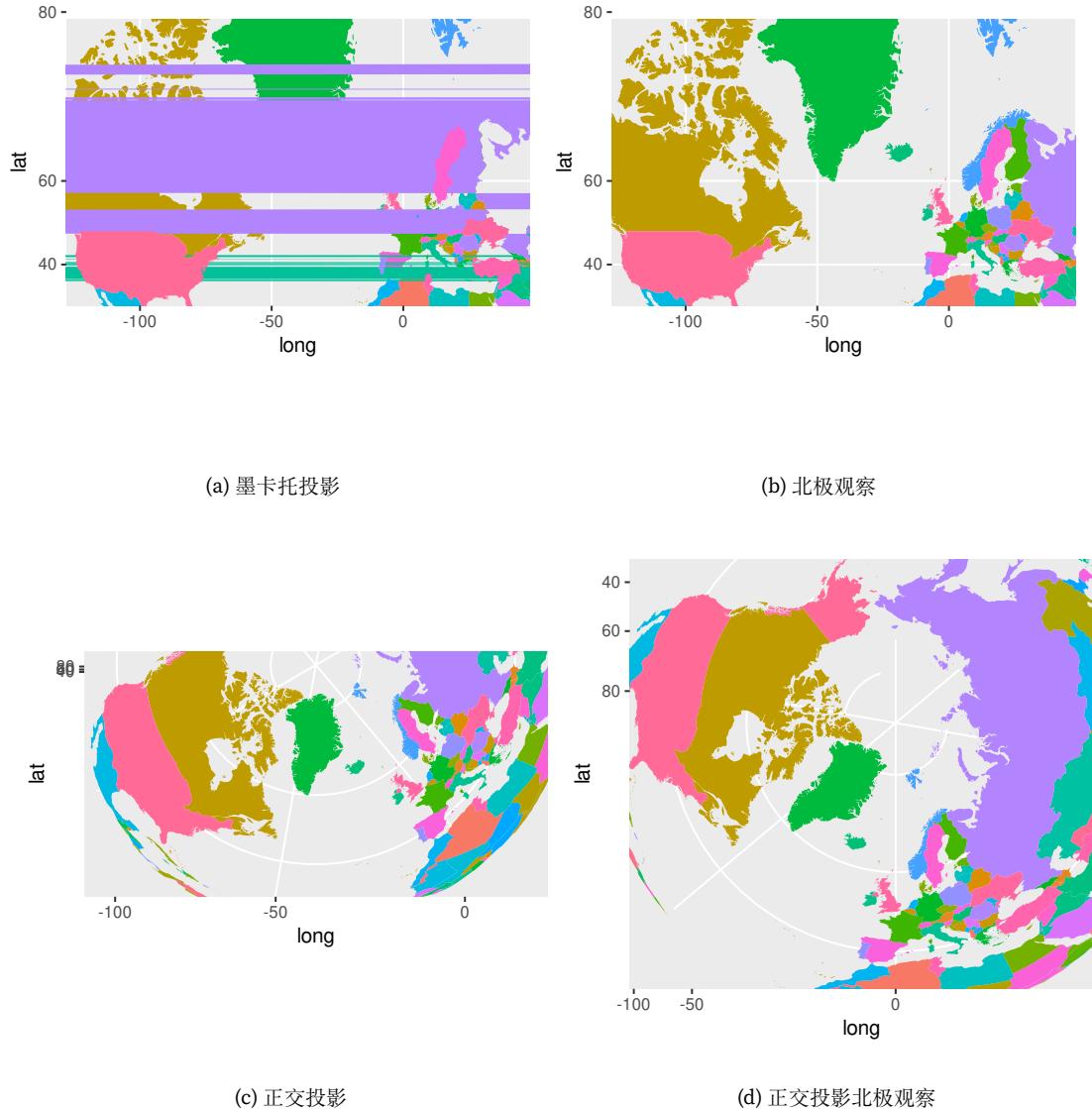


图 11.35: 画地图的正确姿势

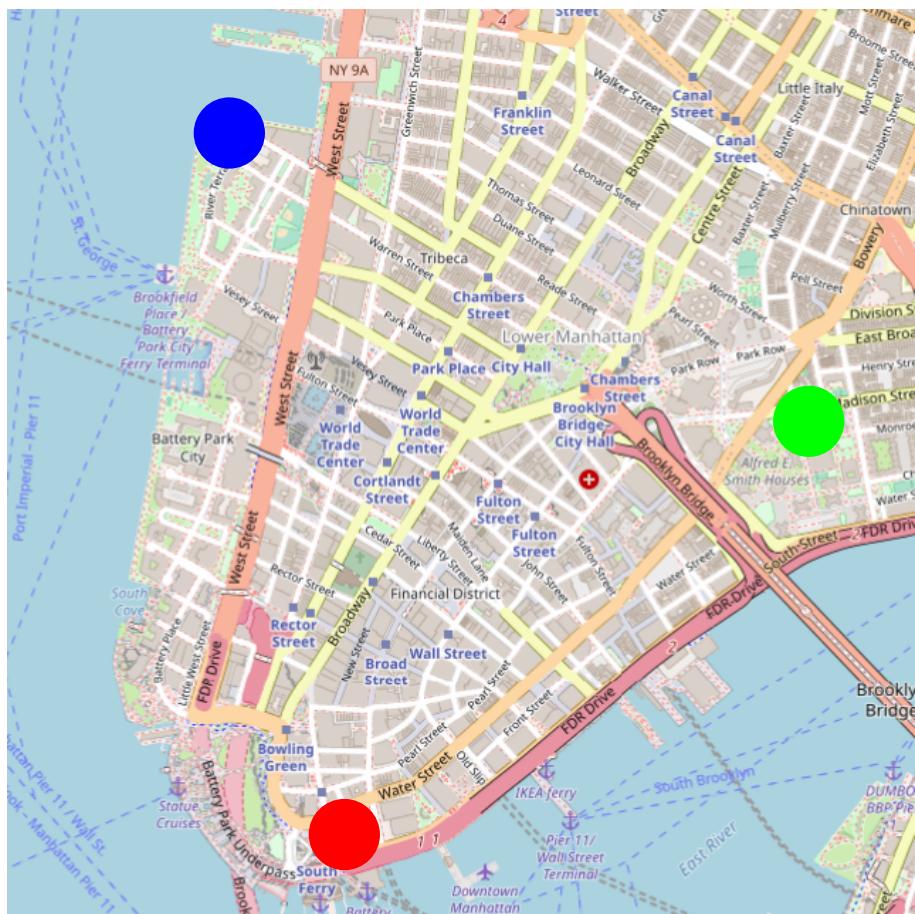


图 11.36: Google 地图示例



**BiocManager** 包,这个包负责管理 Bioconductor 上所有的包,需要先安装它,然后安装 **ComplexHeatmap** 包 [Gu et al., 2016]。

```
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install("ComplexHeatmap")
```



#### 11.4.4 散点图

下面以 diamonds 数据集为例展示 ggplot2 的绘图过程，首先加载 diamonds 数据集，查看数据集的内容

```
data(diamonds)  
str(diamonds)
```

```
## # tibble [53,940 x 10] (S3:tbl_df/tbl/data.frame)
## # $ carat  : num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
## # $ cut    : Ord.factor w/ 5 levels "Fair" < "Good" < ...: 5 4 2 4 2 3 3 3 1 3 ...
## # $ color  : Ord.factor w/ 7 levels "D" < "E" < "F" < "G" < ...: 2 2 2 6 7 7 6 5 2 5 ...
## # $ clarity: Ord.factor w/ 8 levels "I1" < "SI2" < "SI1" < ...: 2 3 5 4 2 6 7 3 4 5 ...
## # $ depth   : num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
## # $ table   : num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...
## # $ price   : int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...
## # $ x       : num [1:53940] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
## # $ y       : num [1:53940] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
## # $ z       : num [1:53940] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

数值型变量 carat 作为 x 轴

```
ggplot(diamonds, aes(x = carat))  
ggplot(diamonds, aes(x = carat, y = price))  
ggplot(diamonds, aes(x = carat, color = cut))  
ggplot(diamonds, aes(x = carat), color = "steelblue")
```

图 11.37 的基础上添加数据图层

```
sub_diamonds <- diamonds[sample(1:nrow(diamonds), 1000), ]  
ggplot(sub_diamonds, aes(x = carat, y = price)) +  
  geom_point()
```

给散点图11.38上色

```
ggplot(sub_diamonds, aes(x = carat, y = price)) +
  geom_point(color = "steelblue")

ggplot(sub_diamonds, aes(x = carat, y = price)) +
  geom_point(color = "steelblue") +
  scale_y_continuous(
    labels = scales::unit_format(unit = "k", scale = 1e-3),
    breaks = seq(0, 20000, 4000)
)
```

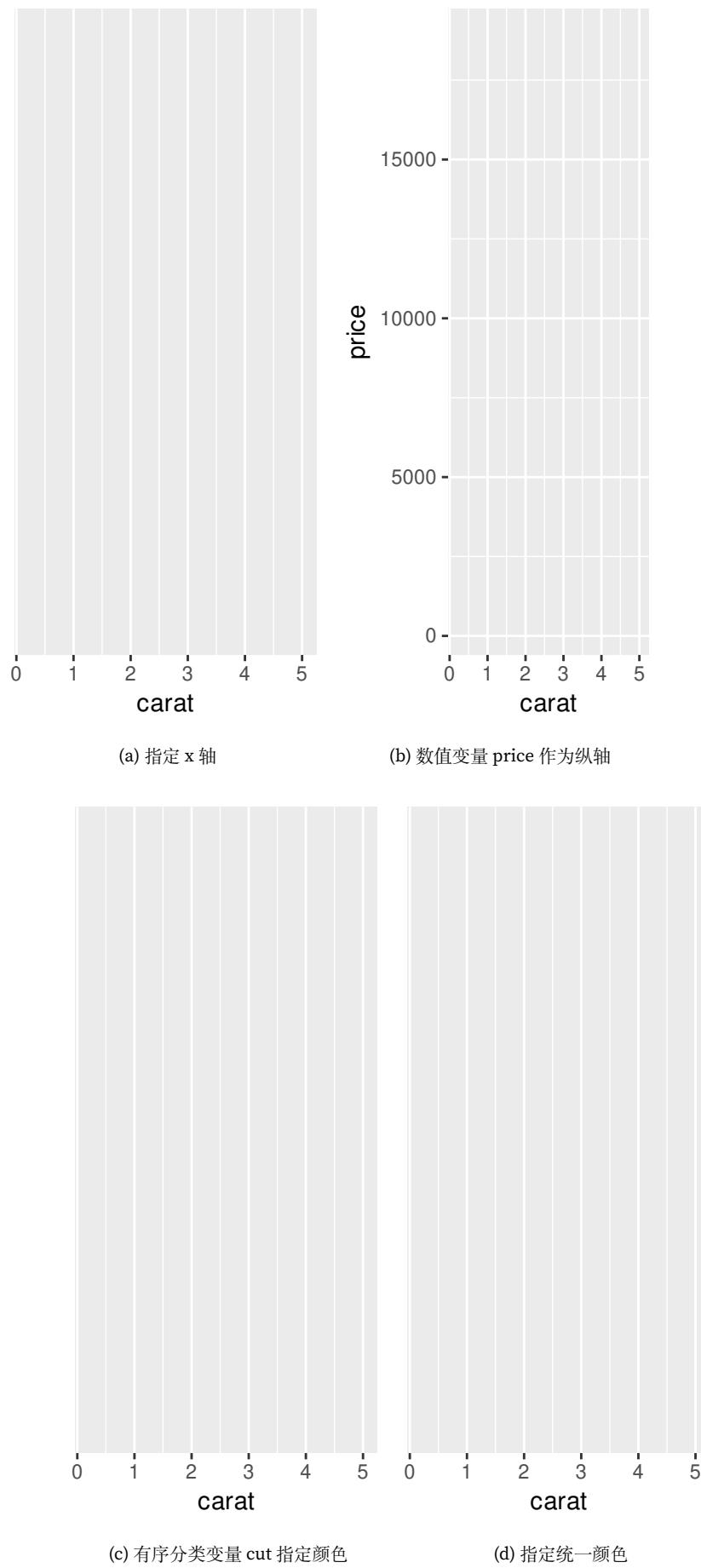


图 11.37: 绘图过程

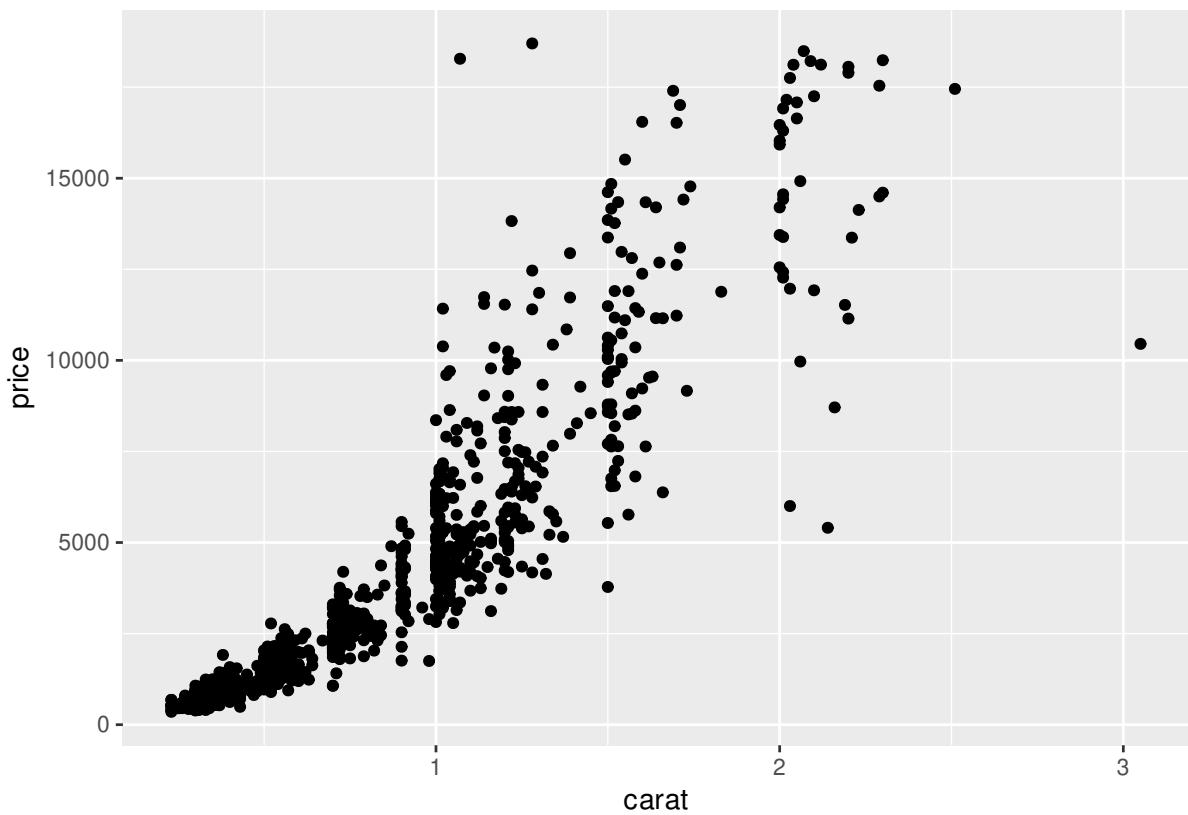


图 11.38: 添加数据图层

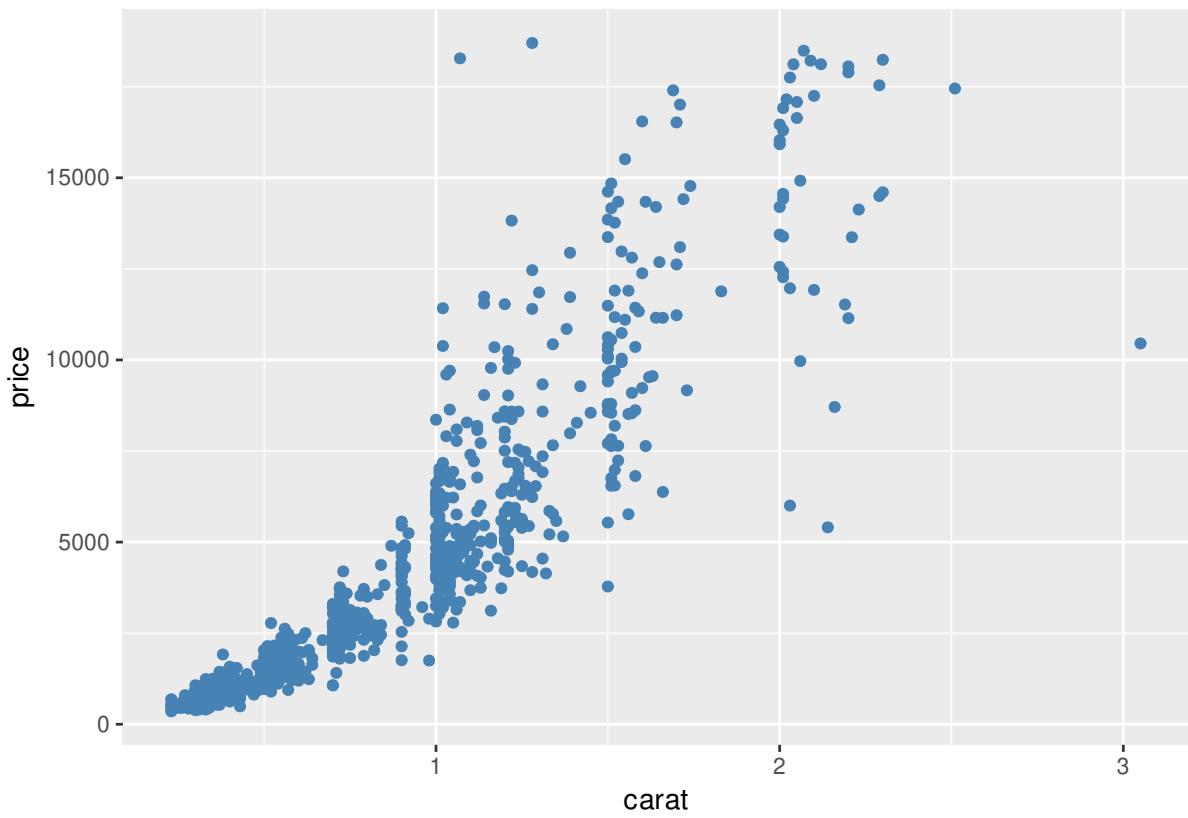


图 11.39: 散点图配色

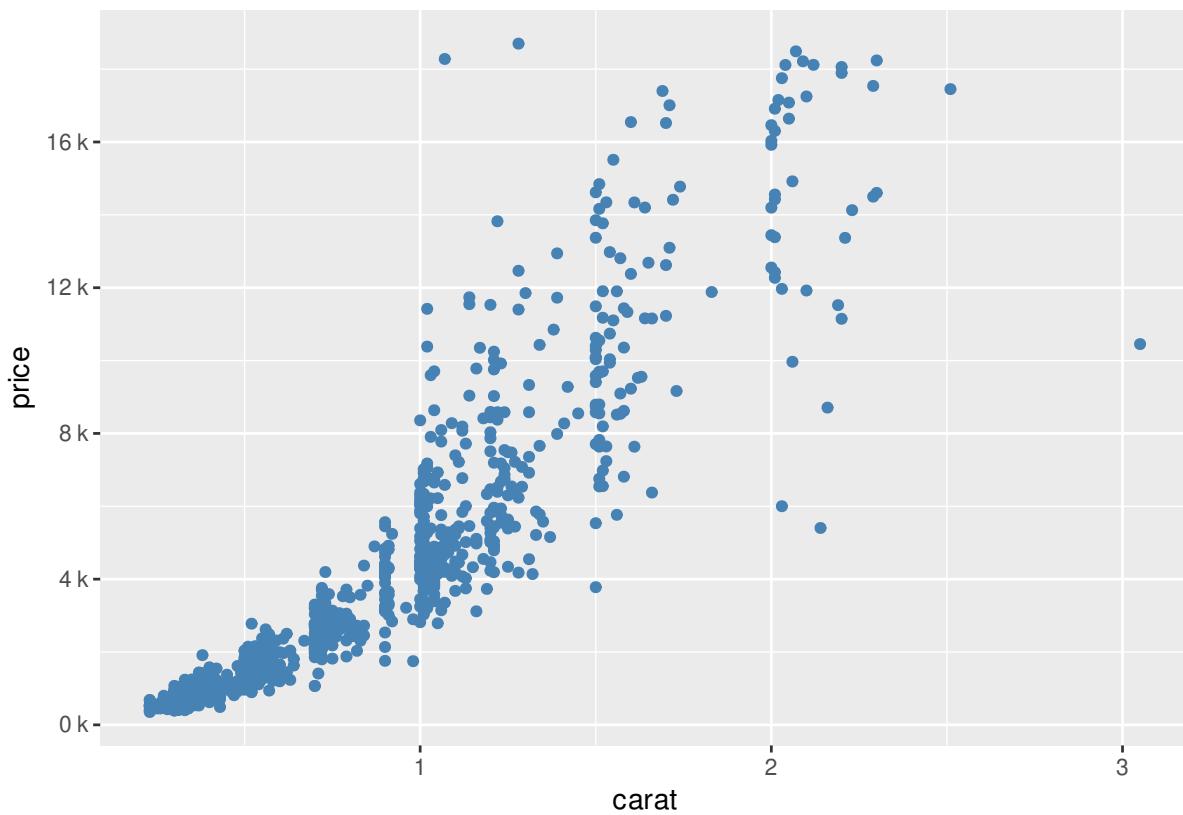


图 11.40: 格式化坐标轴刻度标签

让另一变量 cut 作为颜色分类指标

```
ggplot(sub_diamonds, aes(x = carat, y = price, color = cut)) +
  geom_point()
```

当然还有一种类似的表示就是分组，默认情况下，ggplot2 将所有观测点视为一组，以分类变量 cut 来分组

```
ggplot(sub_diamonds, aes(x = carat, y = price, group = cut)) +
  geom_point()
```

在图11.42 上没有体现出来分组的意思，下面以 cut 分组线性回归为例

```
ggplot(sub_diamonds, aes(x = carat, y = price)) +
  geom_point() +
  geom_smooth(method = "lm")
```

```
ggplot(sub_diamonds, aes(x = carat, y = price, group = cut)) +
  geom_point() +
  geom_smooth(method = "lm")
```

我们当然可以选择更加合适的拟合方式，如局部多项式平滑 loess 但是该方法不太适用观测值比较多的情况，因为它会占用比较多的内存，建议使用广义可加模型作平滑拟合

```
ggplot(sub_diamonds, aes(x = carat, y = price, group = cut)) +
  geom_point() +
```

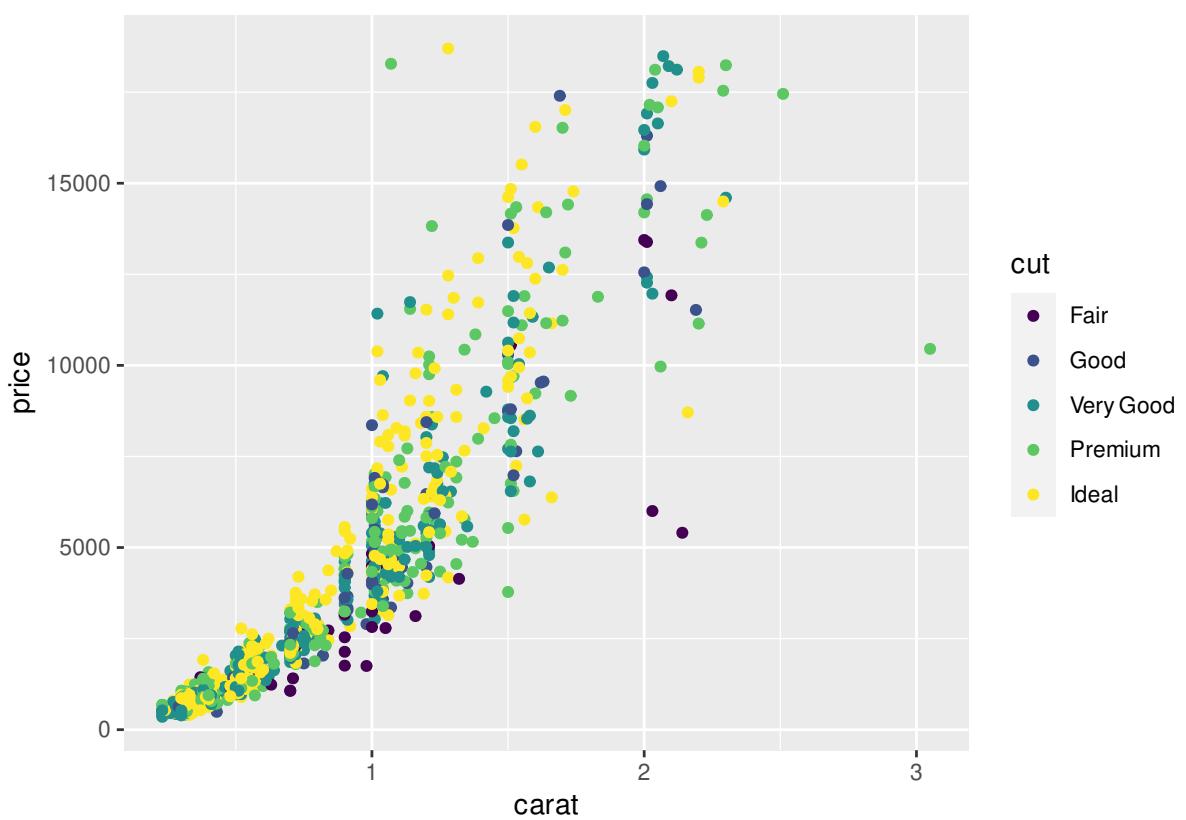


图 11.41：分类散点图

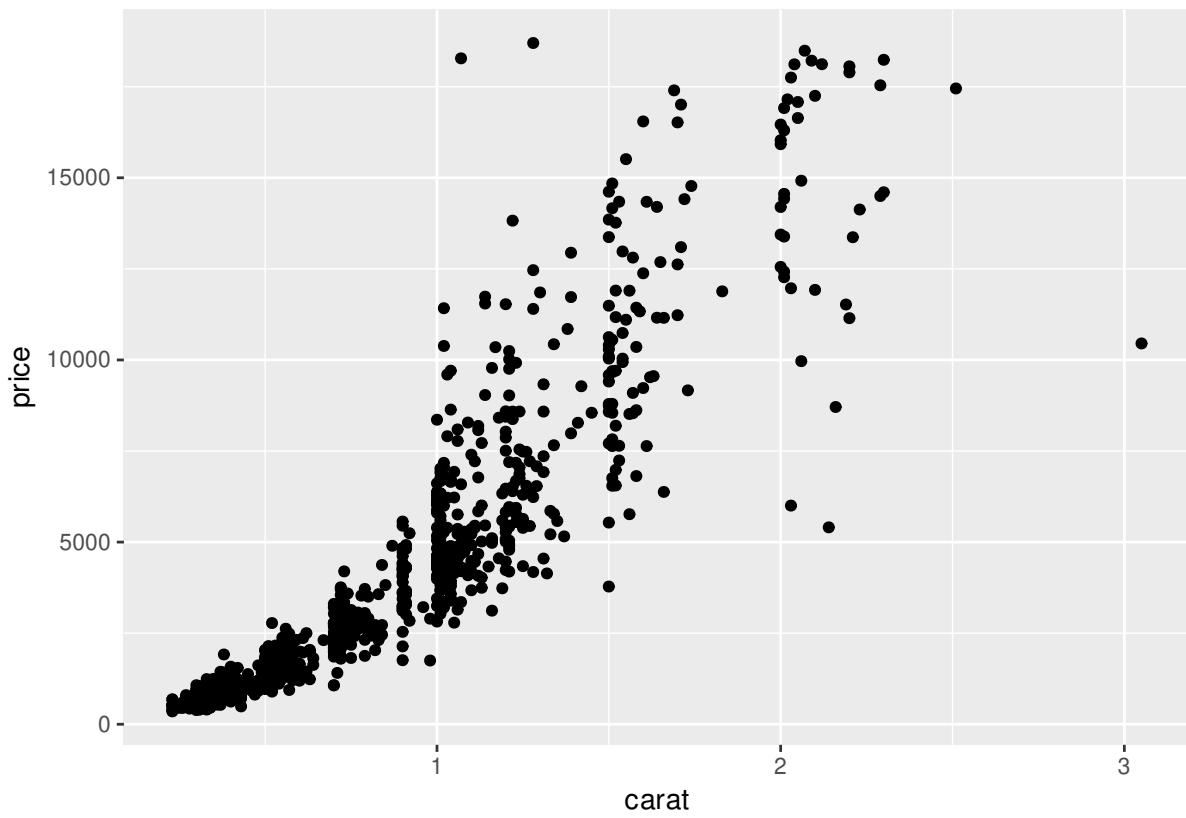


图 11.42：分组

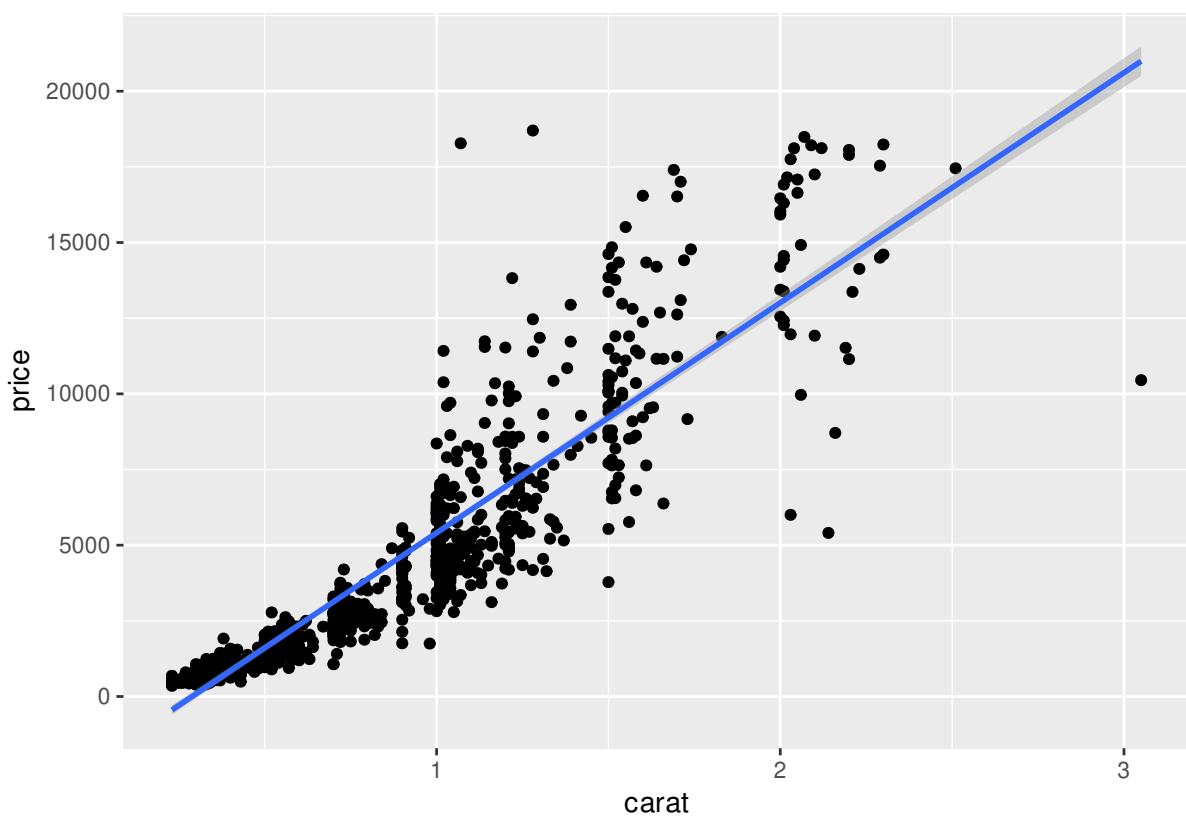


图 11.43: 分组线性回归

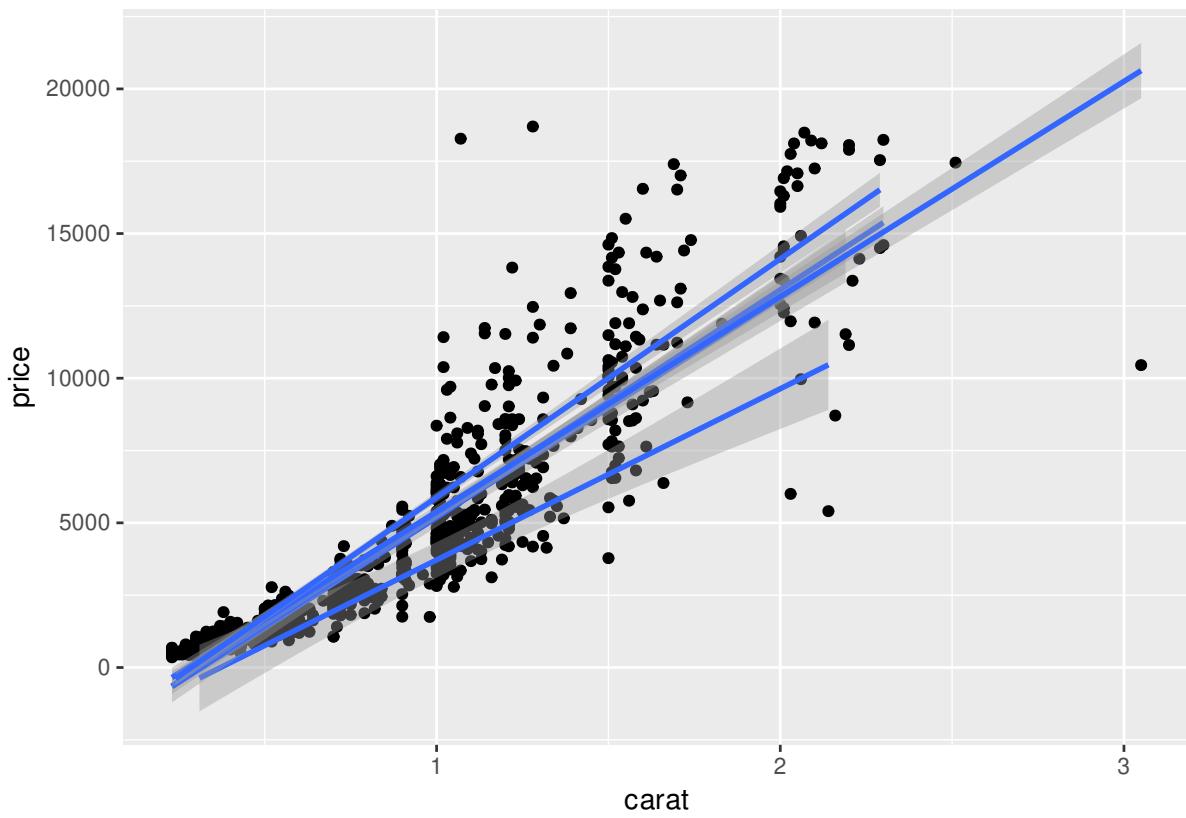


图 11.44: 分组线性回归

```
geom_smooth(method = "loess")
```

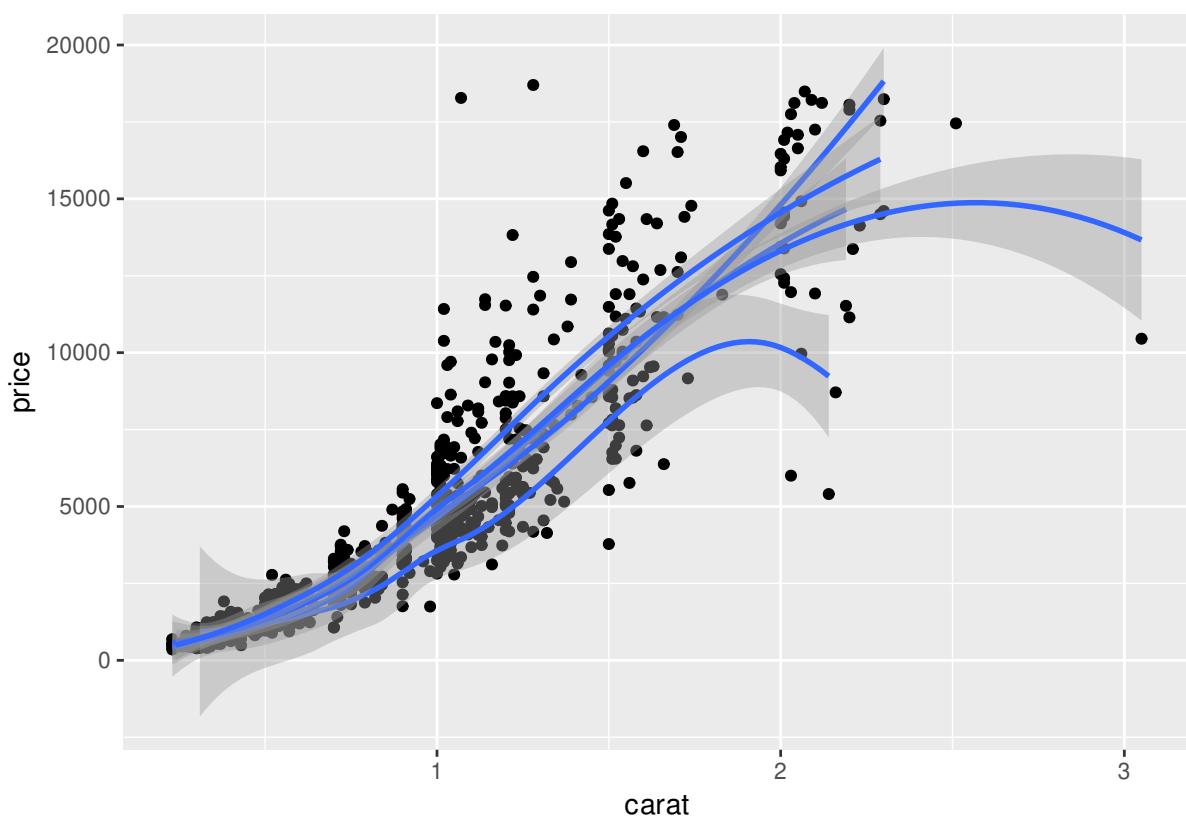


图 11.45: 局部多项式平滑

```
ggplot(sub_diamonds, aes(x = carat, y = price, group = cut)) +
  geom_point() +
  geom_smooth(method = "gam", formula = y ~ s(x, bs = "cs"))
```

[ggfortify](#) 包支持更多的统计分析结果的可视化。

为了更好地区分开组别，我们在图11.46的基础上分面或者配色

```
ggplot(sub_diamonds, aes(x = carat, y = price, group = cut)) +
  geom_point() +
  geom_smooth(method = "gam", formula = y ~ s(x, bs = "cs")) +
  facet_grid(~cut)
```

```
ggplot(sub_diamonds, aes(x = carat, y = price, group = cut, color = cut)) +
  geom_point() +
  geom_smooth(method = "gam", formula = y ~ s(x, bs = "cs"))
```

在分类散点图的另一种表示方法就是分面图，以 cut 变量作为分面的依据

```
ggplot(sub_diamonds, aes(x = carat, y = price)) +
  geom_point() +
  facet_grid(~cut)
```

给图 11.49 上色

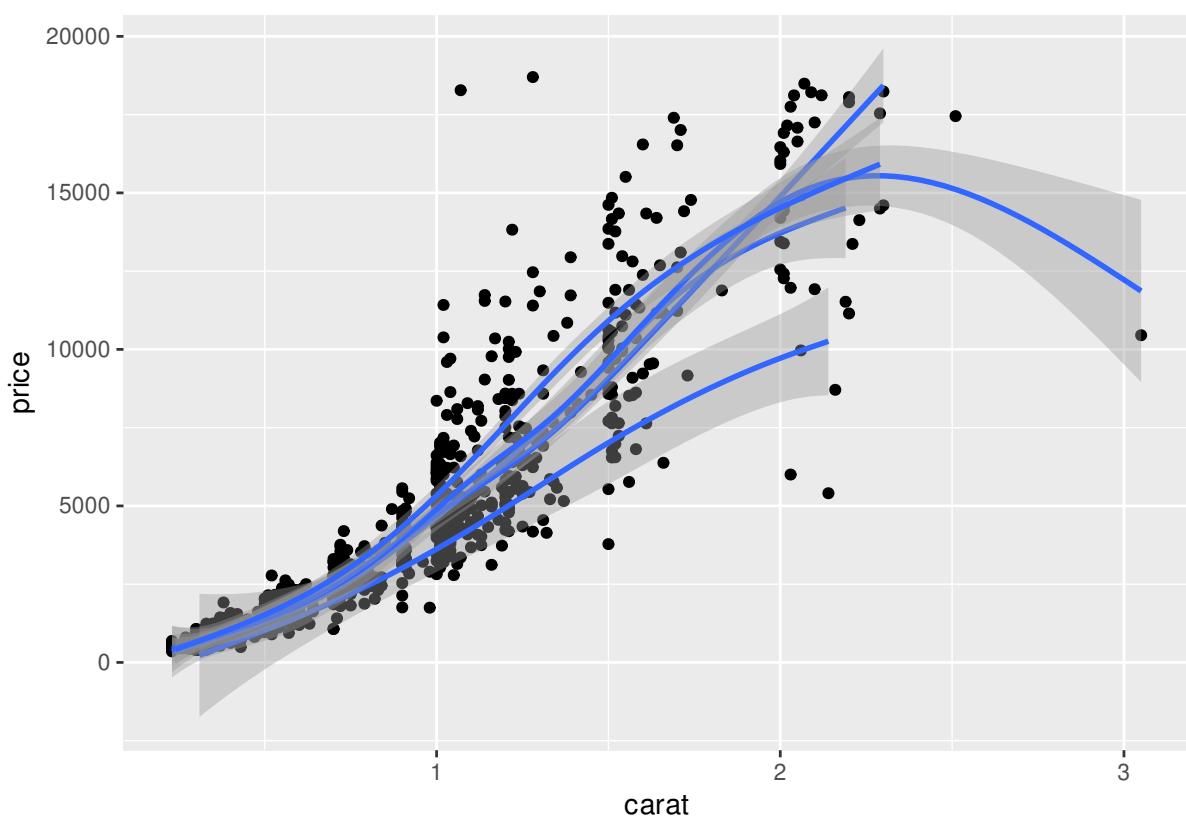


图 11.46: 数据分组应用广义可加平滑

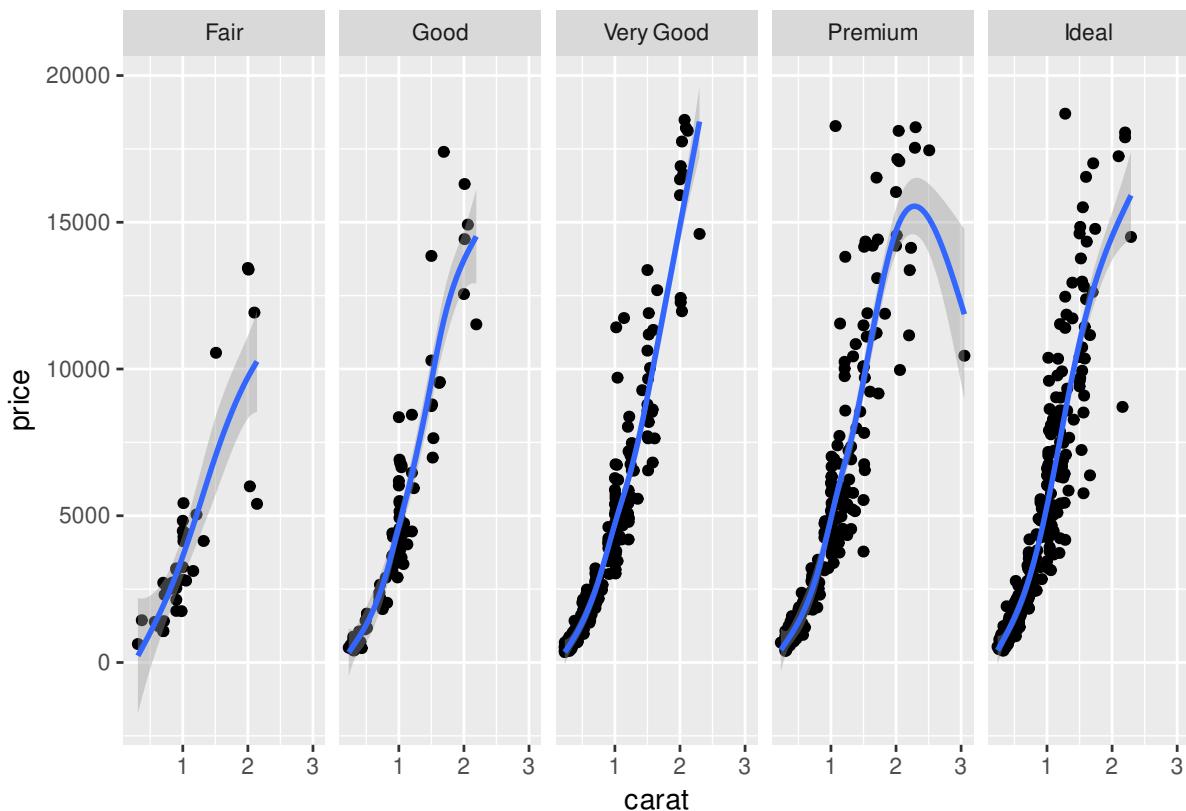


图 11.47: 分组分面

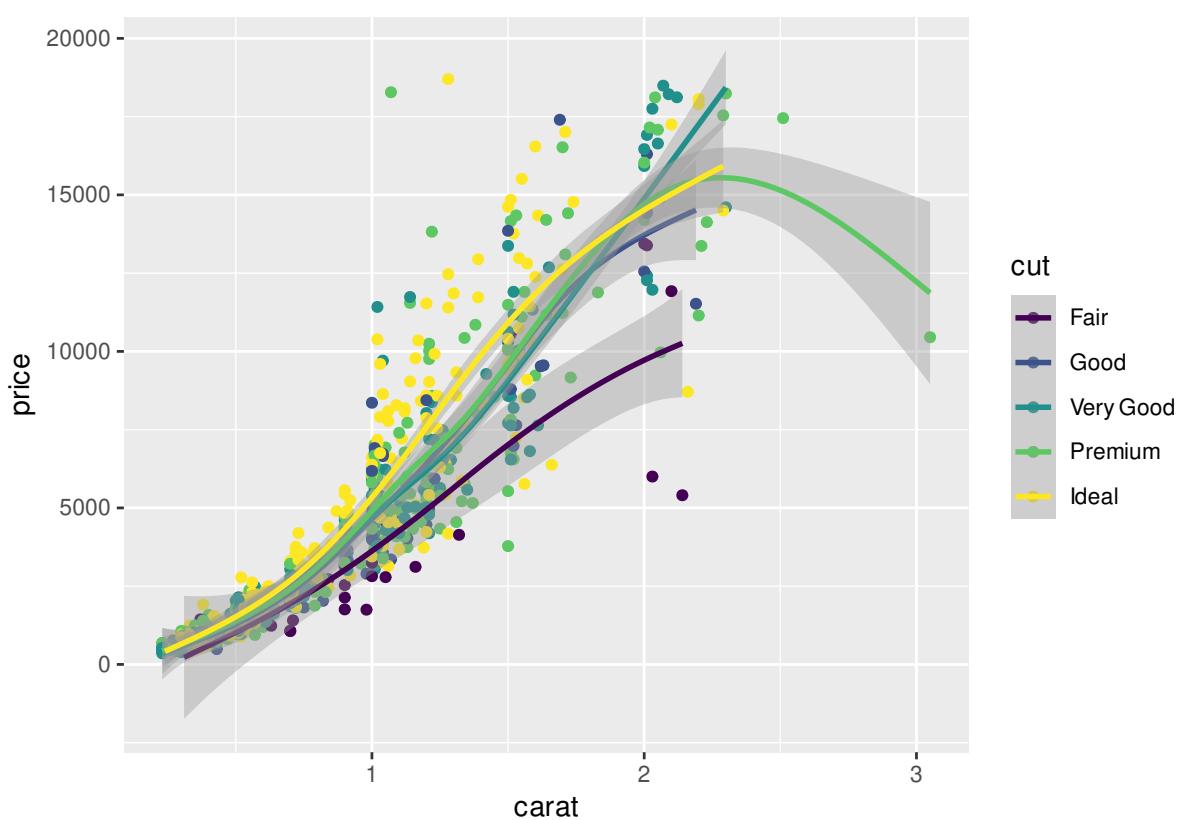


图 11.48: 分组配色

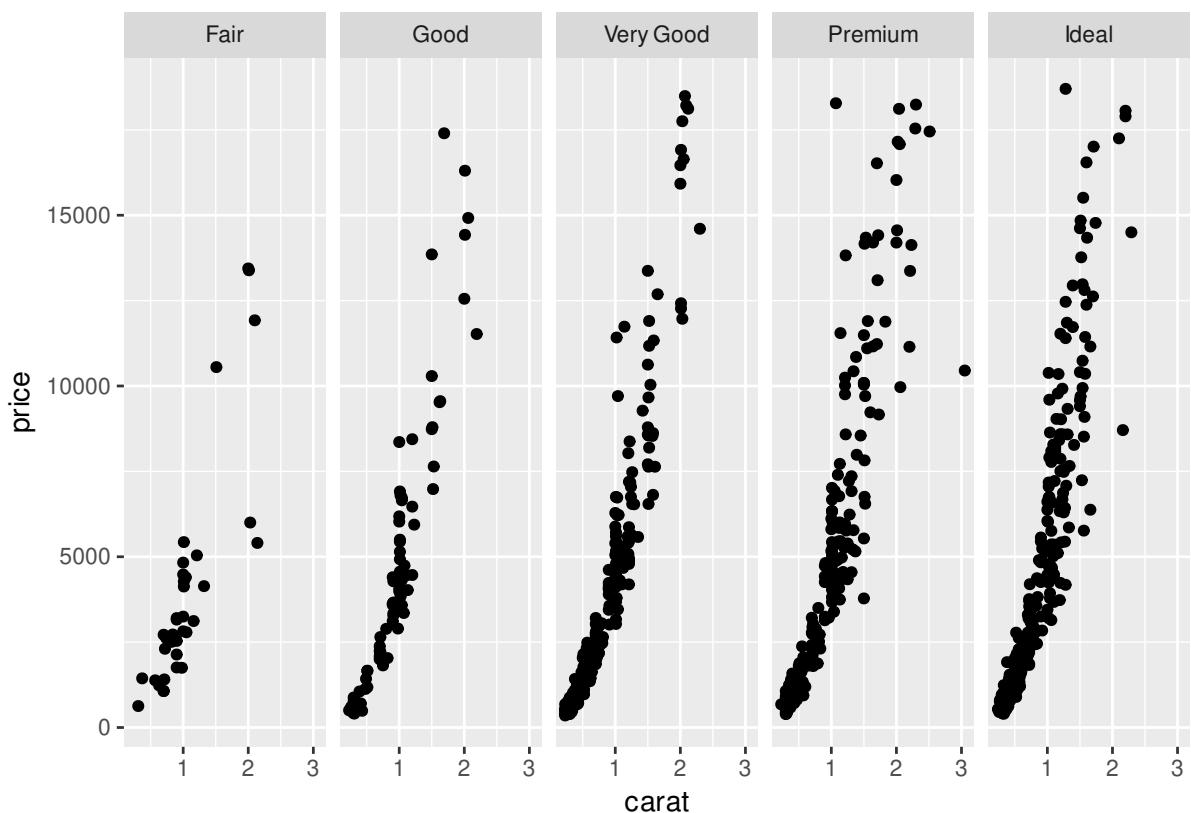


图 11.49: 分面散点图

```
ggplot(sub_diamonds, aes(x = carat, y = price)) +
  geom_point(color = "steelblue") +
  facet_grid(~cut)
```

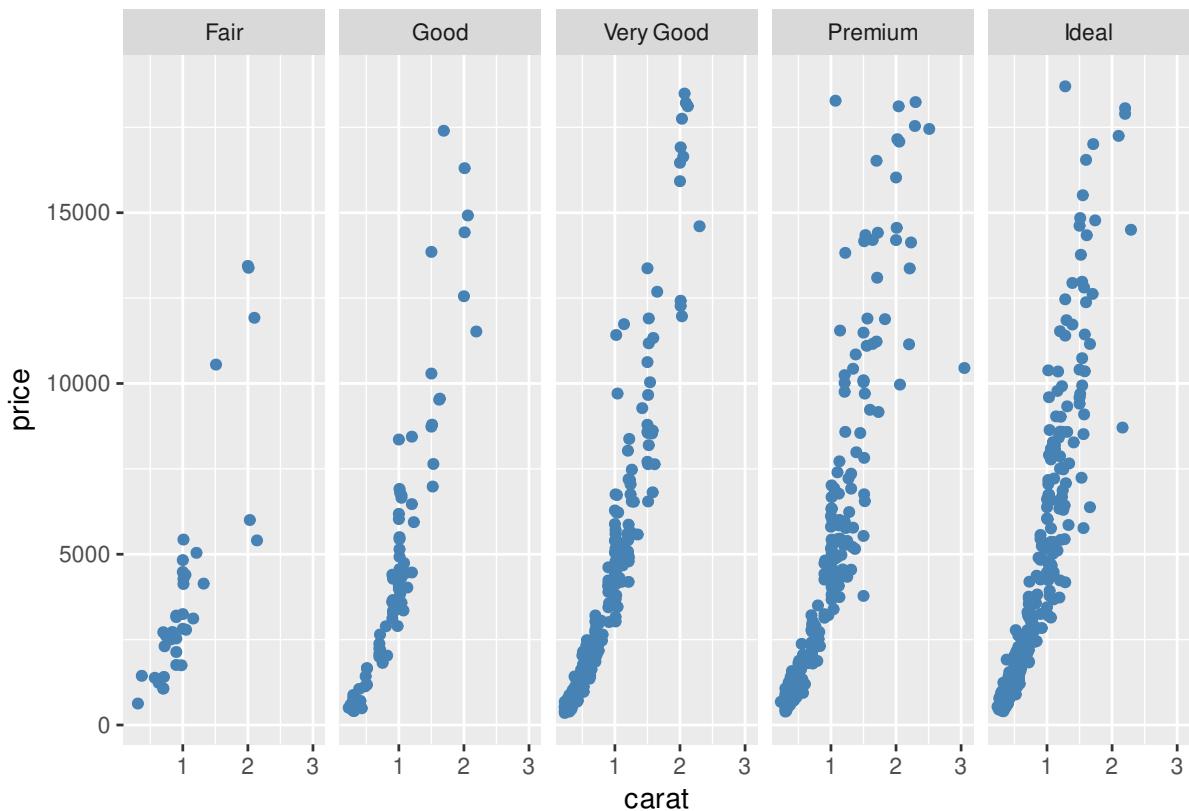


图 11.50: 给分面散点图上色

在图11.50的基础上，给不同的类上不同的颜色

```
ggplot(sub_diamonds, aes(x = carat, y = price, color = cut)) +
  geom_point() +
  facet_grid(~cut)
```

去掉图例，此时图例属于冗余信息了

```
ggplot(sub_diamonds, aes(x = carat, y = price, color = cut)) +
  geom_point(show.legend = FALSE) +
  facet_grid(~cut)
```

四块土地，所施肥料不同，肥力大小顺序  $4 < 2 < 3 < 1$  小麦产量随肥力的变化

```
data(Wheat2, package = "nlme") # Wheat Yield Trials
library(colorspace)
ggplot(Wheat2, aes(longitude, latitude)) +
  geom_point(aes(size = yield, colour = Block)) +
  scale_color_discrete_sequential(palette = "Viridis") +
  scale_x_continuous(breaks = seq(0, 30, 5)) +
  scale_y_continuous(breaks = seq(0, 50, 10))
```

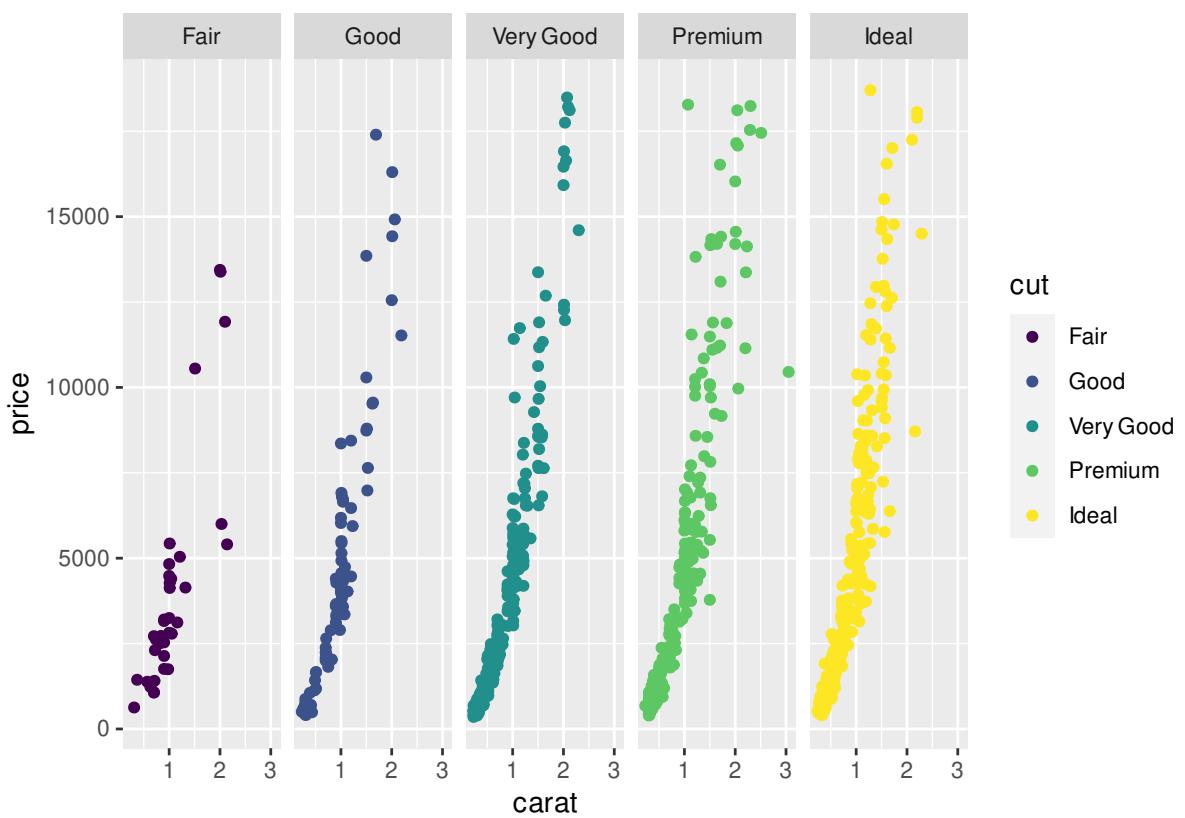


图 11.51: 给不同的类上不同的颜色

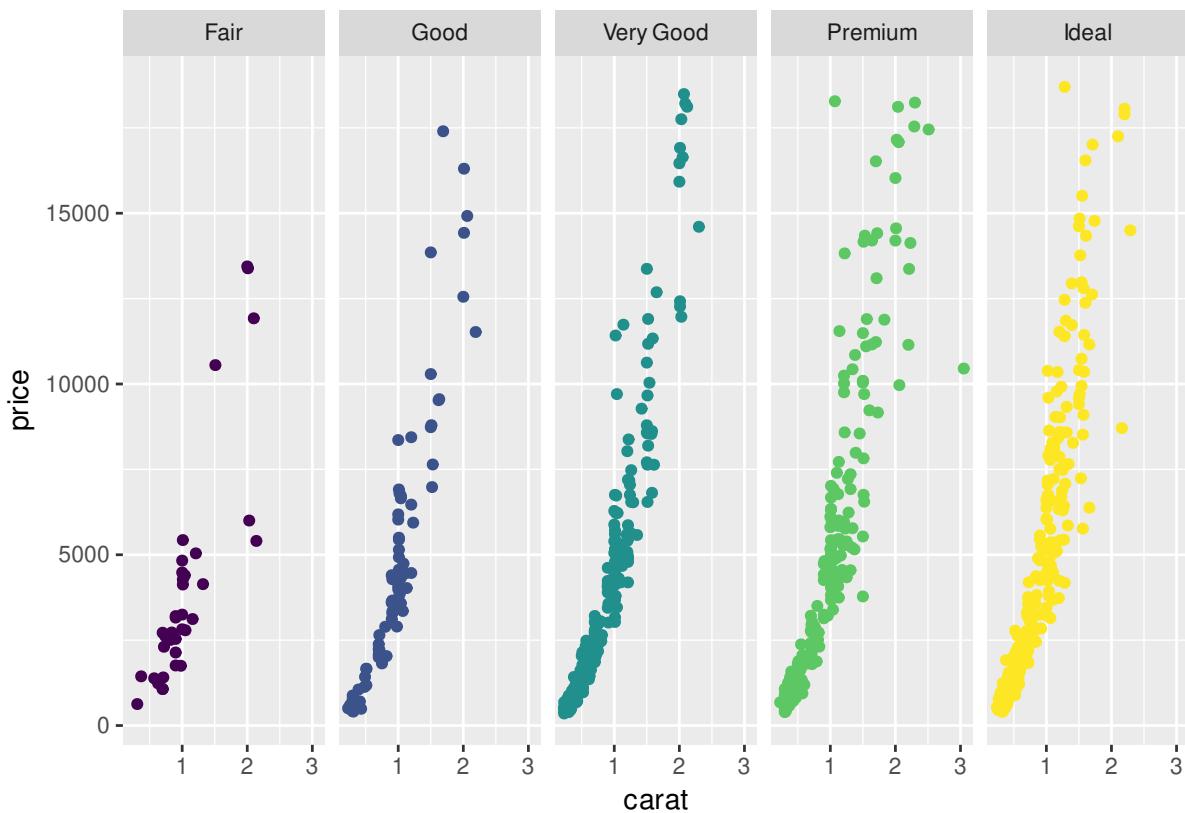


图 11.52: 去掉图例

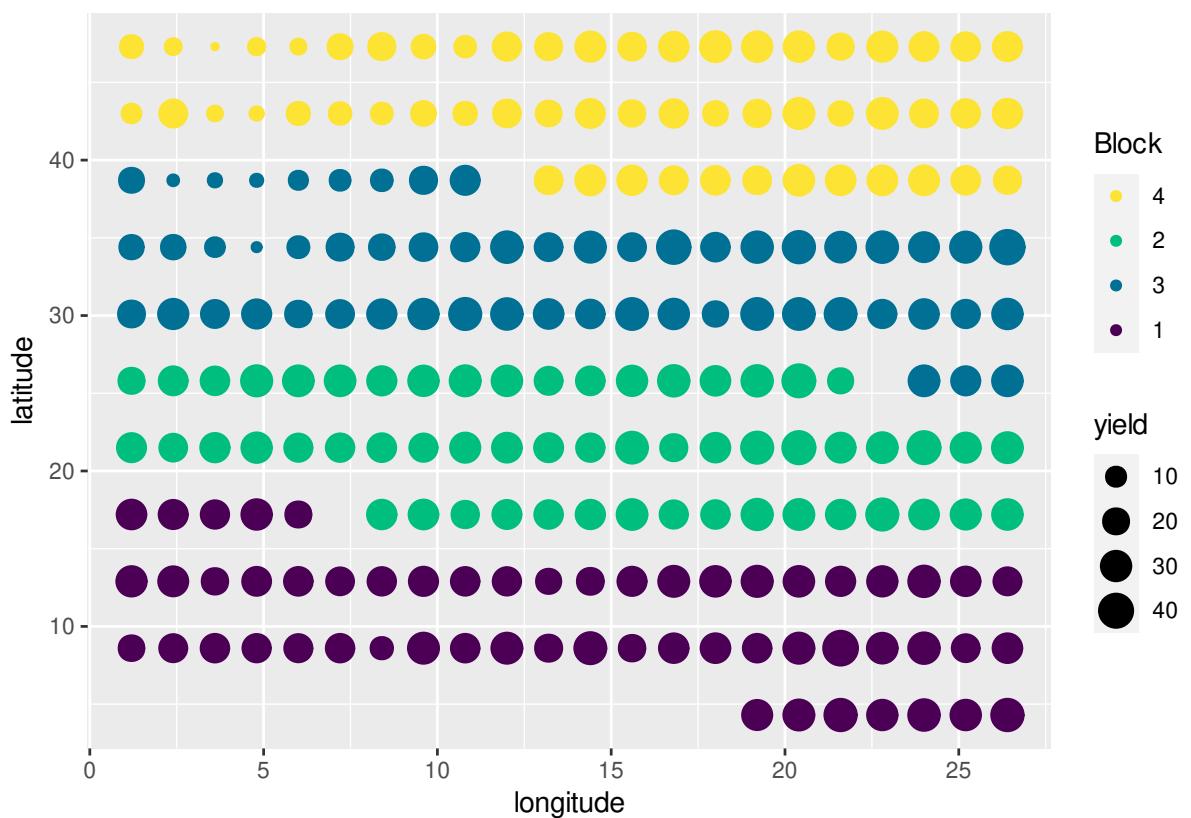


图 11.53: 多个图例

```
ggplot(mtcars, aes(x = hp, y = mpg, color = factor(am))) +
  geom_point()
```

图层、分组、分面和散点图介绍完了，接下来就是其它统计图形，如箱线图，小提琴图和条形图

```
dat <- as.data.frame(cbind(rep(1948 + seq(12), each = 12), rep(seq(12), 12), AirPassengers))
colnames(dat) <- c("year", "month", "passengers")

ggplot(data = dat, aes(x = as.factor(year), y = as.factor(month))) +
  stat_sum(aes(size = passengers), colour = "lightblue") +
  scale_size(range = c(1, 10), breaks = seq(100, 650, 50)) +
  labs(x = "Year", y = "Month", colour = "Passengers") +
  theme_minimal()
```

## 11.4.5 条形图

条形图特别适合分类变量的展示，我们这里展示钻石切割质量 cut 不同等级的数量，当然我们可以直接展示各类的数目，在图层 `geom_bar` 中指定 `stat="identity"`

```
# 需要映射数据框的两个变量，相当于自己先计算了每类的数量
with(diamonds, table(cut))
```

```
## cut
```

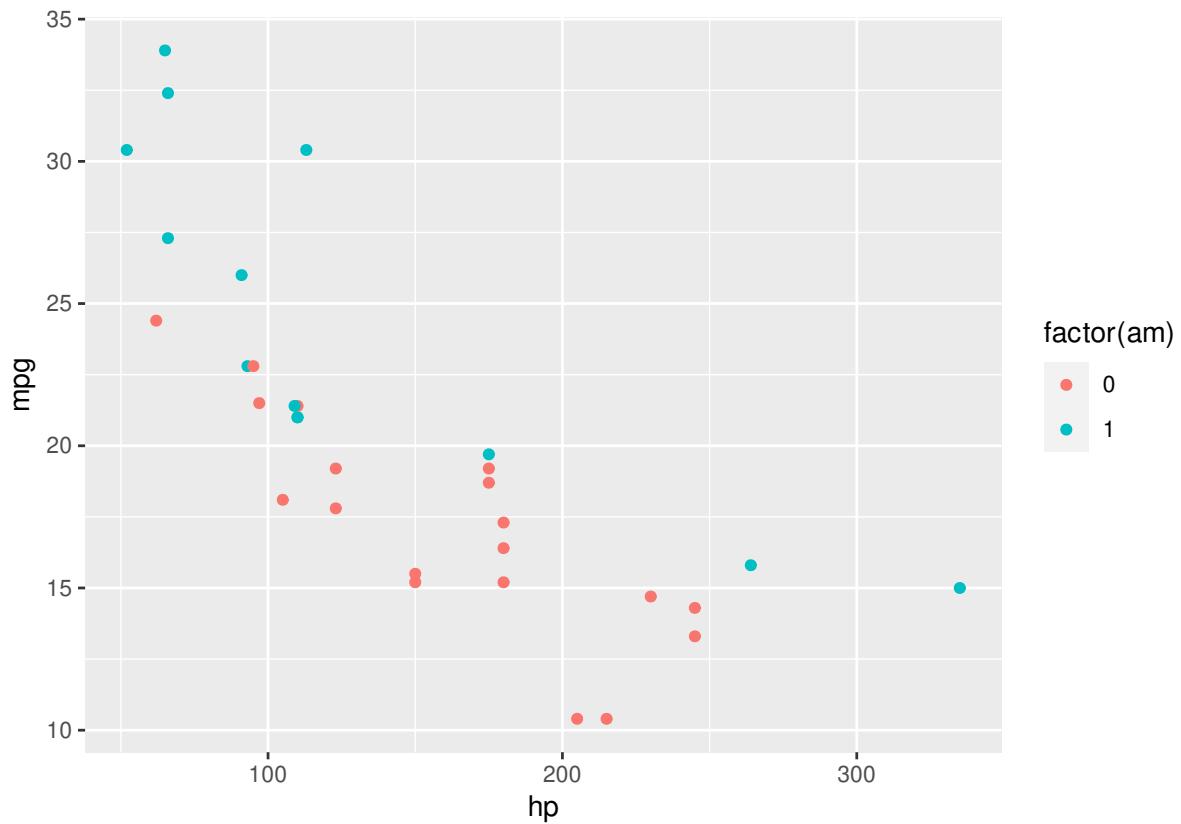


图 11.54: 分类散点图

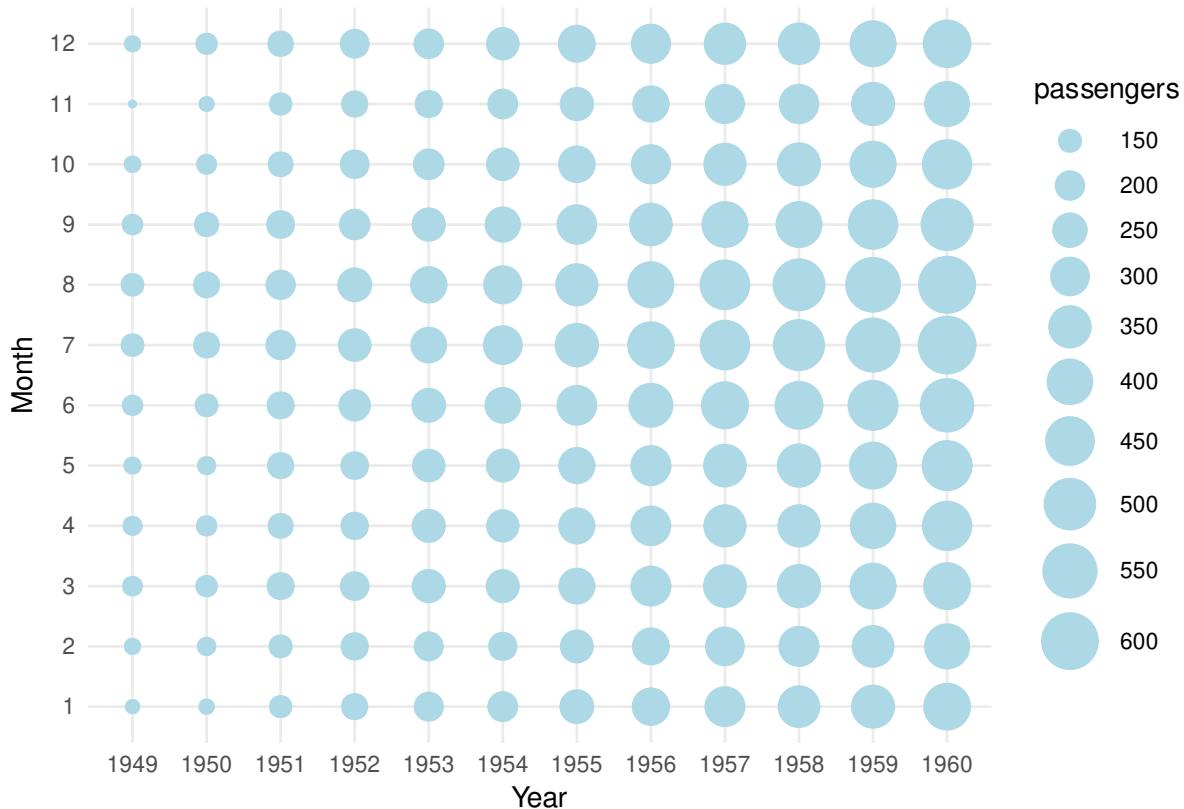
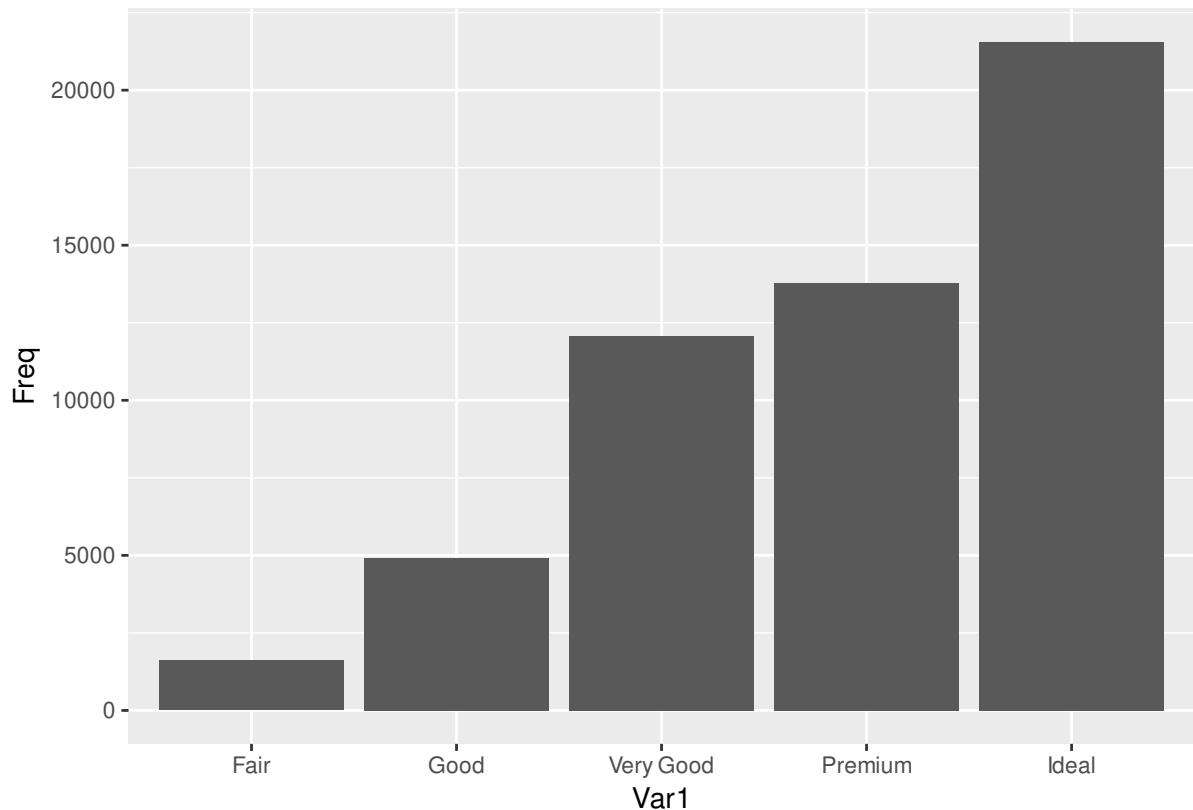


图 11.55: 1948 年至 1960 年航班乘客人数变化

```
##      Fair      Good Very Good Premium Ideal
##      1610      4906     12082    13791   21551
cut_df <- as.data.frame(table(diamonds$cut))
ggplot(cut_df, aes(x = Var1, y = Freq)) + geom_bar(stat = "identity")
```



```
ggplot(diamonds, aes(x = cut)) + geom_bar()
```

还有另外三种表示方法

```
ggplot(diamonds, aes(x = cut)) + geom_bar(stat = "count")
```

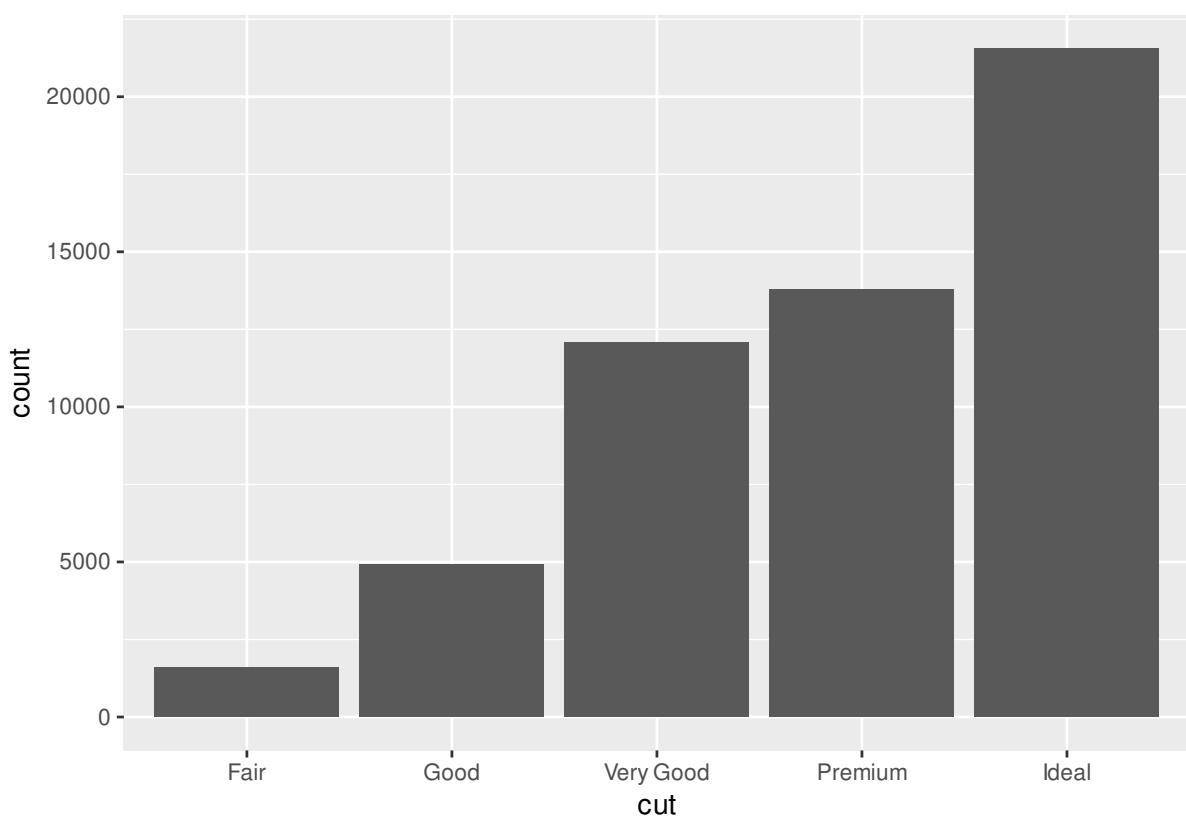
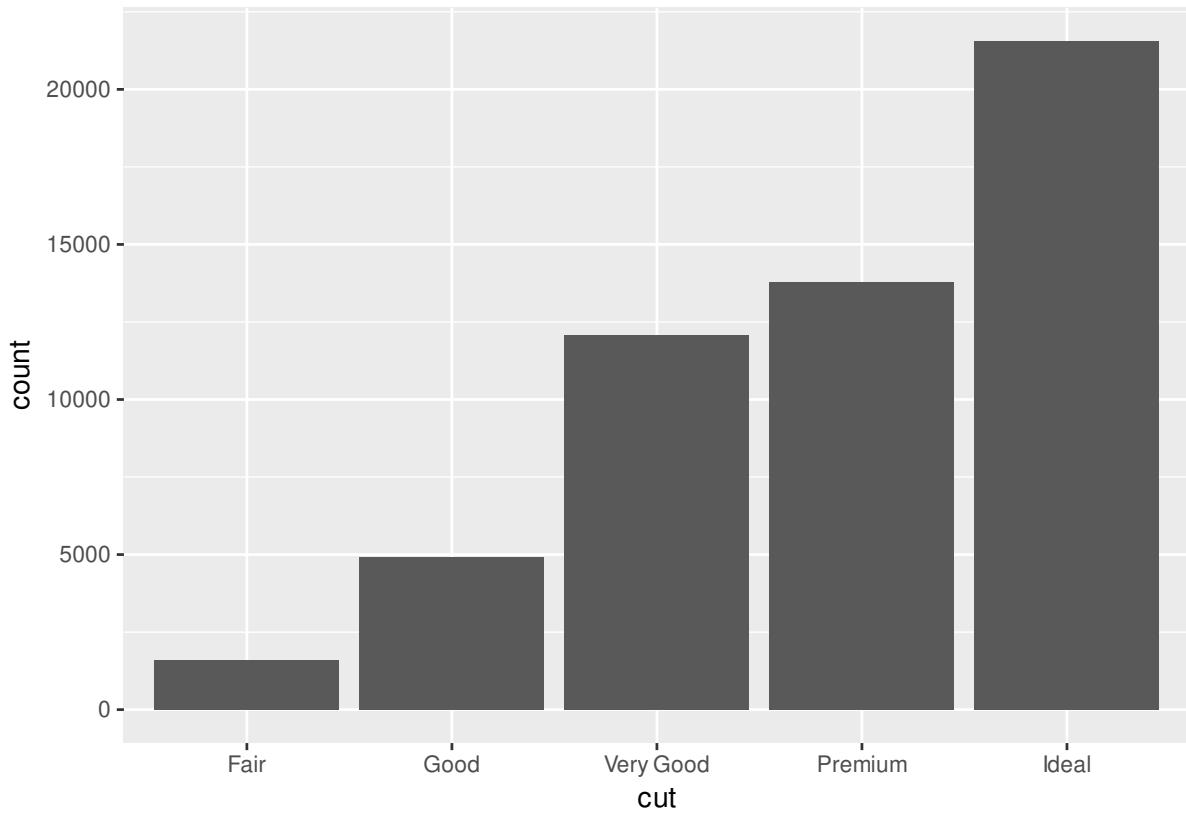
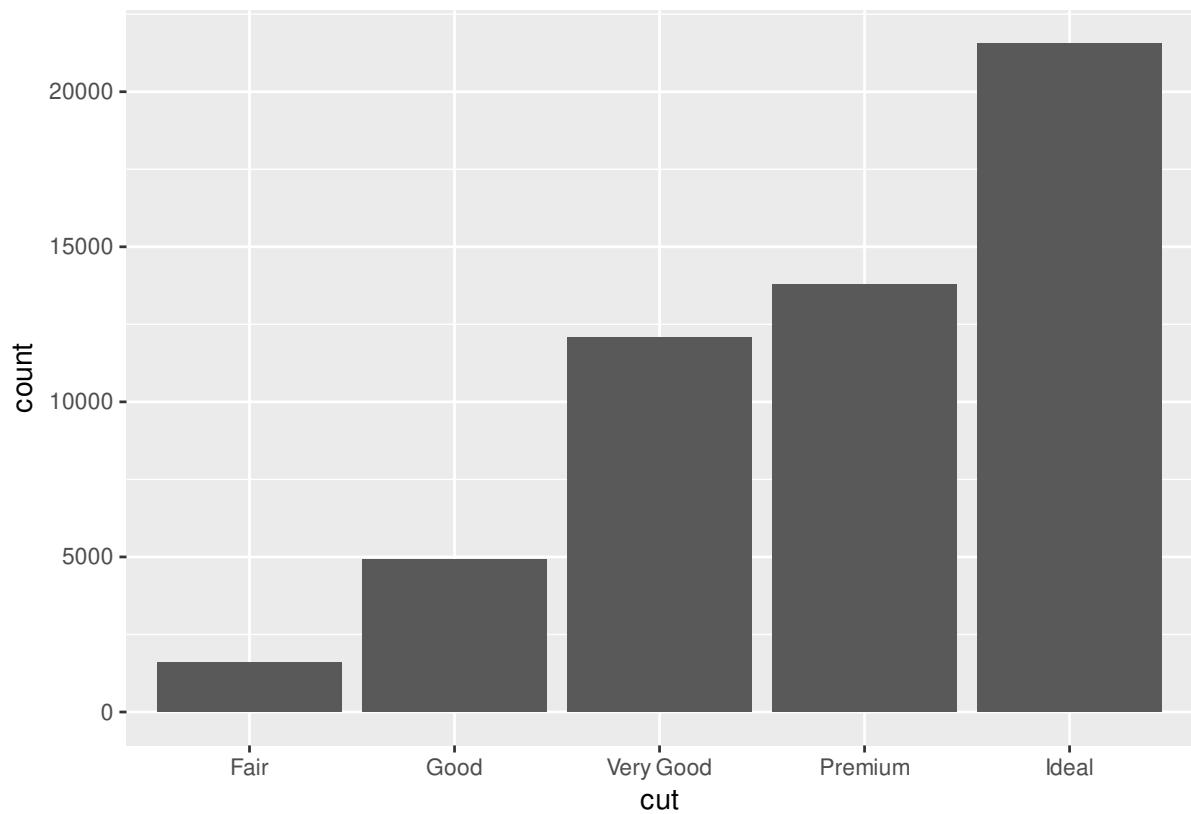


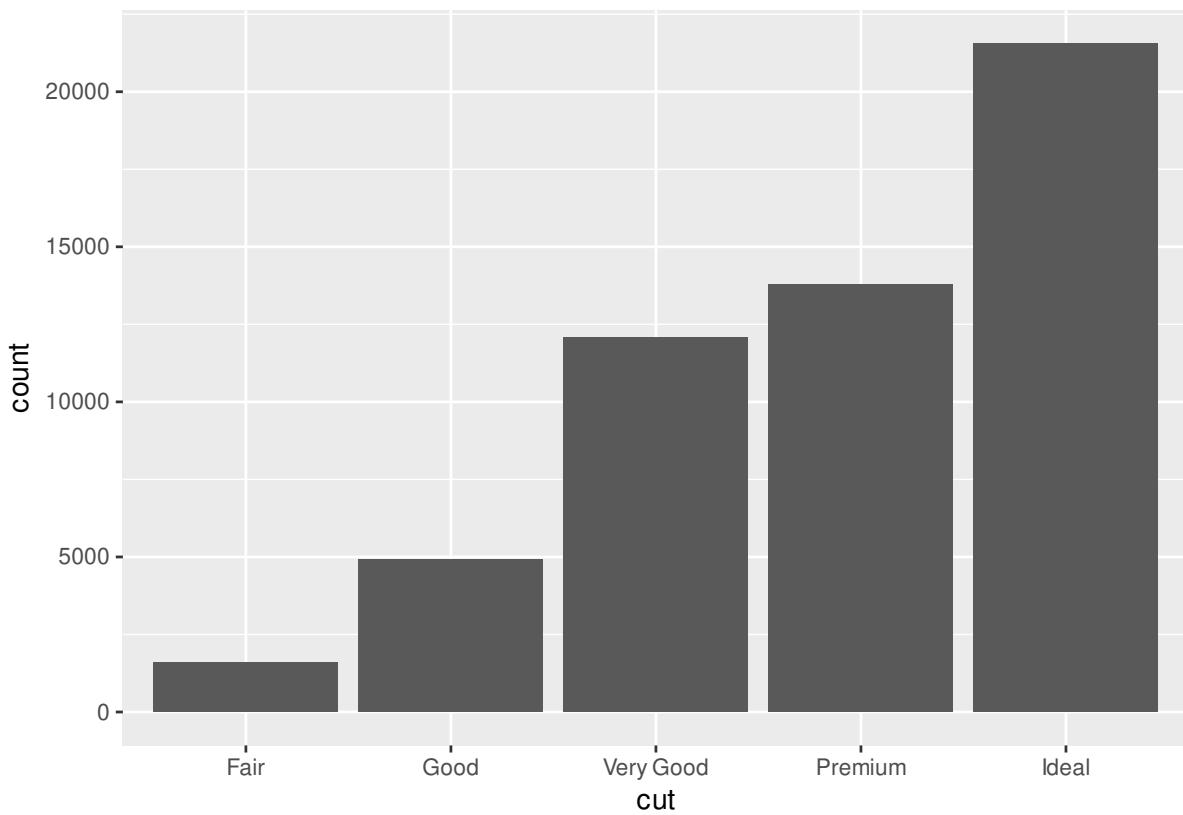
图 11.56: 频数条形图



```
ggplot(diamonds, aes(x = cut, y = ..count..)) + geom_bar()
```



```
ggplot(diamonds, aes(x = cut, y = stat(count))) + geom_bar()
```



我们还可以在图 11.56 的基础上再添加一个分类变量钻石的纯净度 clarity, 形成堆积条形图

```
ggplot(diamonds, aes(x = cut, fill = clarity)) + geom_bar()
```

再添加一个分类变量钻石颜色 color 比较好的做法是分面

```
ggplot(diamonds, aes(x = color, fill = clarity)) +  
  geom_bar() +  
  facet_grid(~cut)
```

实际上, 绘制图11.58包含了对分类变量的分组计数过程, 如下

```
with(diamonds, table(cut, color))
```

```
##          color  
##  cut        D   E   F   G   H   I   J  
##  Fair      163 224 312 314 303 175 119  
##  Good      662 933 909 871 702 522 307  
##  Very Good 1513 2400 2164 2299 1824 1204 678  
##  Premium   1603 2337 2331 2924 2360 1428 808  
##  Ideal     2834 3903 3826 4884 3115 2093 896
```

还有一种堆积的方法是按比例, 而不是按数量, 如图11.59

```
ggplot(diamonds, aes(x = color, fill = clarity)) +  
  geom_bar(position = "fill") +  
  facet_grid(~cut)
```

接下来就是复合条形图

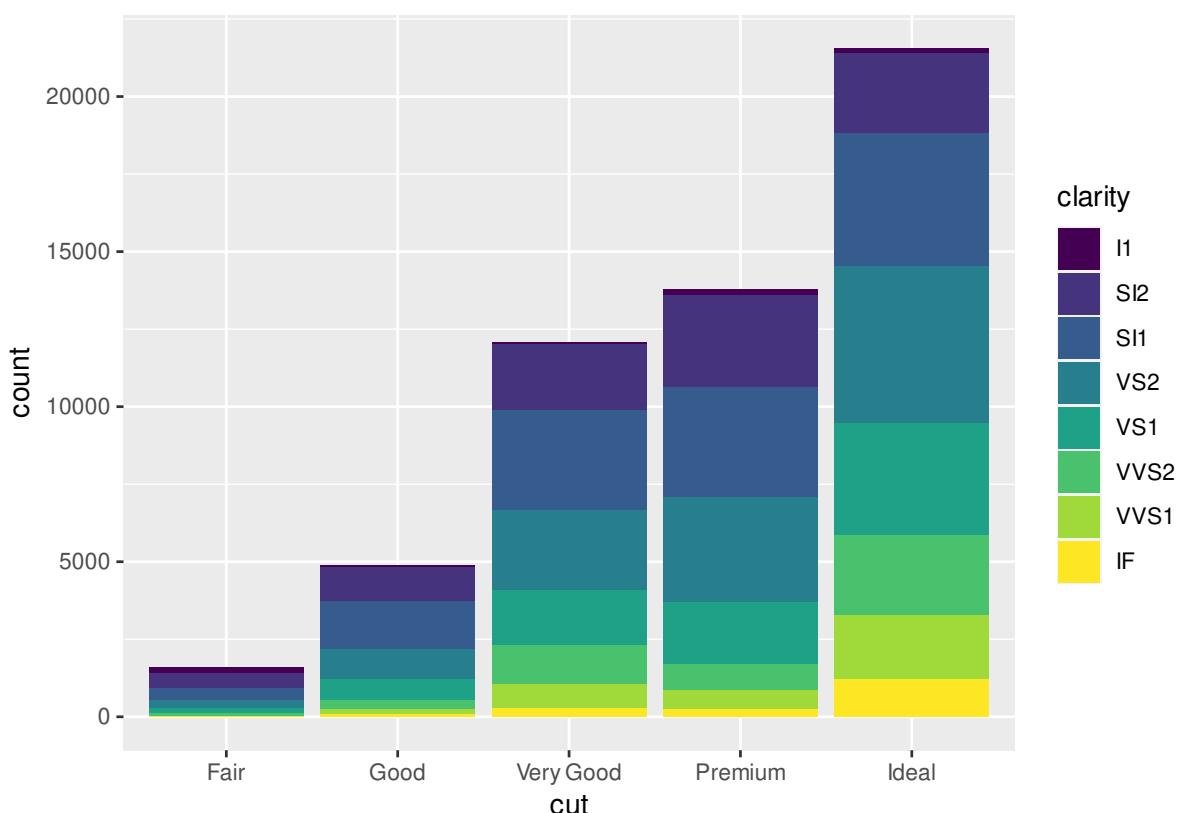


图 11.57: 堆积条形图

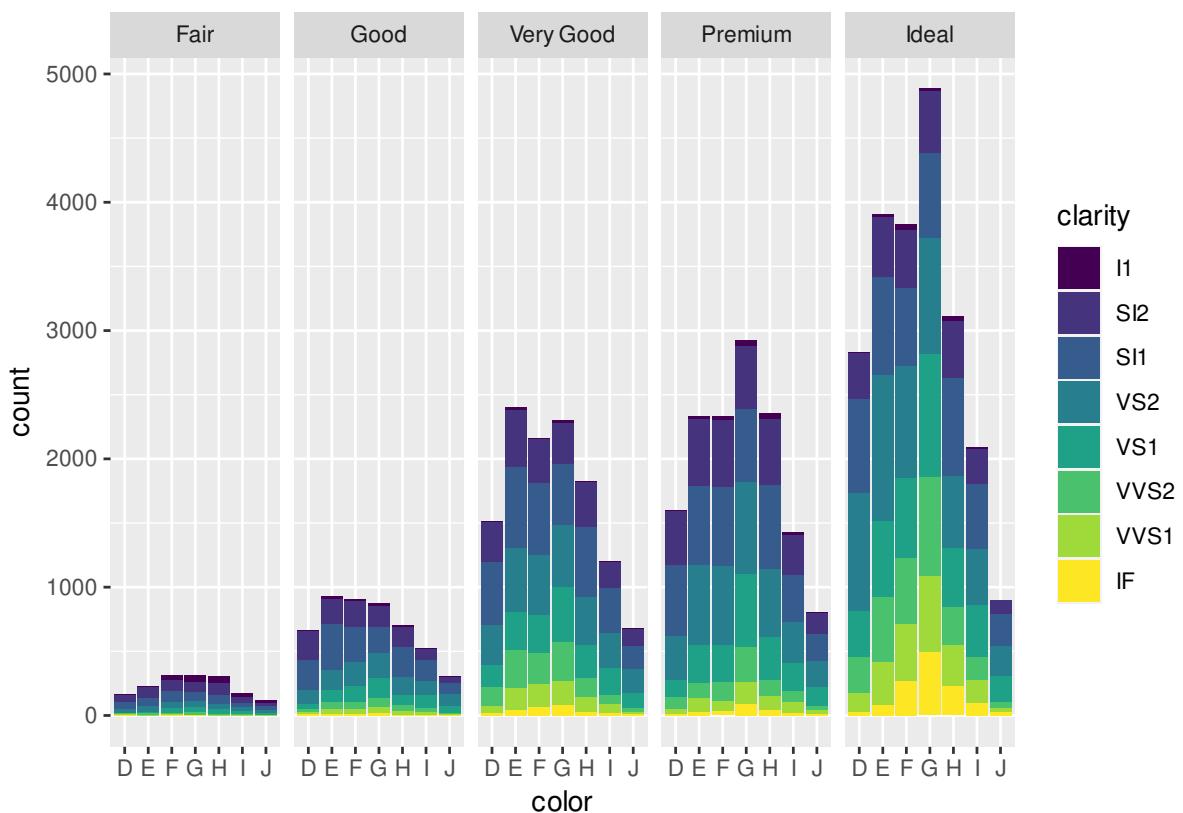


图 11.58: 分面堆积条形图

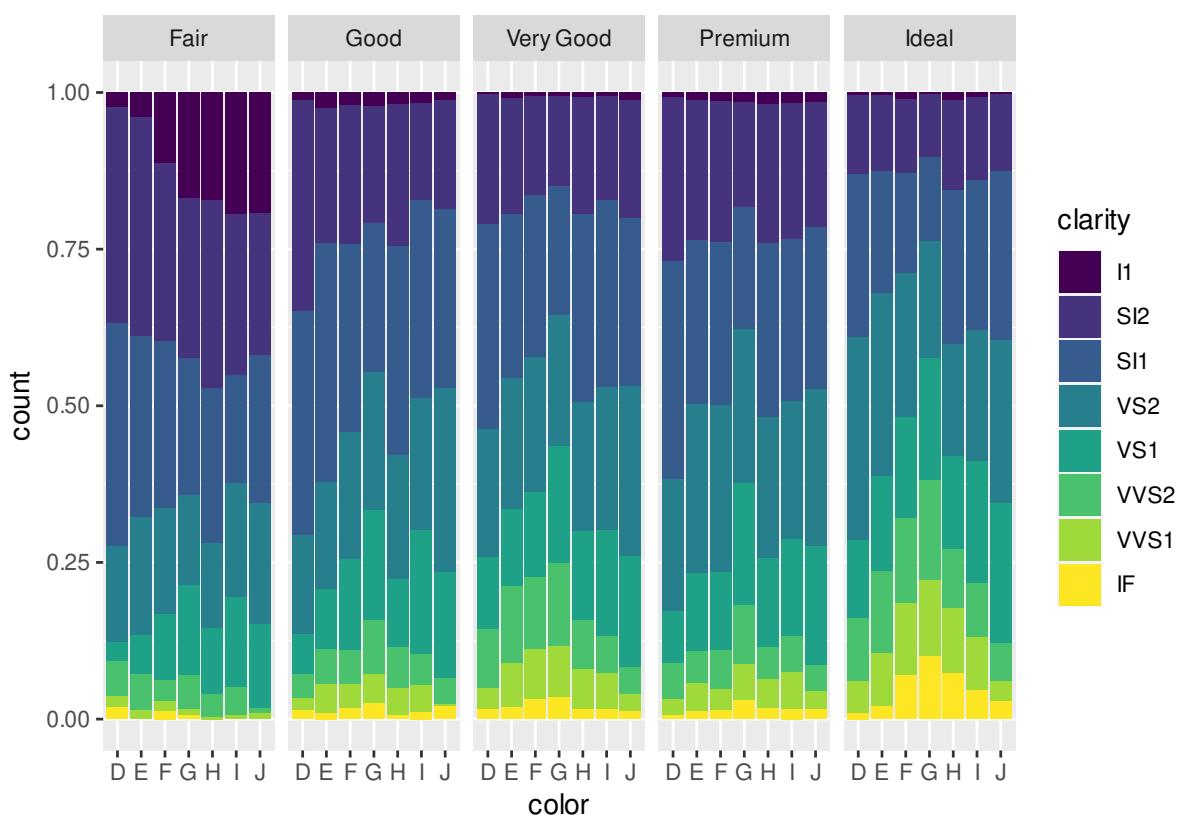


图 11.59: 比例堆积条形图

```
ggplot(diamonds, aes(x = color, fill = clarity)) +
  geom_bar(position = "dodge")
```

再添加一个分类变量，就是需要分面大法了，图 11.60 展示了三个分类变量，其实我们还可以再添加一个分类变量用作分面的列依据

```
ggplot(diamonds, aes(x = color, fill = clarity)) +
  geom_bar(position = "dodge") +
  facet_grid(rows = vars(cut))
```

图 11.61 展示的数据如下

```
with(diamonds, table(color, clarity, cut))
```

```
## , , cut = Fair
##
##      clarity
##      color I1  SI2  SI1  VS2  VS1 VVS2 VVS1  IF
##      D     4   56   58   25    5   9    3    3
##      E     9   78   65   42   14   13   3    0
##      F    35   89   83   53   33   10   5    4
##      G    53   80   69   45   45   17   3    2
##      H    52   91   75   41   32   11   1    0
##      I    34   45   30   32   25    8   1    0
```

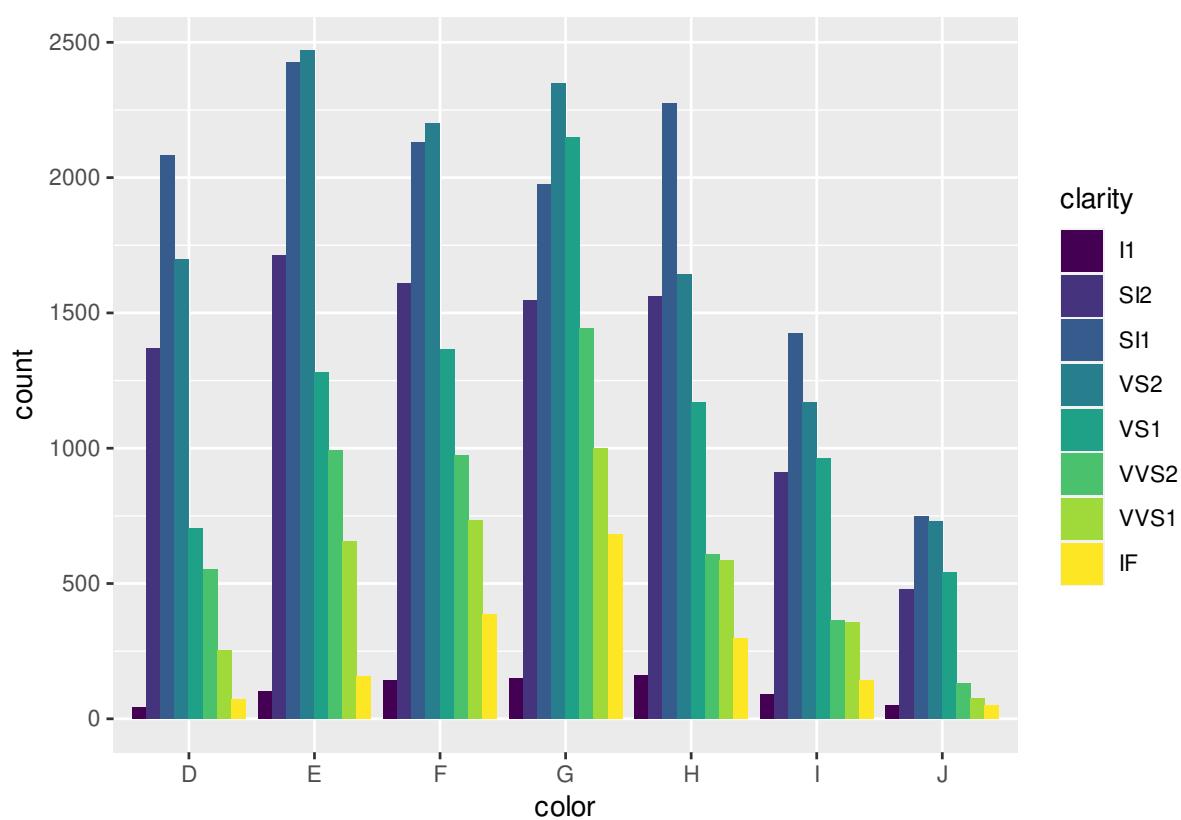


图 11.60: 复合条形图

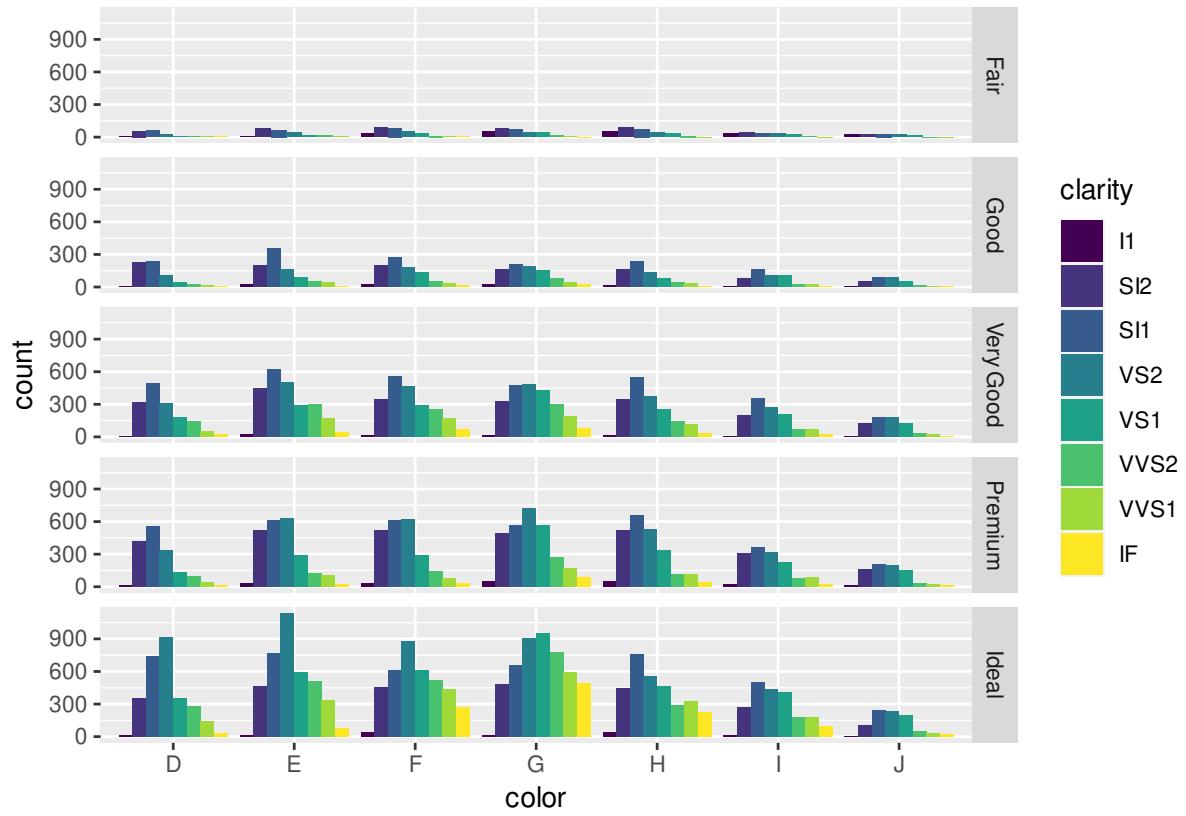


图 11.61: 分面复合条形图

```
##      J  23   27   28   23   16    1    1    0
##
## , , cut = Good
##
##      clarity
## color  I1  SI2  SI1  VS2  VS1  VVS2  VVS1  IF
##   D     8   223  237  104   43   25   13    9
##   E    23   202  355  160   89   52   43    9
##   F    19   201  273  184  132   50   35   15
##   G    19   163  207  192  152   75   41   22
##   H    14   158  235  138   77   45   31    4
##   I     9   81   165  110  103   26   22    6
##   J     4   53   88   90   52   13    1    6
##
## , , cut = Very Good
##
##      clarity
## color  I1  SI2  SI1  VS2  VS1  VVS2  VVS1  IF
##   D     5   314  494  309  175  141   52   23
##   E    22   445  626  503  293  298  170   43
##   F    13   343  559  466  293  249  174   67
##   G    16   327  474  479  432  302  190   79
##   H    12   343  547  376  257  145  115   29
##   I     8   200  358  274  205   71   69   19
##   J     8   128  182  184  120   29   19    8
##
## , , cut = Premium
##
##      clarity
## color  I1  SI2  SI1  VS2  VS1  VVS2  VVS1  IF
##   D    12   421  556  339  131   94   40   10
##   E    30   519  614  629  292  121  105   27
##   F    34   523  608  619  290  146   80   31
##   G    46   492  566  721  566  275  171   87
##   H    46   521  655  532  336  118  112   40
##   I    24   312  367  315  221   82   84   23
##   J    13   161  209  202  153   34   24   12
##
## , , cut = Ideal
##
##      clarity
## color  I1  SI2  SI1  VS2  VS1  VVS2  VVS1  IF
##   D    13   356  738  920  351  284  144   28
##   E    18   469  766 1136  593  507  335   79
##   F    42   453  608  879  616  520  440  268
```



```
##    G    16   486   660   910   953   774   594   491
##    H    38   450   763   556   467   289   326   226
##    I    17   274   504   438   408   178   179   95
##    J     2   110   243   232   201    54    29    25

# 漫谈条形图 https://cosx.org/2017/10/discussion-about-bar-graph
set.seed(2020)
dat <- data.frame(
  age = rep(1:30, 2),
  gender = rep(c("man", "woman"), each = 30),
  num = sample(x = 1:100, size = 60, replace = T)
)
# 重叠
p1 <- ggplot(data = dat, aes(x = age, y = num, fill = gender)) +
  geom_col(position = "identity", alpha = 0.5)
# 堆积
p2 <- ggplot(data = dat, aes(x = age, y = num, fill = gender)) +
  geom_col(position = "stack")
# 双柱
p3 <- ggplot(data = dat, aes(x = age, y = num, fill = gender)) +
  geom_col(position = "dodge")
# 百分比
p4 <- ggplot(data = dat, aes(x = age, y = num, fill = gender)) +
  geom_col(position = "fill") +
  scale_y_continuous(labels = scales::percent_format()) +
  labs(y = "%")
(p1 + p2) / (p3 + p4)
```

以数据集 diamonds 为例，按照纯净度 clarity 和切工 cut 分组统计钻石的数量，再按切工分组统计不同纯净度的钻石数量占比，如表 11.2 所示

```
library(data.table)
diamonds <- as.data.table(diamonds)
dat <- diamonds[, .(cnt = .N), by = .(cut, clarity)] %>%
  .[, pct := cnt / sum(cnt), by = .(cut)] %>%
  .[, pct_pp := paste0(cnt, " (", scales::percent(pct, accuracy = 0.01), ")")]
# 分组计数 with(diamonds, table(clarity, cut))
dcast(dat, formula = clarity ~ cut, value.var = "pct_pp") %>%
  knitr::kable(alignment = "crrrrr", caption = "数值和比例组合呈现")
```

分别以堆积条形图和百分比堆积条形图展示，添加注释到条形图上，见 11.63

```
p1 = ggplot(data = dat, aes(x = cut, y = cnt, fill = clarity)) +
  geom_col(position = "dodge") +
  geom_text(aes(label = cnt), position = position_dodge(1), vjust = -0.5) +
  geom_text(aes(label = scales::percent(pct, accuracy = 0.1)),
            position = position_dodge(1), vjust = 1, hjust = 0.5
  ) +
```

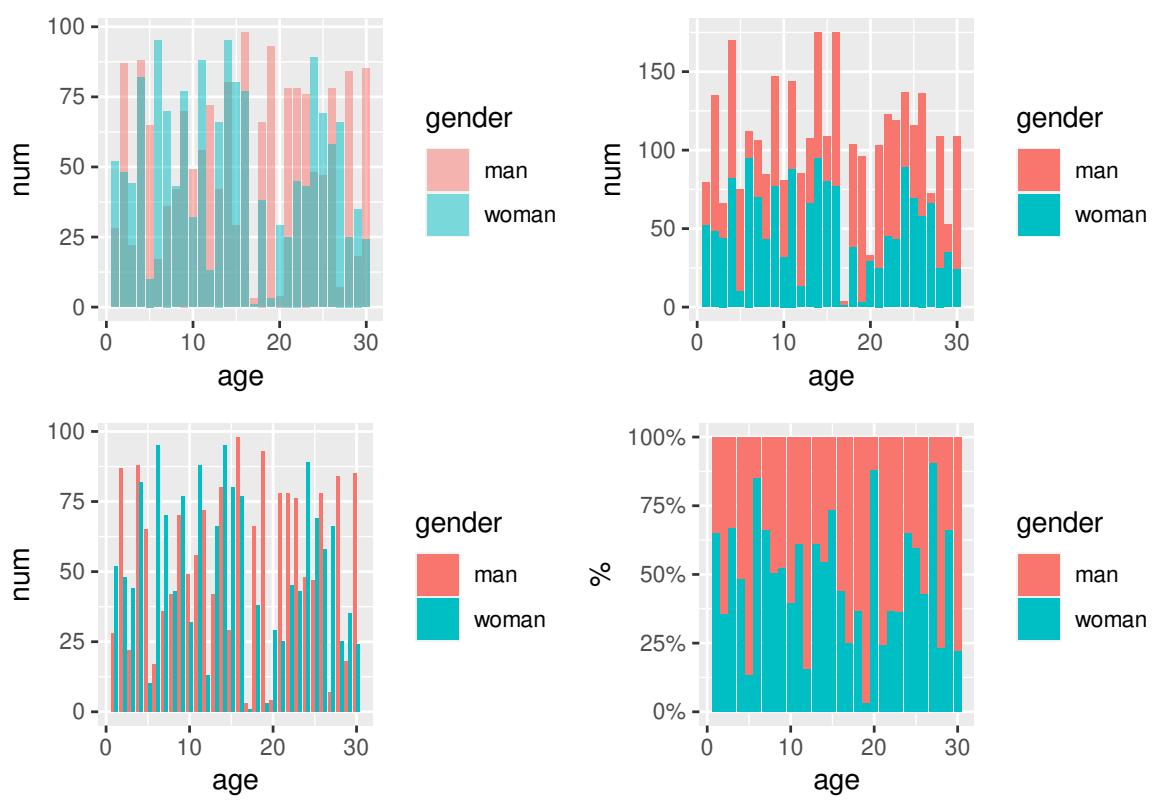


图 11.62: 条形图的四种常见形态

表 11.2: 数值和比例组合呈现

clarity	Fair	Good	Very Good	Premium	Ideal
I1	210 (13.04%)	96 (1.96%)	84 (0.70%)	205 (1.49%)	146 (0.68%)
SI2	466 (28.94%)	1081 (22.03%)	2100 (17.38%)	2949 (21.38%)	2598 (12.06%)
SI1	408 (25.34%)	1560 (31.80%)	3240 (26.82%)	3575 (25.92%)	4282 (19.87%)
VS2	261 (16.21%)	978 (19.93%)	2591 (21.45%)	3357 (24.34%)	5071 (23.53%)
VS1	170 (10.56%)	648 (13.21%)	1775 (14.69%)	1989 (14.42%)	3589 (16.65%)
VVS2	69 (4.29%)	286 (5.83%)	1235 (10.22%)	870 (6.31%)	2606 (12.09%)
VVS1	17 (1.06%)	186 (3.79%)	789 (6.53%)	616 (4.47%)	2047 (9.50%)
IF	9 (0.56%)	71 (1.45%)	268 (2.22%)	230 (1.67%)	1212 (5.62%)



```
scale_fill_brewer(palette = "Spectral") +
  labs(fill = "clarity", y = "", x = "cut") +
  theme_minimal() +
  theme(legend.position = "top")

p2 = ggplot(data = dat, aes(y = cut, x = cnt, fill = clarity)) +
  geom_col(position = "fill") +
  geom_text(aes(label = cnt), position = position_fill(1), vjust = -0.5) +
  geom_text(aes(label = scales::percent(pct, accuracy = 0.1)),
            position = position_fill(1), vjust = 1, hjust = 0.5
  ) +
  scale_fill_brewer(palette = "Spectral") +
  scale_x_continuous(labels = scales::percent) +
  labs(fill = "clarity", y = "", x = "cut") +
  theme_minimal() +
  theme(legend.position = "top")

p1 / p2
```

借助 `plotly` 制作相应的动态百分比堆积条形图

```
ggplot(data = diamonds, aes(x = cut, fill = clarity)) +
  geom_bar(position = "dodge2") +
  scale_fill_brewer(palette = "Spectral")

# 百分比堆积条形图
plotly::plot_ly(dat,
  x = ~cut, color = ~clarity, y = ~pct,
  colors = "Spectral", type = "bar",
  text = ~ paste0(
    cnt, "颗 <br>",
    "占比: ", scales::percent(pct, accuracy = 0.1), "<br>"
  ),
  hoverinfo = "text"
) %>%
  plotly::layout(
    barmode = "stack",
    yaxis = list(tickformat = ".0%")
  ) %>%
  plotly::config(displayModeBar = FALSE)

# `type = "histogram"` 以 cut 和 clarity 分组计数
plotly::plot_ly(diamonds,
  x = ~cut, color = ~clarity,
  colors = "Spectral", type = "histogram"
) %>%
```

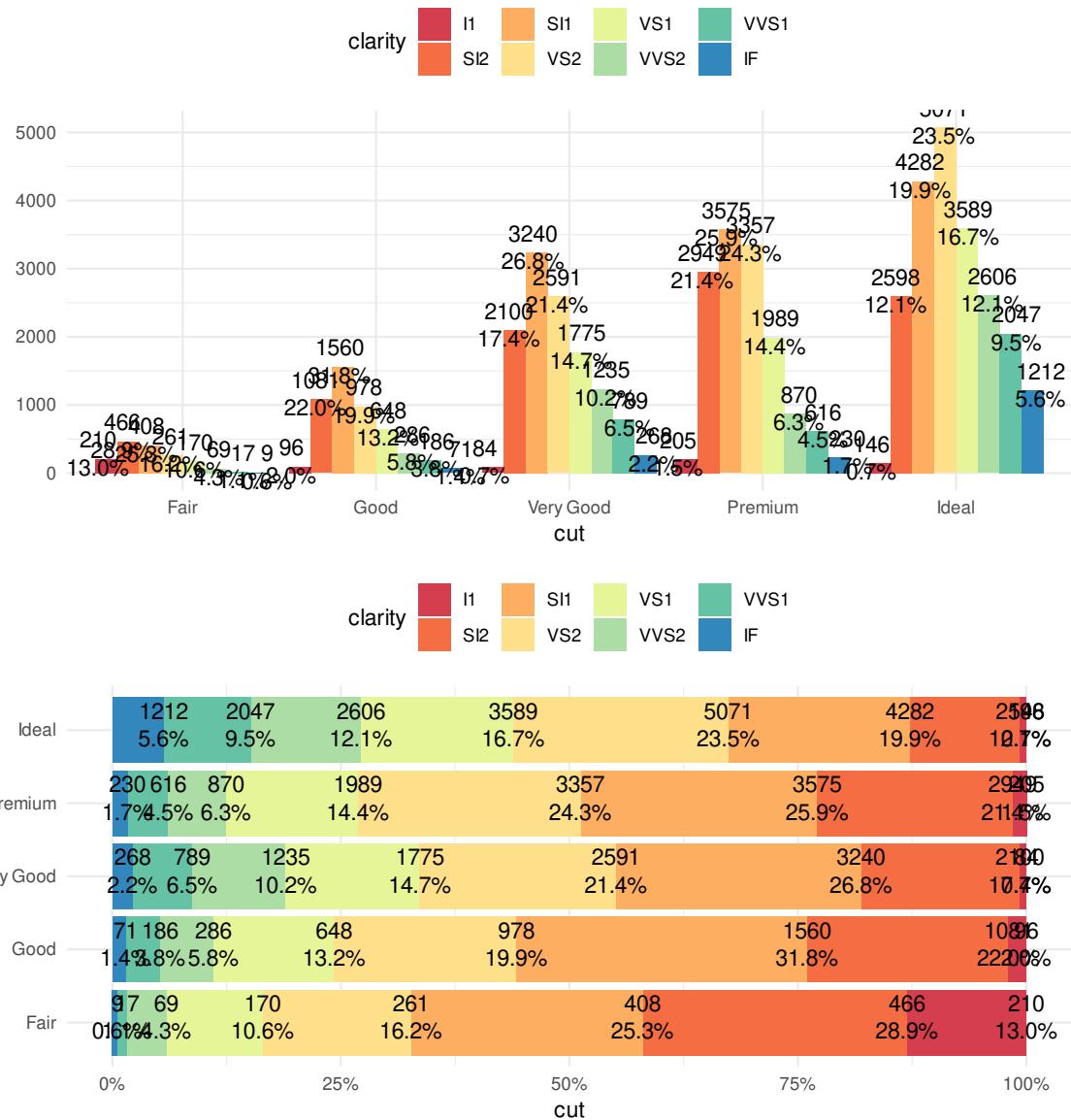


图 11.63: 添加注释到条形图

```
plotly::config(displayModeBar = FALSE)

# 堆积图
plotly::plot_ly(diamonds,
  x = ~cut, color = ~clarity,
  colors = "Spectral", type = "histogram"
) %>%
  plotly::layout(
    barmode = "stack",
    yaxis = list(title = "cnt"),
    legend = list(title = list(text = "clarity"))
) %>%
  plotly::config(displayModeBar = FALSE)
```

#### 11.4.6 直方图

直方图用来查看连续变量的分布

```
ggplot(diamonds, aes(price)) + geom_histogram(bins = 30)
```

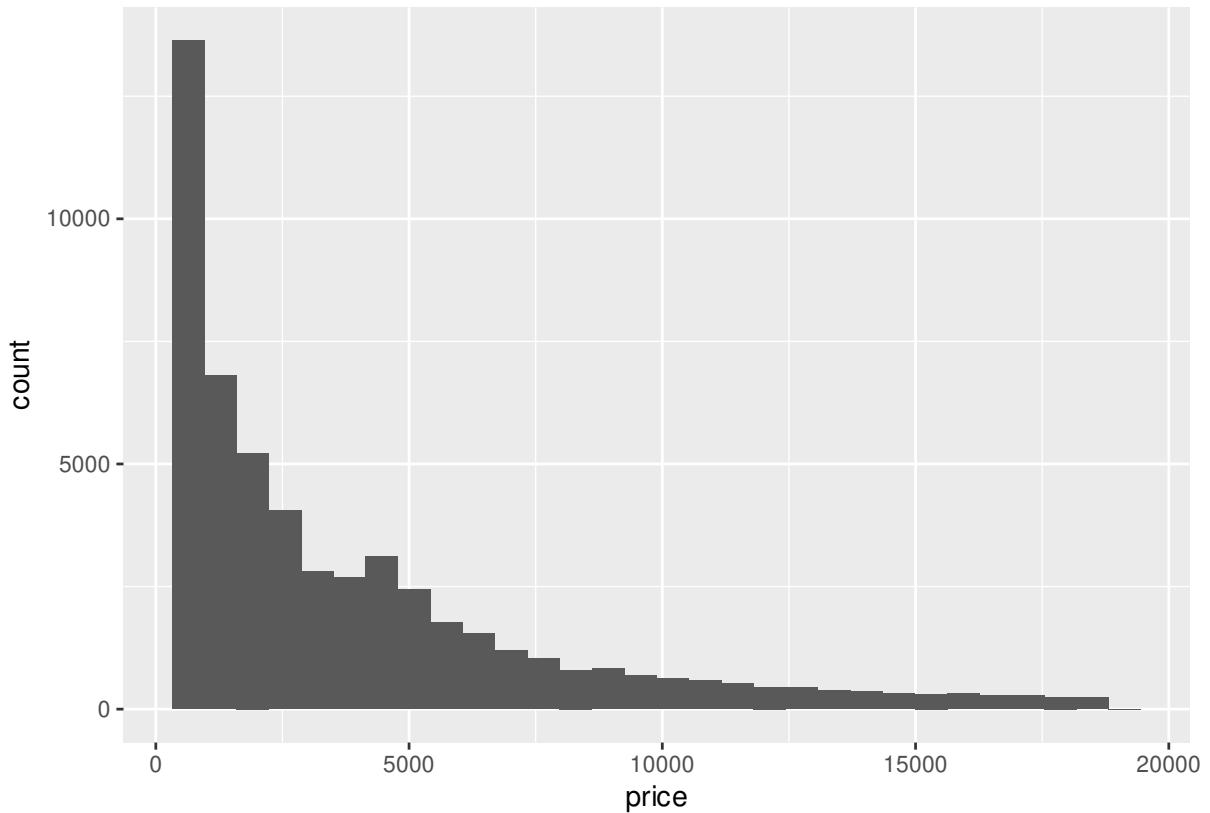


图 11.64: 钻石价格的分布

堆积直方图

```
ggplot(diamonds, aes(x = price, fill = cut)) + geom_histogram(bins = 30)
```

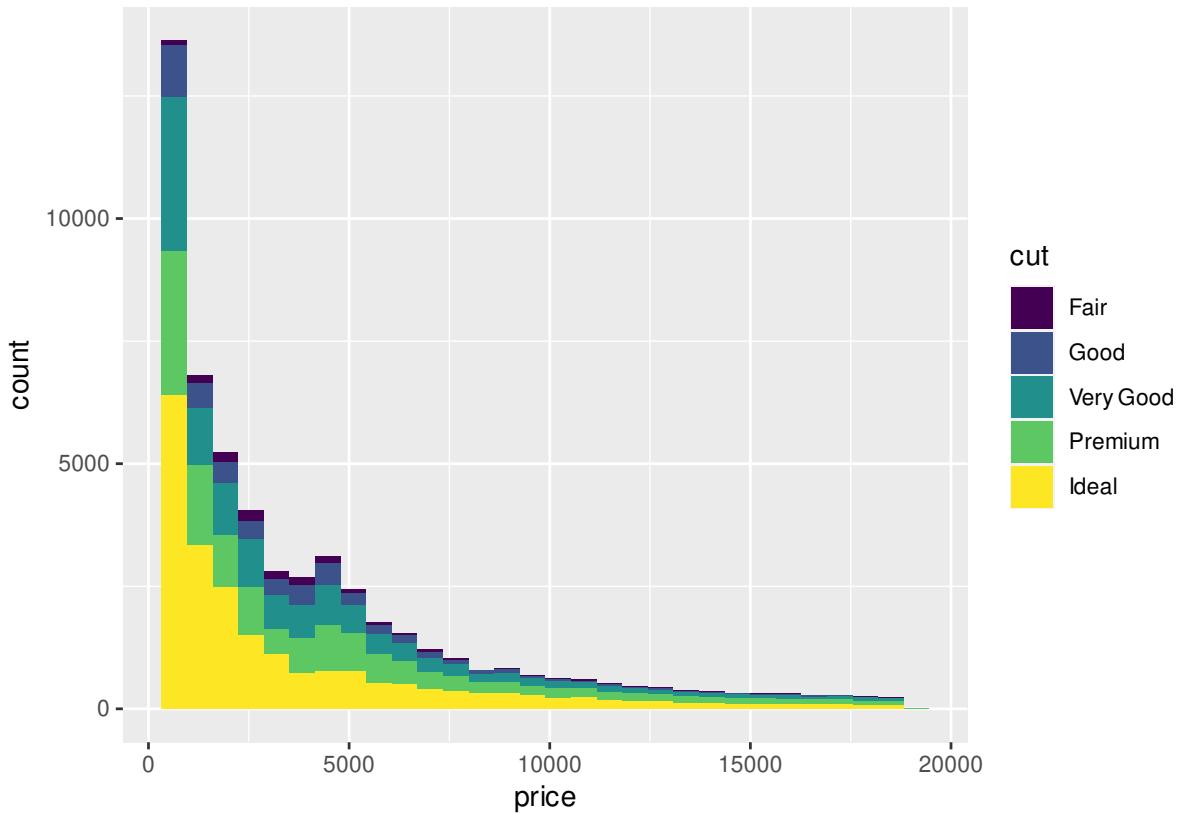


图 11.65: 钻石价格随切割质量的分布

基础 R 包与 Ggplot2 包绘制的直方图的对比，Base R 绘图速度快，代码更加稳定，Ggplot2 代码简洁，更美观

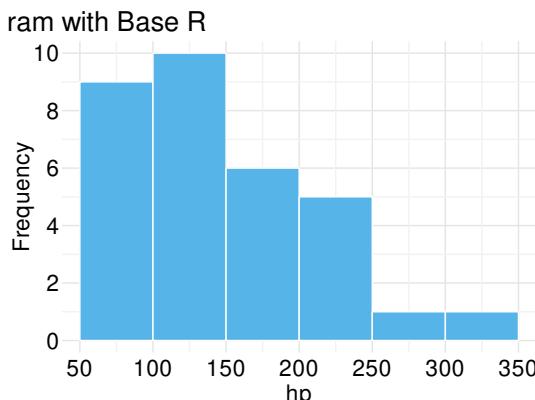
```
par(mar = c(2.1, 2.1, 1.5, 0.5))
plot(c(50, 350), c(0, 10),
  type = "n", font.main = 1,
  xlab = "", ylab = "", frame.plot = FALSE, axes = FALSE,
  # xlab = "hp", ylab = "Frequency",
  main = paste("Histogram with Base R", paste(rep(" ", 60), collapse = "")))
)
axis(
  side = 1, at = seq(50, 350, 50), labels = seq(50, 350, 50),
  tick = FALSE, las = 1, padj = 0, mgp = c(3, 0.1, 0)
)
axis(
  side = 2, at = seq(0, 10, 2), labels = seq(0, 10, 2),
  # col = "white", 坐标轴的颜色
  # col.ticks 刻度线的颜色
  tick = FALSE, # 取消刻度线
  las = 1, # 水平方向
  hadj = 1, # 右侧对齐
```

```

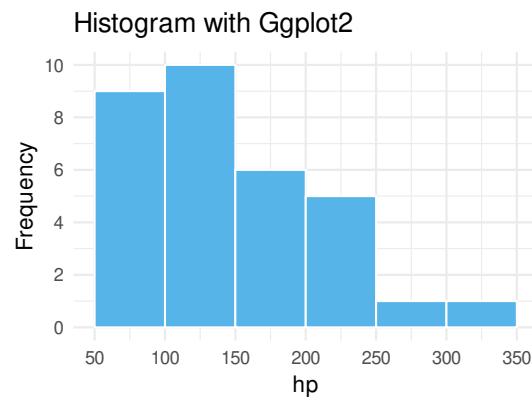
mgp = c(3, 0.1, 0) # 纵轴边距线设置为 0.1
)
abline(h = seq(0, 10, 2), v = seq(50, 350, 50), col = "gray90", lty = "solid")
abline(h = seq(1, 9, 2), v = seq(75, 325, 50), col = "gray95", lty = "solid")
hist(mtcars$hp,
  col = "#56B4E9", border = "white",
  freq = TRUE, add = TRUE
  # labels = TRUE, axes = TRUE, ylim = c(0, 10.5),
  # xlab = "hp", main = "Histogram with Base R"
)
mtext("hp", 1, line = 1.0)
mtext("Frequency", 2, line = 1.0)

ggplot(mtcars) +
  geom_histogram(aes(x = hp), fill = "#56B4E9", color = "white", breaks = seq(50, 350, 50)) +
  scale_x_continuous(breaks = seq(50, 350, 50)) +
  scale_y_continuous(breaks = seq(0, 12, 2)) +
  labs(x = "hp", y = "Frequency", title = "Histogram with Ggplot2") +
  theme_minimal(base_size = 12)

```



(a) Base R 直方图



(b) Ggplot2 直方图

图 11.66: 直方图

#### 11.4.7 箱线图

以 PlantGrowth 数据集为例展示箱线图，在两组不同实验条件下，植物生长的情况，纵坐标是干燥植物的量，横坐标表示不同的实验条件。这是非常典型的适合用箱线图来表达数据的场合，Y 轴对应数值型变量，X 轴对应分类变量，在 R 语言中，分类变量的类型是 factor

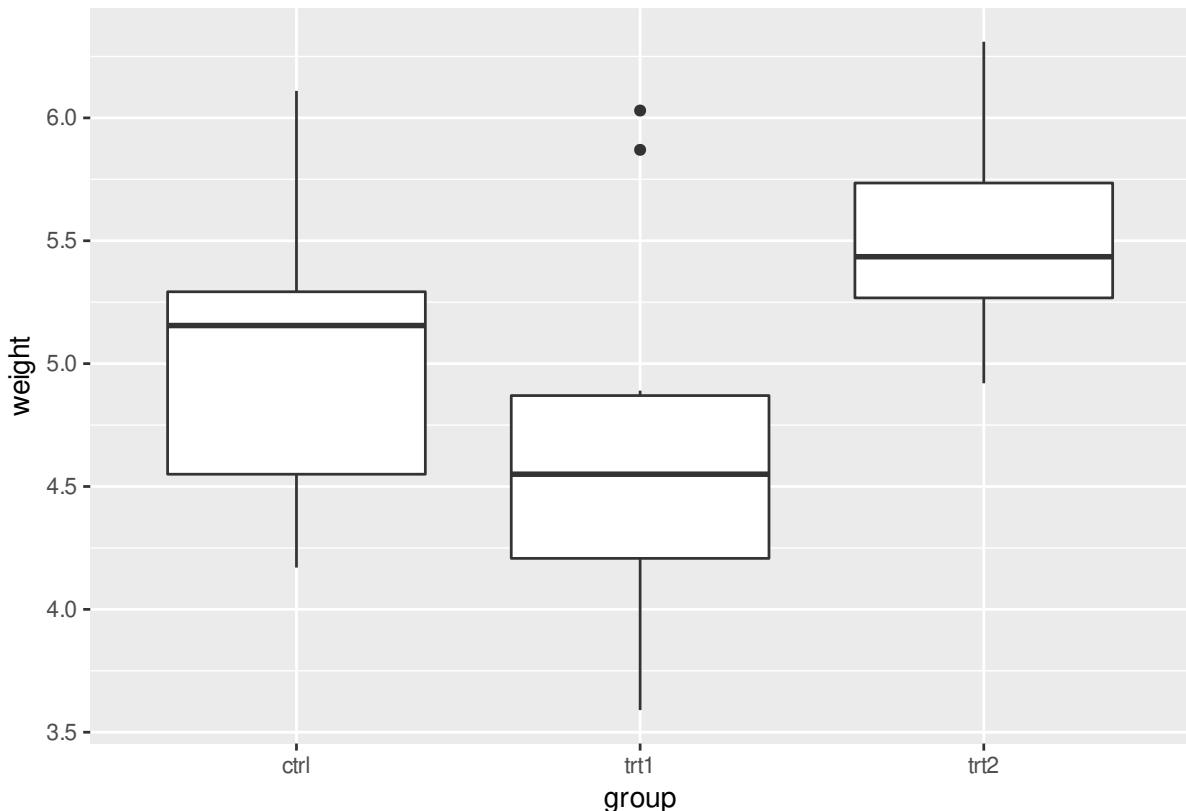
```

data("PlantGrowth")
str(PlantGrowth)

## 'data.frame': 30 obs. of 2 variables:
## $ weight: num 4.17 5.58 5.18 6.11 4.5 4.61 5.17 4.53 5.33 5.14 ...

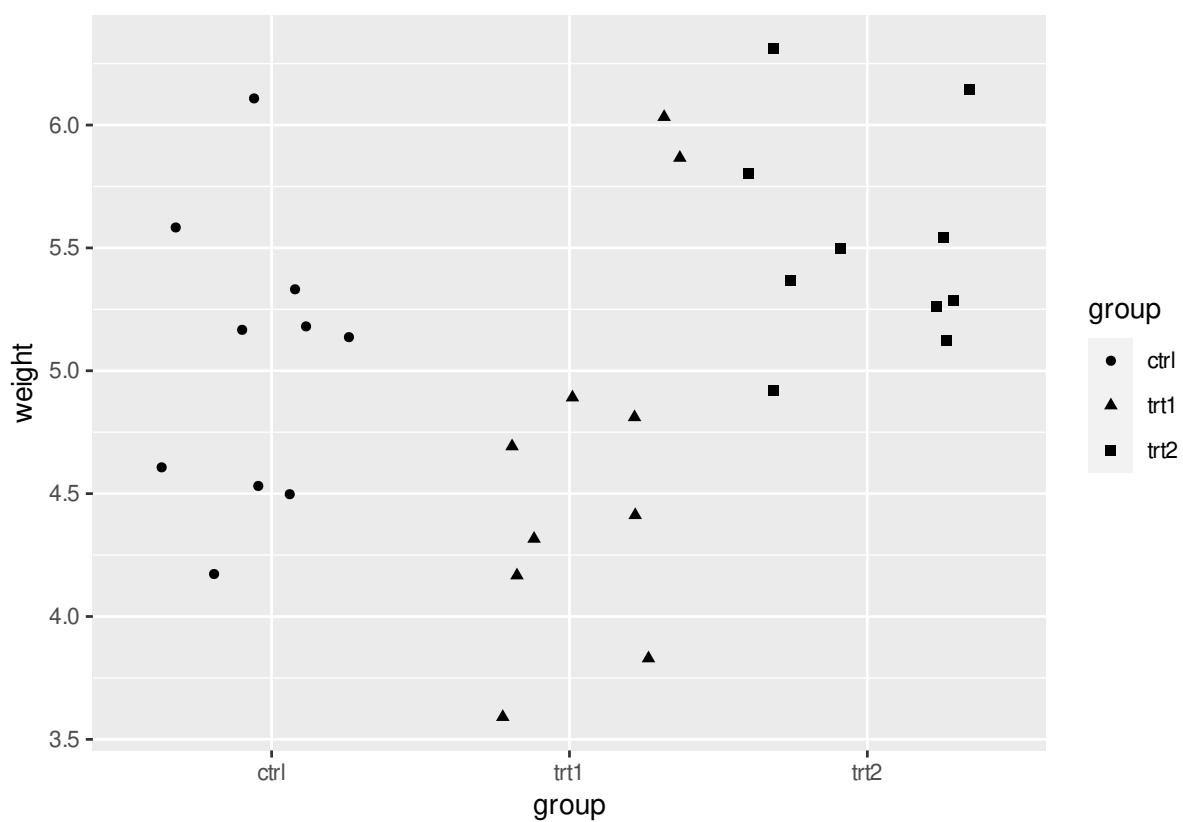
```

```
## $ group : Factor w/ 3 levels "ctrl","trt1",...: 1 1 1 1 1 1 1 1 1 1 ...  
ggplot(data = PlantGrowth, aes(x = group, y = weight)) + geom_boxplot()
```

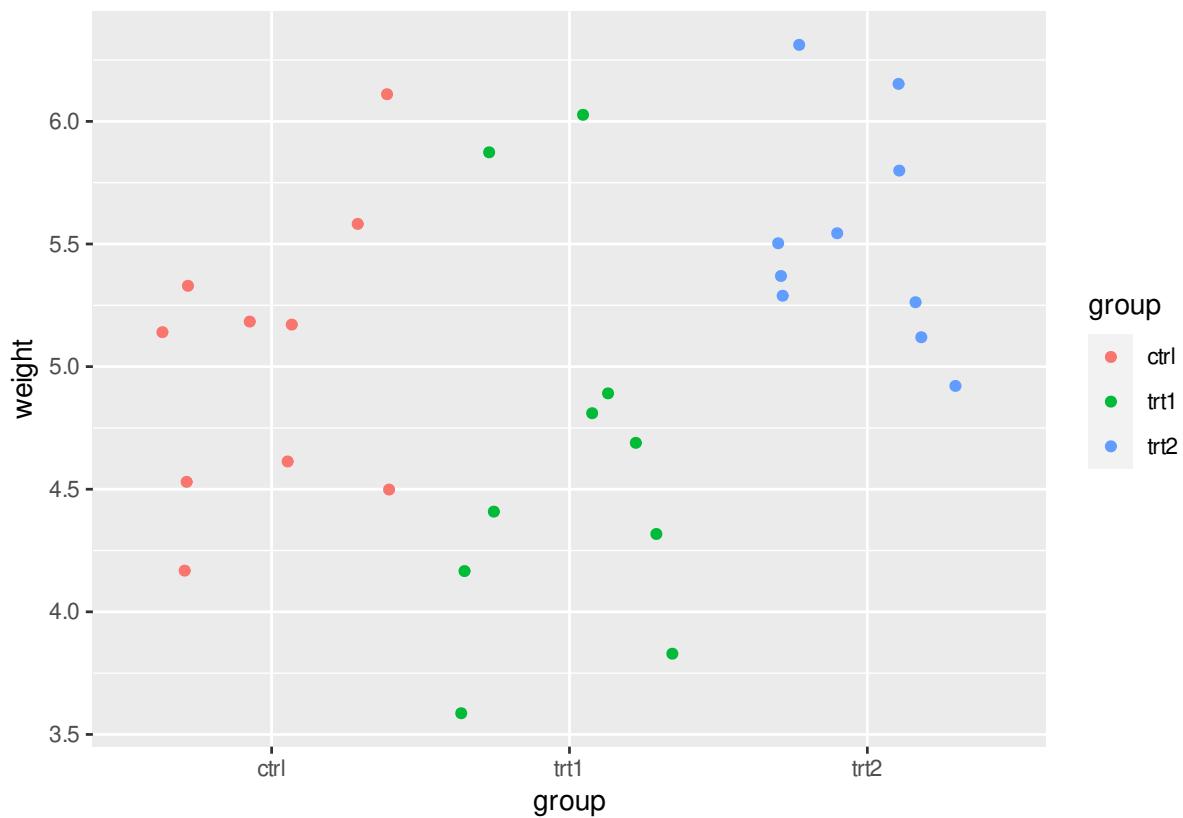


PlantGrowth 数据量比较小，此时比较适合采用抖动散点图，抖动是为了避免点之间相互重叠，为了增加不同类别之间的识别性，我们可以用不同的点的形状或者不同的颜色来表示类别

```
ggplot(data = PlantGrowth, aes(x = group, y = weight, shape = group)) + geom_jitter()
```

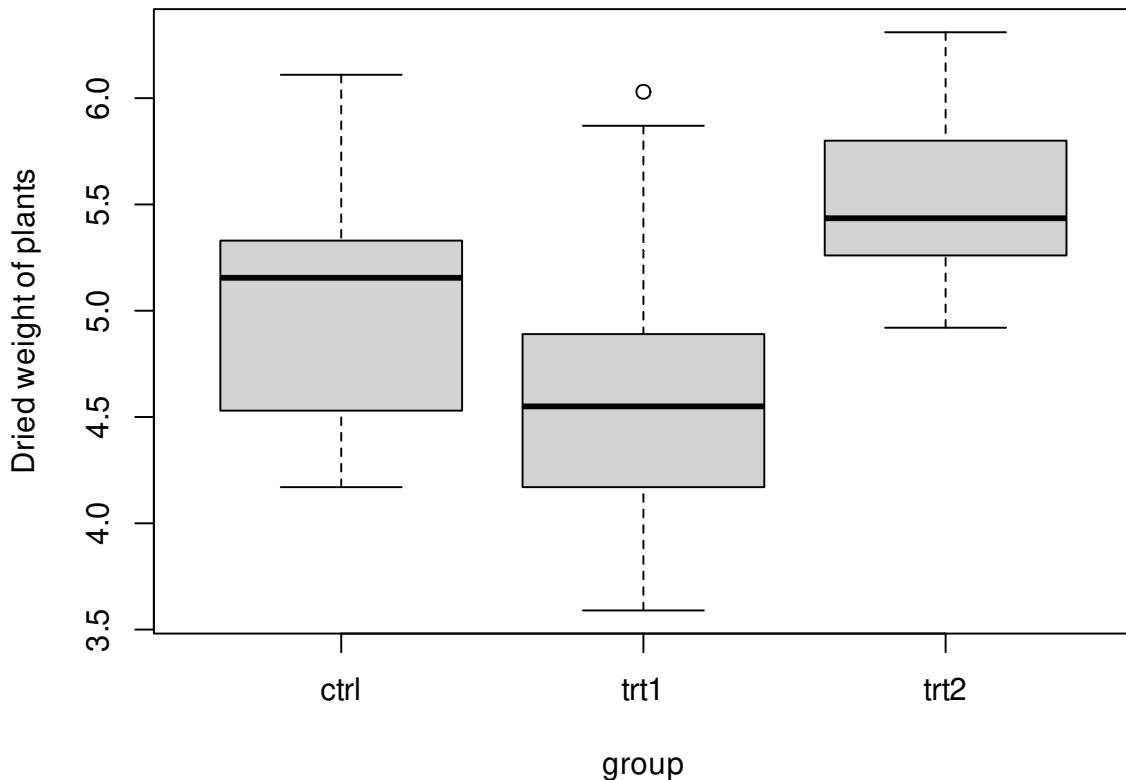


```
ggplot(data = PlantGrowth, aes(x = group, y = weight, color = group)) + geom_jitter()
```



```
boxplot(weight ~ group,
  data = PlantGrowth,
  ylab = "Dried weight of plants", col = "lightgray",
  notch = FALSE, varwidth = TRUE
)
```

C



以钻石切割质量 cut 为分面依据, 以钻石颜色类别 color 为 x 轴, 钻石价格为 y 轴, 绘制箱线图11.67

```
ggplot(diamonds, aes(x = color, y = price, color = cut)) +
  geom_boxplot(show.legend = FALSE) +
  facet_grid(~cut)
```

我们当然还可以添加钻石的纯净度 clarity 作为分面依据, 那么箱线图可以为图 11.68

```
ggplot(diamonds, aes(x = color, y = price, color = cut)) +
  geom_boxplot(show.legend = FALSE) +
  facet_grid(clarity ~ cut)
```

经过观察, 我们发现水平分类过多, 考虑用切割质量 cut 替换钻石颜色 color 绘图, 但是由于分类过细, 图信息展示不简练, 反而不好, 如图 11.69

```
ggplot(diamonds, aes(x = cut, y = price, color = cut)) +
  geom_boxplot(show.legend = FALSE) +
```

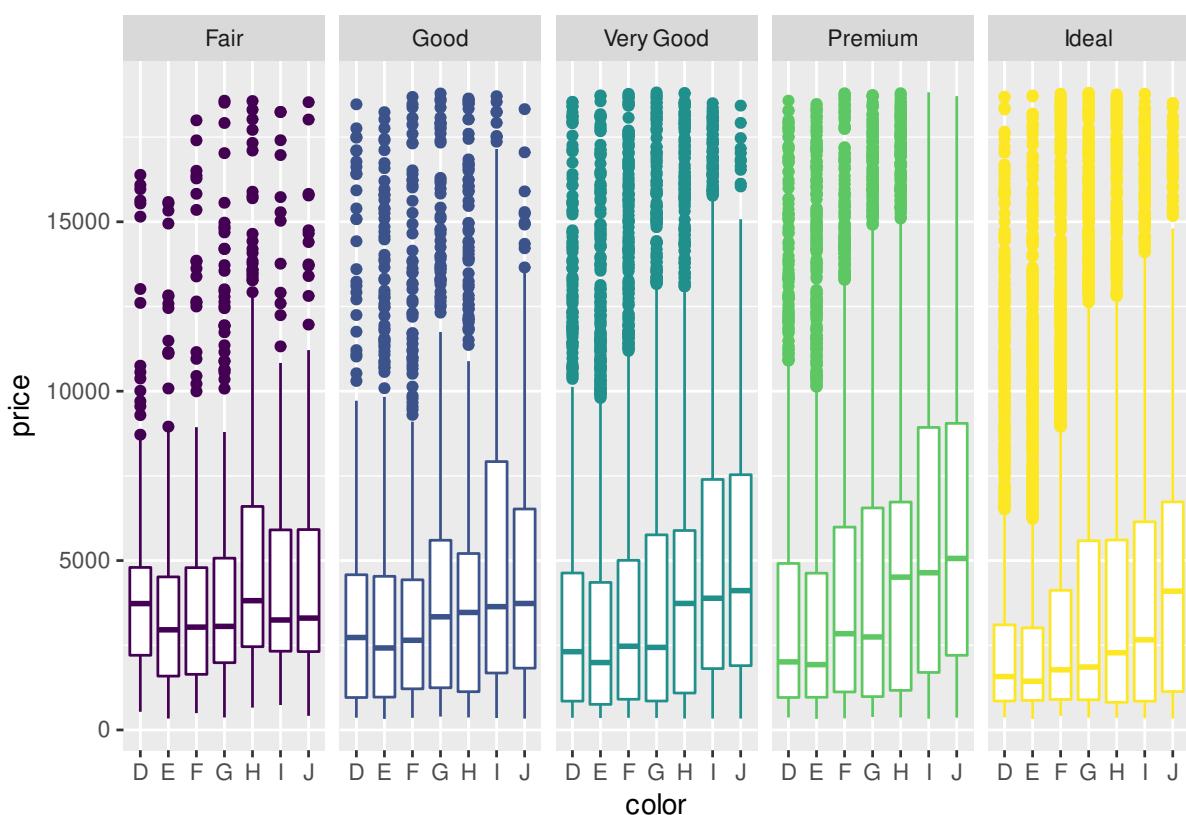


图 11.67: 箱线图

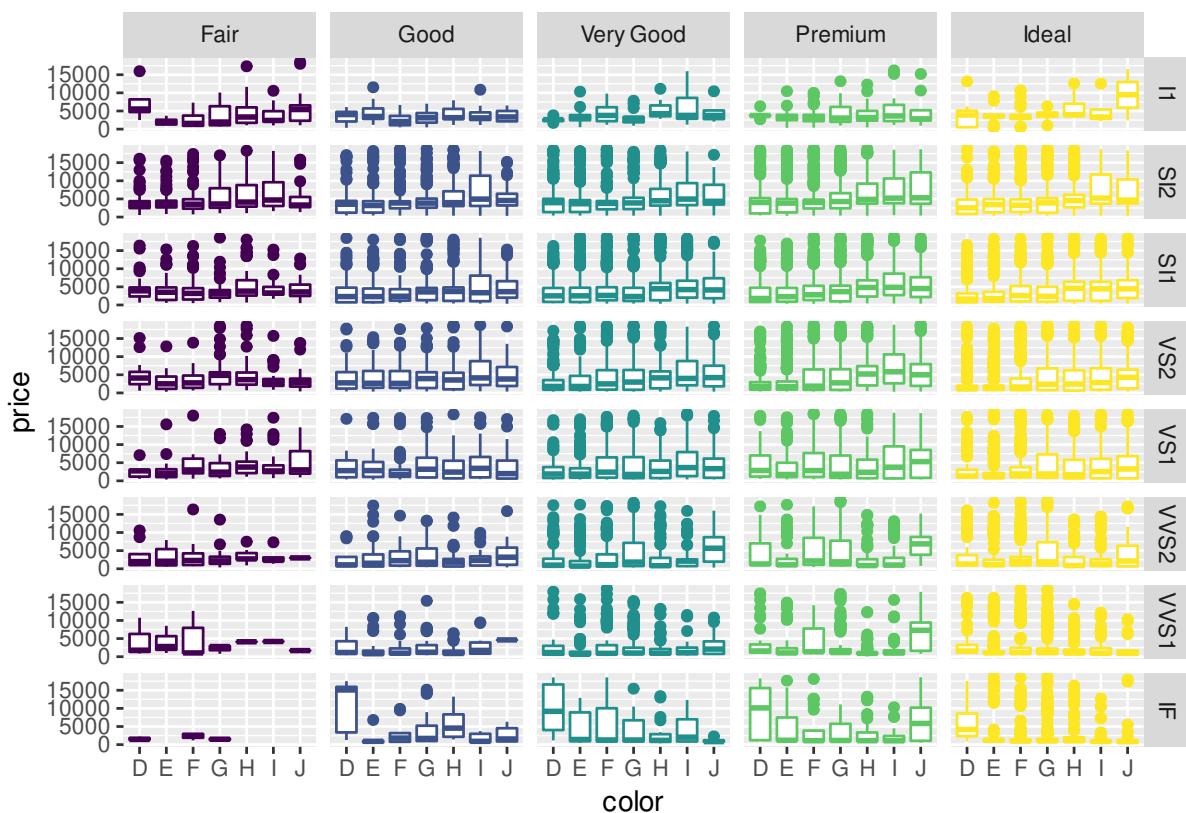


图 11.68: 复合分面箱线图



```
facet_grid(clarity ~ color)
ggplot(diamonds, aes(x = cut, y = price, color = color)) +
  geom_boxplot(show.legend = FALSE) +
  facet_grid(clarity ~ color)
```



### 11.4.8 函数图

蝴蝶图的参数方程如下

$$x = \sin t \left( e^{\cos t} - 2 \cos 4t + \sin^5 \left( \frac{t}{12} \right) \right) \quad (11.1)$$

$$y = \cos t \left( e^{\cos t} - 2 \cos 4t + \sin^5 \left( \frac{t}{12} \right) \right), t \in [-\pi, \pi] \quad (11.2)$$

### 11.4.9 密度图

```
ggplot(mpg, aes(cty)) +
  geom_density(aes(fill = factor(cyl)), alpha = 0.8) +
  labs(
    title = "Density plot",
    subtitle = "City Mileage Grouped by Number of cylinders",
    caption = "Source: mpg",
    x = "City Mileage",
    fill = "# Cylinders"
  )
```

添加透明度，解决遮挡

```
ggplot(diamonds, aes(x = price, fill = cut)) + geom_density()
```

```
ggplot(diamonds, aes(x = price, fill = cut)) + geom_density(alpha = 0.5)
```

堆积密度图

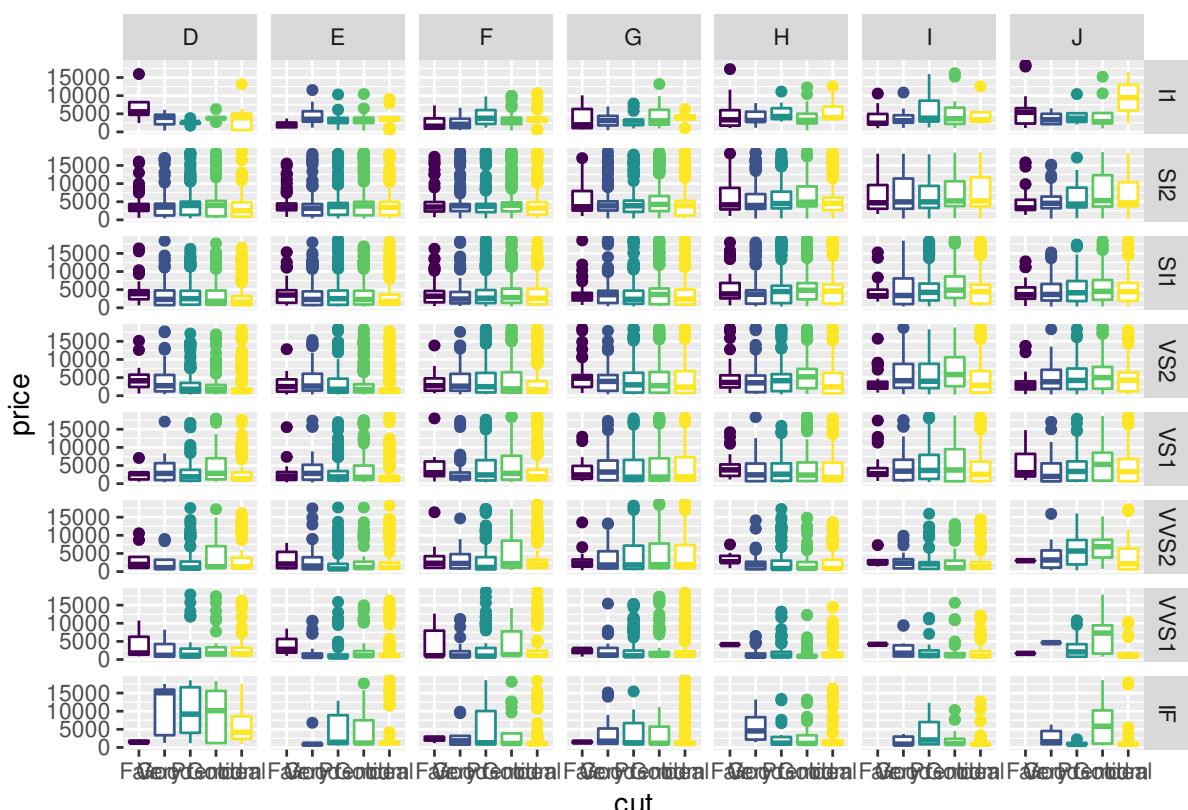
```
ggplot(diamonds, aes(x = price, fill = cut)) +
  geom_density(position = "stack")
```

条件密度估计

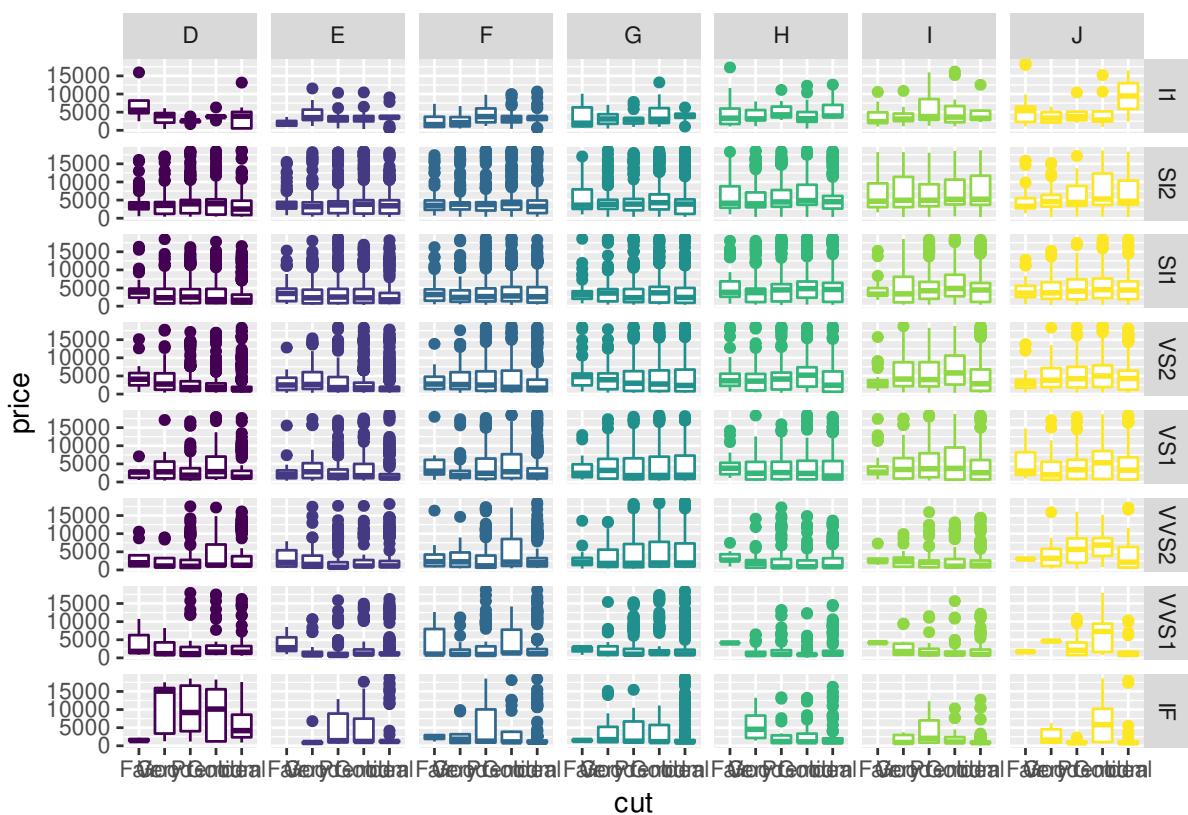
```
# You can use position="fill" to produce a conditional density estimate
ggplot(diamonds, aes(carat, stat(count), fill = cut)) +
  geom_density(position = "fill")
```

岭线图是密度图的一种变体，可以防止密度曲线重叠在一起

```
ggplot(diamonds) +
  ggridges::geom_density_ridges(aes(x = price, y = color, fill = color))
```



(a) 切割质量 cut 上色



(b) 钻石颜色配色

图 11.69: 箱线图配色

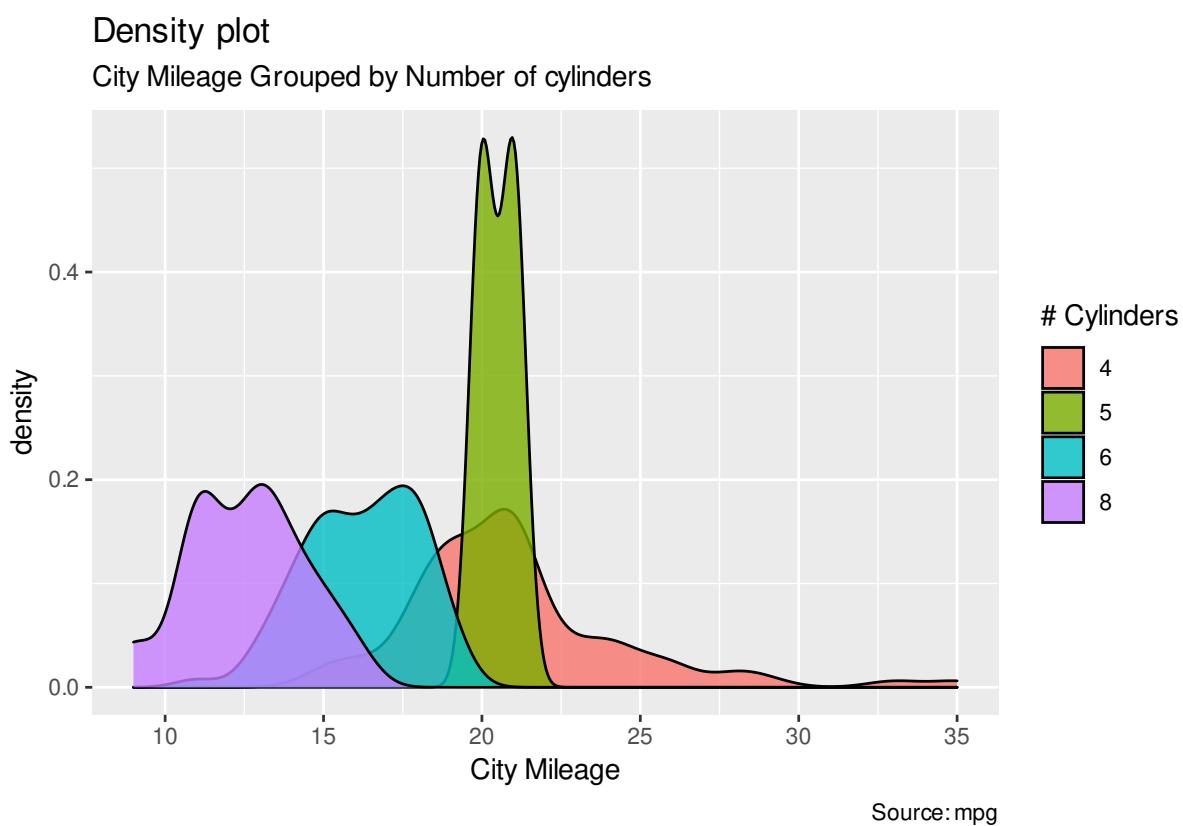


图 11.70: 按汽缸数分组的城市里程

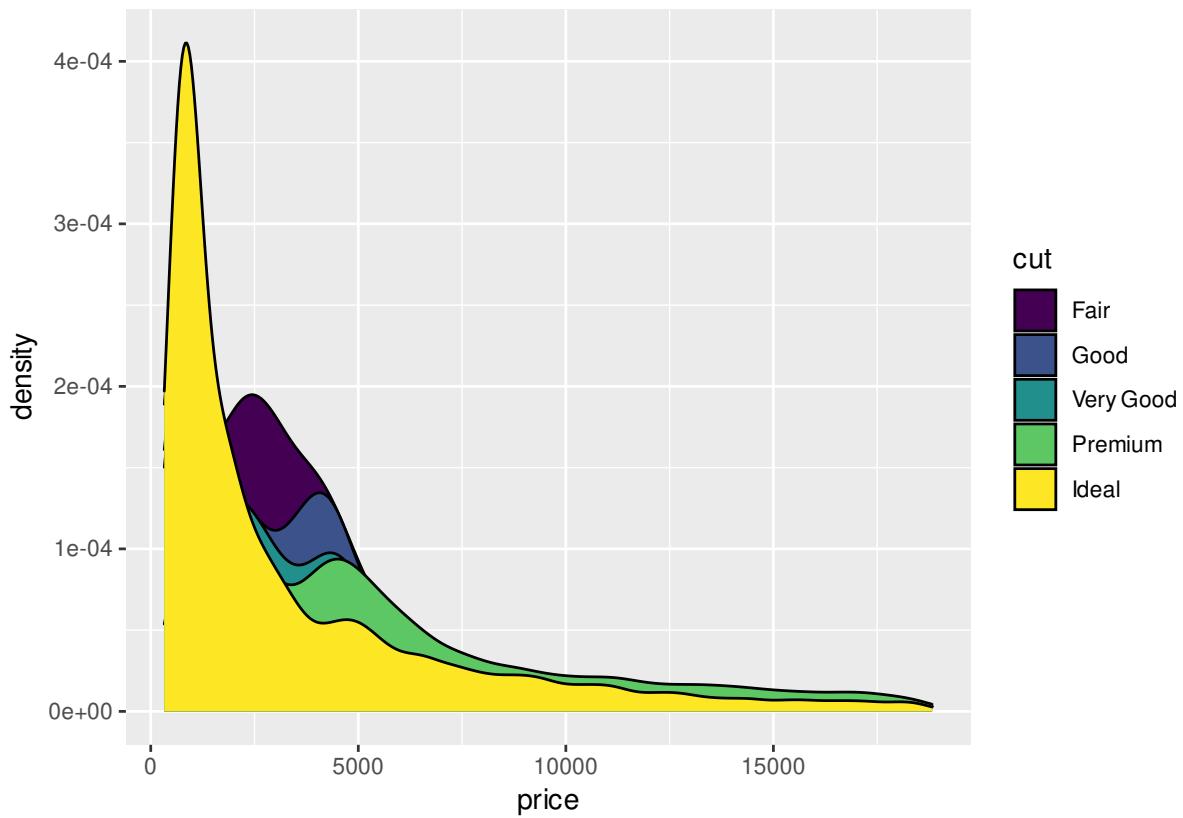


图 11.71: 密度图

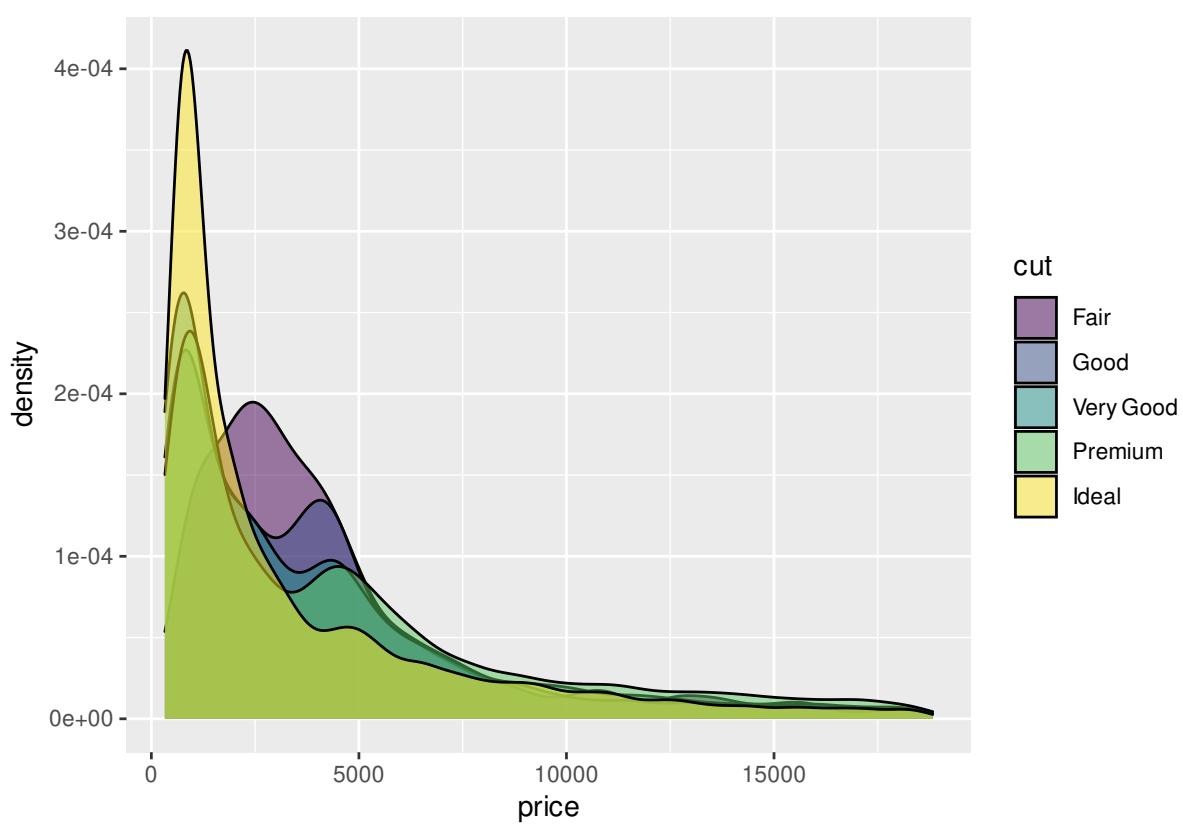


图 11.72: 添加透明度的密度图

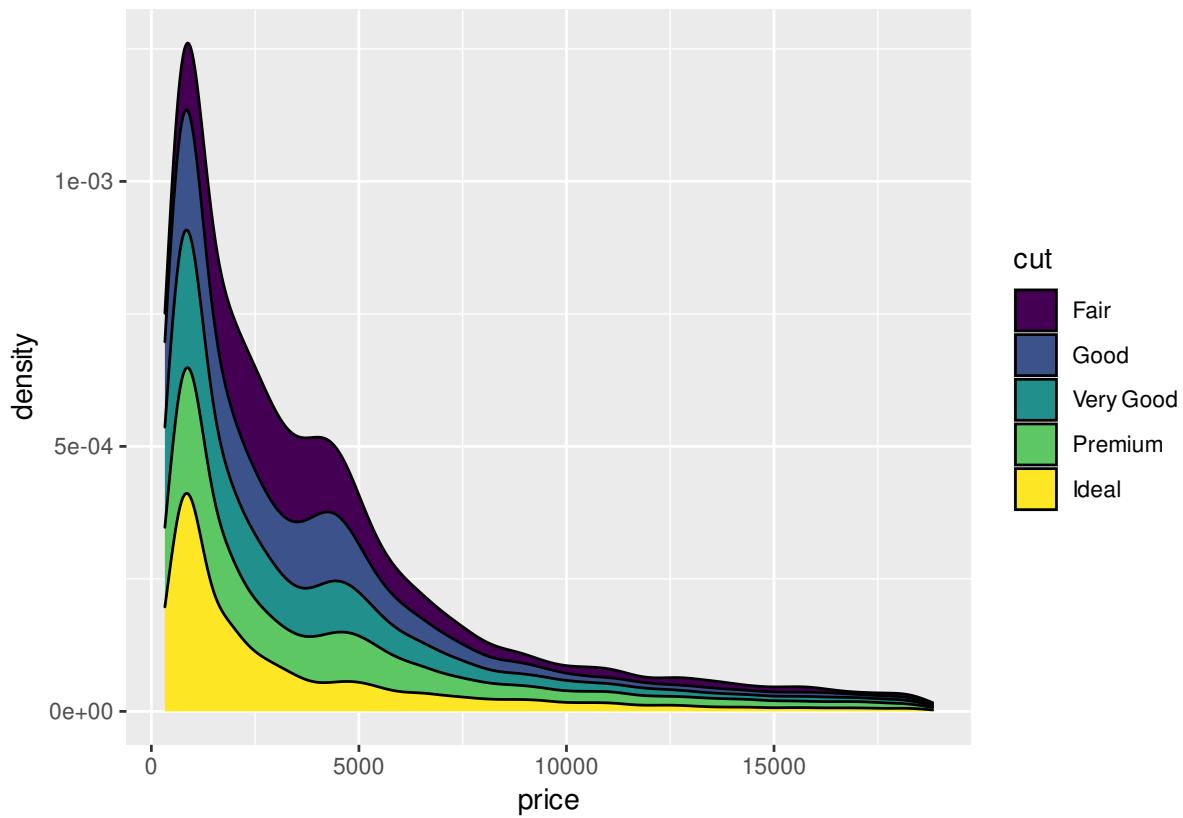


图 11.73: 堆积密度图

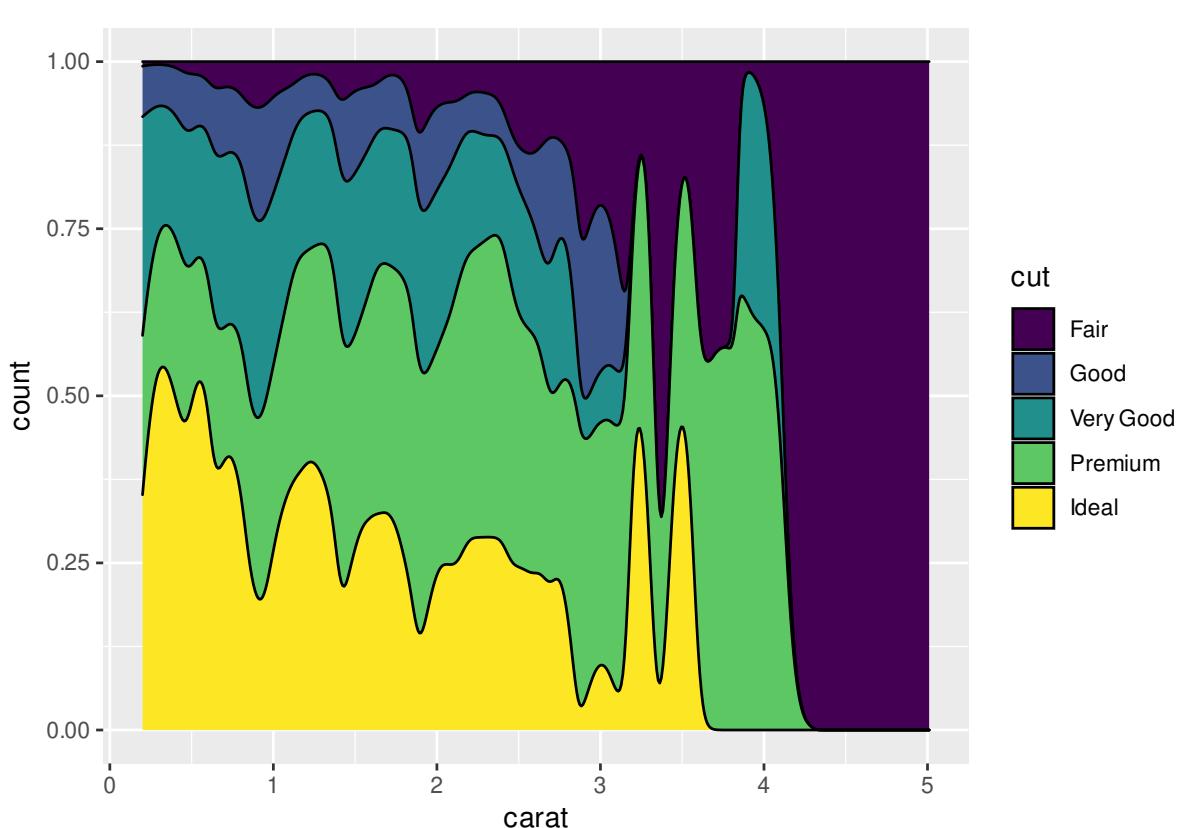
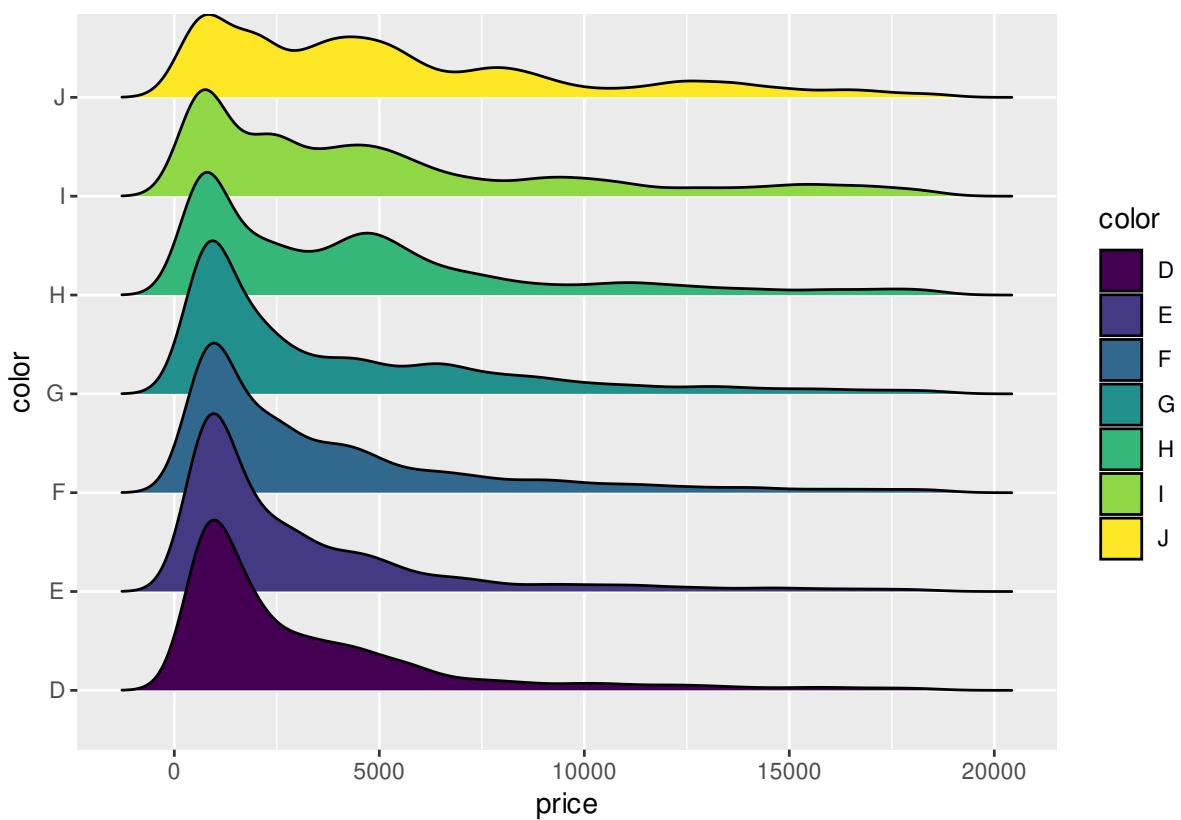
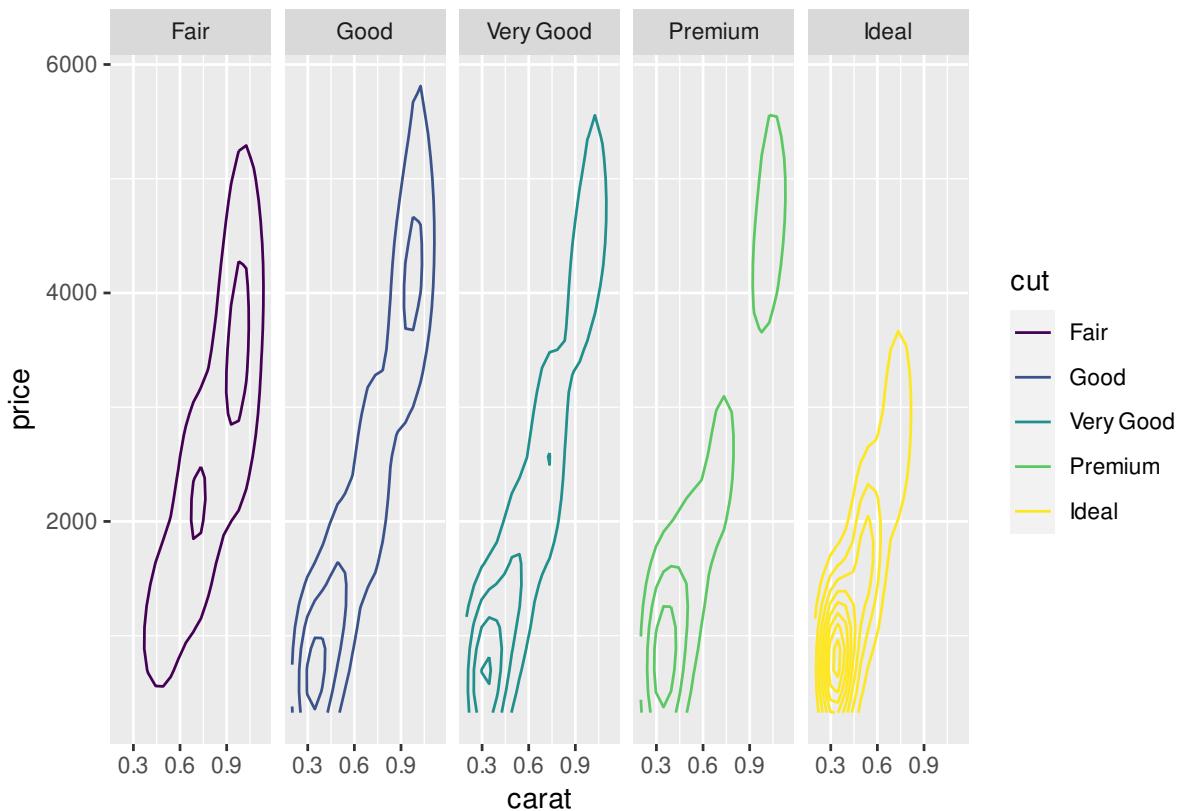


图 11.74: 条件密度估计图



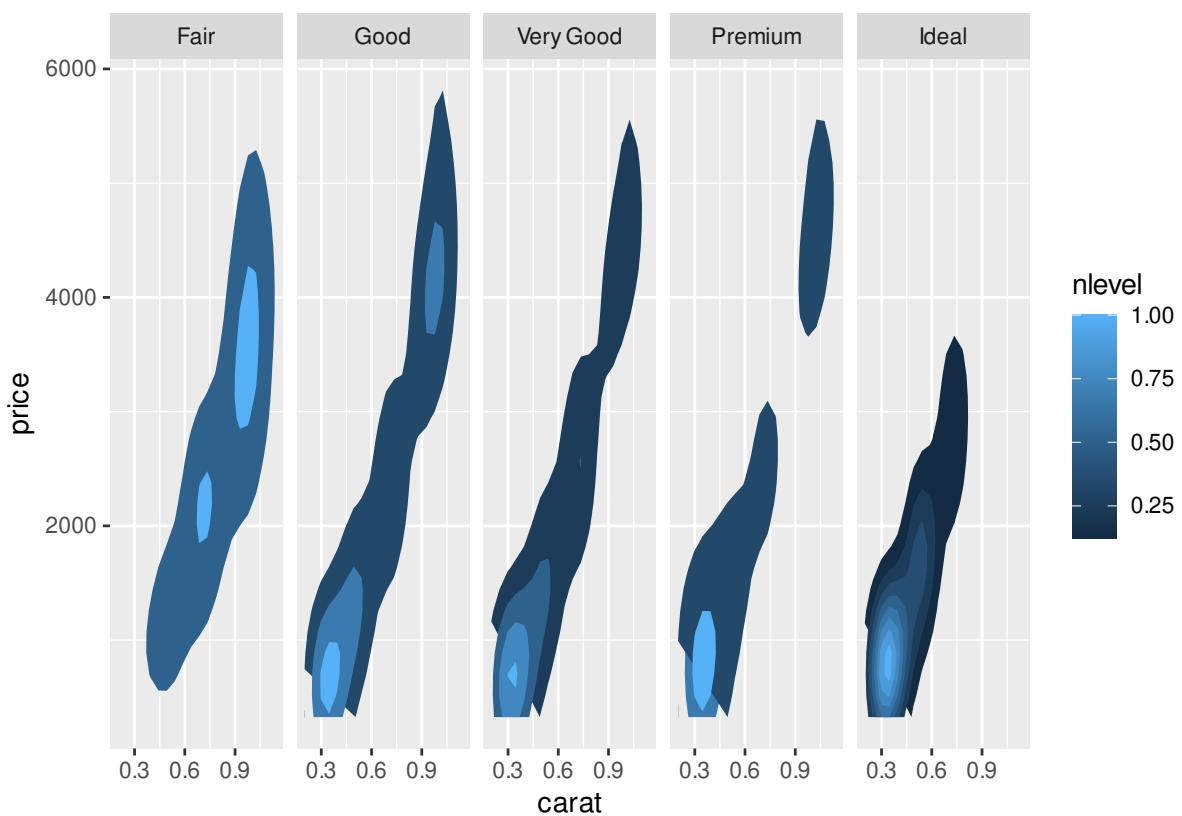
二维的密度图又是一种延伸

```
ggplot(diamonds, aes(x = carat, y = price)) +  
  geom_density_2d(aes(color = cut)) +  
  facet_grid(~cut)
```



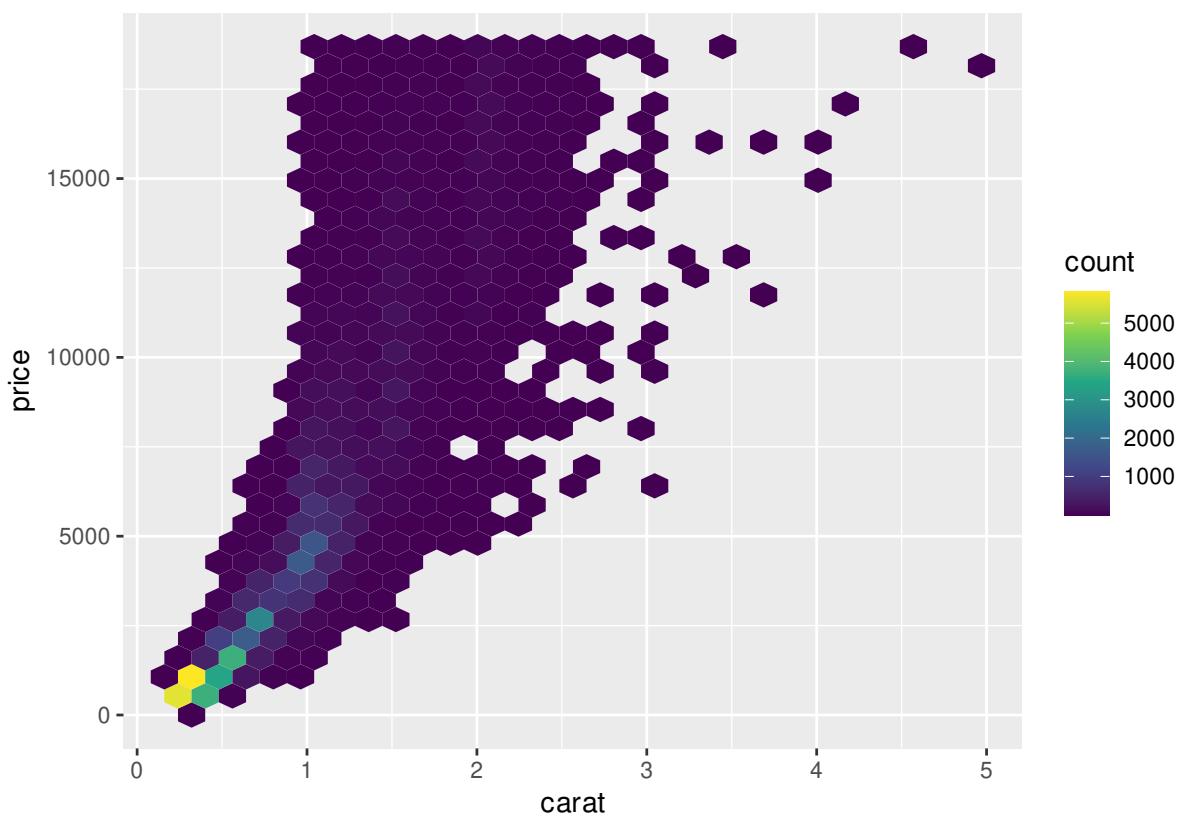
`stat` 函数，特别是 `nlevel` 参数，在密度曲线之间填充我们又可以得到热力图

```
ggplot(diamonds, aes(x = carat, y = price)) +  
  stat_density_2d(aes(fill = stat(nlevel)), geom = "polygon") +  
  facet_grid(. ~ cut)
```



geom\_hex 也是二维密度图的一种变体，特别适合数据量比较大的情形

```
ggplot(diamonds, aes(x = carat, y = price)) + geom_hex() +  
  scale_fill_viridis_c()
```



### heatmaps in ggplot2 二维密度图

```
ggplot(faithful, aes(x = eruptions, y = waiting)) +  
  stat_density_2d(aes(fill = ..level..), geom = "polygon") +  
  xlim(1, 6) +  
  ylim(40, 100)  
  
ggplot(faithful, aes(x = eruptions, y = waiting)) +  
  stat_density2d(aes(fill = stat(level)), geom = "polygon") +  
  scale_fill_viridis_c(option = "viridis") +  
  xlim(1, 6) +  
  ylim(40, 100)
```

提示

MASS:::kde2d() 实现二维核密度估计, ggplot2 包提供了两种等价的绘图方式

1. stat\_density\_2d() 和 ..
2. stat\_density2d() 和 stat()

```
plotly::plot_ly(  
  data = faithful, x = ~eruptions,  
  y = ~waiting, type = "histogram2dcontour"  
) %>%  
  plotly::config(displayModeBar = FALSE)  
  
# plot_ly(faithful, x = ~waiting, y = ~eruptions) %>%
```

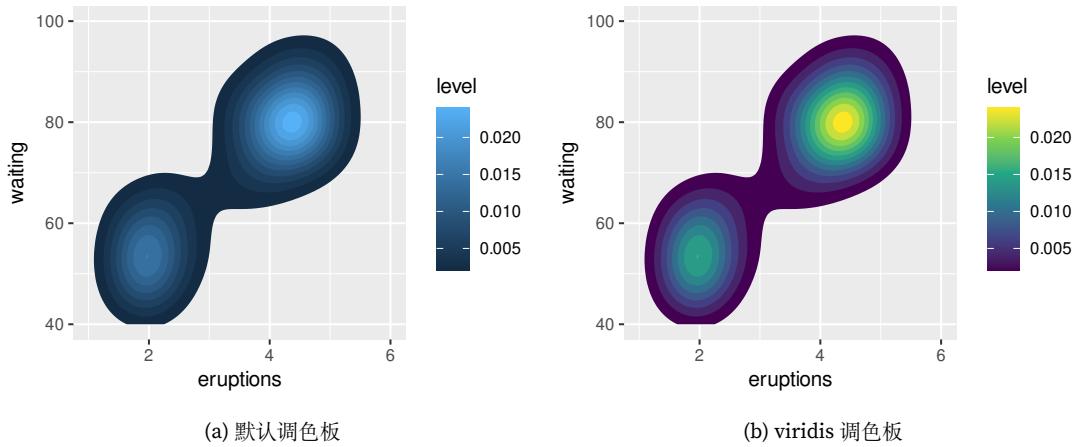


图 11.75: 二维密度图

```
# add_histogram2d() %>%
# add_histogram2dcontour()
```

延伸一下，热力图

```
library(KernSmooth)
den <- bkde2D(x = faithful, bandwidth = c(0.7, 7))
# 热力图
p1 <- plotly::plot_ly(x = den$x1, y = den$x2, z = den$fhat) %>%
  plotly::config(displayModeBar = FALSE) %>%
  plotly::add_heatmap()

# 等高线图
p2 <- plotly::plot_ly(x = den$x1, y = den$x2, z = den$fhat) %>%
  plotly::config(displayModeBar = FALSE) %>%
  plotly::add_contour()

htmltools::tagList(p1, p2)
```

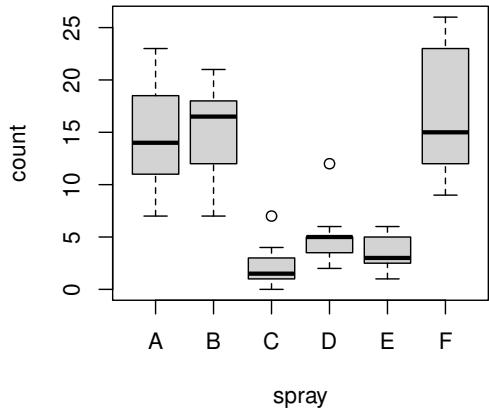
### 11.4.10 提琴图

2004 年 Daniel Adler 开发 `vioplot` 包实现提琴图的绘制，它可能是最早实现此功能的 R 包，随后 10 余年没有更新却一直坚挺在 CRAN 上，非常难得，好在 Thomas Kelly 已经接手维护。另一款绘制提琴图的 R 包是 Peter Kampstra 开发的 `beanplot` [Kampstra, 2008]，也存在很多年了，不过随着时间的变迁，比较现代的方式是 `ggplot2` 带来的 `geom_violin()` 扔掉了很多依赖，也是各种图形的汇集地，可以看作是最佳实践。提琴图比起箱线图优势在于呈现更多的分布信息，其次在于更加美观，但是就目前来说箱线图的受众比提琴图要多很多，毕竟前者是包含更多统计信息，如图11.76 所示。

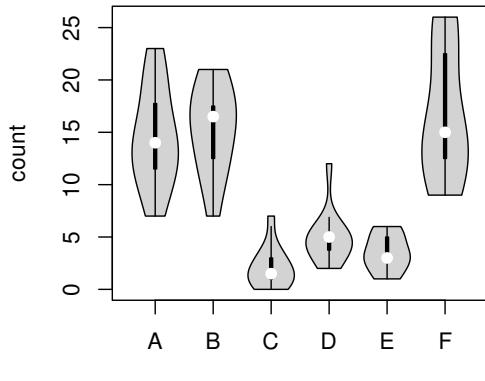
```
boxplot(count ~ spray, data = InsectSprays)
vioplot::vioplot(count ~ spray, data = InsectSprays, col = "lightgray")
ggplot(InsectSprays, aes(x = spray, y = count)) +
```

```
geom_violin(fill = "lightgray") +
  theme_minimal()

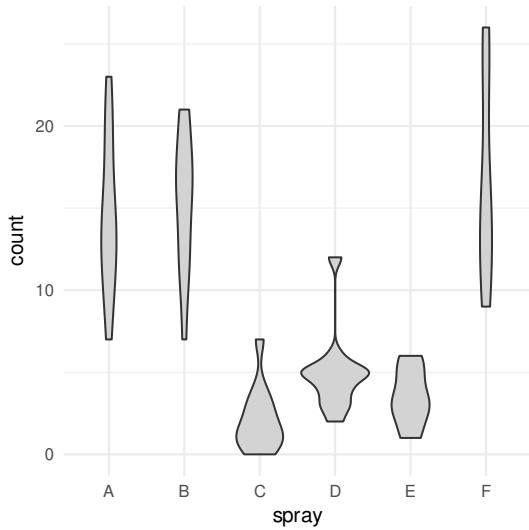
beanplot::beanplot(count ~ spray, data = InsectSprays, col = "lightgray")
```



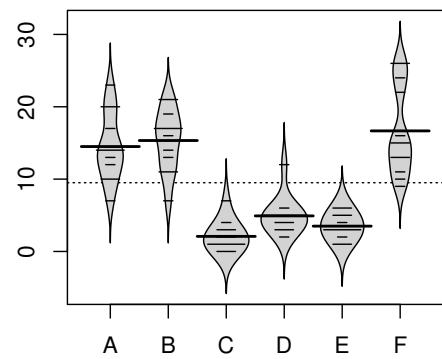
(a) 简单箱线图



(b) vioplot 绘制的提琴图



(c) ggplot2 绘制的提琴图



(d) beanplot 绘制的提琴图

图 11.76: 几种不同的提琴图

`ggnormalviolin` 包在给定均值和标准差的情况下，绘制正态分布的概率密度曲线，如图 11.77 所示。

```
library(ggnormalviolin)
with(
  aggregate(
    data = iris, Sepal.Length ~ Species,
    FUN = function(x) c(dist_mean = mean(x), dist_sd = sd(x))
  ),
  cbind.data.frame(Sepal.Length, Species)
```

```
) %>%  
  ggplot(aes(x = Species, mu = dist_mean, sigma = dist_sd, fill = Species)) +  
  geom_normalviolin() +  
  theme_minimal()
```

C

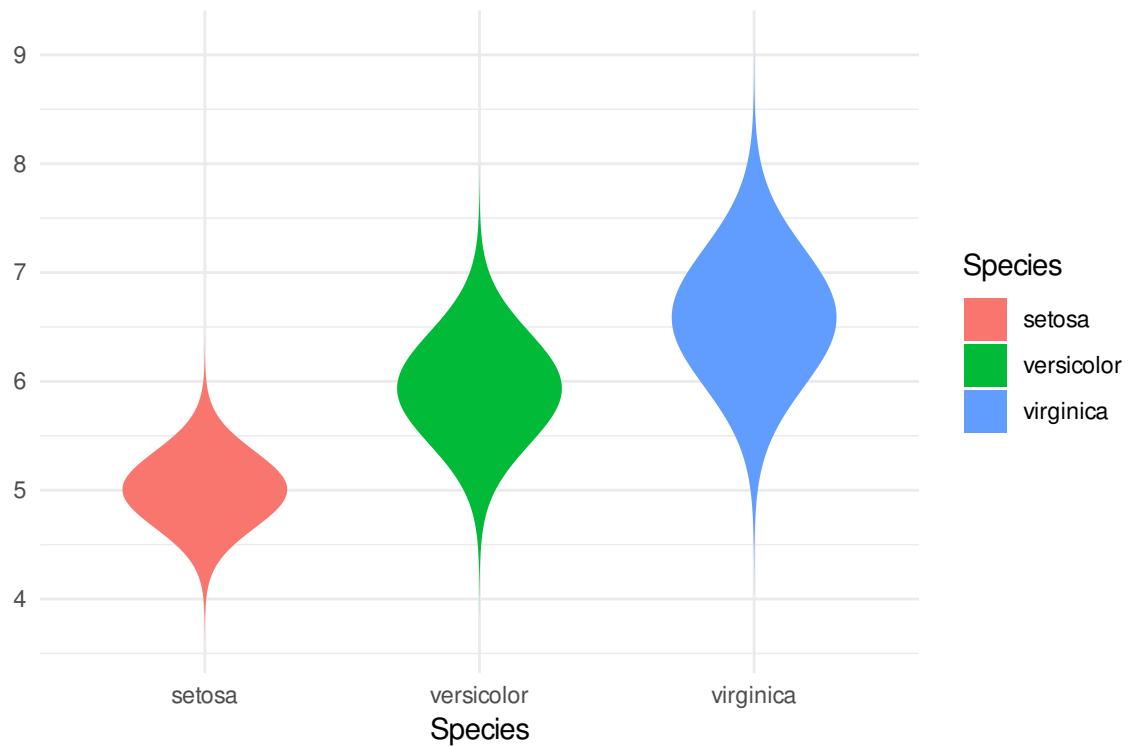
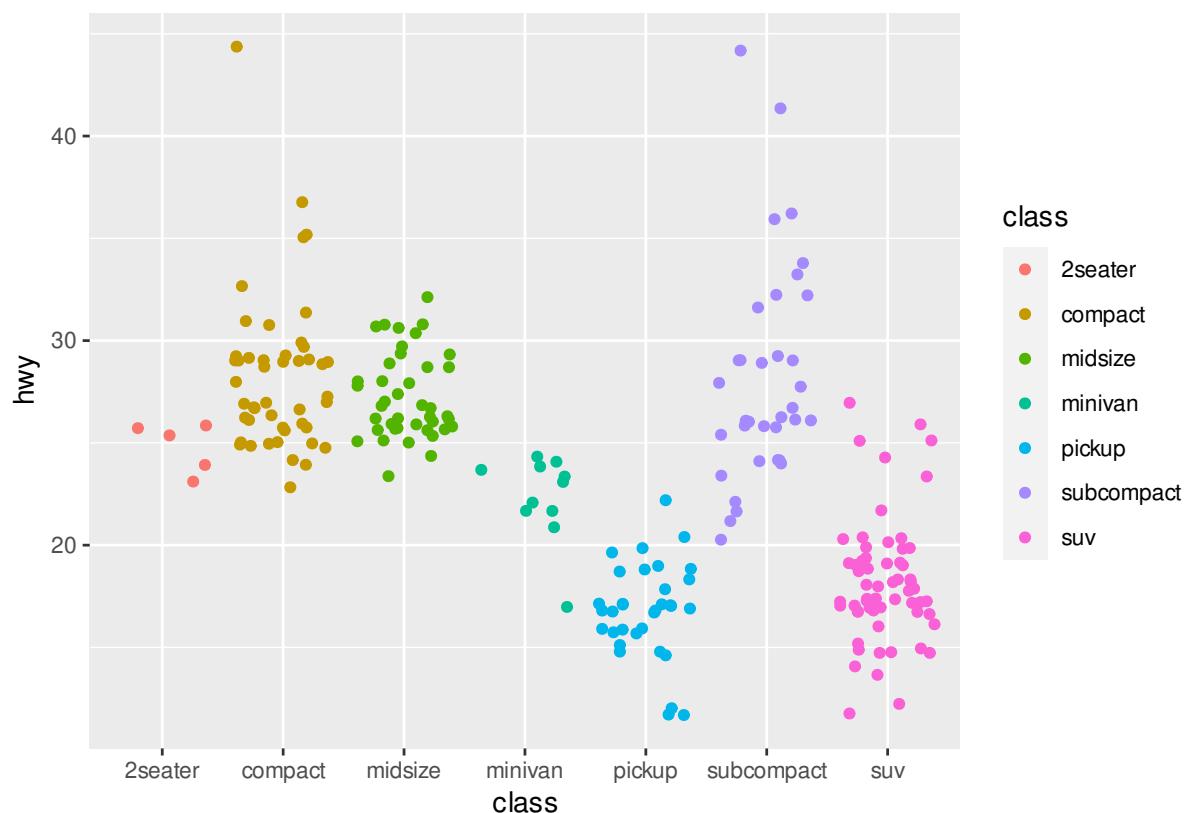


图 11.77: 正态分布的概率密度曲线

#### 11.4.11 抖动图

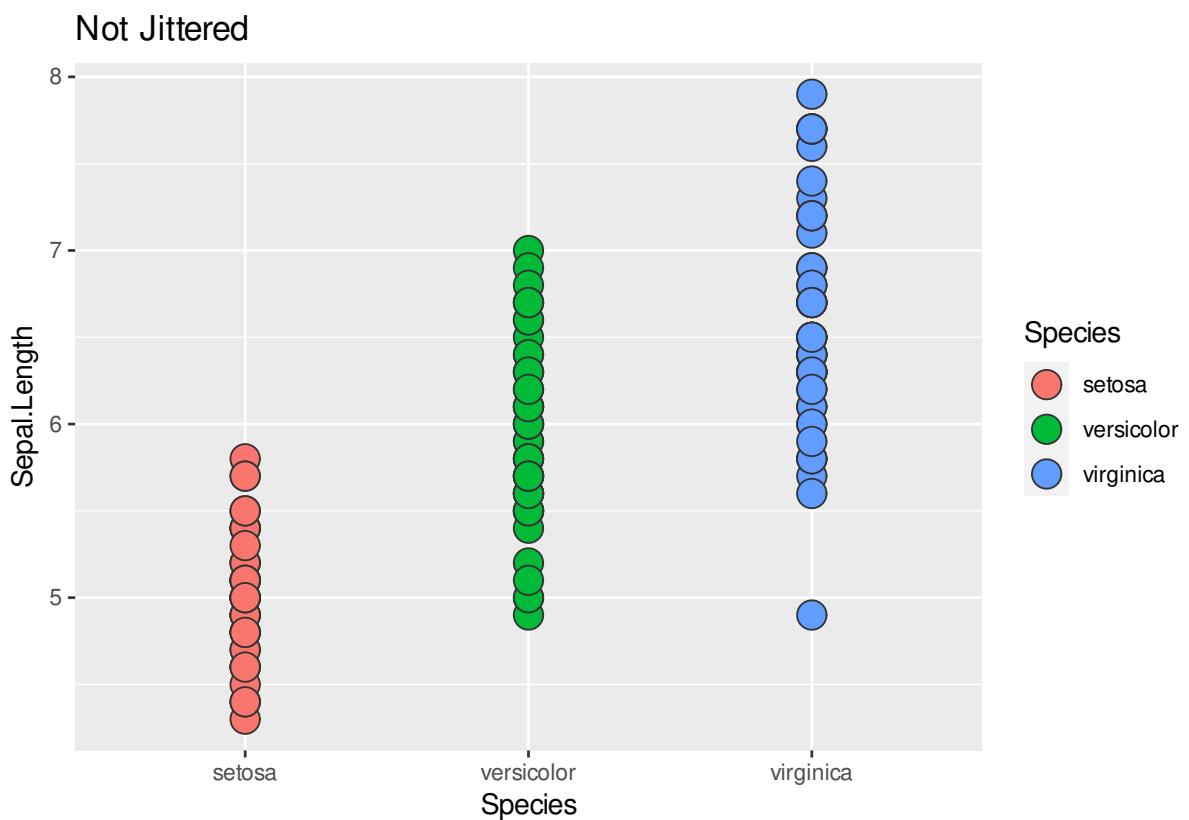
抖动图适合数据量比较小的情况

```
ggplot(mpg, aes(x = class, y = hwy, color = class)) + geom_jitter()
```

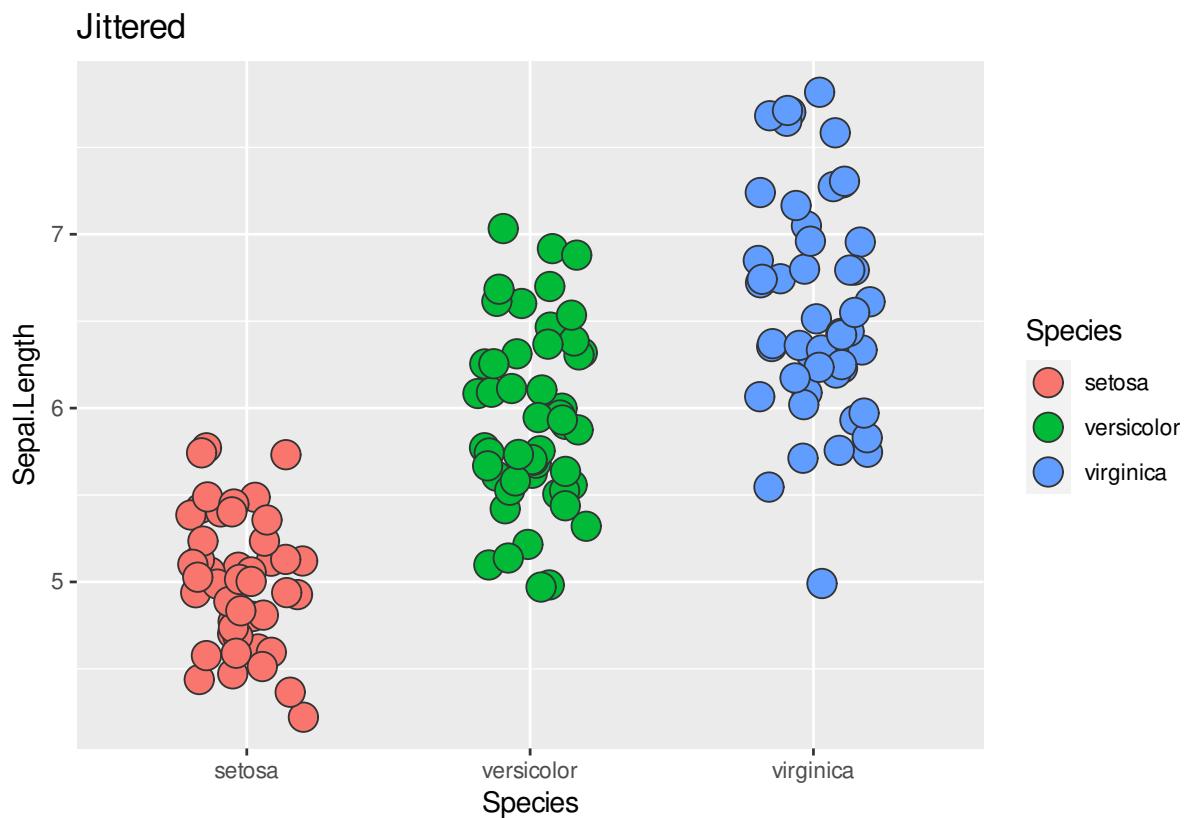


抖不抖，还是抖一下

```
ggplot(iris, aes(x = Species, y = Sepal.Length)) +  
  geom_point(aes(fill = Species), size = 5, shape = 21, colour = "grey20") +  
  # geom_boxplot(outlier.colour = NA, fill = NA, colour = "grey20") +  
  labs(title = "Not Jittered")
```



```
ggplot(iris, aes(x = Species, y = Sepal.Length)) +  
  geom_point(aes(fill = Species),  
             size = 5, shape = 21, colour = "grey20",  
             position = position_jitter(width = 0.2, height = 0.1))  
  ) +  
  # geom_boxplot(outlier.colour = NA, fill = NA, colour = "grey20") +  
  labs(title = "Jittered")
```



在数据量比较大的时候，可以用箱线图、密度图、提琴图

```
ggplot(sub_diamonds, aes(x = cut, y = price)) + geom_jitter()
```

上色和分面都不好使的抖动图，因为区分度变小

```
ggplot(sub_diamonds, aes(x = color, y = price, color = color)) +
  geom_jitter() +
  facet_grid(clarity ~ cut)
```

箱线图此时不宜分的过细

```
ggplot(diamonds, aes(x = color, y = price, color = color)) +
  geom_boxplot() +
  facet_grid(cut ~ clarity)
```

所以这样更好，先按纯净度分面，再对比不同的颜色，钻石价格的差异

```
ggplot(diamonds, aes(x = color, y = price, color = color)) +
  geom_boxplot() +
  facet_grid(~clarity)
```

最好只比较一个维度，不同颜色钻石的价格对比

```
ggplot(diamonds, aes(x = color, y = price, color = color)) +
  geom_boxplot()
```

设置随机数种子，抖动图是可重复的。

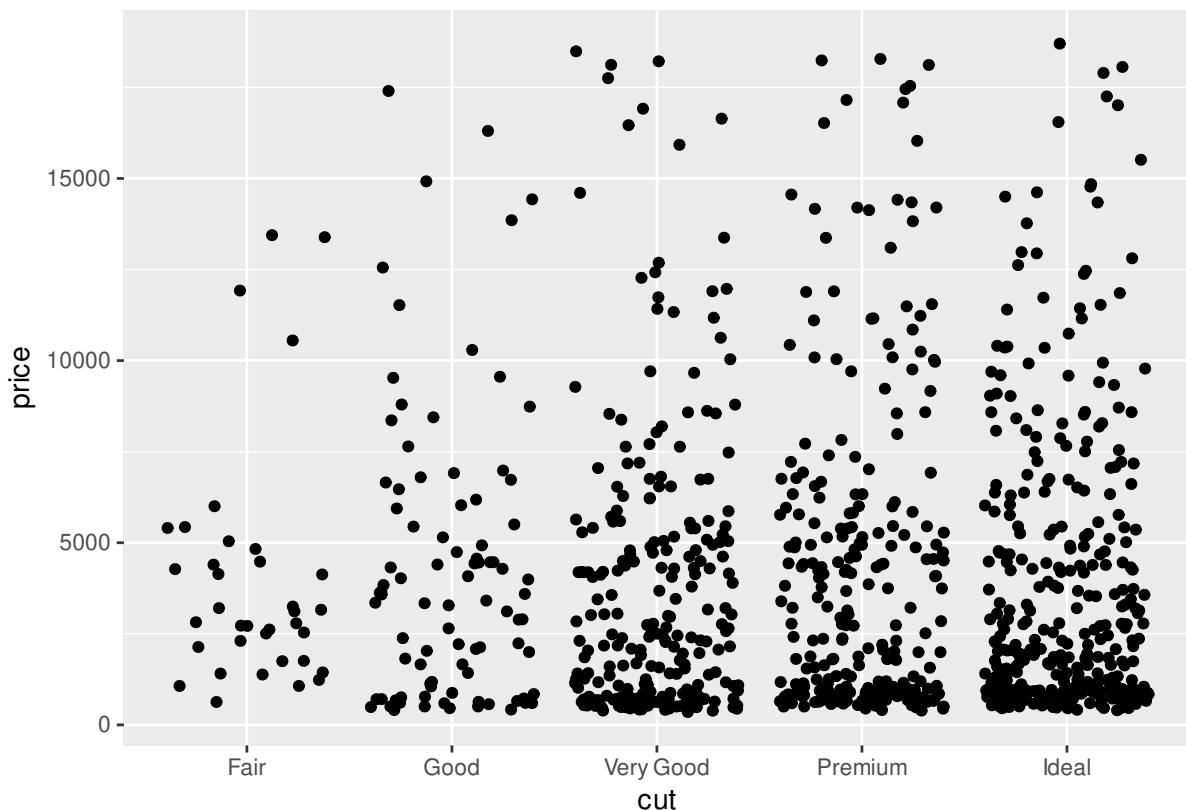


图 11.78: 抖动图的反例

```
ggplot(iris, aes(x = Species, y = Sepal.Width, color = Species)) +  
  geom_boxplot(width = 0.65) +  
  geom_point(position = position_jitter(seed = 37, width = 0.25))
```



图 11.79: 根据钻石颜色上色

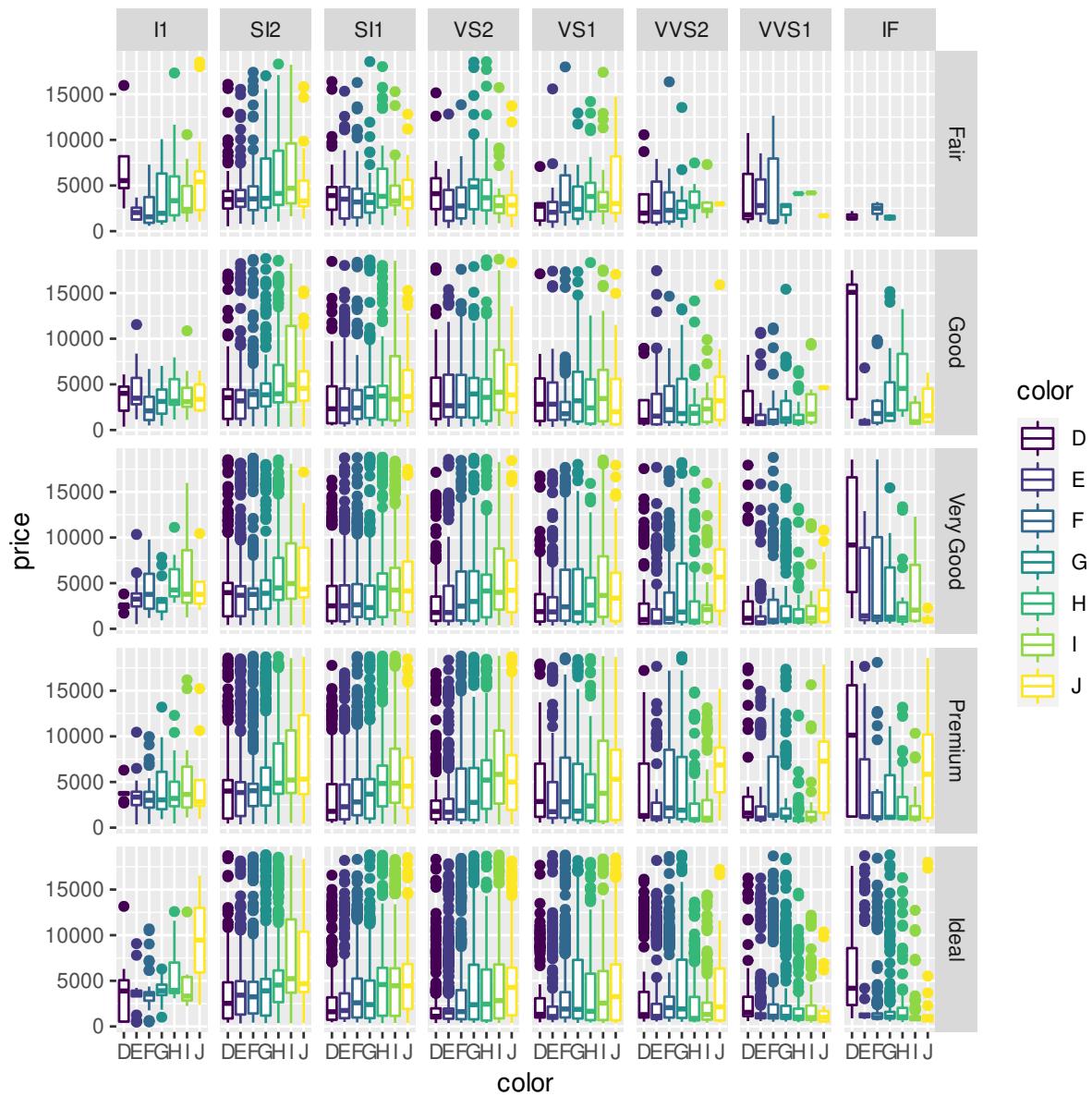


图 11.80: 箱线图

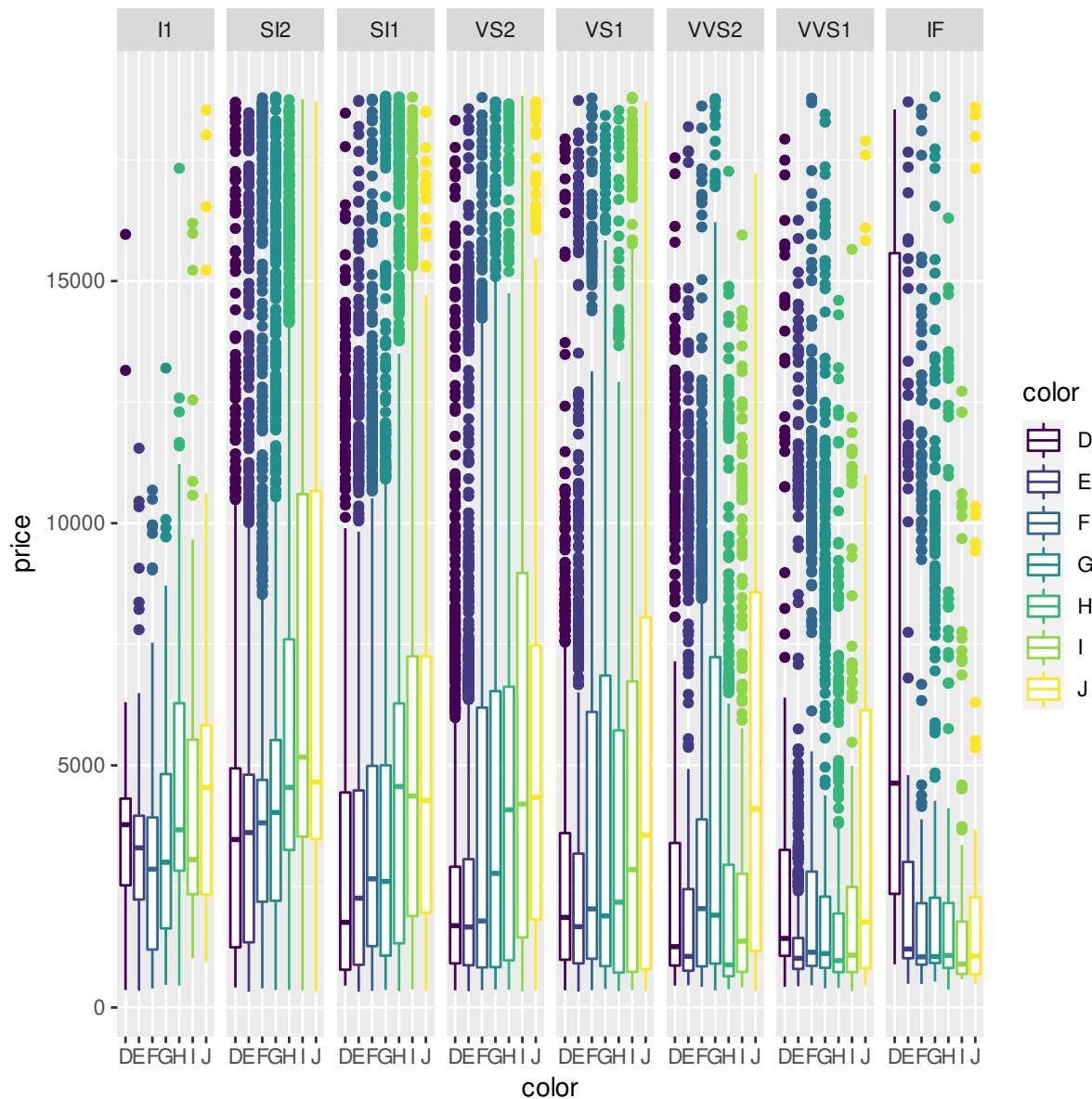


图 11.81: 钻石按纯净度分面

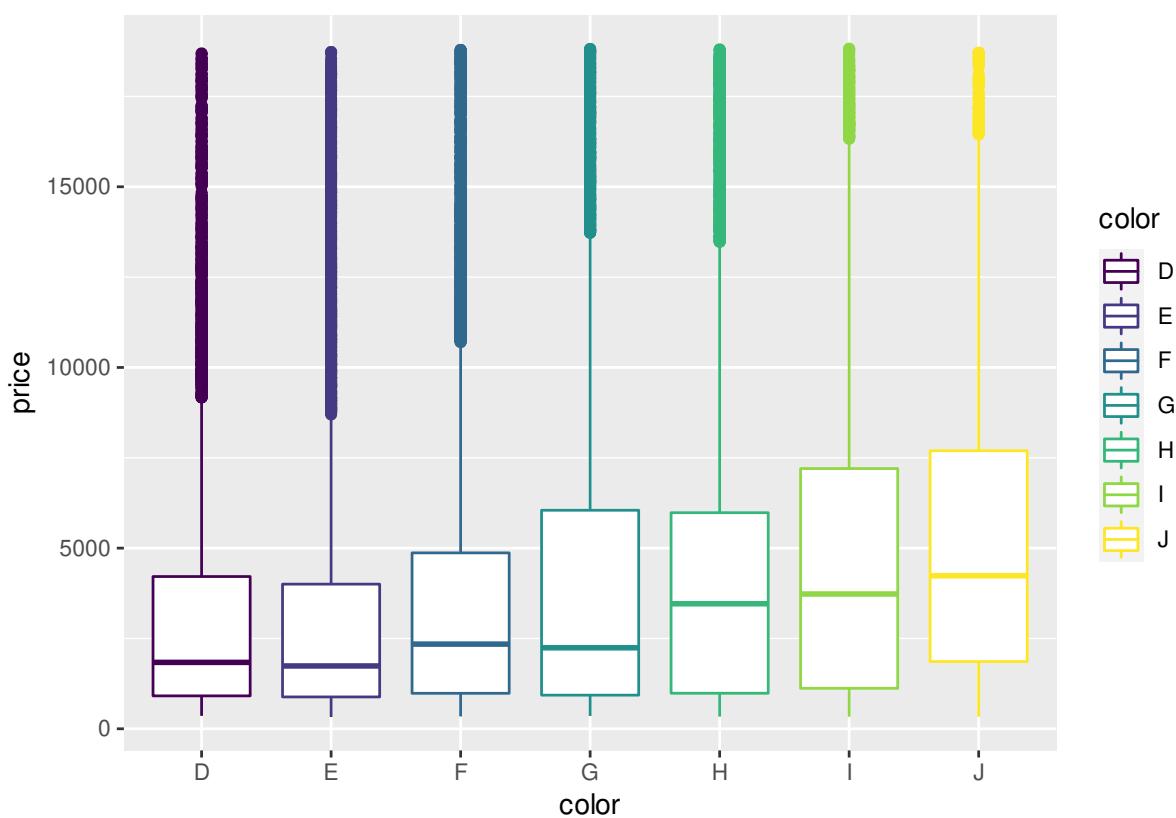
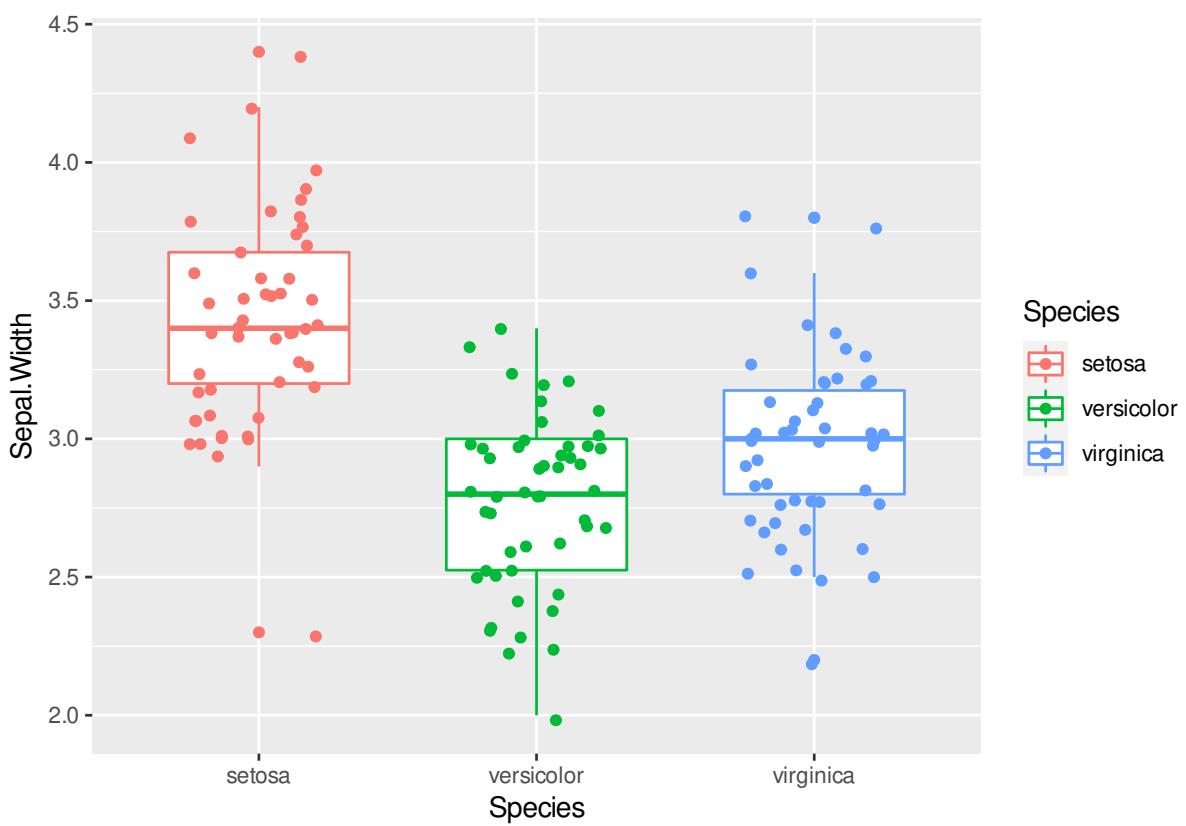


图 11.82: 不同颜色钻石的价格比较



### 11.4.12 蜂群图

在样本点有限的情况下,用蜜蜂图代替普通的抖动图,可视化效果会好很多,如图 11.83 所示。Erik Clarke 开发的 `ggbeeswarm` 包可以将随机抖动的散点图朝着比较规律的方向聚合,又不丢失数据本身的准确性。

```
library(ggbeeswarm)
p1 <- ggplot(iris, aes(Species, Sepal.Length)) +
  geom_jitter() +
  theme_minimal()
p2 <- ggplot(iris, aes(Species, Sepal.Length)) +
  geom_quasirandom() +
  theme_minimal()
p1 + p2
```

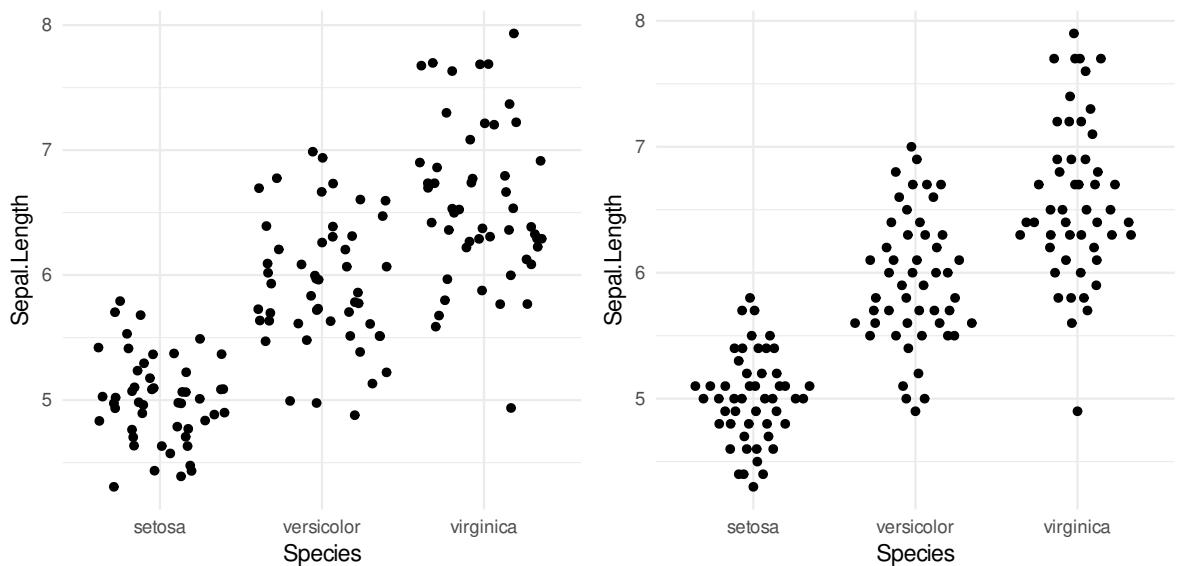


图 11.83: 蜜蜂图可视化效果比抖动图好

### 11.4.13 玫瑰图

南丁格尔风玫瑰图<sup>9</sup>可以作为堆积条形图,分组条形图

```
ggplot(diamonds, aes(x = color, fill = clarity)) +
  geom_bar()

ggplot(diamonds, aes(x = color, fill = clarity)) +
  geom_bar() +
  coord_polar()

# 风玫瑰图 http://blog.csdn.net/Bone\_ACE/article/details/47624987
set.seed(2018)
# 随机生成100次风向, 并汇集到16个区间内
```

<sup>9</sup><https://mbostock.github.io/protovis/ex/crimea-rose-full.html>

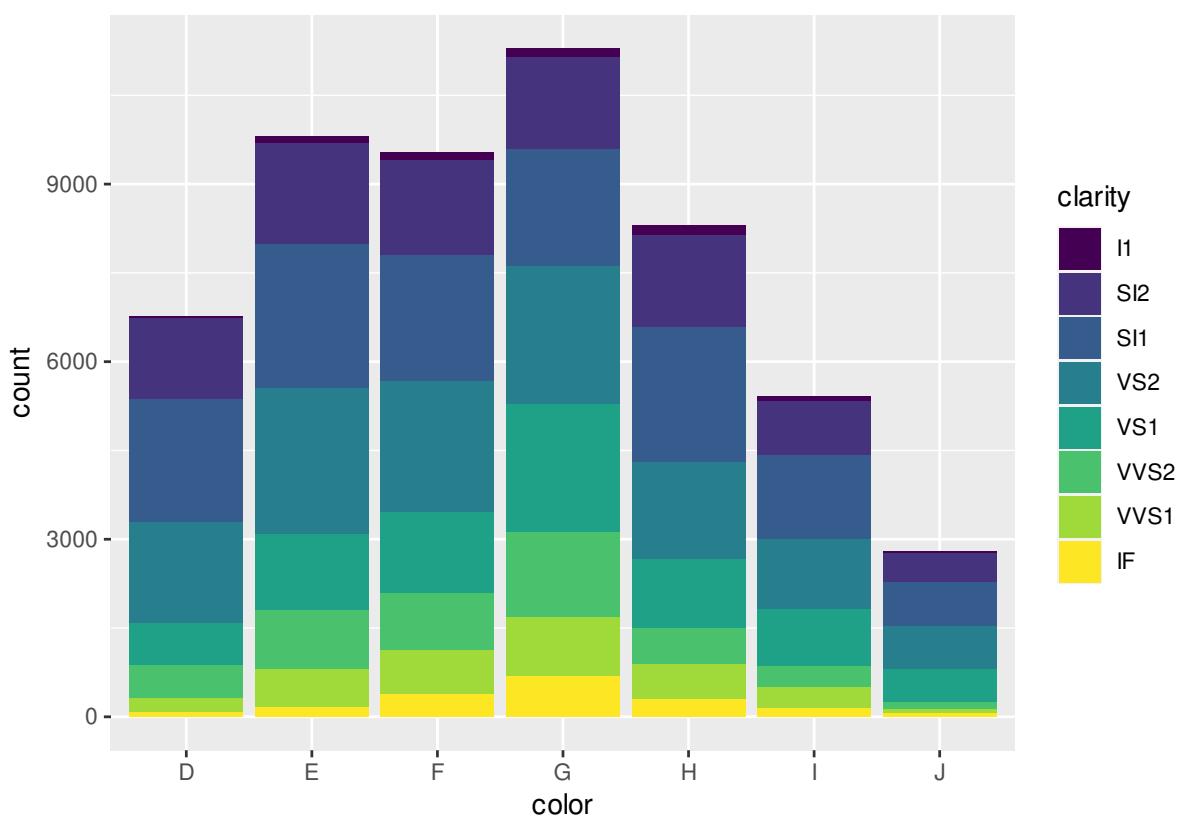


图 11.84: 堆积条形图转风玫瑰图

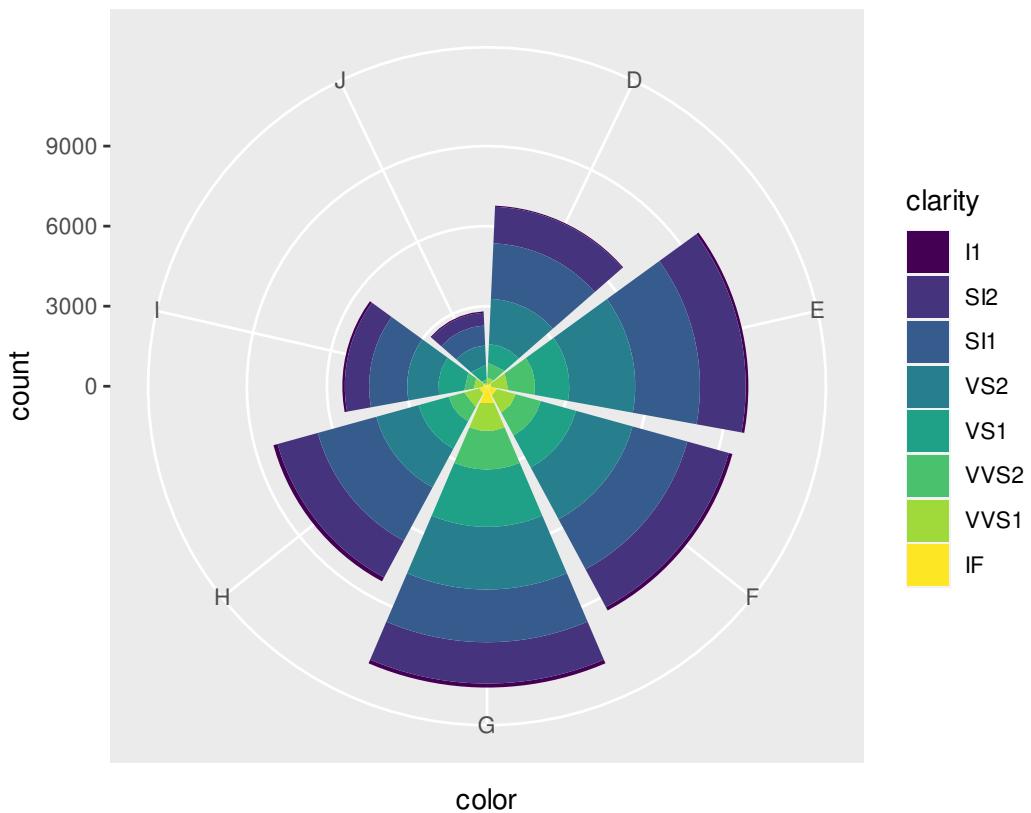


图 11.85: 堆积条形图转风玫瑰图

```
direction <- cut_interval(runif(100, 0, 360), n = 16)
# 随机生成100次风速，并划分成4种强度
mag <- cut_interval(rgamma(100, 15), 4)
dat <- data.frame(direction = direction, mag = mag)
# 将风向映射到X轴，频数映射到Y轴，风速大小映射到填充色，生成条形图后再转为极坐标形式即可
p <- ggplot(dat, aes(x = direction, y = ..count.., fill = mag))
p + geom_bar(colour = "white") +
  coord_polar() +
  theme(axis.ticks = element_blank(), axis.text.y = element_blank()) +
  labs(x = "", y = "", fill = "Magnitude")
```

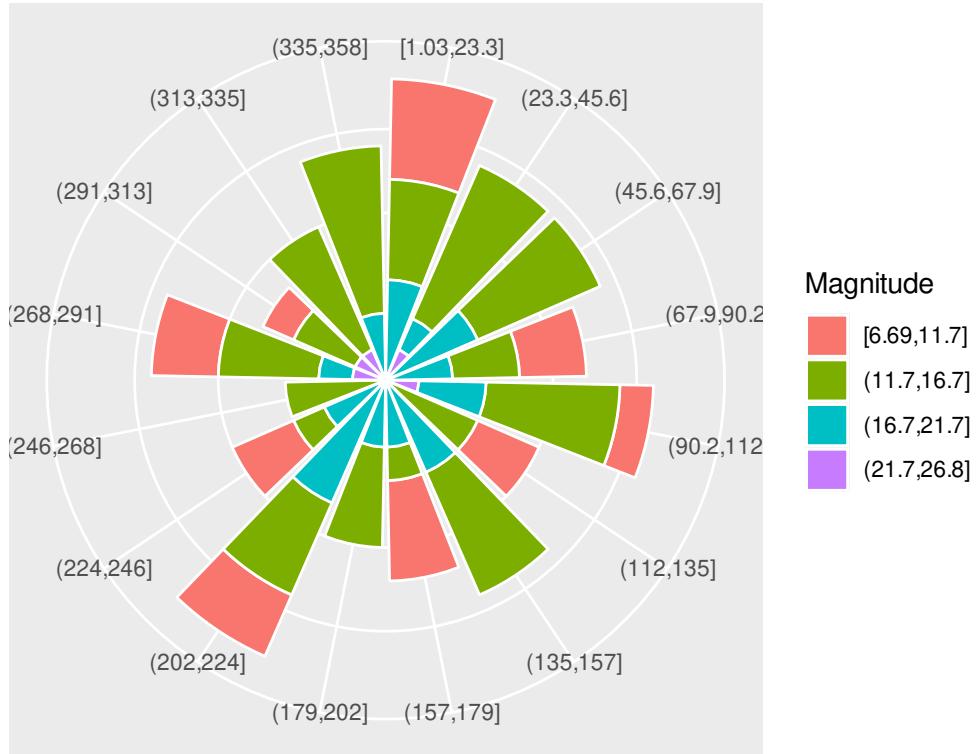
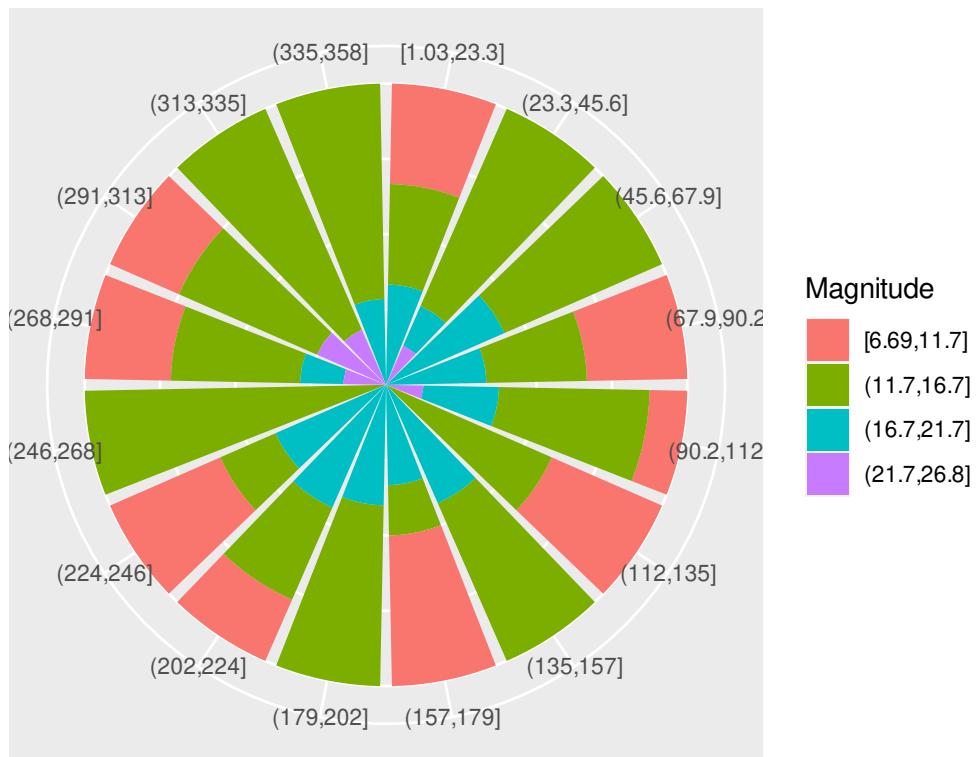


图 11.86: 风玫瑰图

```
p + geom_bar(position = "fill") +
  coord_polar() +
  theme(axis.ticks = element_blank(), axis.text.y = element_blank()) +
  labs(x = "", y = "", fill = "Magnitude")
```



#### 11.4.14 瓦片图

```
p1 <- expand.grid(months = month.abb, years = 1949:1960) %>%
  transform(num = as.vector(AirPassengers)) %>%
  ggplot(aes(x = years, y = months, fill = num)) +
  scale_fill_continuous(type = "viridis") +
  geom_tile(color = "white", size = 0.4) +
  scale_x_continuous(
    expand = c(0.01, 0.01),
    breaks = seq(1949, 1960, by = 1), labels = 1949:1960
  ) +
  theme_minimal(base_size = 10.54, base_family = "Noto Serif SC") +
  theme(legend.position = "top") +
  labs(x = "年", y = "月", fill = "人数")

p2 <- expand.grid(months = month.abb, years = 1949:1960) %>%
  transform(num = as.vector(AirPassengers)) %>%
  ggplot(aes(x = years, y = months, color = num)) +
  geom_point(pch = 15, size = 8) +
  scale_color_distiller(palette = "Spectral") +
  scale_x_continuous(
    expand = c(0.01, 0.01),
    breaks = seq(1949, 1960, by = 1), labels = 1949:1960
```

```

) +
theme_minimal(base_size = 10.54, base_family = "Noto Serif SC") +
theme(legend.position = "top") +
labs(x = "年", y = "月", color = "人数")
p1 + p2

```

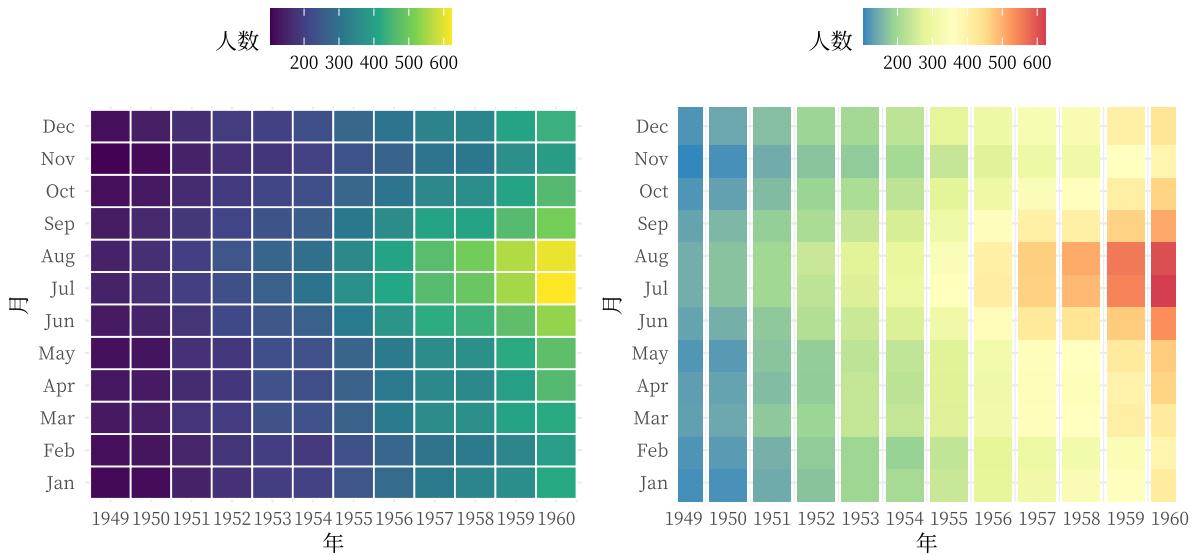


图 11.87: 1949-1960 年国际航线乘客数量的月度趋势

#### 11.4.15 日历图

airquality 数据集记录了 1973 年 5 月至 9 月纽约的空气质量，包括气温（华氏度）、风速（米/小时）、紫外线强度、臭氧含量四个指标，图 11.88 展示了每日的气温变化。

```

airquality %>%
  transform(Date = seq.Date(
    from = as.Date("1973-05-01"),
    to = as.Date("1973-09-30"), by = "day"
  )) %>%
  transform(
    Week = as.integer(format(Date, "%W")),
    Year = as.integer(format(Date, "%Y")),
    Weekdays = factor(weekdays(Date, abbreviate = T),
      levels = c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"))
  )
) %>%
ggplot(aes(x = Week, y = Weekdays, fill = Temp)) +
  scale_fill_distiller(name = "Temp (F)", palette = "Spectral") +
  geom_tile(color = "white", size = 0.4) +
  facet_wrap("Year", ncol = 1) +

```

```
scale_x_continuous(  
  expand = c(0, 0),  
  breaks = seq(1, 52, length = 12),  
  labels = month.abb  
)
```

C

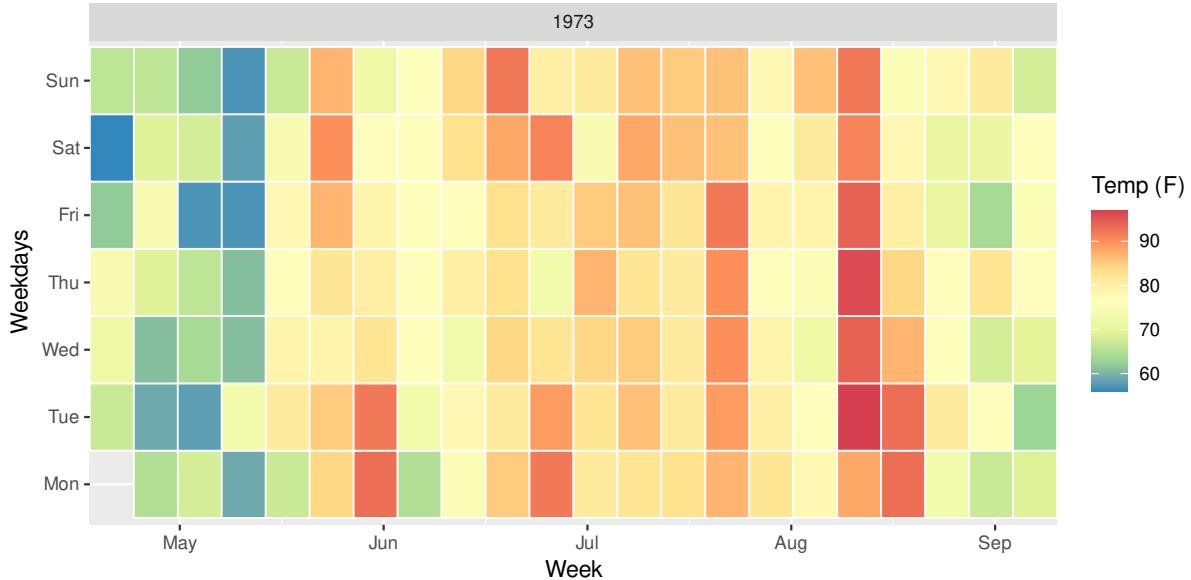


图 11.88: 1973 年 5 月至 9 月纽约的气温变化

## 注意

图 11.88 横轴的刻度标签换成了月份，一个月为四周，一年 52 ~ 53 周，每周的第一天约定为星期一，1973 年 05 月 01 日为星期二。代码中颇为技巧的在于 `format()` 函数从 Date 日期类型的数据提取第几周，用 `weekdays()` 函数提取星期几，而 `month.abb` 则是一个内置常量，12 个月份的英文缩写。在调用其它 R 包处理日期数据时要特别小心，要留意一周的第一天是星期几，有的是星期一，有的是星期日，这往往和宗教信仰相关，星期日在西方也叫礼拜天。上面 Base R 提供的日期函数认为一周的第一天是星期一，而调用 `data.table` 的话，默认一周是从星期日（礼拜天）开始的。

```
# https://d.cosx.org/d/421230  
weekdays(Sys.Date(), abbreviate = TRUE)  
  
## [1] "Thu"  
  
data.table::wday(Sys.Date())  
  
## [1] 5
```

```
library(gert)  
library(ggplot2)  
git_config_set("user.name", "XiangyunHuang")  
git_config_set("user.email", "xiangyunfaith@outlook.com")  
  
dat <- git_log(max = 1000)  
# format(time, "%a") 本地 MacOS 环境 "六" "四" "五" 表示星期  
# Sys.getlocale("LC_TIME") # "zh_CN.UTF-8"
```

```

lvls <- if(!is.na(Sys.getenv("CI", NA))) {
  c("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat")
} else {
  c("日", "一", "二", "三", "四", "五", "六")
}

dat <- transform(dat,
  date = format(time, "%Y-%m-%d"),
  year = format(time, "%Y"),
  month = format(time, "%m"),
  weekday = format(time, "%a"), # factor(format(time, "%a"), levels = lvls),
  week = as.integer(format(time, "%W")))
)

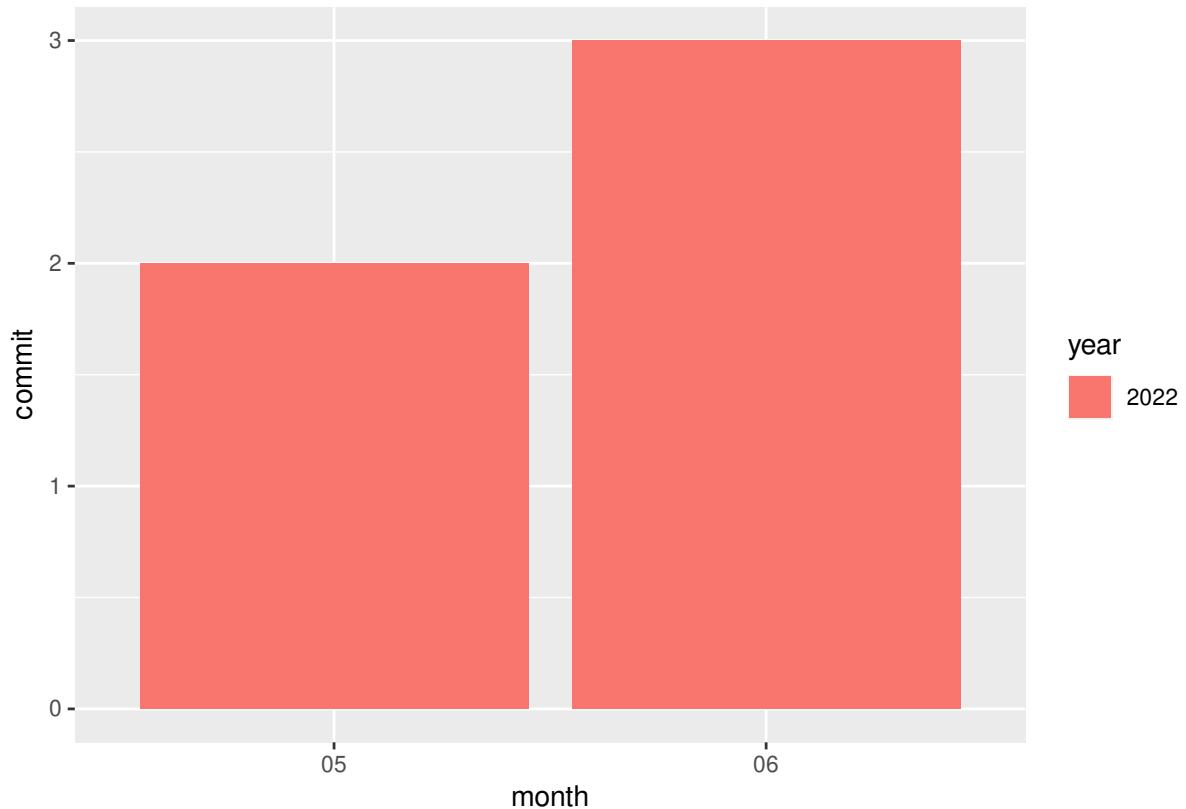
```

本书的活跃情况

```

dat1 <- aggregate(x = commit ~ year + month, data = dat, FUN = length)
# 条形图
ggplot(data = dat1, aes(x = month, y = commit, fill = year)) +
  geom_bar(stat = "identity", position = "identity")

```



```

# 日历图
dat2 <- aggregate(x = commit ~ year + week + weekday, data = dat, FUN = length)

```



```
dat2 <- transform(dat2, colorBin = cut(commit, breaks = c(0, 5, 10, 15, 20, 25)))  
  
ggplot(data = dat2, aes(x = week, y = weekday, fill = colorBin)) +  
  scale_fill_brewer(name = "commit", palette = "Greens") +  
  geom_tile(color = "white", size = 0.4) +  
  facet_wrap("year", ncol = 1) +  
  scale_x_continuous(  
    expand = c(0, 0),  
    breaks = seq(1, 52, length = 12),  
    labels = month.abb  
) +  
  labs(x = "", y = "") +  
  theme_minimal(base_family = "Noto Sans SC")
```

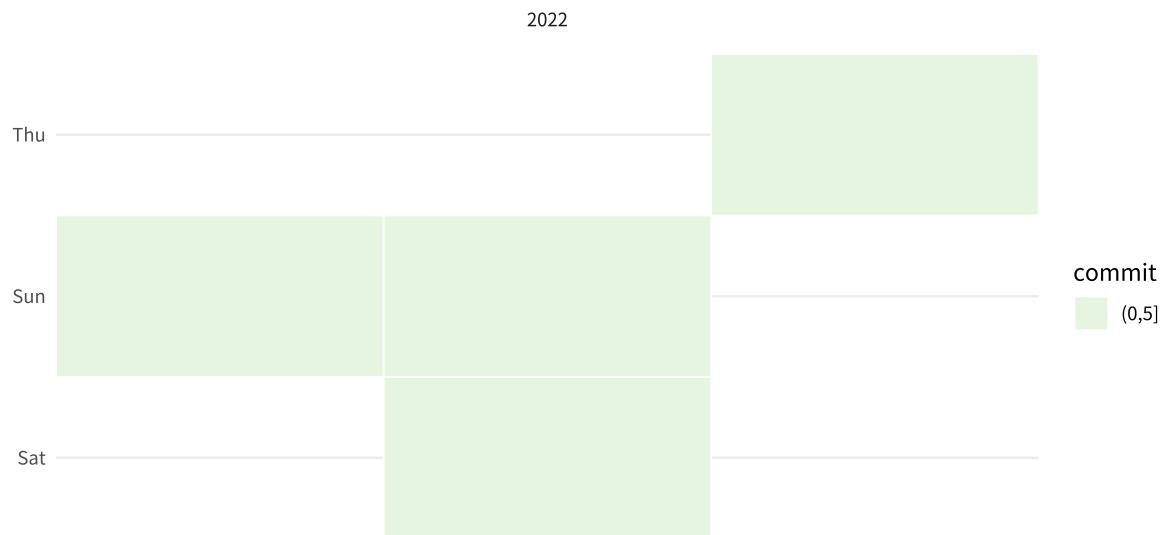


图 11.89: 《R 语言学习笔记》的活跃情况

#### 11.4.16 岭线图

`ggridges` 包, 于淼 对此图形的来龙去脉做了比较系统的阐述, 详见统计之都主站文章[叠嶂图的前世今生](#)

```
library(ggridges)  
ggplot(lincoln_weather, aes(x = `Mean Temperature [F]`, y = Month, fill = stat(x))) +  
  geom_density_ridges_gradient(scale = 3, rel_min_height = 0.01, gradient_lwd = 1.) +  
  scale_x_continuous(expand = c(0, 0)) +  
  scale_y_discrete(expand = expansion(mult = c(0.01, 0.25))) +  
  scale_fill_viridis_c(name = "Temp. [F]", option = "C") +  
  labs(  
    title = 'Temperatures in Lincoln NE',  
    subtitle = 'Mean temperatures (Fahrenheit) by month for 2016'
```

```
) +
  theme_ridges(font_size = 13, grid = TRUE) +
  theme(axis.title.y = element_blank())
```

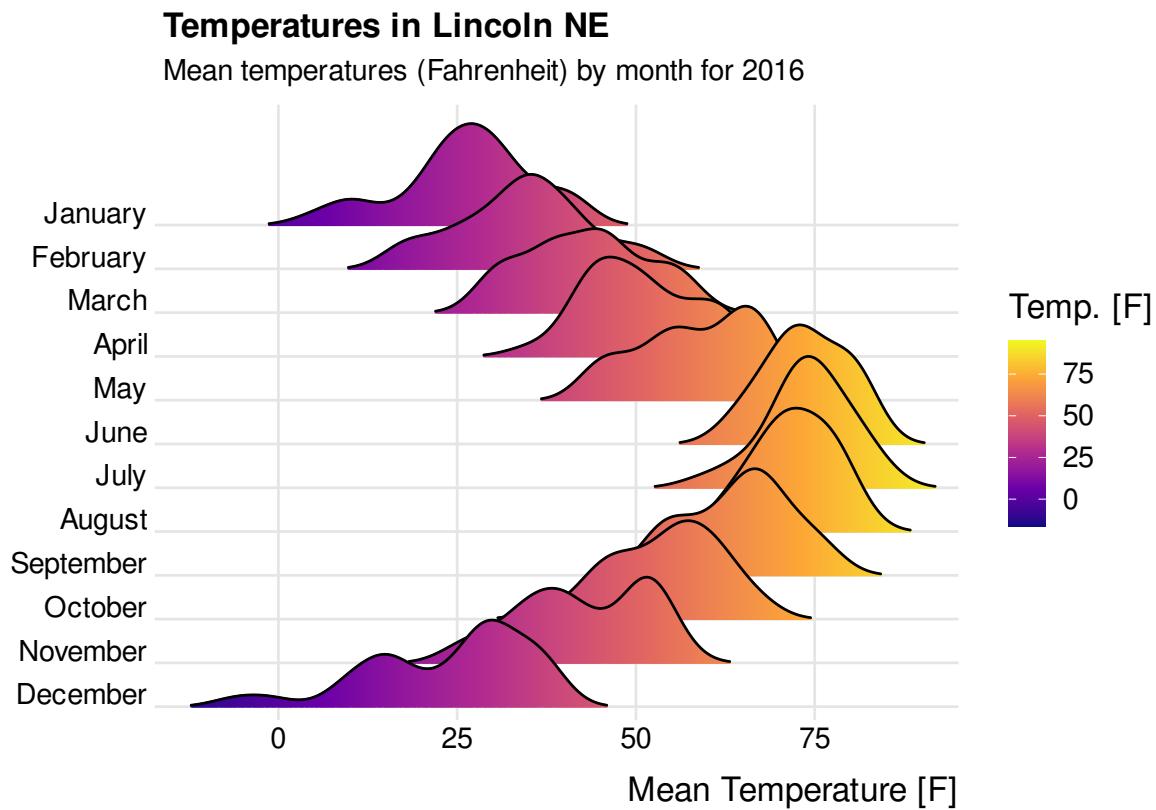


图 11.90: 2016 年在内布拉斯加州林肯市的天气变化

通过数据可视化的手段帮助肉眼检查两组数据的分布

```
p1 <- ggplot(sleep, aes(x = extra, y = group, fill = group)) +
  geom_density_ridges() +
  theme_ridges()

p2 <- ggplot(diamonds, aes(x = price, y = color, fill = color)) +
  geom_density_ridges() +
  theme_ridges()

p1 / p2
```

[ridgeline](#) 提供 Base R 绘图方案

### 11.4.17 椭圆图

type 指定多元分布的类型, type = "t" 和 type = "norm" 分别表示 t 分布和正态分布, geom = "polygon", 以 eruptions > 3 分为两组

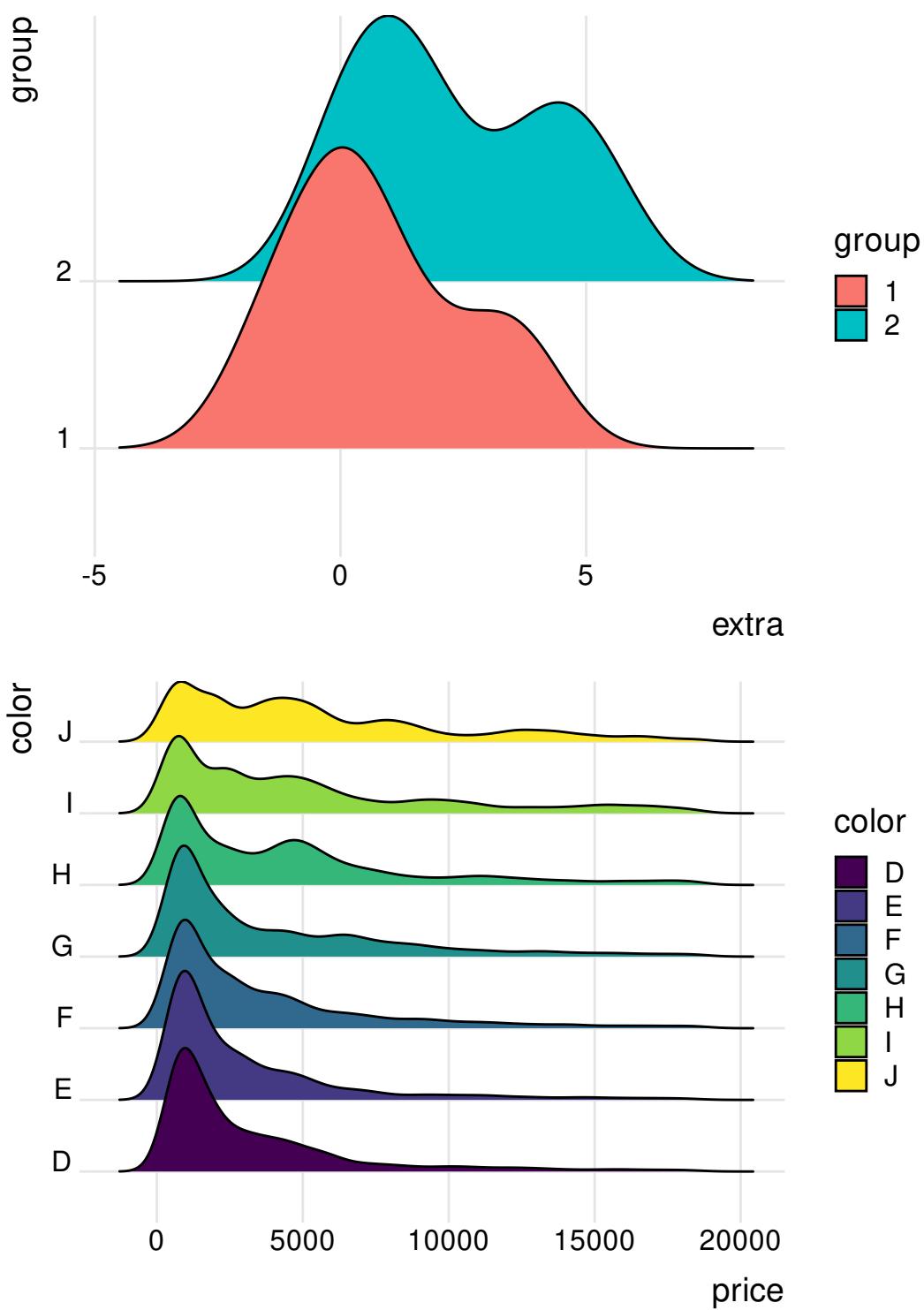


图 11.91: 比较数据的分布

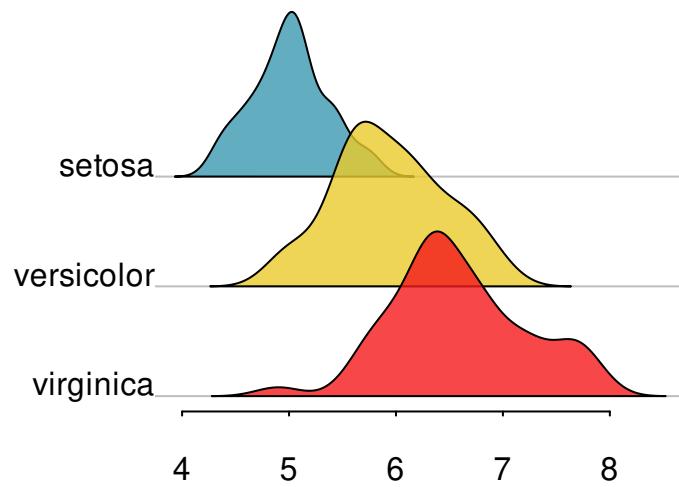


图 11.92: 岭线图

```
ggplot(faithful, aes(x = waiting, y = eruptions)) +  
  geom_point() +  
  stat_ellipse()  
  
ggplot(faithful, aes(waiting, eruptions, color = eruptions > 3)) +  
  geom_point() +  
  stat_ellipse(type = "norm", linetype = 2) +  
  stat_ellipse(type = "t") +  
  theme(legend.position = "none")  
  
ggplot(faithful, aes(waiting, eruptions, fill = eruptions > 3)) +  
  stat_ellipse(geom = "polygon") +  
  theme(legend.position = "none")
```

### 11.4.18 Q-Q 图

quantile-quantile Q-Q 正态分布图的 ggplot2 实现 [qqplotr](#)

### 11.4.19 包络图

ggpubr 包提供了 stat\_chull() 图层

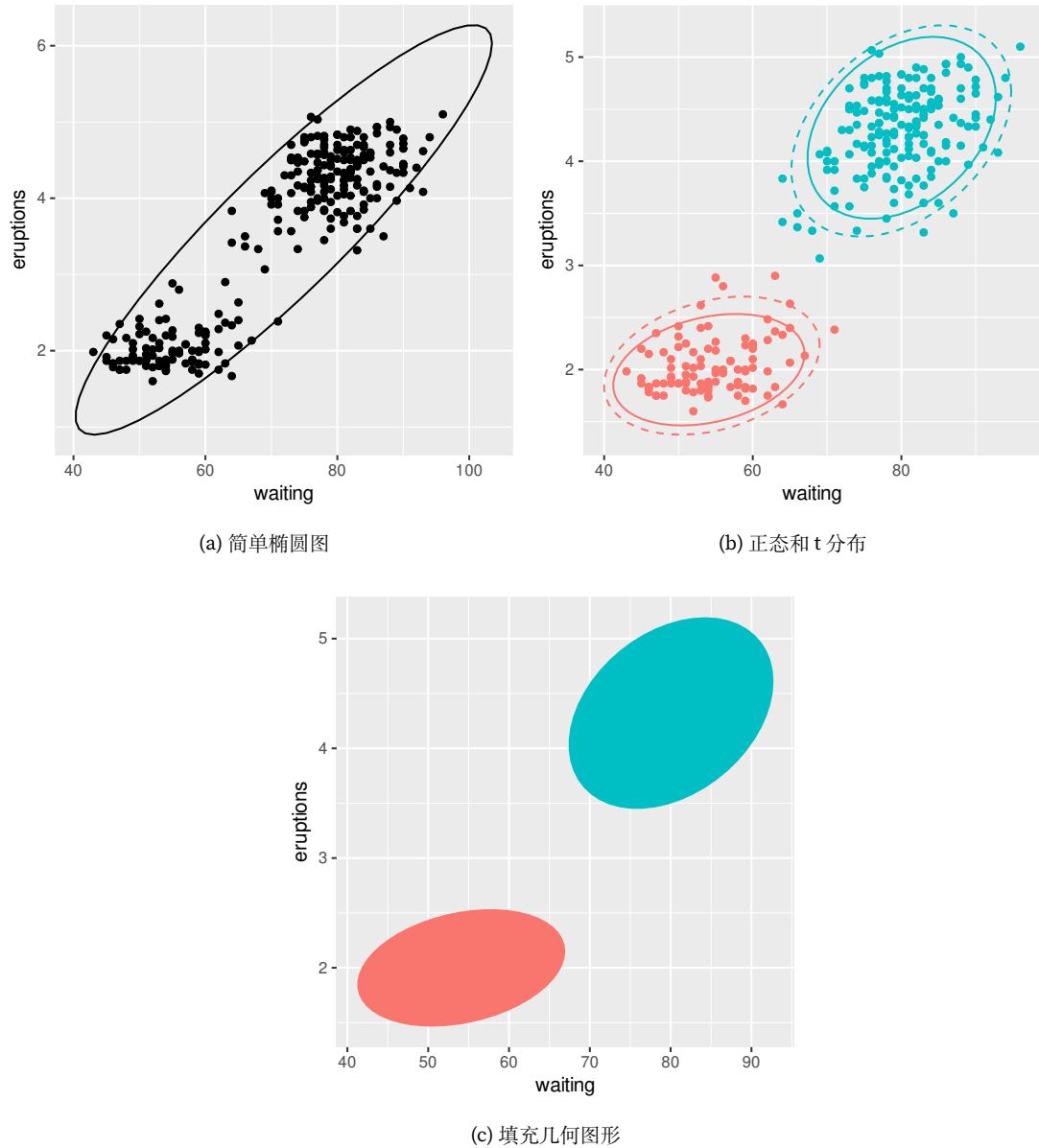


图 11.93: 几种不同的椭圆图

```
library(ggpubr)
ggscatter(mpg, x = "displ", y = "hwy", color = "drv")+
  stat_chull(aes(color = drv, fill = drv), alpha = 0.1, geom = "polygon")
```

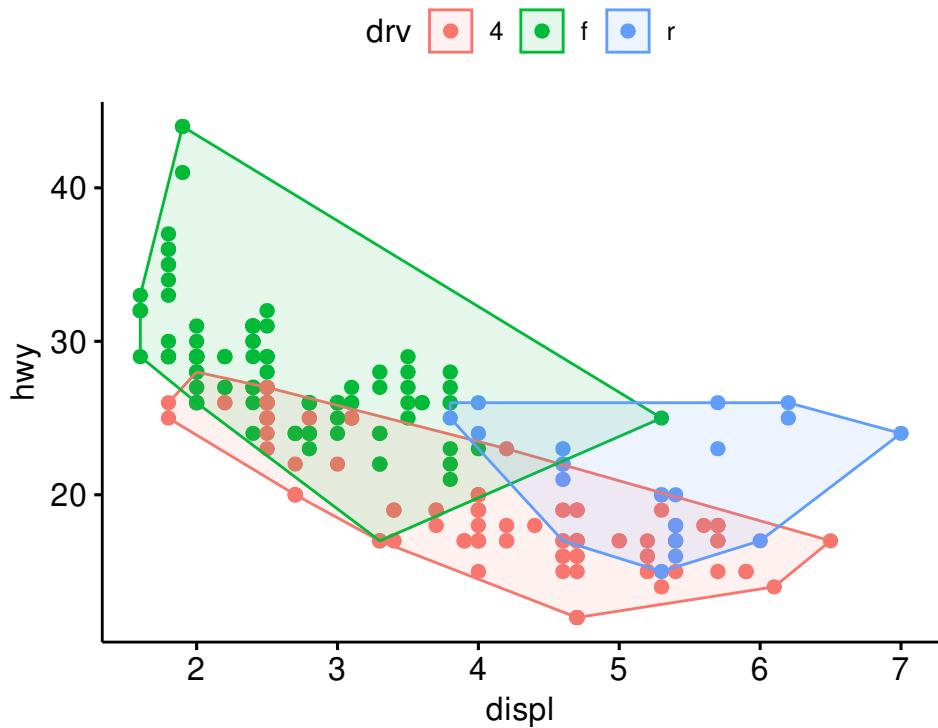


图 11.94: 包络图

其背后的原理如下

```
stat_chull

## function (mapping = NULL, data = NULL, geom = "path", position = "identity",
##           na.rm = FALSE, show.legend = NA, inherit.aes = TRUE, ...)
## {
##   layer(stat = StatChull, data = data, mapping = mapping, geom = geom,
##         position = position, show.legend = show.legend, inherit.aes = inherit.aes,
##         params = list(na.rm = na.rm, ...))
## }
## <bytecode: 0x55e1c14386c8>
## <environment: namespace:ggpubr>

StatChull <- ggproto("StatChull", Stat,
  compute_group = function(data, scales) {
    data[chull(data$x, data$y), , drop = FALSE]
  },
  required_aes = c("x", "y")
)

stat_chull <- function(mapping = NULL, data = NULL, geom = "polygon",
```



```
position = "identity", na.rm = FALSE, show.legend = NA,
inherit.aes = TRUE, ...)

layer(
  stat = StatChull, data = data, mapping = mapping, geom = geom,
  position = position, show.legend = show.legend, inherit.aes = inherit.aes,
  params = list(na.rm = na.rm, ...))
)

ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  stat_chull(fill = NA, colour = "black")

ggplot(mpg, aes(displ, hwy, colour = drv)) +
  geom_point() +
  stat_chull(fill = NA)
```

#### 11.4.20 拟合图

```
xx <- -9:9
yy <- sqrt(abs(xx))
plot(xx, yy,
  col = "red",
  xlab = expression(x),
  ylab = expression(sqrt(abs(x))))
)
lines(spline(xx, yy, n = 101, method = "fmm", ties = mean), col = "pink")

myspline <- function(formula, data, ...) {
  dat <- model.frame(formula, data)
  res <- splinefun(dat[[2]], dat[[1]])
  class(res) <- "myspline"
  res
}

predict.myspline <- function(object, newdata, ...) {
  object(newdata[[1]])
}

data.frame(x = -9:9) %>%
  transform(y = sqrt(abs(x))) %>%
  ggplot(aes(x = x, y = y)) +
  geom_point(color = "red", pch = 1, size = 2) +
  stat_smooth(method = myspline, formula = y~x, se = F, color = "pink") +
```

```
labs(x = expression(x), y = expression(sqrt(abs(x)))) +
  theme_minimal()
```

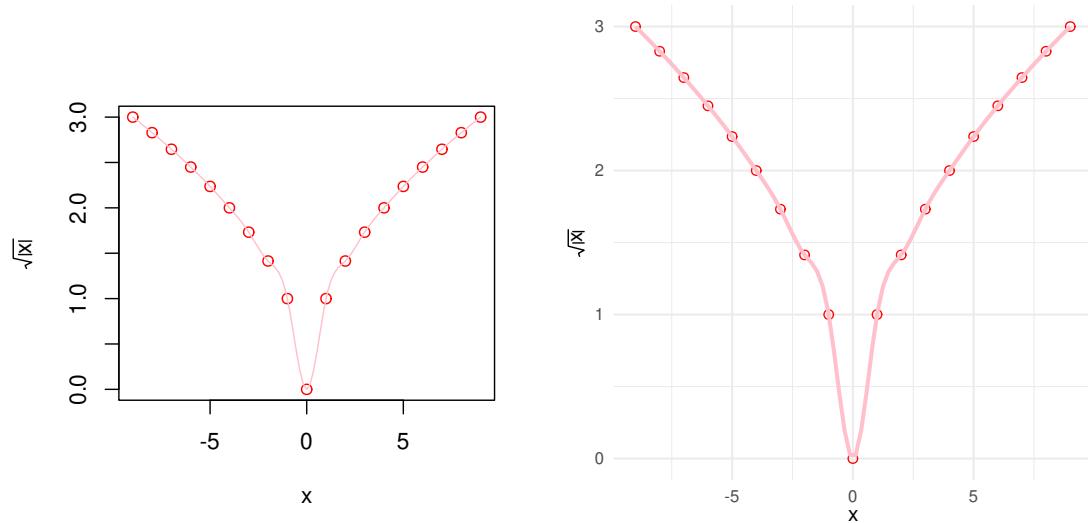


图 11.95: 自定义样条函数

下面以真实数据集 `trees` 为例, 介绍 `geom_smooth()` 支持的拟合方法, 比如 "`lm`" 线性回归和 "`nls`" 非线性回归

```
ggplot(trees, aes(x = log(Girth), y = log(Volume))) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE)

ggplot(trees, aes(x = Girth, y = Volume)) +
  geom_point() +
  geom_smooth(
    method = "nls", formula = y ~ a * x^2 + b, se = F,
    method.args = list(start = list(a = 5, b = -36))
  )
```

### 11.4.21 地形图

区域之间以轮廓分割, 轮廓之间以相同的颜色填充, Cleveland 把这个叫做 level plot, `lattice` 包中 `levelplot()` 函数正来源于此。

[Auckland's Maunga Whau Volcano](#) 是火山喷发后留下的渣堆, 位于新西兰奥克兰伊甸山郊区。Ross Ihaka 收集了它的地形数据, 命名为 `volcano`, 打包在 R 软件环境中, 见图 11.97

```
filled.contour(volcano,
  color.palette = terrain.colors,
  plot.title = title(
    main = "The Topography of Maunga Whau",
    xlab = "Meters North", ylab = "Meters West"
),
```

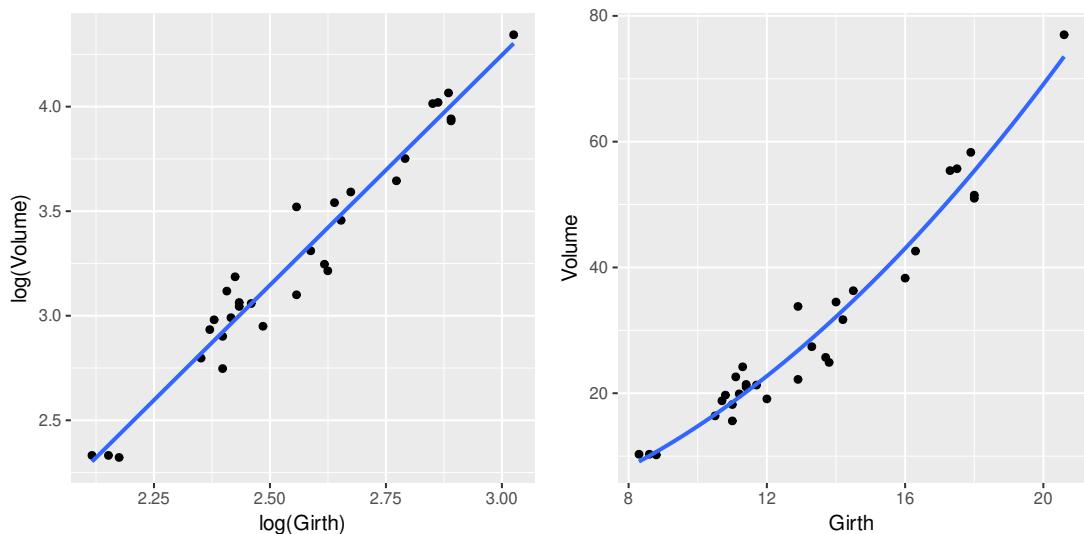


图 11.96: 平滑方法

```
plot.axes = {
  axis(1, seq(100, 800, by = 100))
  axis(2, seq(100, 600, by = 100))
},
key.title = title(main = "Height\n(meters)"),
key.axes = axis(4, seq(90, 190, by = 10))
)
```

## 11.4.22 树状图

数据集 GNI2014 来自 **treemap** 包，是一个 `data.frame` 类型的数据对象，记录了 2014 年每个国家的人口总数 `population` 和国民人均收入 `GNI`，数据样例见下方：

```
library(treemap)
data(GNI2014, package = "treemap")
subset(GNI2014, subset = grepl(x = country, pattern = 'China'))
```

```
##   iso3           country continent population    GNI
## 7  MAC      Macao SAR, China     Asia  559846 76270
## 33 HKG Hong Kong SAR, China     Asia 7061200 40320
## 87 CHN           China     Asia 1338612970  7400
```

数据呈现明显的层级结构，从大洲到国家记录人口数量和人均收入，矩阵树图以方块大小表示人口数量，以颜色深浅表示人均收入，见图11.98

```
treemap(GNI2014,
  index = c("continent", "iso3"),
  vSize = "population",
  vColor = "GNI",
  type = "value",
```

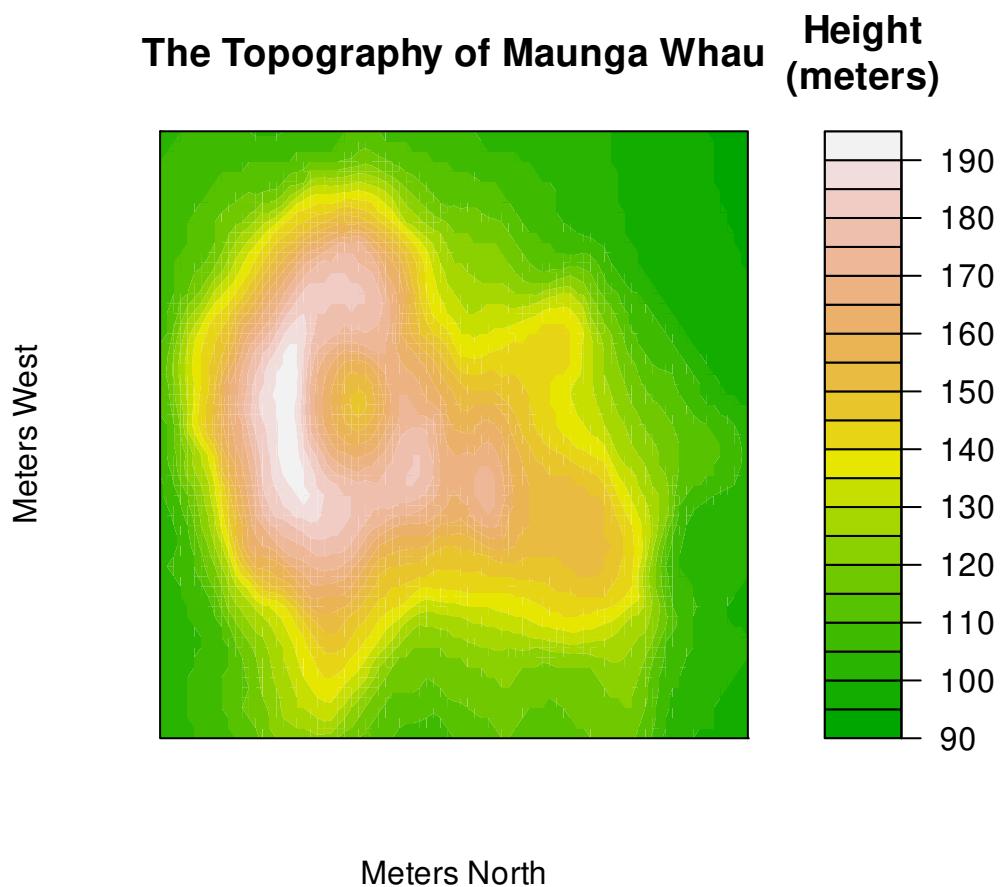


图 11.97: image 图形

```
format.legend = list(scientific = FALSE, big.mark = " ")
```

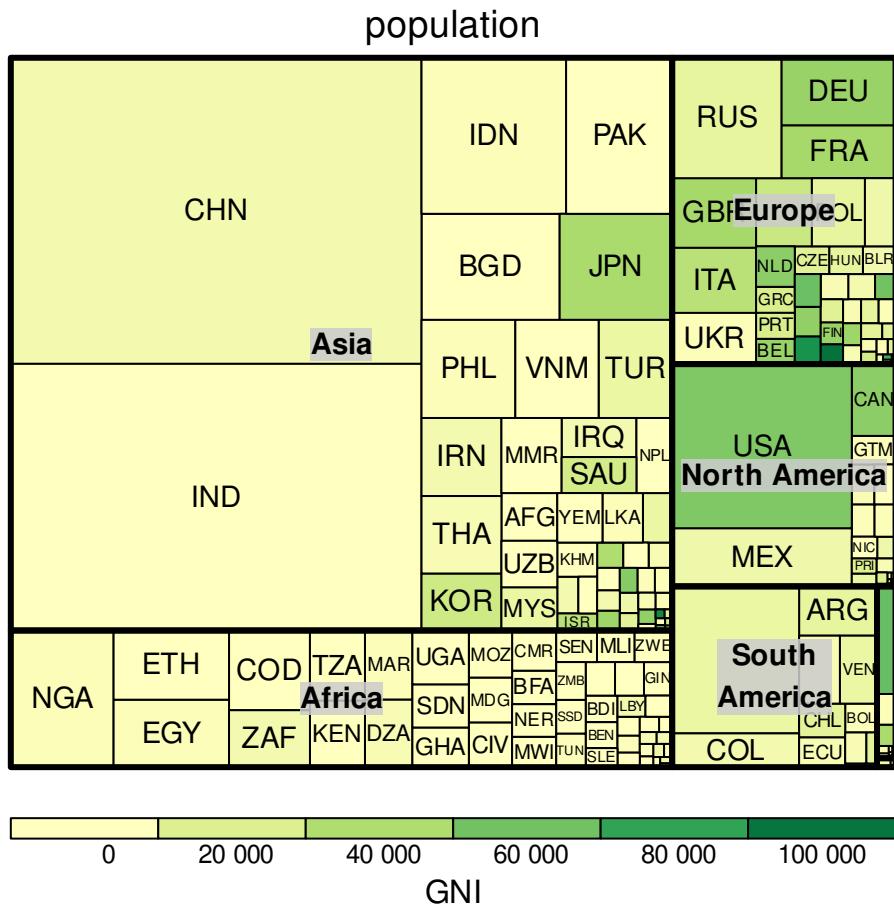


图 11.98: 矩阵树图

**treemapify** 包基于 **ggplot2** 制作树状图, 类似地, 该 R 包内置了数据集 **G20**, 记录了世界主要经济体 **G20** (<https://en.wikipedia.org/wiki/G20>) 的经济和人口信息, 国家 GDP (单位: 百万美元) **gdp\_mil\_usd** 和人类发展指数 **hdi**。相比于 **GNI2014**, 它还包含了两列标签信息: 经济发展阶段和所处的半球。图 @**(fig:treemap-  
ggplot2)** 以南北半球 **hemisphere** 分面, 以色彩填充区域 **region**, 以 **gdp\_mil\_usd** 表示区域大小

```
library(treemapify)
ggplot(G20, aes(
  area = gdp_mil_usd, fill = region,
  label = country, subgroup = region
)) +
  geom_treemap() +
  geom_treemap_text(grow = T, reflow = T, colour = "black") +
  facet_wrap(~hemisphere) +
  scale_fill_brewer(palette = "Set1") +
  theme(legend.position = "bottom") +
  labs(
    title = "The G-20 major economies by hemisphere",
```

```
caption = "The area of each tile represents the country's GDP as a
proportion of all countries in that hemisphere",
fill = "Region"
)
```

The G-20 major economies by hemisphere

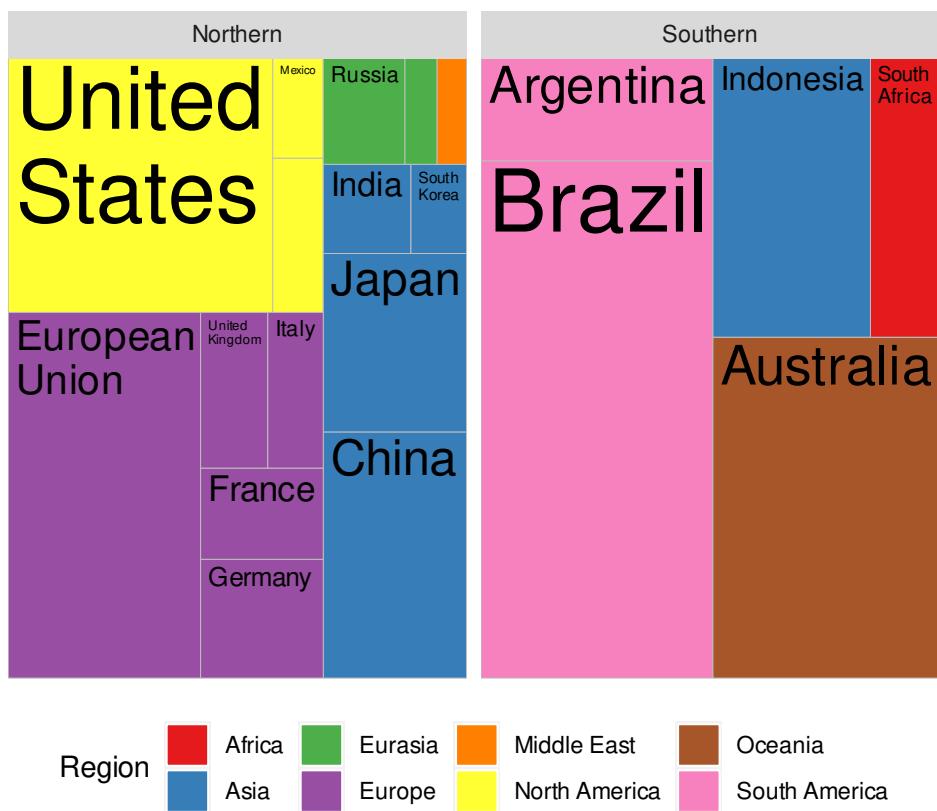
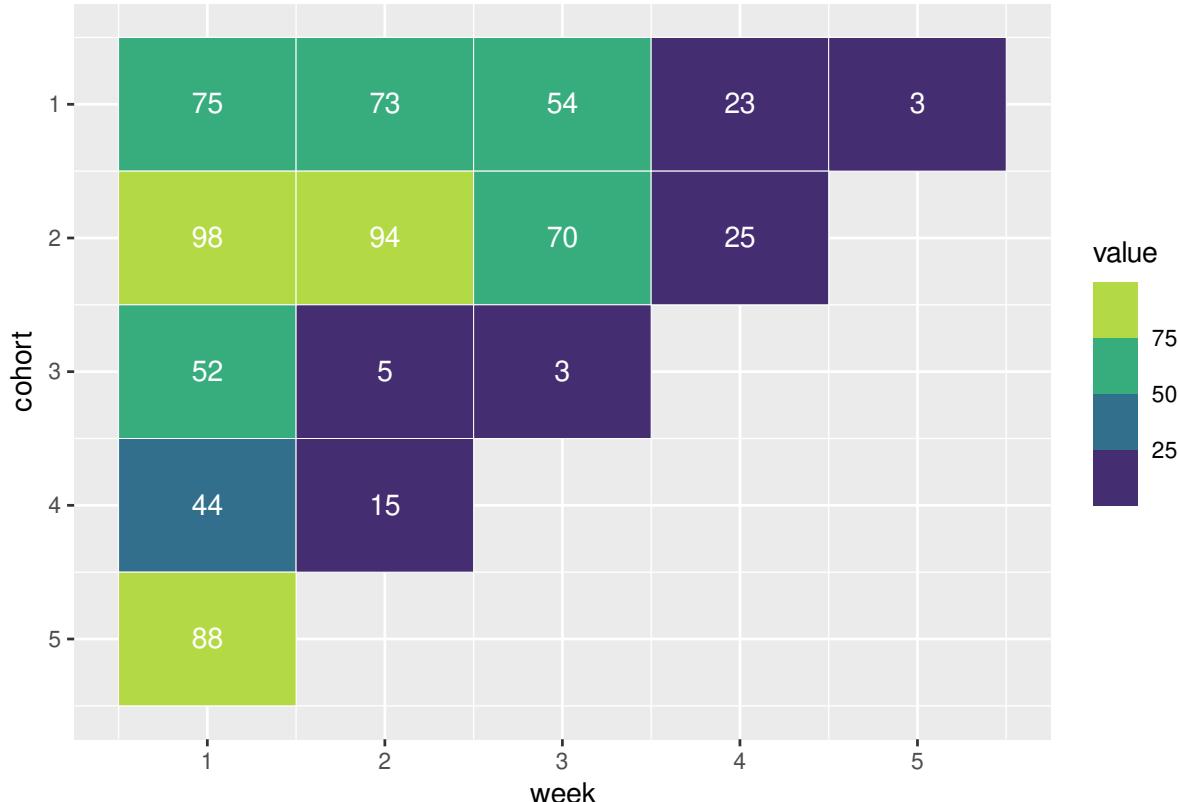


图 11.99: 世界主要经济体 G20 的人口和经济信息

#### 11.4.23 留存图

```
cohort <- data.frame(
  cohort = rep(1:5, times = 5:1),
  week = c(1:5, 1:4, 1:3, 1:2, 1),
  value = c(
    75, 73, 54, 23, 3,
    98, 94, 70, 25,
    52, 5, 3,
    44, 15,
    88
  )
)
```

```
ggplot(cohort, aes(x = week, y = cohort, fill = value)) +  
  geom_tile(color = "white") +  
  geom_text(aes(label = value), color = "white") +  
  scale_y_reverse() +  
  scale_fill_binned(type = "viridis")
```



留存是 [Cohort 分析](#) 中的一种情况，还有转化等，首先定义你的问题，确定度量问题的指标，确定和问题相关的 Cohort（比如时间、空间和用户属性等关键的影响因素），然后数据处理、可视化获得 Cohort 分析结果，最后在实际决策和行动中检验分析结论。

#### 11.4.24 瀑布图

瀑布图 waterfall 与上月相比，谁增谁减，用瀑布图分别表示占比和绝对数值。[瀑布图 waterfall](#)

```
balance <- data.frame(  
  event = c(  
    "Starting\nCash", "Sales", "Refunds",  
    "Payouts", "Court\nLosses", "Court\nWins", "Contracts", "End\nCash"  
  change = c(2000, 3400, -1100, -100, -6600, 3800, 1400, -2800)  
)  
  
balance$balance <- cumsum(c(0, balance$change[-nrow(balance)])) # 累计值
```

```
balance$time <- 1:nrow(balance)
balance$flow <- factor(sign(balance$change)) # 变化为正还是为负

ggplot(balance) +
  geom_hline(yintercept = 0, colour = "white", size = 2) +
  geom_rect(aes(
    xmin = time - 0.45, xmax = time + 0.45,
    ymin = balance, ymax = balance + change, fill = flow
  )) +
  geom_text(aes(
    x = time,
    y = pmin(balance, balance + change) - 50,
    label = scales::dollar(change)
  ),
  hjust = 0.5, vjust = 1, size = 3
) +
  scale_x_continuous(
    name = "",
    breaks = balance$time,
    labels = balance$event
  ) +
  scale_y_continuous(
    name = "Balance",
    labels = scales::dollar
  ) +
  scale_fill_brewer(palette = "Spectral") +
  theme_minimal()
```

```
library(ggplot2)
# AtherEnergy/ggTimeSeries
# 个人收入，国家地区收入
library(ggTimeSeries) # https://github.com/AtherEnergy/ggTimeSeries
dat <- data.frame(year = 2000:2021, dpc = 10:31)
ggplot(data = dat, aes(x = year, y = dpc)) +
  stat_waterfall()
```

### 11.4.25 桑基图

#### ggalluvial

```
titanic_wide <- data.frame(Titanic)
head(titanic_wide)

##   Class     Sex   Age Survived Freq
## 1   1st   Male Child       No     0
```

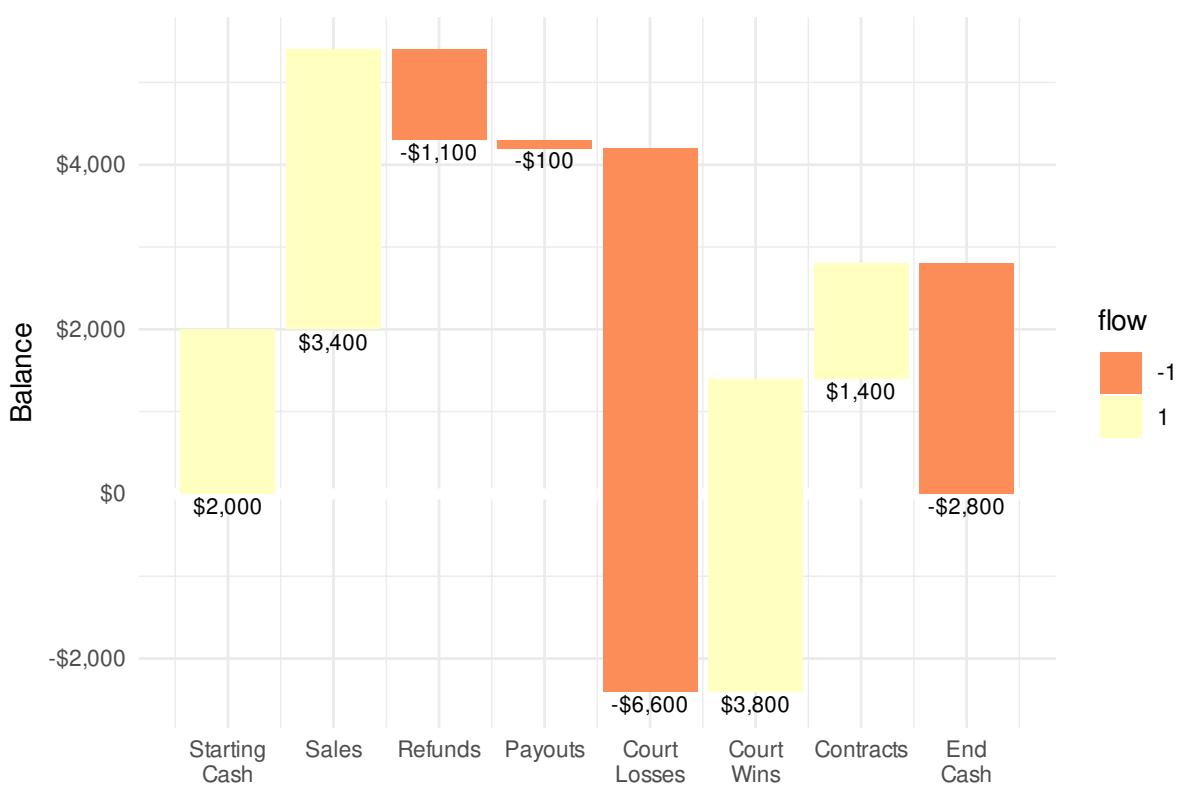


图 11.100: 瀑布图

```

## 2 2nd Male Child No 0
## 3 3rd Male Child No 35
## 4 Crew Male Child No 0
## 5 1st Female Child No 0
## 6 2nd Female Child No 0

library(ggalluvial)
ggplot(data = titanic_wide,
       aes(axis1 = Class, axis2 = Sex, axis3 = Age,
            y = Freq)) +
  scale_x_discrete(limits = c("Class", "Sex", "Age"),
                   expand = c(.2, .05)) +
  xlab("Demographic") +
  geom_alluvium(aes(fill = Survived)) +
  geom_stratum() +
  geom_text(stat = "stratum", aes(label = after_stat(stratum))) +
  theme_minimal() +
  ggtitle("passengers on the maiden voyage of the Titanic",
          "stratified by demographics and survival")

```

#### 11.4.26 词云图

词云 [ggwordcloud](#)

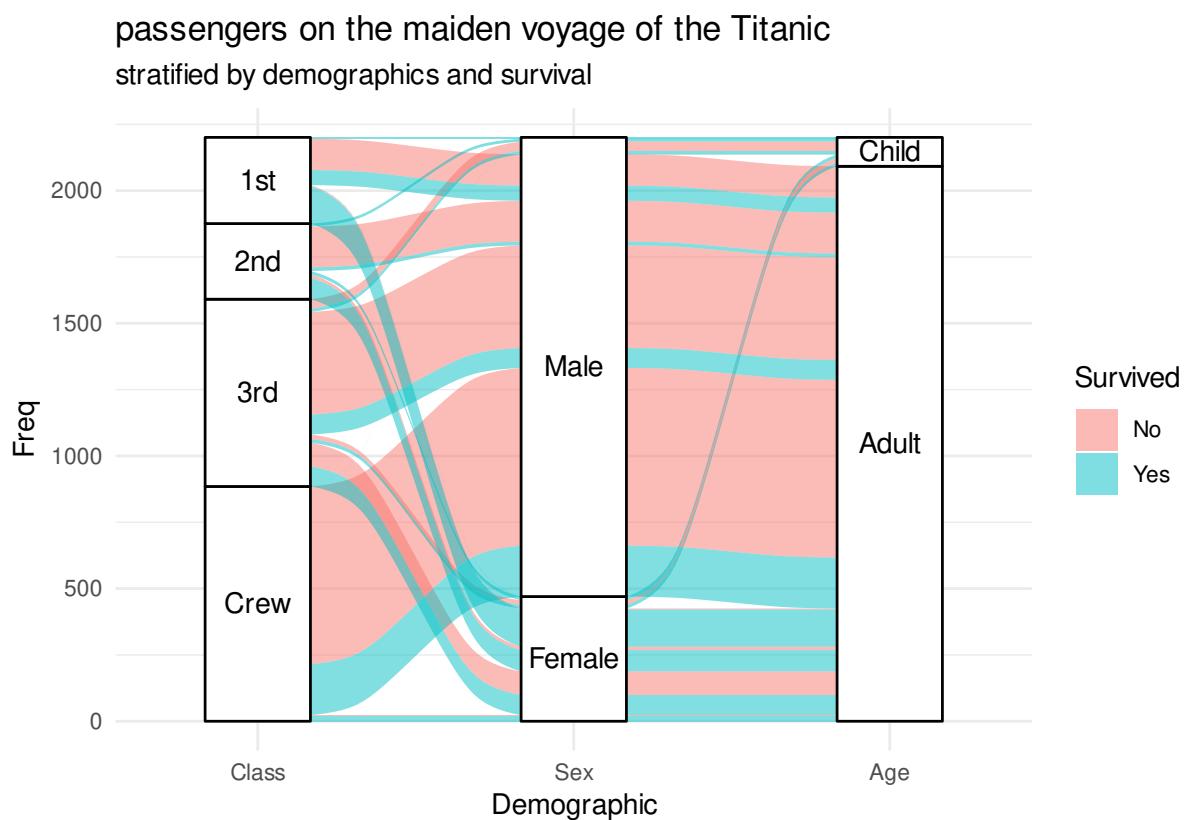


图 11.101: 桑基图

#### 11.4.27 甘特图

描述项目进展的甘特图 [gantrify](#)

#### 11.4.28 马赛克图

```
library(ggmosaic)
ggplot(data = as.data.frame(UCBAdmissions)) +
  geom_mosaic(aes(weight = Freq, x = product(Gender, Admit), fill = Dept)) +
  coord_flip() +
  theme_minimal() +
  labs(x = "Admit", y = "Gender")

## Warning: `unite_()` was deprecated in tidyverse 1.2.0.
## Please use `unite()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

#### 11.4.29 凹凸图

[ggbump](#) 排序随位置的变化

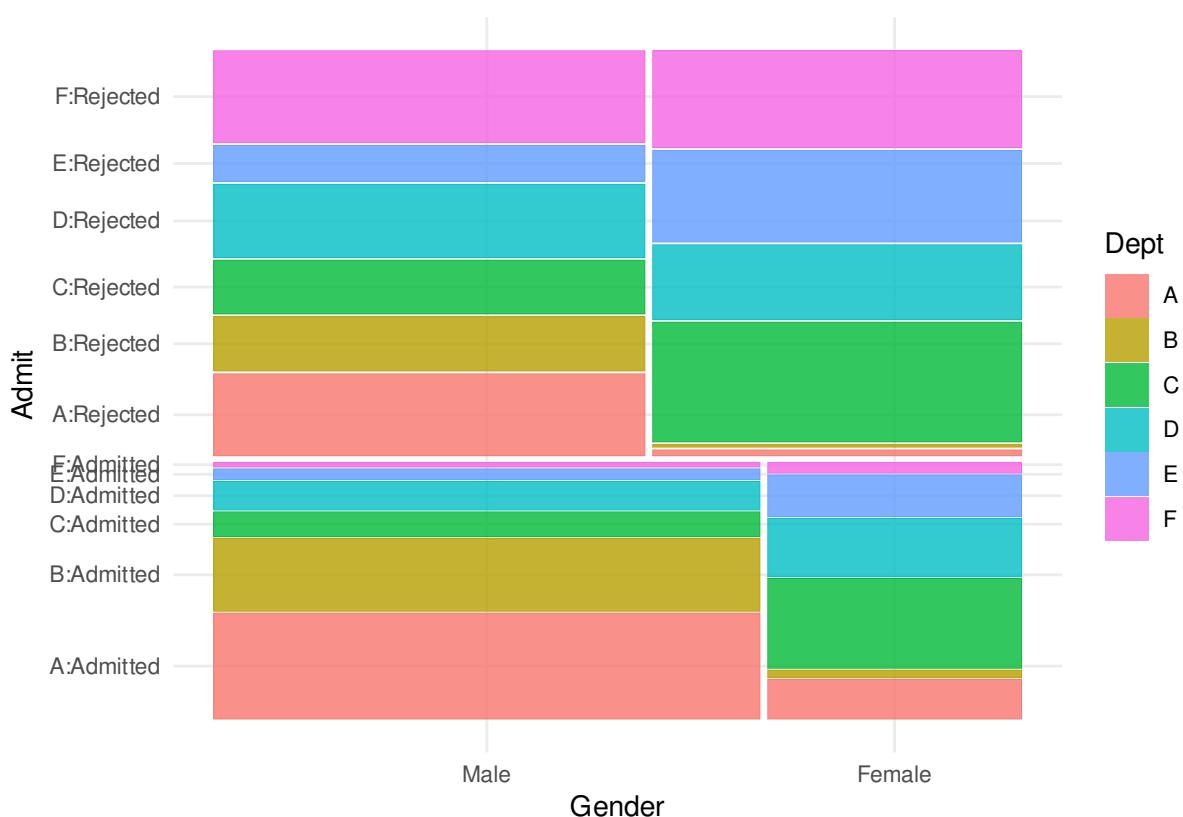


图 11.102: UCBAdmissions 马赛克图

```
# remotes::install_github("davidsjoberg/ggbump")
library(ggbump)
# 代码修改自 https://github.com/davidsjoberg/ggbump
df <- data.frame(
  season = c(
    "Spring", "Pre-season", "Summer", "Season finale", "Autumn", "Winter",
    "Spring", "Pre-season", "Summer", "Season finale", "Autumn", "Winter",
    "Spring", "Pre-season", "Summer", "Season finale", "Autumn", "Winter",
    "Spring", "Pre-season", "Summer", "Season finale", "Autumn", "Winter"
  ),
  rank = c(
    1, 3, 4, 2, 1, 4,
    2, 4, 1, 3, 2, 3,
    4, 1, 2, 4, 4, 1,
    3, 2, 3, 1, 3, 2
  ),
  player = c(
    rep("David", 6),
    rep("Anna", 6),
    rep("Franz", 6),
    rep("Ika", 6)
  )
)
```

```

)
)

# Create factors and order factor
df <- transform(df, season = factor(season, levels = unique(season)))

# Add manual axis labels to plot
ggplot(df, aes(season, rank, color = player)) +
  geom_bump(size = 2, smooth = 20, show.legend = F) +
  geom_point(size = 5, aes(shape = player)) +
  theme_minimal(base_size = 10, base_line_size = 0) +
  theme(panel.grid.major = element_blank(),
        axis.ticks = element_blank()) +
  scale_color_manual(values = RColorBrewer::brewer.pal(name = "Set2", n = 4))

```

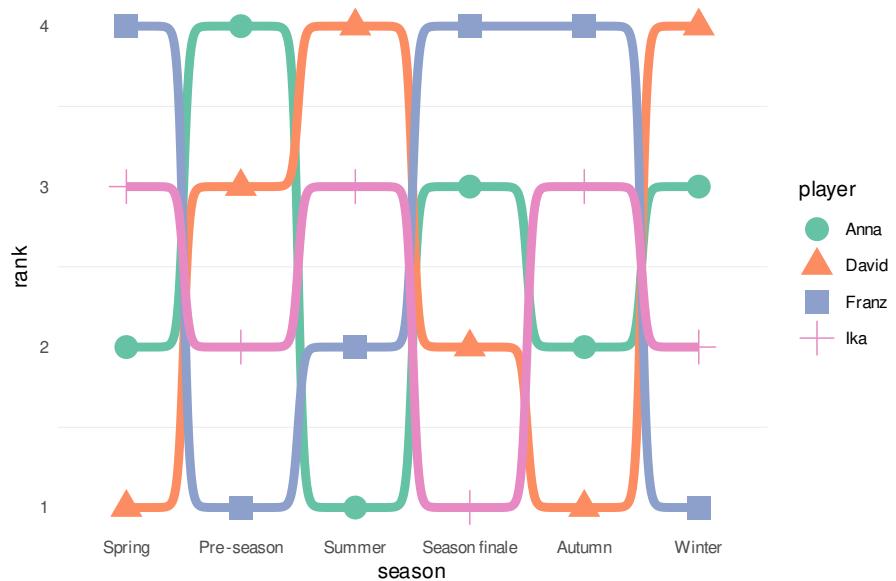


图 11.103: 凸凹图

### 11.4.30 水流图

常用于时间序列数据展示的堆积区域图, [ggstream](#) 和 [streamgraph](#)

```

library(ggstream)

ggplot(blockbusters, aes(year, box_office, fill = genre)) +
  geom_stream() +
  theme_minimal()

```

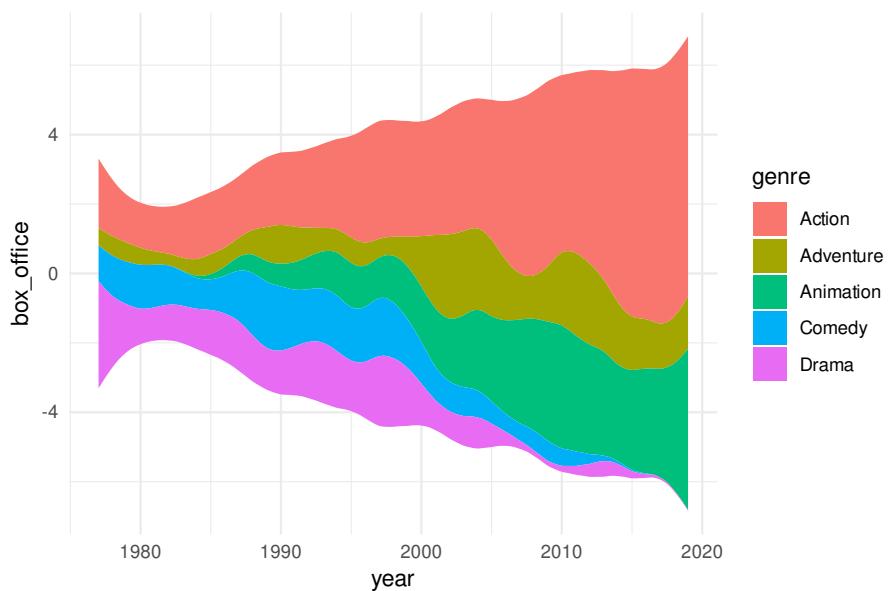


图 11.104: 堆积区域图

#### 11.4.31 时间线

```
# 交互动态图 https://github.com/shosaco/vistime
# 刘思F 2018 数据科学的时间轴 https://bjt.name/2018/11/18/timeline.html
x <- read.table(
  textConnection("
The Future of Data Analysis,1962
Relational Database,1970
Data science(Peter Naur),1974
Two-Way Communication,1975
Exploratory Data Analysis,1977
Business Intelligence,1989
The First Database Report,1992
The World Wide Web Explodes,1995
Data Mining and Knowledge Discovery,1997
S(ACM Software System Award),1998
Statistical Modeling: The Two Cultures,2001
Hadoop,2006
Data scientist,2008
NOSQL,2009
Deep Learning,2015
"),
  sep = ","
)
names(x) <- c("Event", "EventDate")
x$EventDate <- as.Date(paste(x$EventDate, "/01/01", sep = ""))
```

```
library(timelines)
timelines(x,
  labels = paste(x[[1]], format(x[[2]], "%Y")),
  line.color = "blue", label.angle = 15
)
```

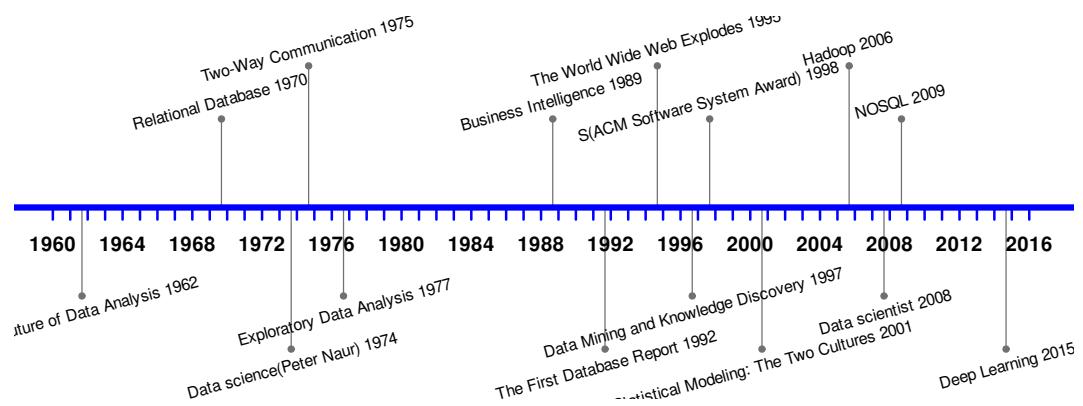


图 11.105: 数据科学的时间轴

```
library(timeline)
data(ww2, package = 'timeline')
timeline(ww2, ww2.events, event.spots=2, event.label='', event.above=FALSE)
```

```
# 适合放在动态幻灯片
# 美团风格的写轮眼
# 时间线
library(vistime)
# presidents and vice presidents
pres <- data.frame(
  Position = rep(c("President", "Vice"), each = 3),
  Name = c("Washington", rep(c("Adams", "Jefferson"), 2), "Burr"),
  start = c("1789-03-29", "1797-02-03", "1801-02-03"),
  end = c("1797-02-03", "1801-02-03", "1809-02-03"),
  color = c("#cbb69d", "#603913", "#c69c6e")
)

hc_vistime(pres, col.event = "Position", col.group = "Name",
  title = "Presidents of the USA")
```



### 11.4.32 三元图

`Ternary` 使用基础图形库, 而 `ggtern` 使用 `ggplot2` 绘制

```
library(ggtern)
library(ggalt)
data("Fragments")
ggtern(Fragments, aes(
  x = Qm, y = Qp, z = Rf + M,
  fill = GrainSize, shape = GrainSize
)) +
  geom_encircle(alpha = 0.5, size = 1) +
  geom_point() +
  labs(
    title = "Example Plot",
    subtitle = "using geom_encircle"
  ) +
  theme_bw() +
  theme_legend_position("tr")
```

### 11.4.33 向量场图

```
library(ggquiver)
```

### 11.4.34 四象限图

```
dat <- data.frame(
  perc = c(54, 18, 5, 15),
  wall_policy = c("oppose", "favor", "oppose", "favor"),
  dreamer_policy = c("favor", "favor", "oppose", "oppose"),
  stringsAsFactors = FALSE
) %>%
  transform(
    xmin = ifelse(wall_policy == "oppose", -sqrt(perc), 0),
    xmax = ifelse(wall_policy == "favor", sqrt(perc), 0),
    ymin = ifelse(dreamer_policy == "oppose", -sqrt(perc), 0),
    ymax = ifelse(dreamer_policy == "favor", sqrt(perc), 0)
  )

ggplot(data = dat) +
  geom_rect(aes(
    xmin = xmin, xmax = xmax,
    ymin = ymin, ymax = ymax
  ), fill = "grey") +
```

```
geom_text(aes(  
  x = xmin + 0.5 * sqrt(perc),  
  y = ymin + 0.5 * sqrt(perc),  
  label = perc  
)  
,  
  color = "white", size = 10  
) +  
  coord_equal() +  
  geom_hline(yintercept = 0) +  
  geom_vline(xintercept = 0) +  
  theme_minimal() +  
  labs(x = "", y = "", title = "")
```

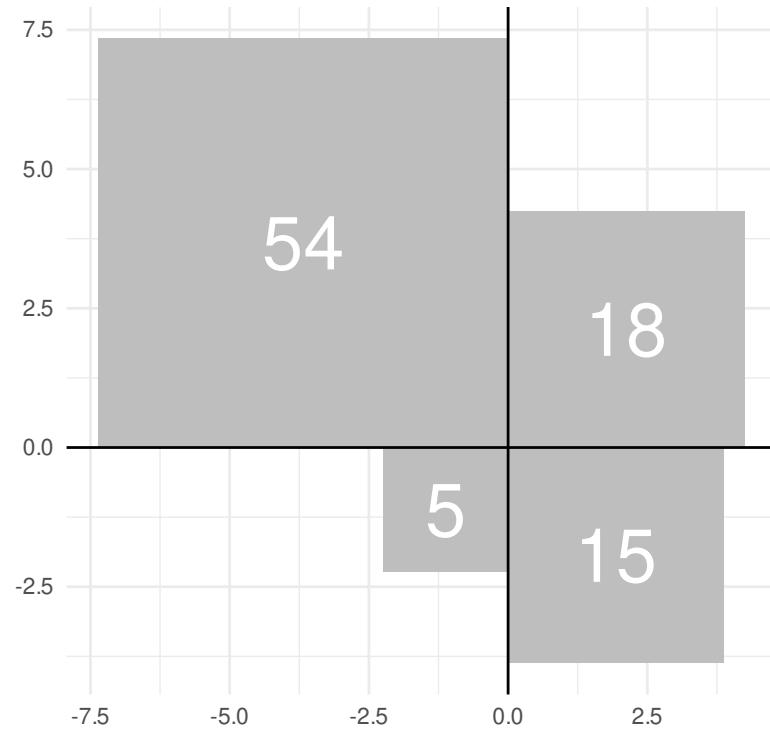


图 11.106: 四象限图

#### 11.4.35 韦恩图

[ggVennDiagram](#)

#### 11.4.36 龙卷风图

```
dat <- data.frame(  
  variable = c("A", "B", "A", "B"),  
  Level = c("Top-2", "Top-2", "Bottom-2", "Bottom-2"),  
  value = c(.8, .7, -.2, -.3)  
)  
ggplot(dat, aes(x = variable, y = value, fill = Level)) +  
  geom_bar(position = "identity", stat = "identity") +  
  scale_y_continuous(labels = abs) +  
  coord_flip() +  
  theme_minimal()
```

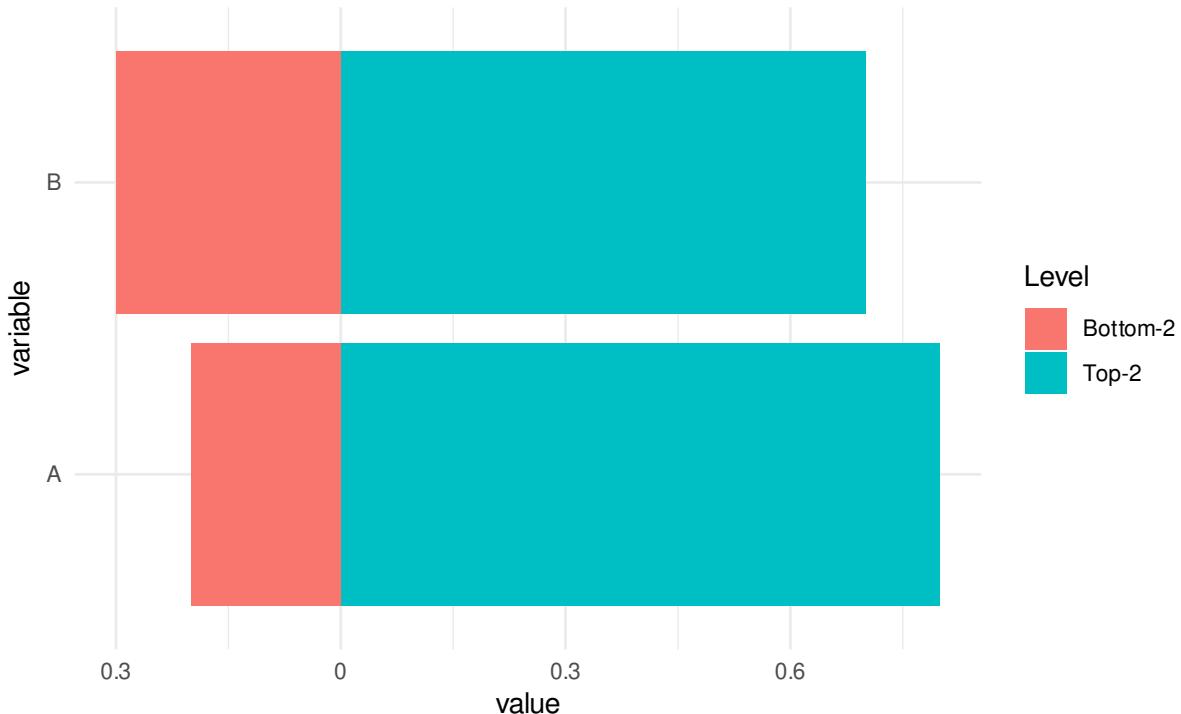


图 11.107: 龙卷风图展示变量重要性

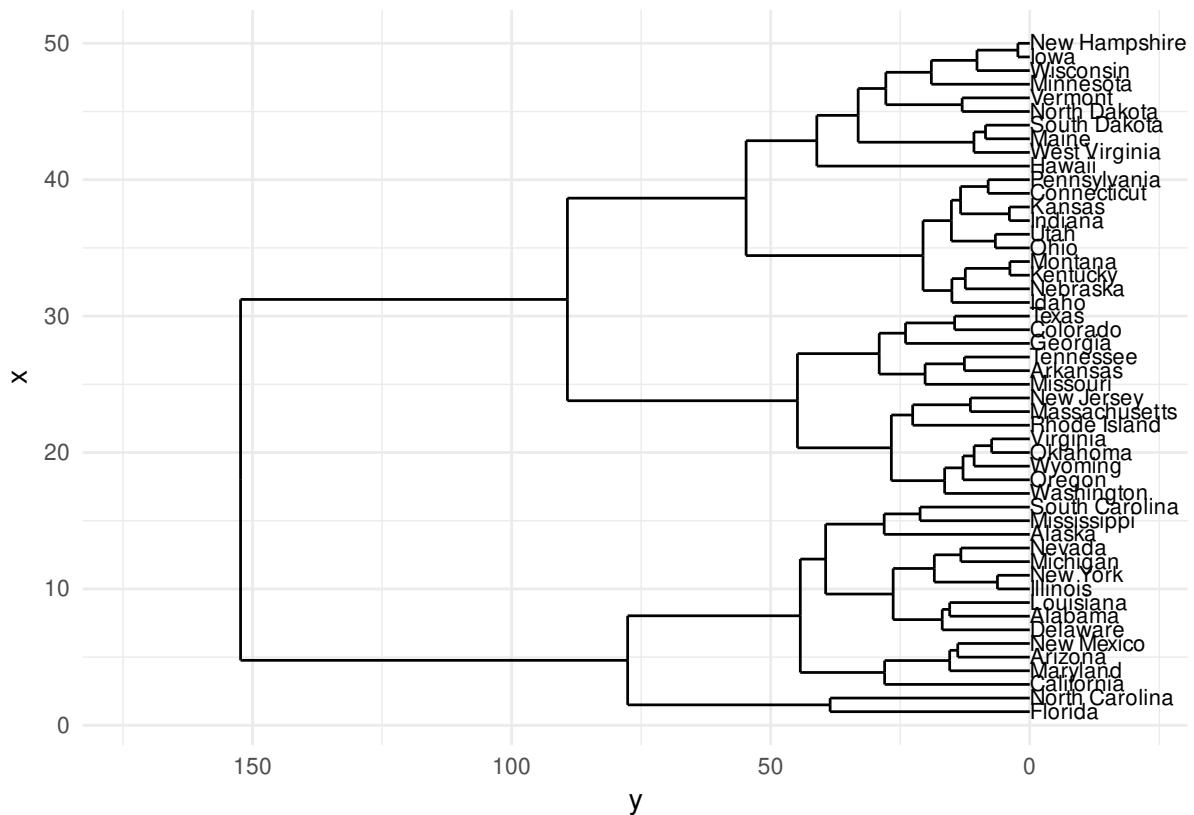
**Tornado diagram** 主要用于敏感性分析, 比较不同变量的重要性程度。条形图 `geom_bar()` 图层的变体, 模型权重可视化的手段, 仅限于广义线性模型。

### 11.4.37 聚类图

`ggdendro` 的 `dendro_data()` 函数支持 `tree`、`hclust`、`dendrogram` 和 `rpart` 结果的整理, 进而绘图

```
library(ggdendro)  
hc <- hclust(dist(USArrests), "ave")  
hcdata <- dendro_data(hc, type = "rectangle")  
ggplot() +  
  geom_segment(data = segment(hcdata),  
               aes(x = x, y = y, xend = xend, yend = yend)
```

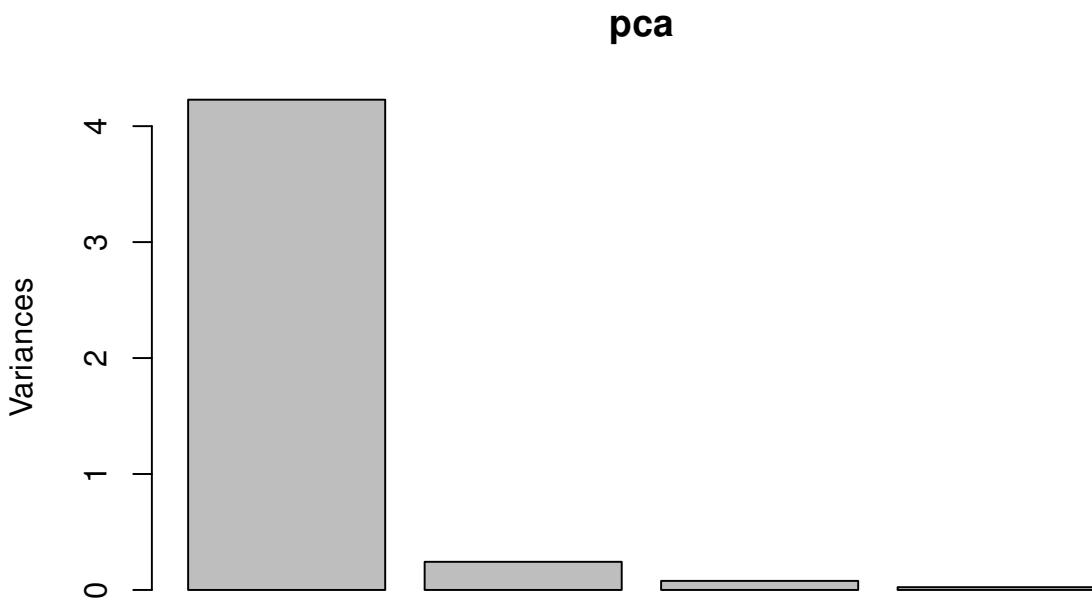
```
) +  
  geom_text(data = label(hcdata),  
            aes(x = x, y = y, label = label, hjust = 0),  
            size = 3  
) +  
  coord_flip() +  
  scale_y_reverse(expand = c(0.2, 0)) +  
  theme_minimal()
```



#### 11.4.38 主成分图

借助 **autoplotly** 包 [Tang, 2018] 可将函数 `stats::prcomp` 生成的结果转化为交互图形

```
pca <- prcomp(iris[c(1, 2, 3, 4)])  
plot(pca)
```



```
library(autoplotly)
autoplotly(pca,
  data = iris, colour = "Species",
  label = TRUE, label.size = 3, frame = TRUE
)
```

ggfortify [Tang et al., 2016] 包将主成分分析图转化为静态图形

```
library(ggfortify)
autoplot(pca, data = iris, colour = 'Species')
```

#### 11.4.39 组合图

组合的意思是将不同种类的图形绘制在一个区域中，比如密度曲线和地毡图<sup>10</sup>组合。GGally、ggupset、ggcharts 和 ggpubr 高度定制了一些组合统计图形，以 ggpubr 为例，见图 11.109。

```
library(ggpubr)
ggdensity(sleep,
  x = "extra", add = "mean", rug = TRUE, color = "group",
  fill = "group", palette = c("#00AFBB", "#E7B800")
)
```

上面介绍的都是已经固化的组合方式，一般地，将多个图形组合到一个图中，可以有很多办法，比如 Claus Wilke 开发的 cowplot，在他的书里 Fundamentals of Data Visualization 大量使用，后起之秀 patchwork

<sup>10</sup>其实是轴须图 rug plot，只因样子看起来像铺在地上的毛毯，故而称之为地毡图，对应于 R 内置的 rug() 函数或 ggplot2 提供的图层 geom\_rug()，更多解释详见 [https://en.wikipedia.org/wiki/Rug\\_plot](https://en.wikipedia.org/wiki/Rug_plot)。

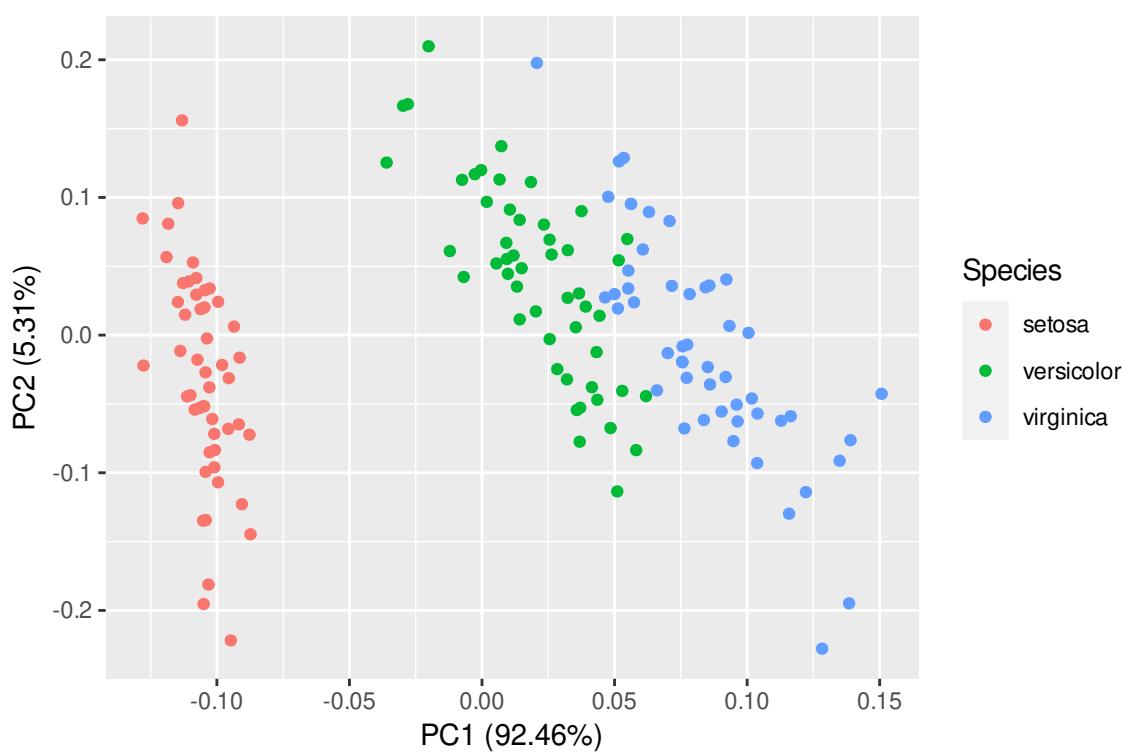


图 11.108: 主成分分析

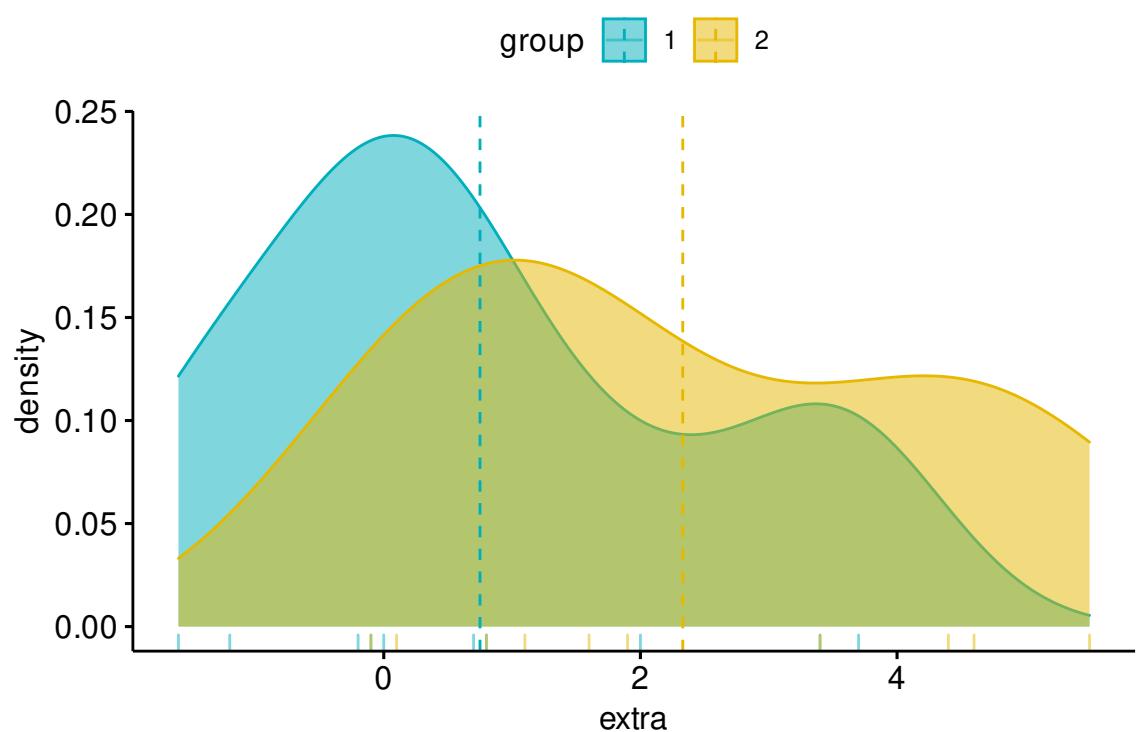


图 11.109: 组合图形

表 11.3: 哌哚美辛在人体中的代谢情况

Subject	0.25	0.5	0.75	1	1.25	2	3	4	5	6	8
1	1.50	0.94	0.78	0.48	0.37	0.19	0.12	0.11	0.08	0.07	0.05
2	2.03	1.63	0.71	0.70	0.64	0.36	0.32	0.20	0.25	0.12	0.08
3	2.72	1.49	1.16	0.80	0.80	0.39	0.22	0.12	0.11	0.08	0.08
4	1.85	1.39	1.02	0.89	0.59	0.40	0.16	0.11	0.10	0.07	0.07
5	2.05	1.04	0.81	0.39	0.30	0.23	0.13	0.11	0.08	0.10	0.06
6	2.31	1.44	1.03	0.84	0.64	0.42	0.24	0.17	0.13	0.10	0.09

则提供更加简洁的组合语法，非常受欢迎，更加底层的拼接方法可以去看 [一页多图](#) 和 R 内置的 grid 系统。

#### 11.4.40 动态图

**av** 包基于 **FFmpeg** 将静态图片合成视频，而 **gifski** 包基于 **gifski** 将静态图片合成 GIF 动画，**animation** 包 [Xie, 2013] 将 Base R 绘制的图形转化为动画或视频，**mapmate** 制作地图相关的三维可视化图形，**gganimate** 包支持将 **ggplot2** 生成的图形，**magick** 可以将一系列静态图形合成功态图形，借助 **gifski** 包转化为动态图片或视频。推荐读者从 **gganimate** 案例合集 开始制作动态图形。**rgl** 可以制作真三维动态图形，支持缩放、拖拽、旋转等操作，**rayshader** 还支持转化 **ggplot2** 对象为 3D 图形。

数据集 **Indometh** 记录了药物在人体中的代谢情况，给 6 个人分别静脉注射了哌哚美辛，每隔一段时间抽血检查药物在血浆中的浓度，收集的数据见表 11.3

```
reshape(Indometh, v.names = "conc", idvar = "Subject",
       timevar = "time", direction = "wide", sep = "") %>%
knitr::kable(.,  
  caption = "哌哚美辛在人体中的代谢情况",  
  row.names = FALSE, col.names = gsub("(conc)", "", names(.)),  
  align = "c"  
)
```

如图 11.110 所示，药物在人体中浓度变化情况

```
p <- ggplot(  
  data = Indometh,  
  aes(x = time, y = conc, color = Subject))  
+  
  geom_point() +  
  geom_line() +  
  theme_minimal() +  
  labs(  
    x = "time (hr)",  
    y = "plasma concentrations of indometacin (mcg/ml)")  
)  
p
```

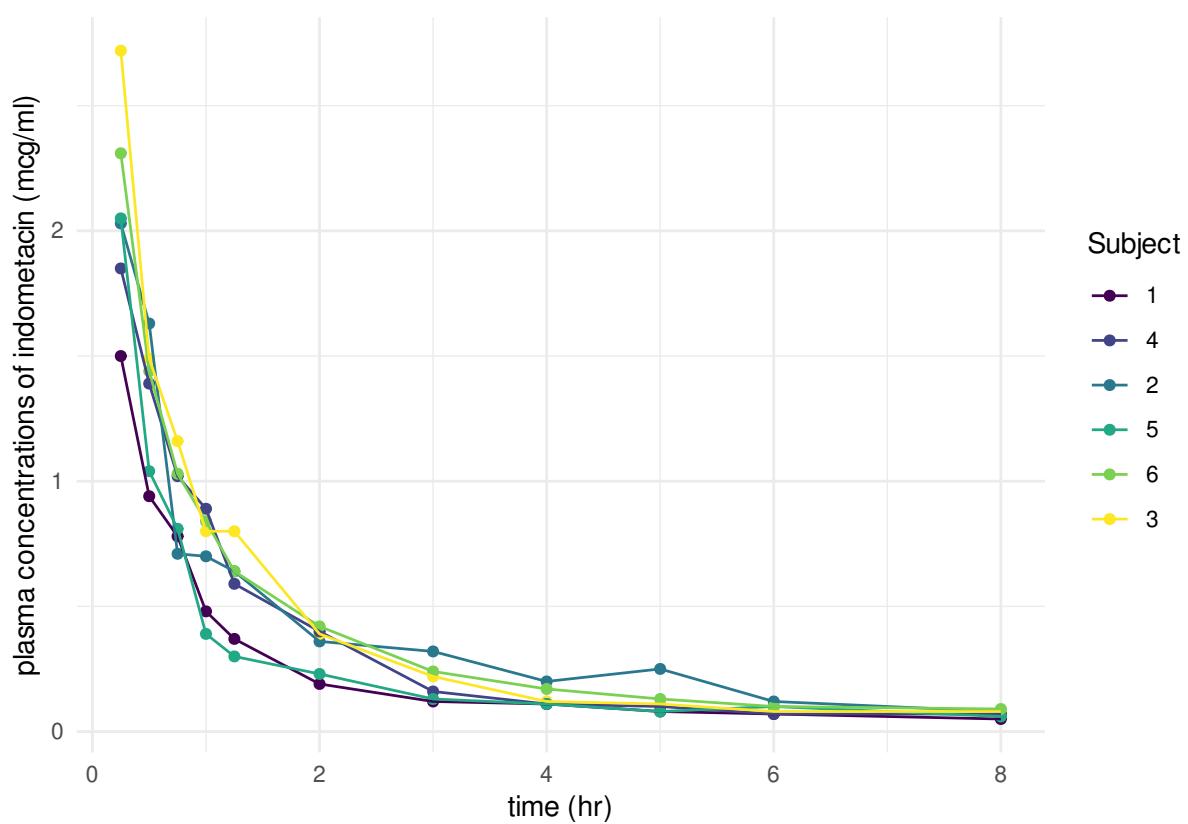


图 11.110: 药物在人体中的代谢情况

```
library(gganimate)
p + transition_reveal(time)
```

## 提示

书籍目标输出格式是 PDF，则在代码块选项设置里必须指定参数 `fig.show='animate'` 否则插入的只是图片而不是动画，目标格式是 HTML 网页，就不必指定参数，默认会将图片合成 GIF 动态图，嵌入 PDF 里面的动画需要 Acrobat Reader 阅读器才能正确地显示。

动态图形制作的原理，简单来说，就是将一帧帧静态图形以较快的速度播放，人眼形成视觉残留，以为是连续的画面，相比于 `animation`，`ggridanimate` 借助 `tweenr` 包添加了过渡效果，动态图形显得非常自然。下面以 `cup` 函数<sup>11</sup>为例

$$f(x; \theta, \phi) = \theta x \log(x) - \frac{1}{\phi} e^{-\phi^4(x - \frac{1}{e})^4}, \quad \theta \in (2, 3), \phi \in (30, 50), x \in (0, 1)$$

函数图像随着  $\theta$  和  $\phi$  的变化情况见图 11.111。

```
library(tweenr)
cup_curve <- function(n = 100, theta = 3, phi = 30, cup = "A") {
  data.frame(x = seq(0.00001, 1, length.out = n), cup = cup) %>%
    transform(y = theta * x * log(x, base = 10)
              - 1 / phi * exp(-(phi * x - phi / exp(1))^4))
}
mapply(
  FUN = cup_curve, theta = c(E = 3, D = 2.8, C = 2.5, B = 2.2, A = 2),
  phi = c(30, 33, 36, 40, 50), cup = c("E", "D", "C", "B", "A"),
  MoreArgs = list(n = 50), SIMPLIFY = FALSE, USE.NAMES = TRUE
```

<sup>11</sup>函数来自余光创的博客 — [3D 版邪恶的曲线](#)，此处借用 `ggridanimate` 将其动态化，前方高能，少儿不宜，R 还能这么不正经的玩。

```
) %>%
  tween_states(
    data = .,
    tweenlength = 2, statelength = 1,
    ease = rep("cubic-in-out", 4), nframes = 100
  ) %>%
  ggplot(data = ., aes(x, y, color = cup, frame = .frame)) +
  geom_path() +
  coord_flip() +
  theme_void()
```

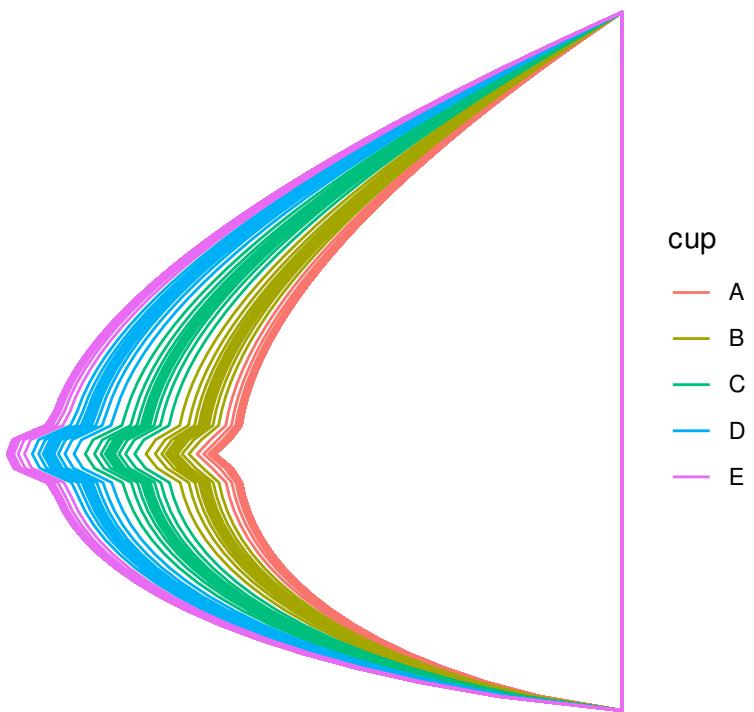


图 11.111: 添加过渡效果

## 第十二章 交互图形

`plotly` 是一个功能非常强大的绘制交互式图形的 R 包。它支持下载图片、添加水印、自定义背景图片、工具栏和注释<sup>1</sup>等一系列细节的自定义控制。下面结合 JavaScript 库 `plotly.js` 一起介绍，帮助文档 `?config` 没有太详细地介绍，所以我们看看 `config()` 函数中参数 ... 和 JavaScript 库 `plot_config.js` 中的功能函数是怎么对应的。图中图片下载按钮对应 `toImageButtonOptions` 参数，看 `toImageButtonOptions` 源代码，可知，它接受任意数据类型，对应到 R 里面就是列表。`watermark` 和 `displaylogo` 都是传递布尔值 (TRUE/FALSE)，具体根据 JavaScript 代码中的 `valType` (参数值类型) 决定，其它参数类似。另一个函数 `layout` 和函数 `config()` 是类似的，怎么传递参数值是根据 JavaScript 代码来的。

```
toImageButtonOptions: {
  valType: 'any',
  dflt: {},
  description: [
    'Statically override options for toImage modebar button',
    'allowed keys are format, filename, width, height, scale',
    'see ../components/modebar/buttons.js'
  ].join(' ')
},
displaylogo: {
  valType: 'boolean',
  dflt: true,
  description: [
    'Determines whether or not the plotly logo is displayed',
    'on the end of the mode bar.'
  ].join(' ')
},
watermark: {
  valType: 'boolean',
  dflt: false,
  description: 'watermark the images with the company\'s logo'
},
library(plotly, warn.conflicts = FALSE)
plot_ly(diamonds,
  x = ~clarity, y = ~price,
  color = ~clarity, colors = "Set1", type = "box"
```

<sup>1</sup><https://plotly.com/r/reference/#layout-scene-annotations-items-annotation-font>

```
) %>%  
  config(  
    toImageButtonOptions = list(  
      format = "svg", width = 450, height = 300,  
      filename = paste("plot", Sys.Date(), sep = "_")  
    ),  
    modeBarButtons = list(list("toImage")),  
    watermark = FALSE,  
    displaylogo = FALSE,  
    locale = "zh-CN",  
    staticPlot = TRUE,  
    showLink = FALSE,  
    modeBarButtonsToRemove = c(  
      "hoverClosestCartesian", "hoverCompareCartesian",  
      "zoom2d", "zoomIn2d", "zoomOut2d",  
      "autoScale2d", "resetScale2d", "pan2d",  
      "toggleSpikelines"  
    )  
) %>%  
  layout(  
    template = "plotly_dark",  
    images = list(  
      source = "https://images.plot.ly/language-icons/api-home/r-logo.png",  
      xref = "paper",  
      yref = "paper",  
      x = 1.00,  
      y = 0.25,  
      sizex = 0.2,  
      sizey = 0.2,  
      opacity = 0.5  
    ),  
    annotations = list(  
      text = "DRAFT",          # 水印文本  
      textangle = -30,          # 逆时针旋转 30 度  
      font = list(  
        size = 40,              # 字号  
        color = "gray",          # 颜色  
        family = "Times New Roman" # 字族  
      ),  
      opacity = 0.2,            # 透明度  
      xref = "paper",  
      yref = "paper",  
      x = 0.5,  
      y = 0.5,  
      showarrow = FALSE         # 去掉箭头指示  
    )
```



```
  )
  )
```

表 12.1: 交互图形的设置函数 config() 各个参数及其作用 (部分)

参数	作用
displayModeBar	是否显示交互图形上的工具条, 默认显示 TRUE <sup>2</sup> 。
modeBarButtons	工具条上保留的工具, 如下载 "toImage", 缩放 "zoom2d" <sup>3</sup> 。
modeBarButtonsToRemove	工具条上要移除的工具, 如下载和缩放图片 c("toImage", "zoom2d")。
toImageButtonOptions	工具条上下载图片的选项设置, 包括名称、类型、尺寸等。 <sup>4</sup>
displaylogo	是否显示交互图形上 Plotly 的图标, 默认显示 TRUE <sup>5</sup> 。
staticPlot	是否将交互图形转为静态图形, 默认 FALSE。
locale	本土化语言设置, 比如 "zh-CN" 表示中文。

## 12.1 散点图

表 12.2: 散点图类型

类型	名称
scattercarpet	地毯图
scatterternary	三元图
scatter3d	三维散点图
scattergeo	地图散点图
scattermapbox	地图散点图 Mapbox
scatter	散点图
scattergl	散点图 GL
scatterpolar	极坐标散点图
scatterpolargl	极坐标散点图 GL

plotly.js 提供很多图层用于绘制各类图形 <https://github.com/plotly/plotly.js/tree/master/src/traces>

```
# 折线图
plot_ly(Orange,
  x = ~age, y = ~circumference, color = ~Tree,
  type = "scatter", mode = "markers"
)
```

<sup>2</sup><https://plotly-r.com/control-modebar.html>。

<sup>3</sup>完整的列表见 <https://github.com/plotly/plotly.js/blob/master/src/components/modebar/buttons.js>。

<sup>4</sup>设置下载图片的尺寸, 还可设置为 PNG 格式, SVG 格式图片, 可借助 rsvg 的 rsvg\_pdf() 函数转化为 PDF 格式 <https://github.com/ropensci/plotly/issues/1556#issuecomment-505833092>。

<sup>5</sup><https://plotly.com/r/logos/>。



## 12.2 条形图

日常使用最多的图形无外乎散点图、柱形图（分组、堆积、百分比堆积等）

```
# 简单条形图
library(data.table)
diamonds <- as.data.table(diamonds)

p11 <- diamonds[, .(cnt = .N), by = .(cut)] %>%
  plot_ly(x = ~cut, y = ~cnt, type = "bar") %>%
  add_text(
    text = ~ scales::comma(cnt), y = ~cnt,
    textposition = "top middle",
    cliponaxis = FALSE, showlegend = FALSE
  )
# 分组条形图
p12 <- plot_ly(diamonds,
  x = ~cut, color = ~clarity,
  colors = "Accent", type = "histogram"
)
# 堆积条形图
p13 <- plot_ly(diamonds,
  x = ~cut, color = ~clarity,
  colors = "Accent", type = "histogram"
) %>%
  layout(barmode = "stack")
# 百分比堆积条形图
# p14 <- plot_ly(diamonds,
#   x = ~cut, color = ~clarity,
#   colors = "Accent", type = "histogram"
# ) %>%
#   layout(barmode = "stack", barnorm = "percent") %>%
#   config(displayModeBar = F)

# 推荐使用如下方式绘制堆积条形图
dat = diamonds[, .(cnt = length(carat)), by = .(clarity, cut)] %>%
  .[, pct := round(100 * cnt / sum(cnt), 2), by = .(cut)]

p14 <- plot_ly(
  data = dat, x = ~cut, y = ~pct, color = ~clarity,
  colors = "Set3", type = "bar"
) %>%
  layout(barmode = "stack")

htmltools::tagList(p11, p12, p13, p14)
```



## 12.3 折线图

其它常见的图形还要折线图、直方图、箱线图和提琴图

```
# 折线图
plot_ly(Orange,
  x = ~age, y = ~circumference, color = ~Tree,
  type = "scatter", mode = "markers+lines"
)
```

## 12.4 双轴图

### 双轴图

模拟一组数据

```
set.seed(2020)
dat <- data.frame(
  dt = seq(from = as.Date("2020-01-01"), to = as.Date("2020-01-31"), by = "day"),
  search_qv = sample(100000:1000000, size = 31, replace = T)
) %>%
  transform(valid_click_qv = sapply(search_qv, rbinom, n = 1, prob = 0.5)) %>%
  transform(qv_ctr = valid_click_qv / search_qv)
```

hoverinfo = "text" 表示 tooltips 使用指定的 text 映射, 而 visible = "legendonly" 表示图层默认隐藏不展示, 只在图例里显示, 有时候很多条线, 默认只是展示几条而已。举例如下

```
plot_ly(data = dat) %>%
  add_bars(
    x = ~dt, y = ~search_qv, color = I("gray80"), name = "搜索 QV",
    text = ~ paste0(
      "日期: ", dt, "<br>",
      "点击 QV: ", format(valid_click_qv, big.mark = ","), "<br>",
      "搜索 QV: ", format(search_qv, big.mark = ","), "<br>",
      "QV_CTR: ", scales::percent(qv_ctr, accuracy = 0.01), "<br>"
    ),
    hoverinfo = "text"
) %>%
  add_bars(
    x = ~dt, y = ~valid_click_qv, color = I("gray60"), name = "点击 QV",
    text = ~ paste0(
      "日期: ", dt, "<br>",
      "点击 QV: ", format(valid_click_qv, big.mark = ","), "<br>",
      "搜索 QV: ", format(search_qv, big.mark = ","), "<br>",
      "QV_CTR: ", scales::percent(qv_ctr, accuracy = 0.01), "<br>"
    ), visible = "legendonly",
    hoverinfo = "text"
)
```



```
) %>%
add_lines(
  x = ~dt, y = ~qv_ctr, name = "QV_CTR", yaxis = "y2", color = I("gray40"),
  text = ~ paste("QV_CTR: ", scales::percent(qv_ctr, accuracy = 0.01), "<br>"),
  hoverinfo = "text",
  line = list(shape = "spline", width = 3, dash = "line")
) %>%
layout(
  title = "",
  yaxis2 = list(
    tickfont = list(color = "black"),
    overlaying = "y",
    side = "right",
    title = "QV_CTR (%)",
    # ticksuffix = "%", # 设置坐标轴单位
    tickformat = '.1%', # 设置坐标轴刻度
    showgrid = F, automargin = TRUE
  ),
  xaxis = list(title = "日期", showgrid = F, showline = F),
  yaxis = list(title = " ", showgrid = F, showline = F),
  margin = list(r = 20, autoexpand = T),
  legend = list(
    x = 0, y = 1, orientation = "h",
    title = list(text = " ")
  )
)
```

## 12.5 直方图

```
plot_ly(iris,
  x = ~Sepal.Length, colors = "Greys",
  color = ~Species, type = "histogram"
)
```

## 12.6 箱线图

```
# 箱线图
plot_ly(diamonds,
  x = ~clarity, y = ~price, colors = "Greys",
  color = ~clarity, type = "box"
)
```

表 12.3: 图层

A	B	C
add_annotations	add_histogram	add_polygons
add_area	add_histogram2d	add_ribbons
add_bars	add_histogram2dcontour	add_scattergeo
add_boxplot	add_image	add_segments
add_choropleth	add_lines	add_sf
add_contour	add_markers	add_surface
add_data	add_mesh	add_table
add_fun	add_paths	add_text
add_heatmap	add_pie	add_trace

## 12.7 提琴图

```
plot_ly(sleep,
  x = ~group, y = ~extra, split = ~group,
  type = "violin",
  box = list(visible = T),
  meanline = list(visible = T)
)
```

plotly 包含图层 27 种, 见表 12.3

## 12.8 气泡图

简单图形 scatter, 分布图几类, 其中 scatter、heatmap、scatterpolar 支持 WebGL 绘图引擎

```
# https://plotly.com/r/bubble-charts/
dat <- diamonds[, .(
  carat = mean(carat),
  price = sum(price),
  cnt = .N
), by = .(cut)]
plot_ly(
  data = dat, colors = "Greys",
  x = ~carat, y = ~price, color = ~cut, size = ~cnt,
  type = "scatter", mode = "markers",
  marker = list(
    symbol = "circle", sizemode = "diameter",
    line = list(width = 2, color = "#FFFFFF"), opacity = 0.4
  ),
  text = ~ paste(
```

```
sep = " ", "重量: ", round(carat, 2), "克拉",
      "<br>价格:", round(price / 10^6, 2), "百万"
    ),
  hoverinfo = 'text'
) %>%
add_annotations(
  x = ~carat, y = ~price, text = ~cnt,
  showarrow = F, font = list(family = "sans")
) %>%
layout(
  xaxis = list(hoverformat = ".2f"),
  yaxis = list(hoverformat = ".0f")
)
```

## 12.9 曲线图

```
plot_ly(
  x = c(1, 2.2, 3), y = c(5.3, 6, 7),
  type = "scatter", color = I("gray40"),
  mode = "markers+lines", line = list(shape = "spline")
) %>%
add_annotations(
  x = 2, y = 6, size = I(100),
  text = TeX("x_i \sim N(\mu, \sigma^2)")
) %>%
layout(
  xaxis = list(showgrid = F, title = TeX("\mu")),
  yaxis = list(showgrid = F, title = TeX("\alpha"))
) %>%
config(mathjax = 'cdn')
```

## 12.10 堆积图

```
plot_ly(
  data = PlantGrowth, y = ~weight,
  color = ~group, colors = "Greys",
  type = "scatter", line = list(shape = "spline"),
  mode = "lines", fill = "tozeroY"
)
```



## 12.11 热力图

其他基础图形

```
plot_ly(z = volcano, type = 'heatmap', colors = "Greys")
```



## 12.12 地图 I

plot\_mapbox() 使用 Mapbox 提供的地图服务，因此，需要注册一个账户，获取 MAPBOX\_TOKEN

```
data("quakes")
plot_mapbox(
  data = quakes, colors = "Greys",
  lon = ~long, lat = ~lat,
  color = ~mag, size = 2,
  type = "scattermapbox",
  mode = "markers",
  marker = list(opacity = 0.5)
) %>%
  layout(
    title = "Fiji Earthquake",
    mapbox = list(
      zoom = 3,
      center = list(
        lat = ~ median(lat - 5),
        lon = ~ median(long)
      )
    )
  )
) %>%
  config(
    mapboxAccessToken = Sys.getenv("MAPBOX_TOKEN")
  )
```

```
plotly::plot_ly(
  data = quakes,
  lon = ~long, lat = ~lat,
  type = "scattergeo", mode = "markers",
  text = ~ paste0(
    "站点: ", stations, "<br>",
    "震级: ", mag
  ),
  marker = list(
    color = ~mag,
    size = 10, opacity = 0.8,
    line = list(color = "white", width = 1)
```

```
)  
) %>%  
  plotly::layout(geo = list(  
    showland = TRUE,  
    landcolor = plotly::toRGB("gray95"),  
    subunitcolor = plotly::toRGB("gray85"),  
    countrycolor = plotly::toRGB("gray85"),  
    countrywidth = 0.5,  
    subunitwidth = 0.5,  
    lonaxis = list(  
      showgrid = TRUE,  
      gridwidth = 0.5,  
      range = c(160, 190),  
      dtick = 5  
,  
    lataxis = list(  
      showgrid = TRUE,  
      gridwidth = 0.5,  
      range = c(-40, -10),  
      dtick = 5  
)  
)  
  
dat <- data.frame(state.x77,  
  stats = rownames(state.x77),  
  stats_abbr = state.abb  
)  
  
plotly::plot_ly(  
  data = dat,  
  type = "choropleth",  
  locations = ~stats_abbr,  
  locationmode = "USA-states",  
  colorscale = "Viridis",  
  z = ~Income  
) |>  
  plotly::layout(  
    geo = list(scope = "usa"),  
    title = "1974年美国各州的人均收入",  
    legend = list(title = "收入")  
)
```

### 12.13 拟合图

```
plot_ly(economics,
  type = "scatter",
  x = ~date,
  y = ~uempmed,
  name = "observed unemployment",
  mode = "markers+lines",
  marker = list(
    color = "red"
  ),
  line = list(
    color = "red",
    dash = "dashed"
  )
) %>%
add_trace(
  x = ~date,
  y = ~fitted(loess(uempmed ~ as.numeric(date))),
  name = "fitted unemployment",
  mode = "markers+lines",
  marker = list(
    color = "orange"
  ),
  line = list(
    color = "orange"
  )
) %>%
layout(
  title = "失业时间",
  xaxis = list(
    title = "日期",
    showgrid = F
  ),
  yaxis = list(
    title = "失业时间 (周)"
  ),
  legend = list(
    x = 0, y = 1, orientation = "v",
    title = list(text = "")
  )
)
```

## 12.14 轨迹图

**rasterly** 百万量级的散点图

```
library(rasterly)
plot_ly(quakes, x = ~long, y = ~lat) %>%
  add_rasterly_heatmap()

quakes %>%
  rasterly(mapping = aes(x = long, y = lat)) %>%
  rasterly_points()

library(plotly)
# 读取数据
# uber 轨迹数据来自 https://github.com/plotly/rasterly
ridesDf <- readRDS(file = 'data/uber.rds')

ridesDf %>%
  rasterly(mapping = aes(x = Lat, y = Lon)) %>%
  rasterly_points()
```

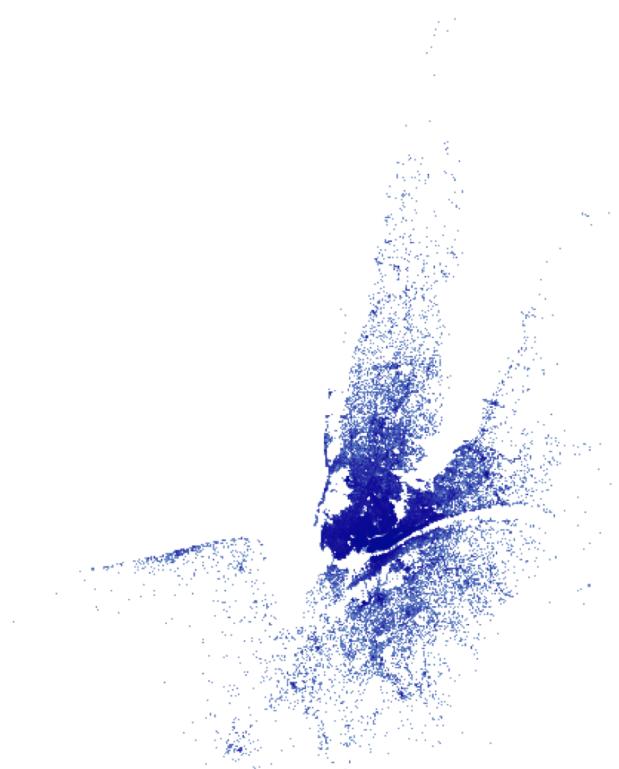


图 12.1: 轨迹数据



## 12.15 三维图 (plotly)

```
plot_ly(z = ~volcano) %>%
  add_surface()

plot_ly(x = c(0, 0, 1), y = c(0, 1, 0), z = c(0, 0, 0)) %>%
  add_mesh()

# https://plot.ly/r/reference/#scatter3d
transform(mtcars, am = ifelse(am == 0, "Automatic", "Manual")) %>%
  plot_ly(x = ~wt, y = ~hp, z = ~qsec,
          color = ~am, colors = c("#BF382A", "#0C4B8E")) %>%
  add_markers() %>%
  layout(scene = list(
    xaxis = list(title = "Weight"),
    yaxis = list(title = "Gross horsepower"),
    zaxis = list(title = "1/4 mile time")
  ))

```

## 12.16 甘特图

项目管理必备，如图所示，本项目拆分成 7 个任务，一共使用 3 种项目资源

```
# https://plotly.com/r/gantt/
# 项目拆解为一系列任务，每个任务的开始时间，持续时间和资源类型
df <- data.frame(
  task = paste("Task", 1:8),
  start = as.Date(c(
    "2016-01-01", "2016-02-20", "2016-01-01",
    "2016-04-10", "2016-06-09", "2016-04-10",
    "2016-09-07", "2016-11-26"
  )),
  duration = c(50, 25, 100, 60, 30, 150, 80, 10),
  resource = c("A", "B", "C", "C", "C", "A", "B", "B")
) %>%
  transform(end = start + duration) %>%
  transform(y = 1:nrow(.))

plot_ly(data = df) %>%
  add_segments(
    x = ~start, xend = ~end,
    y = ~y, yend = ~y,
    color = ~resource,
    mode = "lines",
```



```
colors = "Greys",
line = list(width = 20),
showlegend = F,
hoverinfo = "text",
text = ~ paste(
  "任务: ", task, "<br>",
  "启动时间: ", start, "<br>",
  "周期: ", duration, "天<br>",
  "资源: ", resource
)
) %>%
layout(
  xaxis = list(
    showgrid = F,
    title = list(text = ""))
),
yaxis = list(
  showgrid = F,
  title = list(text = ""),
  tickmode = "array",
  tickvals = 1:nrow(df),
  ticktext = unique(df$task),
  domain = c(0, 0.9)
),
annotations = list(
  list(
    xref = "paper", yref = "paper",
    x = 0.80, y = 0.1,
    text = paste0(
      "项目周期: ", sum(df$duration), " 天<br>",
      "资源类型: ", length(unique(df$resource)), " 个<br>"
    ),
    font = list(size = 12),
    ax = 0, ay = 0,
    align = "left"
),
  list(
    xref = "paper", yref = "paper",
    x = 0.1, y = 1,
    xanchor = "left",
    text = "项目资源管理",
    font = list(size = 20),
    ax = 0, ay = 0,
    align = "left",
    showarrow = FALSE
)
```



## 12.17 帕雷托图

帕雷托图 20/80 法则

```
# 数据来自 https://github.com/plotly/datasets
dat <- data.frame(
  complaint = c(
    "Small portions", "Overpriced",
    "Wait time", "Food is tasteless", "No atmosphere", "Not clean",
    "Too noisy", "Food is too salty", "Unfriendly staff", "Food not fresh"
  ),
  count = c( 621L, 789L, 109L, 65L, 45L, 30L, 27L, 15L, 12L, 9L)
)

dat <- dat[order(-dat$count), ] %>%
  transform(cumulative = round(100 * cumsum(count) / sum(count), digits = 2))

# complaint 按 count 降序排列
dat$complaint <- reorder(x = dat$complaint, x = dat$count, FUN = function(x) 1/(1 + x))

plot_ly(data = dat) %>%
  add_bars(
    x = ~complaint, y = ~count,
    showlegend = F, color = I("gray60")
  ) %>%
  add_lines(
    x = ~complaint, y = ~cumulative, yaxis = "y2",
    showlegend = F, color = I("gray40")
  ) %>%
  layout(
    yaxis2 = list(
      tickfont = list(color = "black"),
      overlaying = "y",
      side = "right",
      title = "累积百分比 (%)",
      showgrid = F
    ),
    xaxis = list(title = "投诉类型", showgrid = F, showline = F),
    yaxis = list(title = "数量", showgrid = F, showline = F)
  )
```

提示

reorder() 对 complaint 按照降序还是升序由 FUN 函数的单调性决定，单调增对应升序，单调减对应降序

## 12.18 时间线

```
library(vistime)

pres <- data.frame(
  Position = rep(c("President", "Vice"), each = 3),
  Name = c("Washington", rep(c("Adams", "Jefferson"), 2), "Burr"),
  start = c("1789-03-29", "1797-02-03", "1801-02-03"),
  end = c("1797-02-03", "1801-02-03", "1809-02-03"),
  color = c("#cbb69d", "#603913", "#c69c6e"),
  fontcolor = c("black", "white", "black")
)

vistime(pres, col.event = "Position", col.group = "Name")
```

## 12.19 漏斗图

```
dat <- data.frame(
  category = c("访问", "下载", "潜客", "报价", "下单"),
  value = c(39, 27.4, 20.6, 11, 2)
) %>%
  transform(percent = value / cumsum(value))

plot_ly(data = dat) %>%
  add_trace(
    type = "funnel",
    y = ~category,
    x = ~value,
    color = ~category,
    colors = "Set2",
    text = ~ paste0(value, "<br>", sprintf("%.2f%%", 100*percent)) ,
    hoverinfo = "text",
    showlegend = FALSE
) %>%
  layout(yaxis = list(
    categoryarray = ~category,
    title = ""
```



```
 )))
plotly::plot_ly(data = dat) %>%
  plotly::add_trace(
    type = "funnel",
    y = ~category,
    x = ~value,
    marker = list(color = RColorBrewer::brewer.pal(n = 5, name = "Set2")),
    textposition = "auto",
    textinfo = "value+percent previous",
    hoverinfo = "none"
  ) %>%
  plotly::layout(yaxis = list(categoryarray = ~category, title = ""))
```

## 12.20 雷达图

```
plot_ly(
  type = "scatterpolar", mode = "markers", fill = "toself"
) %>%
  add_trace(
    r = c(39, 28, 8, 7, 28, 39), color = I("gray40"),
    theta = c("数学", "物理", "化学", "英语", "生物", "数学"),
    name = "学生 A"
  ) %>%
  add_trace(
    r = c(1.5, 10, 39, 31, 15, 1.5), color = I("gray80"),
    theta = c("数学", "物理", "化学", "英语", "生物", "数学"),
    name = "学生 B"
  ) %>%
  layout(
    polar = list(
      radialaxis = list(
        visible = T,
        range = c(0, 50)
      )
    )
  )
```

## 12.21 瀑布图



```
library(plotly)
library(dplyr)

dat <- data.frame(
  x = c(
    "销售", "咨询", "净收入",
    "购买", "其他费用", "税前利润"
  ),
  y = c(60, 80, 10, -40, -20, 0),
  measure = c(
    "relative", "relative", "relative",
    "relative", "relative", "total"
  )
) %>%
  mutate(text = case_when(
    y > 0 ~ paste0("+", y),
    y == 0 ~ "",
    y < 0 ~ as.character(y)
  )) %>%
  mutate(x = factor(x, levels = c(
    "销售", "咨询", "净收入",
    "购买", "其他费用", "税前利润"
  )))

n_rows <- nrow(dat)
dat[nrow(dat), "text"] <- "累计"

# measure 取值为 'relative'/'total'/'absolute'
plotly::plot_ly(dat,
  x = ~x, y = ~y, measure = ~measure, type = "waterfall",
  text = ~text, textposition = "outside",
  name = "收支", hoverinfo = "final",
  connector = list(line = list(color = "gray")),
  increasing = list(marker = list(color = "#66C2A5")),
  decreasing = list(marker = list(color = "#FC8D62")),
  totals = list(marker = list(color = "#8DA0CB"))
) %>%
  plotly::layout(
    title = "2018 年收支状态",
    xaxis = list(title = "业务"),
    yaxis = list(title = "金额"),
    showlegend = FALSE
)
```



## 12.22 树状图

plotly 绘制 treemap 和 sunburst 图比较复杂，接口不友好，[plotme](#) 正好弥补不足。

## 12.23 旭日图

[plotme](#)

## 12.24 调色板

```
plot_ly(iris,
  x = ~Petal.Length, y = ~Petal.Width,
  mode = "markers", type = "scatter",
  color = ~ Sepal.Length > 6, colors = c("#132B43", "#56B1F7")
)
plot_ly(iris,
  x = ~Petal.Length, y = ~Petal.Width, color = ~ Sepal.Length > 6,
  mode = "markers", type = "scatter"
)

plot_ly(iris,
  x = ~Petal.Length, y = ~Petal.Width, color = ~ Sepal.Length > 6,
  mode = "markers", type = "scatter", colors = "Set2"
)

plot_ly(iris,
  x = ~Petal.Length, y = ~Petal.Width, color = ~ Sepal.Length > 6,
  mode = "markers", type = "scatter", colors = "Set1"
)
```

构造 20 个类别超出 Set1 调色板的范围，会触发警告说 Set1 没有那么多色块，但还是返回足够多的色块，也可以使用 viridis、plasma、magma 或 inferno 调色板

```
dat <- data.frame(
  dt = rep(seq(
    from = as.Date("2021-01-01"),
    to = as.Date("2021-01-31"), by = "day"
  ), each = 20),
  bu = rep(LETTERS[1:20], 31),
  qv = rbinom(n = 20 * 31, size = 10000, prob = runif(20 * 31))
)
# viridis
plot_ly(dat,
```

```
x = ~dt, y = ~qv, color = ~bu,  
mode = "markers", type = "scatter", colors = "viridis"  
)
```

## 12.25 导出静态图形

orca (Open-source Report Creator App) 软件针对 plotly.js 库渲染的图形具有很强的导出功能，[安装 orca](#) 后，`plotly::orca()` 函数可以将基于 `htmlwidgets` 的 `plotly` 图形对象导出为 PNG、PDF 和 SVG 等格式的高质量静态图片。

```
p <- plot_ly(x = 1:10, y = 1:10, color = 1:10)  
orca(p, "plot.svg")
```

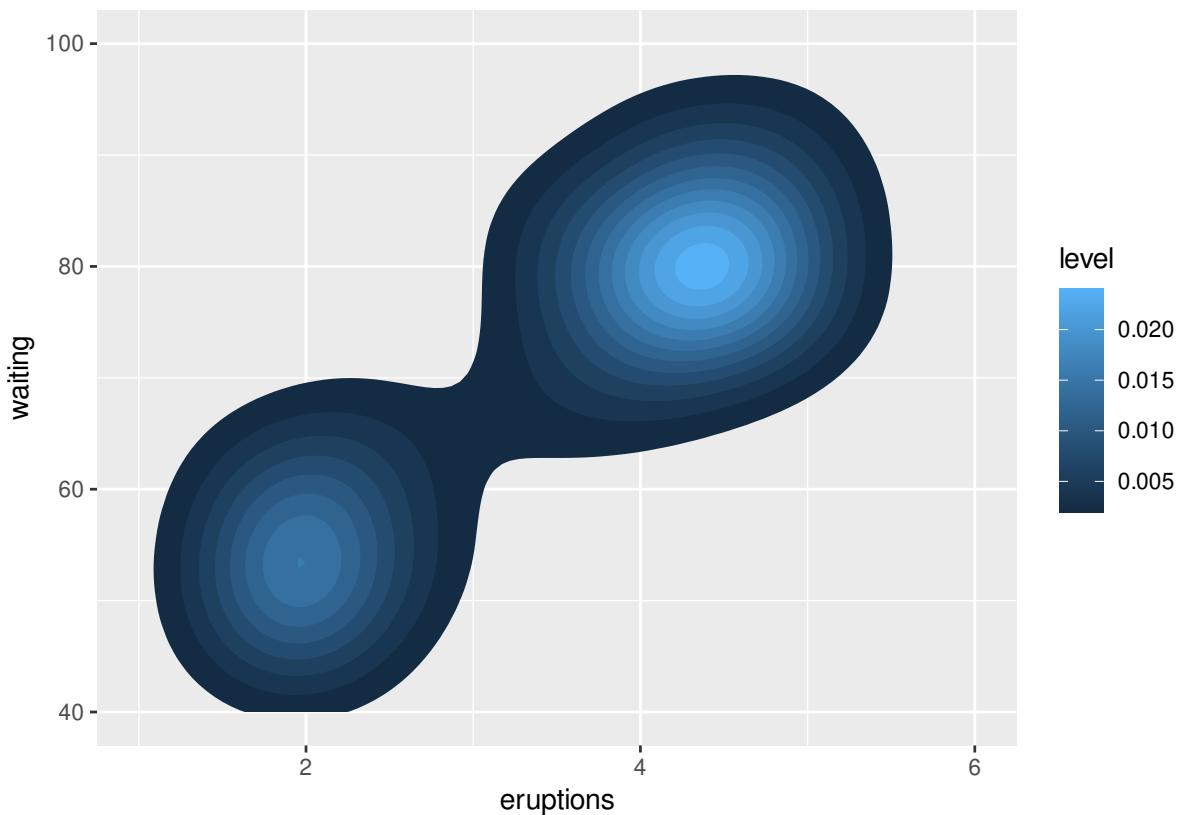
## 12.26 静态图形转交互图形

函数 `ggplotly()` 将 `ggplot` 对象转化为交互式 `plotly` 对象

```
gg <- ggplot(faithful, aes(x = eruptions, y = waiting)) +  
  stat_density_2d(aes(fill = ..level..), geom = "polygon") +  
  xlim(1, 6) +  
  ylim(40, 100)
```

静态图形

```
gg
```



转化为 plotly 对象

```
ggplotly(gg)
```

添加动态点的注释，比如点横纵坐标、坐标文本，整个注释标签的样式（如背景色）

```
ggplotly(gg, dynamicTicks = "y") %>%
  style(., hoveron = "points", hoverinfo = "x+y+text",
        hoverlabel = list(bgcolor = "white"))
```

## 12.27 地图 II

**leaflet** 包制作地图，斐济是太平洋上的一个岛国，处于板块交界处，经常发生地震，如下图所示，展示 1964 年来 1000 次震级大于 4 级的地震活动。

```
library(leaflet)
data(quakes)
# Pop 提示
quakes$popup_text <- lapply(paste(
  "编号:", "<strong>", quakes$stations, "</strong>", "<br>",
  "震深:", quakes$depth, "<br>",
  "震级:", quakes$mag
), htmltools::HTML)
# 构造调色板
pal <- colorBin("Spectral", bins = pretty(quakes$mag), reverse = TRUE)
```

```
p <- leaflet(quakes) |>
  addProviderTiles(providers$CartoDB.Positron) |>
  addCircles(lng = ~long, lat = ~lat, color = ~ pal(mag), label = ~popup_text) |>
  addLegend("bottomright",
    pal = pal, values = ~mag,
    title = "地震震级"
  ) |>
  addScaleBar(position = c("bottomleft"))

p
```

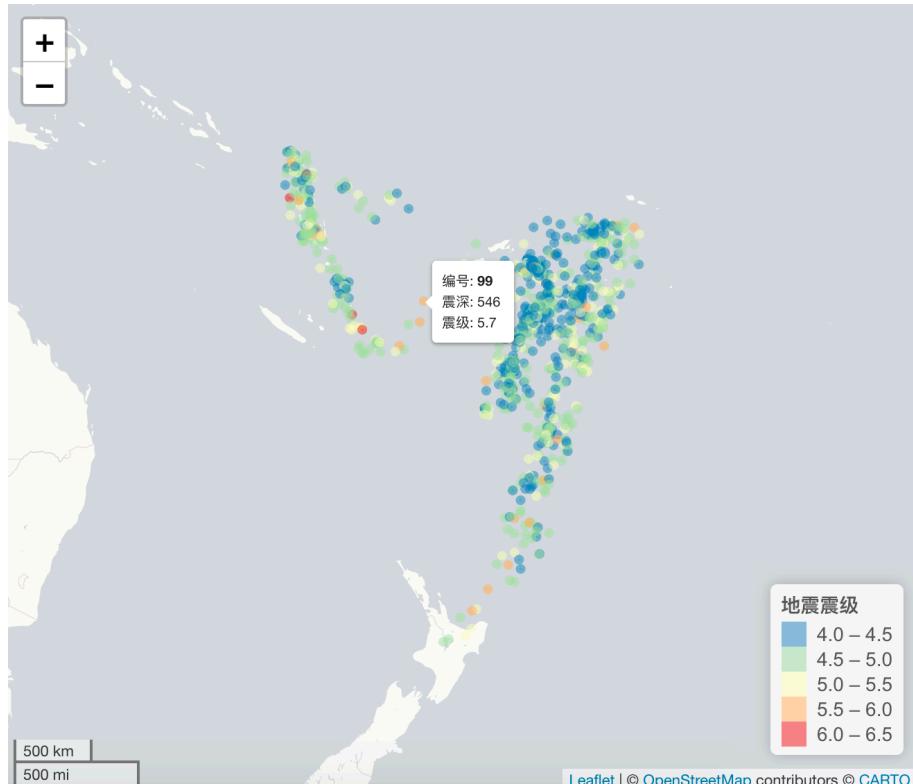


图 12.2: 斐济地震带

将上面的绘图部分保存为独立的 HTML 网页文件

```
library(htmlwidgets)
# p 就是绘图部分的数据对象
saveWidget(p, "fiji-map.html", selfcontained = T)
```

```
library(leaflet)
library(leaflet.extras)

quakes |>
  leaflet() |>
  addTiles() |>
  addProviderTiles(providers$OpenStreetMap.DE) |>
  addHeatmap(
    lng = ~long, lat = ~lat, intensity = ~mag,
```

```
max = 100, radius = 20, blur = 10
)
```

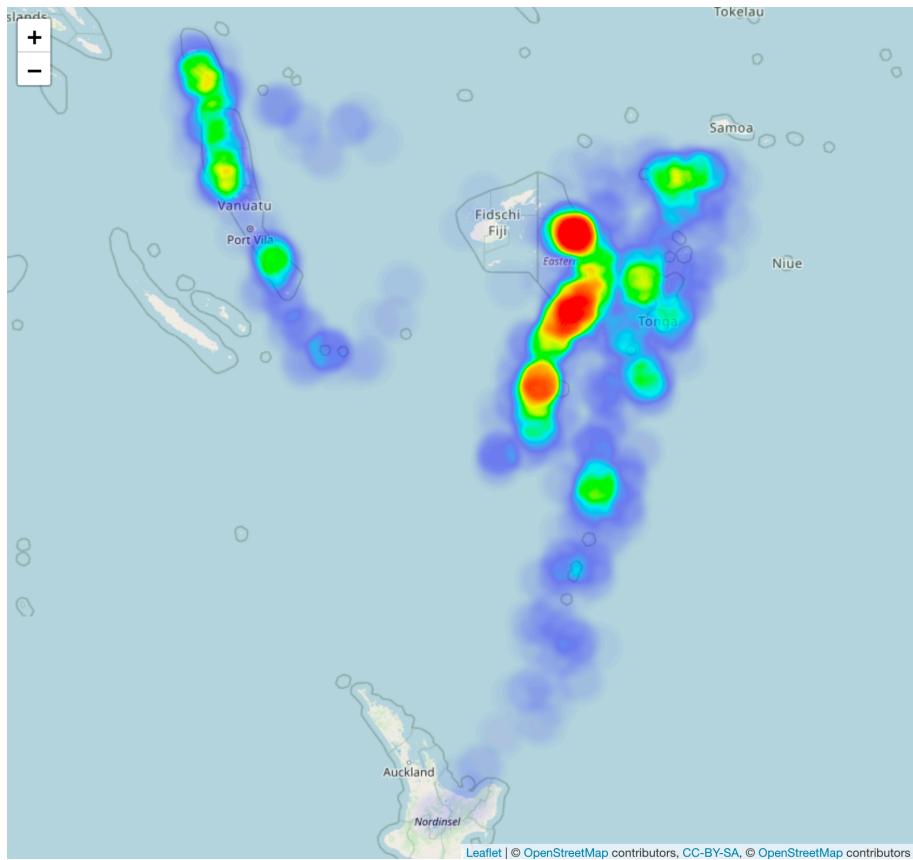


图 12.3: 斐济地震带热力图

leafletCN 提供汉化

```
# 地图默认放大倍数
zoom      <- 4
# 地图可以放大的倍数区间
minZoom   <- 1
maxZoom   <- 18

library(leaflet)
library(leafletCN)
library(maptools)
library(leaflet.extras)

# 热力图 heatmap
leaflet(res, options = leafletOptions(minZoom = minZoom, maxZoom = maxZoom)) |>
  amap() |>
  # setView(lng = mean(data$long), lat = mean(data$lat), zoom = zoom) |>
  setView(lng = 109, lat = 38, zoom = 4) |>
  addHeatmap(
```



```
  lng = ~long2, lat = ~lat2, intensity = ~uv, max = max(res$uv),
  blur = blur, minOpacity = minOpacity, radius = radius
)

quakes$popup_text <- lapply(paste(
  "编号:", "<strong>", quakes$stations, "</strong>", "<br>",
  "震深:", quakes$depth, "<br>",
  "震级:", quakes$mag
), htmltools:::HTML)
# 构造调色板
pal <- colorBin("Spectral", bins = pretty(quakes$mag), reverse = TRUE)

leaflet(quakes) |>
  addProviderTiles(providers$CartoDB.Positron) |>
  addCircles(
    lng = ~long, lat = ~lat,
    color = ~ pal(mag), label = ~popup_text
  ) |>
  setView(178, -20, 5) |>
  addHeatmap(
    lng = ~long, lat = ~lat, intensity = ~mag,
    blur = 20, max = 0.05, radius = 15
  ) |>
  addLegend("bottomright",
    pal = pal, values = ~mag,
    title = "地震震级"
  ) |>
  addScaleBar(position = c("bottomleft"))
```

## 12.28 动画

```
# https://d.cosx.org/d/422311
library(echarts4r)

data("gapminder", package = "gapminder")

titles <- lapply(unique(gapminder$year), function(x) {
  list(
    text = "Gapminder",
    left = "center"
  )
})
```



```
years <- lapply(unique(gapminder$year), function(x) {
  list(
    subtext = x,
    left = "center",
    top = "center",
    z = 0,
    subtextStyle = list(
      fontSize = 100,
      color = "rgb(170, 170, 170, 0.5)",
      fontWeight = "bolder"
    )
  )
})
```

# 添加一列颜色，各大洲和颜色的对应关系可自定义，调整 `levels` 或 `labels` 里面的顺序即可，也可不指定 `levels`，  
gapminder <- within(gapminder, {  
 color <- factor(  
 continent,  
 levels = c("Asia", "Africa", "Americas", "Europe", "Oceania"),  
 labels = RColorBrewer::brewer.pal(n = 5, name = "Spectral")  
 )  
})

```
gapminder |>  
  group_by(year) |>  
  e_charts(x = gdpPercap, timeline = TRUE) |>  
  e_scatter(  
    serie = lifeExp, size = pop, bind = country,  
    symbol_size = 5, name = ""  
  ) |>  
  e_add("itemStyle", color) |>  
  e_y_axis(  
    min = 20, max = 85, nameGap = 30,  
    name = "Life Exp", nameLocation = "center"  
  ) |>  
  e_x_axis(  
    type = "log", min = 100, max = 100000,  
    nameGap = 30, name = "GDP / Cap", nameLocation = "center"  
  ) |>  
  e_timeline_serie(title = titles) |>  
  e_timeline_serie(title = years, index = 2) |>  
  e_timeline_opts(playInterval = 1000) |>  
  e_grid(bottom = 100) |>  
  e_tooltip()
```

## 12.29 网络图

gephi 探索和可视化网络图 GraphViz

```
# library(igraph)
```

### 12.29.1 networkD3

networkD3 D3 非常适合绘制网络图，如网络、树状、桑基图

```
library(networkD3)
data(MisLinks, MisNodes) # 加载数据
head(MisLinks) # 边
```

```
##   source target value
## 1     1      0     1
## 2     2      0     8
## 3     3      0    10
## 4     3      2     6
## 5     4      0     1
## 6     5      0     1
```

```
head(MisNodes) # 节点
```

```
##           name group size
## 1      Myriel    1    15
## 2    Napoleon    1    20
## 3  Mlle.Baptistine    1    23
## 4  Mme.Magloire    1    30
## 5 CountessdeLo    1    11
## 6    Geborand    1     9
```

构造网络图

```
forceNetwork(
  Links = MisLinks, Nodes = MisNodes, Source = "source",
  Target = "target", Value = "value", NodeID = "name",
  Group = "group", opacity = 0.4
)
```

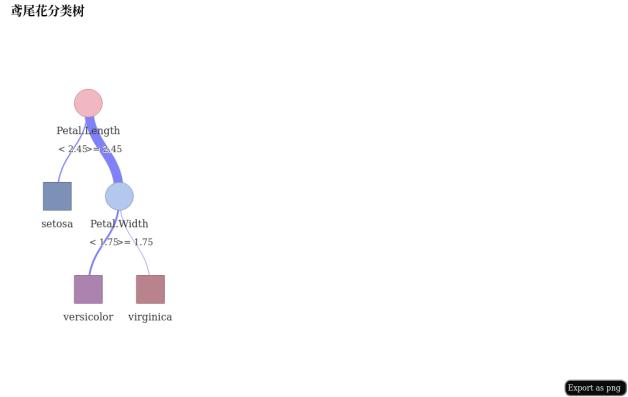
### 12.29.2 visNetwork

visNetwork 使用 vis-network.js 库绘制网络关系图 <https://datastorm-open.github.io/visNetwork>

```
library(visNetwork)
```

调用函数 visTree() 可视化分类模型结果

```
library(rpart)
library(sparkline) # 函数 visTree 需要导入 sparkline 包
res <- rpart(Species~., data=iris)
visTree(res, main = "鸢尾花分类树", width = "100%")
```



节点、边的属性都可以映射数据指标

### 12.29.3 r2d3

D3 是非常流行的 JavaScript 库, r2d3 提供了 R 接口

```
library(r2d3)
```

更加具体的使用介绍, 一个复杂的案例, 如何从简单配置过来, 以条形图为例, D3 是一个相当强大且成熟的库, 提供的案例功能要覆盖 plotly

r2d3 提供了两个样例 JS 库 baranims.js 和 barchart.js

```
list.files(system.file("examples/", package = "r2d3"))

## [1] "baranims.js" "barchart.js"

library(r2d3)
r2d3(
  data = c(0.3, 0.6, 0.8, 0.95, 0.40, 0.20),
  script = system.file("examples/barchart.js", package = "r2d3")
)
```

```
r2d3(  
  data = c(0.3, 0.6, 0.8, 0.95, 0.40, 0.20),  
  script = system.file("examples/baranims.js", package = "r2d3")  
)
```

## 12.30 运行环境

```
sessionInfo()  
  
## R version 4.2.0 (2022-04-22)  
## Platform: x86_64-pc-linux-gnu (64-bit)  
## Running under: Ubuntu 20.04.4 LTS  
##  
## Matrix products: default  
## BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0  
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0  
##  
## locale:  
## [1] LC_CTYPE=en_US.UTF-8          LC_NUMERIC=C  
## [3] LC_TIME=en_US.UTF-8          LC_COLLATE=en_US.UTF-8  
## [5] LC_MONETARY=en_US.UTF-8       LC_MESSAGES=en_US.UTF-8  
## [7] LC_PAPER=en_US.UTF-8          LC_NAME=C  
## [9] LC_ADDRESS=C                 LC_TELEPHONE=C  
## [11] LC_MEASUREMENT=en_US.UTF-8   LC_IDENTIFICATION=C  
##  
## attached base packages:  
## [1] stats      graphics   grDevices  utils      datasets   methods    base  
##  
## other attached packages:  
## [1] sparkline_2.0     rpart_4.1.16    visNetwork_2.1.0  networkD3_0.4  
## [5] r2d3_0.2.6       plotly_4.10.0   ggplot2_3.3.6  
##  
## loaded via a namespace (and not attached):  
## [1] tidyselect_1.1.2  xfun_0.31      purrr_0.3.4      colorspace_2.0-3  
## [5] vctrs_0.4.1      generics_0.1.2   htmltools_0.5.2   viridisLite_0.4.0  
## [9] yaml_2.3.5       utf8_1.2.2     rlang_1.0.2      isoband_0.2.5  
## [13] pillar_1.7.0     glue_1.6.2     withr_2.5.0     DBI_1.1.2  
## [17] lifecycle_1.0.1   stringr_1.4.0   munsell_0.5.0    gtable_0.3.0  
## [21] htmlwidgets_1.5.4 evaluate_0.15  labeling_0.4.2   knitr_1.39  
## [25] callr_3.7.0     fastmap_1.1.0  ps_1.7.0       curl_4.3.2  
## [29] fansi_1.0.3     scales_1.2.0    webshot_0.5.3   jsonlite_1.8.0  
## [33] sysfonts_0.8.8   farver_2.1.0   png_0.1-7     digest_0.6.29  
## [37] stringi_1.7.6   processx_3.5.3 bookdown_0.26   dplyr_1.0.9  
## [41] grid_4.2.0       cli_3.3.0     tools_4.2.0    magrittr_2.0.3
```

## [45] lazyeval\_0.2.2 tibble\_3.1.7 crayon\_1.5.1 tidyR\_1.2.0  
## [49] pkgconfig\_2.0.3 ellipsis\_0.3.2 MASS\_7.3-57 data.table\_1.14.2  
## [53] assertthat\_0.2.1 rmarkdown\_2.14 httr\_1.4.3 rstudioapi\_0.13  
## [57] R6\_2.5.1 igraph\_1.3.1 compiler\_4.2.0

©

## 第十三章 数值优化

R 语言提供了相当多的优化求解器，比较完整的概览见[优化视图](#)。本章介绍一些常用的优化算法及其 R 实现，涵盖线性规划、整数规划、二次规划、非线性规划等。

商业优化求解器的能力都覆盖非线性规划 (NLP)，线性 (LP)、二次 (QP) 和锥规划 (SOCP)，混合整数线性规划 (MILP)，多目标优化，最小二乘和方程求解。此外，还有很多文档介绍，[LINGO](#)提供[用户手册](#)，[Matlab 优化工具箱](#)提供[Optimization](#)工具箱使用指南，[MOSEK](#) (<https://www.mosek.com/>) 提供[MOSEK 建模食谱](#)，[LocalSolver](#) 提供[基本使用手册](#)，[Gurobi](#) 提供[Gurobi 参考手册](#)，[CPLEX Optimization Studio](#)。

开源社区有不少工具，也能求解常见的优化问题，如 Julia 的 [JuMP](#) (<https://jump.dev/>)，[Octave](#) (<https://www.gnu.org/software/octave/>) 内置的优化函数，Python 模块 [SciPy](#) 提供[Optimization](#) 优化求解器，[cvxopt](#) 凸优化求解器，主要基于内点法，提供 Julia、Python、Matlab 接口，算法介绍见[锥优化](#) 机器学习优化。课程见[Optimization for Machine Learning](#)，书籍见[Convex Optimization](#)，相关综述见[Convex Optimization: Algorithms and Complexity](#)。

Berwin A. Turlach 开发的 [quadprog](#) 主要用于求解二次规划问题。Anqi Fu 开发的 [CVXR](#) 可解很多凸优化问题 [Fu et al., 2020]，详见网站 <https://cvxr.rbind.io/>，Jelmer Ypma 开发的 [nloptr](#) 可解无约束和有约束的非线性规划问题 [Ypma, 2020]，[GParato](#) 求解多目标优化问题，帕雷托前沿优化和估计 [Binois and Picheny, 2019]。[igraph](#) 可以用来解决最短路径、最大网络流、最小生成树等图优化相关的问题。[https://palomar.home.ece.ust.hk/MAFS6010R\\_lectures/Rsession\\_solvers.html](https://palomar.home.ece.ust.hk/MAFS6010R_lectures/Rsession_solvers.html) 提供了一般的求解器介绍。ROI 包力图统一各个求解器的调用接口，打造一个优化算法的基础设施平台。Theußl et al. [2020] 详细介绍了目前优化算法发展情况及 R 社区提供的优化能力。GA 包实现了遗传算法，支持连续和离散的空间搜索，可以并行 [Scrucca, 2013, 2017]，是求解 TSP 问题的重要方法。NMOF 包实现了差分进化、遗传算法、粒子群算法、模拟退火算法等启发式优化算法，还提供网格搜索和贪婪搜索工具，Gilli et al. [2019] 提供了详细的介绍。Nash [2014] 总结了 R 语言环境下最优化问题的最佳实践。RcppEnsmalleen 数值优化通用标准的优化方法，前沿最新的优化方法，包含小批量/全批量梯度下降技术、无梯度优化器，约束优化技术。RcppNumerical 无约束数值优化，一维/多维数值积分。

谷歌开源的运筹优化工具 [or-tools](#) 提供了约束优化、线性优化、混合整数优化、装箱和背包算法、TSP (Traveling Salesman Problem)、VRP (Vehicle Routing Problem)、图算法 (最短路径、最小成本流、最大流等) 等算法和求解器。「运筹 OR 帷幄」社区开源的[线性规划](#)一书值得一看。

```
# 加载 ROI 时不要自动加载插件
Sys.setenv(ROI_LOAD_PLUGINS = FALSE)
library(lpSolve)      # 线性规划求解器
library(ROI)          # 优化工具箱
library(ROI.plugin.alabama) # 注册 alabama 求解非线性规划
library(ROI.plugin.nloptr) # 注册 nloptr 求解非线性规划
```

表 13.1: ROI 插件按优化问题分类

	Linear	Quadratic	Conic	Functional
No				
Box				optimx
Linear	clp*, cbc*+, ipop, glpk*+, quadprog*, lpsolve*+, qpoases msbinlp*+, symphony*+			
Quadratic		cplex+, gurobi*+, mosek*+, neos+		
Conic			ecos*+, scs*	
Functional				alabama, deoptim, nlminb, nloptr

\* 求解器受限于凸优化问题

+ 求解器可以处理整型约束

```
library(ROI.plugin.lpsolve) # 注册 lpsolve 求解线性规划
library(ROI.plugin.quadprog) # 注册 quadprog 求解二次规划
library(ROI.plugin.scs)      # 注册 scs 求解凸锥规划

library(lattice)      # 图形绘制
library(kernlab)      # 优化问题和机器学习的关系

library(rootSolve)      # 非线性方程
library(BB)            # 非线性方程组
library(deSolve)        # ODE 常微分方程
library(scatterplot3d)  # 三维曲线图

library(shape)
library(ReacTran)       # PDE 偏微分方程
library(PBSddesolve)    # DAE 延迟微分方程

library(nlme)           # 混合效应模型
# library(nlmeODE)      # ODE 应用于混合效应模型
# library(Sim.DiffProc)  # SDE 随机微分方程 种群 ODE 建模
# library(nlmixr)        # Population ODE modeling
```

表 13.1 对目前的优化器按优化问题做了分类

### 13.1 线性规划

`clpAPI` 线性规划求解器。`glpk` 的两个 R 接口 – `glpkAPI` 和 `Rglpk` 提供线性规划和混合整数规划的求解能力。`lp_solve` 的两个 R 接口 – `lpSolveAPI` 和 `lpSolve` 也提供类似的能力。`ompr` 求解混合整数线性规划问题。

举个例子, 如下

$$\begin{aligned} & \min_x \quad -6x_1 - 5x_2 \\ s.t. \quad & \begin{cases} x_1 + 4x_2 \leq 16 \\ 6x_1 + 4x_2 \leq 28 \\ 2x_1 - 5x_2 \leq 6 \end{cases} \end{aligned}$$

写成矩阵形式

$$\begin{aligned} & \min_x \quad \begin{bmatrix} -6 \\ -5 \end{bmatrix}^T x \\ s.t. \quad & \begin{cases} \begin{bmatrix} 1 & 4 \\ 6 & 4 \\ 2 & -5 \end{bmatrix} x \leq \begin{bmatrix} 16 \\ 28 \\ 6 \end{bmatrix} \end{cases} \end{aligned}$$

对应成 R 代码如下

```
# lpSolve 添加约束条件
library(lpSolve)
# 目标
f.obj <- c(-6, -5)
# 约束
f.con <- matrix(c(1, 4, 6, 4, 2, -5), nrow = 3, byrow = TRUE)
# 方向
f.dir <- c("<=", "<=", "<=")
# 右手边
f.rhs <- c(16, 28, 6)
res <- lp("min", f.obj, f.con, f.dir, f.rhs)
res$objval
```

```
## [1] -31.4
```

```
res$solution
```

```
## [1] 2.4 3.4
```

## 13.2 整数规划

### 13.2.1 一般整数规划

$$\begin{aligned} & \max_x \quad 0.2x_1 + 0.6x_2 \\ s.t. \quad & \begin{cases} 5x_1 + 3x_2 \leq 250 \\ -3x_1 + 2x_2 \leq 4 \\ x_1, x_2 \geq 0, \quad x_1, x_2 \in \mathbb{Z} \end{cases} \end{aligned}$$

```
# 目标
f.obj <- c(0.2, 0.6)
# 约束
f.con <- matrix(c(5, 3, -3, 2), nrow = 2, byrow = TRUE)
# 方向
f.dir <- c("<=", "<=")
# 右手边
f.rhs <- c(250, 4)
# 限制两个变量都是整数
res <- lp("max", f.obj, f.con, f.dir, f.rhs, int.vec=1:2)
res$objval
```

```
## [1] 29.2
```

```
res$solution
```

```
## [1] 26 40
```

### 13.2.2 0-1整数规划

$$\begin{aligned} & \max_x \quad 0.2x_1 + 0.6x_2 \\ s.t. \quad & \begin{cases} 5x_1 + 3x_2 \leq 250 \\ -3x_1 + 2x_2 \leq 4 \\ x_1, x_2 \in \{0, 1\} \end{cases} \end{aligned}$$

```
# 目标
f.obj <- c(0.2, 0.6)
# 约束
f.con <- matrix(c(5, 3, -3, 2), nrow = 2, byrow = TRUE)
# 方向
f.dir <- c("<=", "<=")
# 右手边
f.rhs <- c(250, 4)
# 限制两个变量都是0-1整数
res <- lp("max", f.obj, f.con, f.dir, f.rhs, int.vec=1:2, all.bin = TRUE)
res$objval
```



```
## [1] 0.8
res$solution

## [1] 1 1
```

### 13.2.3 混合整数规划

`Rsymphony` 是混合整数规划求解器 `SYMPHONY` 的 R 语言接口<sup>1</sup>。

```
library(Rsymphony)
## Simple linear program.
## maximize: 2 x_1 + 4 x_2 + 3 x_3
## subject to: 3 x_1 + 4 x_2 + 2 x_3 <= 60
## 2 x_1 + x_2 + x_3 <= 40
## x_1 + 3 x_2 + 2 x_3 <= 80
## x_1, x_2, x_3 are non-negative real numbers

# 简单线性规划
obj <- c(2, 4, 3)
mat <- matrix(c(3, 2, 1, 4, 1, 3, 2, 1, 2), nrow = 3)
dir <- c("<=", "<=", "<=")
rhs <- c(60, 40, 80)
max <- TRUE
Rsymphony_solve_LP(obj, mat, dir, rhs, max = max)

# 混合整数规划
obj <- c(3, 1, 3)
mat <- matrix(c(-1, 0, 1, 2, 4, -3, 1, -3, 2), nrow = 3)
dir <- c("<=", "<=", "<=")
rhs <- c(4, 2, 3)
max <- TRUE
types <- c("I", "C", "I")
Rsymphony_solve_LP(obj, mat, dir, rhs, types = types, max = max)

# 有边界约束的混合整数规划
## Same as before but with bounds replaced by
## -Inf < x_1 <= 4
## 0 <= x_2 <= 100
## 2 <= x_3 < Inf
bounds <- list(
  lower = list(ind = c(1L, 3L), val = c(-Inf, 2)),
```

<sup>1</sup>以 MacOS 为例安装 `symphony` 软件

```
brew tap coin-or-tools/coinor
brew install symphony
```

```

    upper = list(ind = c(1L, 2L), val = c(4, 100))
)
Rsymphony_solve_LP(obj, mat, dir, rhs,
  types = types, max = max,
  bounds = bounds
)

```

一部分变量要求是整数

$$\begin{aligned} \max_x \quad & 3x_1 + 7x_2 - 12x_3 \\ \text{s.t.} \quad & \begin{cases} 5x_1 + 7x_2 + 2x_3 \leq 61 \\ 3x_1 + 2x_2 - 9x_3 \leq 35 \\ x_1 + 3x_2 + x_3 \leq 31 \\ x_1, x_2 \geq 0, \quad x_2, x_3 \in \mathbb{Z}, \quad x_3 \in [-10, 10] \end{cases} \end{aligned}$$

矩阵形式如下

$$\begin{aligned} \min_x \quad & \begin{bmatrix} 3 \\ 7 \\ -12 \end{bmatrix}^T x \\ \text{s.t.} \quad & \begin{bmatrix} 5 & 7 & 2 \\ 3 & 2 & -9 \\ 1 & 3 & 1 \end{bmatrix} x \leq \begin{bmatrix} 61 \\ 35 \\ 31 \end{bmatrix} \end{aligned}$$

```

op <- OP(
  objective = L_objective(c(3, 7, -12)),
  # 指定变量类型: 第1个变量是连续值, 第2、3个变量是整数
  types = c("C", "I", "I"),
  constraints = L_constraint(
    L = matrix(c(
      5, 7, 2,
      3, 2, -9,
      1, 3, 1
    ), ncol = 3, byrow = TRUE),
    dir = c("<=", "<=", "<="),
    rhs = c(61, 35, 31)
  ),
  # 添加约束: 第3个变量的下、上界分别是 -10 和 10
  bounds = V_bound(li = 3, ui = 3, lb = -10, ub = 10, nobj = 3),
  maximum = TRUE
)
op

## ROI Optimization Problem:
##
## Maximize a linear objective function of length 3 with

```



```

## - 1 continuous objective variable,
## - 2 integer objective variables,
##
## subject to
## - 3 constraints of type linear.
## - 1 lower and 1 upper non-standard variable bound.

res <- ROI_solve(op, solver = "lpsolve")
res$solution

## [1] 0.3333333 8.0000000 -2.0000000

res$objval

## [1] 81

```

## 13.3 二次规划

### 13.3.1 凸二次规划

在 R 中使用 **quadprog** [Turlach, 2019] 包求解二次规划<sup>2</sup>，**quadprogXT** 包用来求解带绝对值约束的二次规划，**pracma** [Borchers, 2021] 包提供 **quadprog()** 函数就是对 **quadprog** 包的 **solve.QP()** 进行封装，调用风格更像 Matlab。**quadprog** 包实现了 Goldfarb and Idnani (1982, 1983) 提出的对偶方法，主要用来求解带线性约束的严格凸二次规划问题。**quadprog** 求解的二次型的形式如下：

$$\min_b -d^\top b + \frac{1}{2} b^\top D b, \quad A^\top b \geq b_0$$

```
solve.QP(Dmat, dvec, Amat, bvec, meq = 0, factorized = FALSE)
```

参数 **Dmat**、**dvec**、**Amat**、**bvec** 分别对应二次规划问题中的  $D, d, A, b_0$ 。下面举个二次规划的具体例子

$$D = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, \quad d = (-3, 2), \quad A = \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 0 & -1 \end{bmatrix}, \quad b_0 = (2, -2, -3)$$

即目标函数

$$Q(x, y) = x^2 + y^2 - xy + 3x - 2y + 4$$

它的可行域如图13.1所示

```

plot(0, 0,
  xlim = c(-2, 5.5), ylim = c(-1, 3.5), type = "n",
  xlab = "x", ylab = "y", main = "Feasible Region"
)
polygon(c(2, 5, -1), c(0, 3, 3), border = TRUE, lwd = 2, col = "gray")

```

调用 **quadprog** 包的 **solve.QP()** 函数求解此二次规划问题

<sup>2</sup><https://rwalk.xyz/solving-quadratic-programs-with-rs-quadprog-package/>

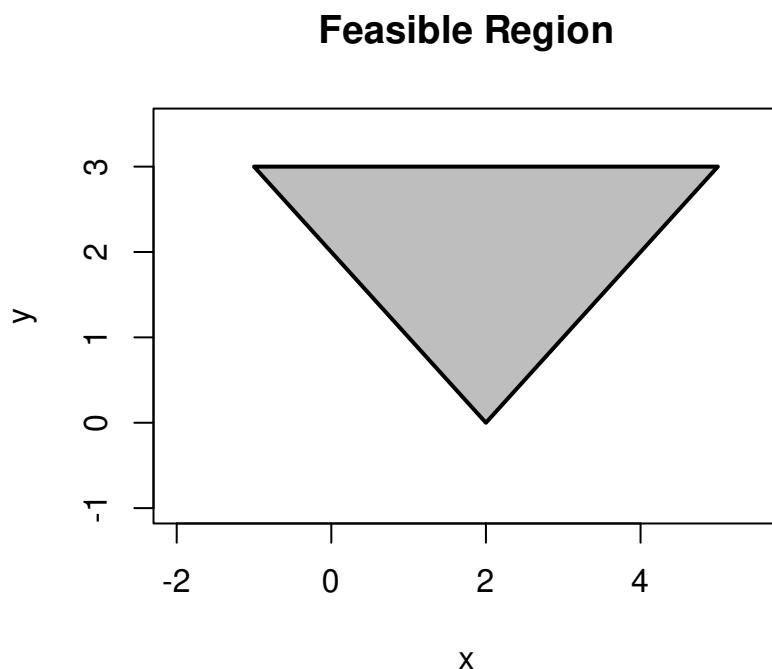


图 13.1: 可行域

```
library(quadprog)
Dmat <- matrix(c(2, -1, -1, 2), nrow = 2, byrow = TRUE)
dvec <- c(-3, 2)
A <- matrix(c(1, 1, -1, 1, 0, -1), ncol = 2, byrow = TRUE)
bvec <- c(2, -2, -3)
Amat <- t(A)
sol <- solve.QP(Dmat = Dmat, dvec = dvec, Amat = Amat, bvec = bvec)
sol

## $solution
## [1] 0.1666667 1.8333333
##
## $value
## [1] -0.08333333
##
## $unconstrained.solution
## [1] -1.3333333  0.3333333
##
## $iterations
## [1] 2 0
##
## $Lagrangian
## [1] 1.5 0.0 0.0
##
```



```
## $iact
## [1] 1
```

ROI 默认的二次规划的标准形式为  $\frac{1}{2}x^\top Qx + a^\top x$ , 在传递参数值的时候注意和上面的区别。

```
library(ROI)
op <- OP(
  objective = Q_objective(Q = Dmat, L = -dvec),
  constraints = L_constraint(A, rep(">=", 3), bvec),
  maximum = FALSE # 默认求最小
)
nlp <- ROI_solve(op, solver = "nloptr.slsqp", start = c(1, 2))
nlp$objval
```

```
## [1] -0.08333333
```

```
nlp$solution
```

```
## [1] 0.1666667 1.8333333
```

对变量  $x$  添加整型约束, 原二次规划即变成混合整数二次规划 (Mixed Integer Quadratic Programming, 简称 MIQP)

```
# 目前开源的求解器都不能处理 MIQP 问题
op <- OP(
  objective = Q_objective(Q = Dmat, L = -dvec),
  constraints = L_constraint(A, rep(">=", 3), bvec),
  types = c("I", "C"),
  maximum = FALSE # 默认求最小
)
nlp <- ROI_solve(op, solver = "nloptr.slsqp", start = c(1, 2))
nlp$objval
nlp$solution
```

在可行域上画出等高线, 标记目标解的位置, 图中红点表示无约束下的解, 黄点表示线性约束下的解

```
qp_sol <- sol$solution # 二次规划的解
uc_sol <- sol$unconstrained.solution # 无约束情况下的解
# 画图
library(lattice)
x <- seq(-2, 5.5, length.out = 500)
y <- seq(-1, 3.5, length.out = 500)
grid <- expand.grid(x = x, y = y)
# 二次规划的目标函数
grid$z <- with(grid, x^2 + y^2 - x * y + 3 * x - 2 * y + 4)
levelplot(z ~ x * y, grid,
  cuts = 40,
  panel = function(...) {
    panel.levelplot(...)
    panel.polygon(c(2, 5, -1), c(0, 3, 3),
```

```
border = TRUE,  
lwd = 2, col = "transparent"  
)  
panel.points(  
  c(uc_sol[1], qp_sol[1]),  
  c(uc_sol[2], qp_sol[2]),  
  lwd = 5, col = c("red", "yellow"), pch = 19  
)  
,  
colorkey = TRUE,  
col.regions = terrain.colors(40)  
)
```

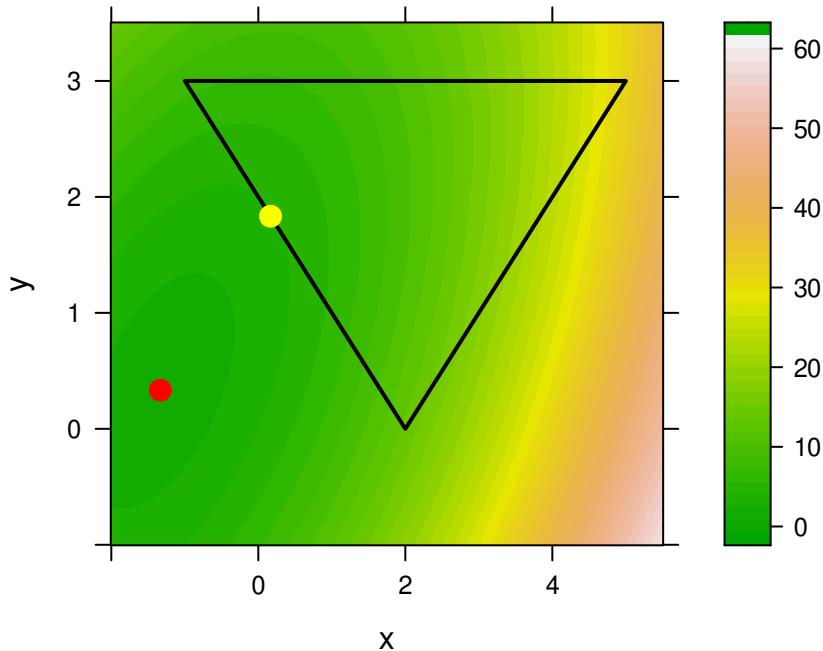


图 13.2: 无约束和有约束条件下的解

### 13.3.2 半正定二次优化

kernlab 提供基于核的机器学习方法，可用于分类、回归、聚类、异常检测、分位回归、降维等场景，包含支撑向量机、谱聚类、核 PCA、高斯过程和二次规划求解器，将优化方法用于机器学习，展示二者的关系。

R 包 kernlab 的函数 `ipop()` 实现内点法可以求解半正定的二次规划问题，对应到上面的例子，就是要求  $A \geq 0$ ，而 R 包 `quadprog` 只能求解正定的二次规划问题，即要求  $A > 0$ 。

以二分类问题为例，采用 SMO (Sequential Minimization Optimization) 求解器，将 SVM 的二次优化问题分解。

```
library(kernlab)
set.seed(123)
x <- rbind(matrix(rnorm(120), 60, 2), matrix(rnorm(120, mean = 3), 60, 2))
y <- matrix(c(rep(1, 60), rep(-1, 60)))
svm <- ksvm(x, y, type = "C-svc")
plot(svm, data = x)
```

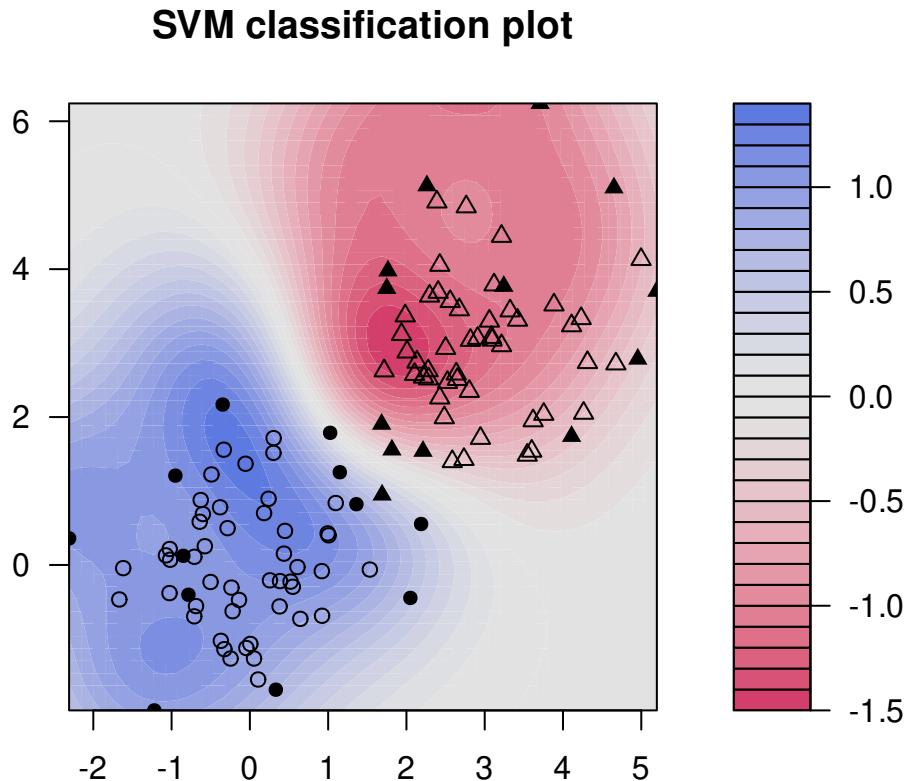


图 13.3: 二分类问题

## 13.4 非线性规划

开源的非线性优化求解器，推荐使用 nloptr，它支持全局优化，同时推荐 ROI，它有统一的接口函数。

### 13.4.1 一元非线性优化

下面考虑一个稍微复杂的一元函数优化问题，求复合函数的极值

$$g(x) = \int_0^x -\sqrt{t} \exp(-t^2) dt, \quad f(y) = \int_0^y g(s) \exp(-s) ds$$



```
g <- function(x) {
  integrate(function(t) {
    -sqrt(t) * exp(-t^2)
  }, lower = 0, upper = x)$value
}

f <- function(y) {
  integrate(function(s) {
    Vectorize(g, "x")(s) * exp(-s)
  }, lower = 0, upper = y)$value
}

optimize(f, interval = c(10, 100), maximum = FALSE)

## $minimum
## [1] 66.84459
##
## $objective
## [1] -0.3201572
```

提示

计算积分的时候，输入了一系列  $s$  值，参数是向量，而函数  $g$  只支持输入参数是单个值， $g(c(1, 2))$  会报错，因此上面对函数  $g()$  用了向量化函数 `Vectorize()` 操作。

```
g(1)

## [1] -0.453392

类似地，同时计算多个目标函数  $f(y)$  的值，也需要 Vectorize() 实现向量化操作。

Vectorize(f, "y")(c(1, 2))

## [1] -0.1103310 -0.2373865
```

## 13.4.2 多元非线性无约束优化

下面这些用来测试优化算法的函数来自[维基百科](#)

### 13.4.2.1 Himmelblau 函数

Himmelblau 函数是一个多模函数，常用于比较优化算法的优劣。

$$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

它在四个位置取得一样的极小值，分别是  $f(-3.7793, -3.2832) = 0$ ,  $f(-2.8051, 3.1313) = 0$ ,  $f(3, 2) = 0$ ,  $f(3.5844, -1.8481) = 0$ 。函数图像见图 13.4。

```
# 目标函数
fn <- function(x) {
```

```
(x[1]^2 + x[2] - 11)^2 + (x[1] + x[2]^2 - 7)^2
}

df <- expand.grid(
  x = seq(-5, 5, length = 101),
  y = seq(-5, 5, length = 101)
)

df$fnxy = apply(df, 1, fn)

library(lattice)
# 减少三维图形的边空
lattice.options(
  layout.widths = list(
    left.padding = list(x = -.6, units = "inches"),
    right.padding = list(x = -1.0, units = "inches")
  ),
  layout.heights = list(
    bottom.padding = list(x = -.8, units = "inches"),
    top.padding = list(x = -1.0, units = "inches")
  )
)

wireframe(
  data = df, fnxy ~ x * y,
  shade = TRUE, drape = FALSE,
  xlab = expression(x[1]),
  ylab = expression(x[2]),
  zlab = list(expression(italic(f) ~ group("(, list(x[1], x[2]), "))), rot = 90),
  scales = list(arrows = FALSE, col = "black"),
  par.settings = list(axis.line = list(col = "transparent")),
  screen = list(z = -240, x = -70, y = 0)
)

# 梯度函数
gr <- function(x) {
  numDeriv::grad(fn, c(x[1], x[2]))
}

optim(par = c(-1.2, 1), fn = fn, gr = gr, method = "BFGS")

## $par
## [1] -2.805118  3.131313
##
## $value
## [1] 2.069971e-27
```

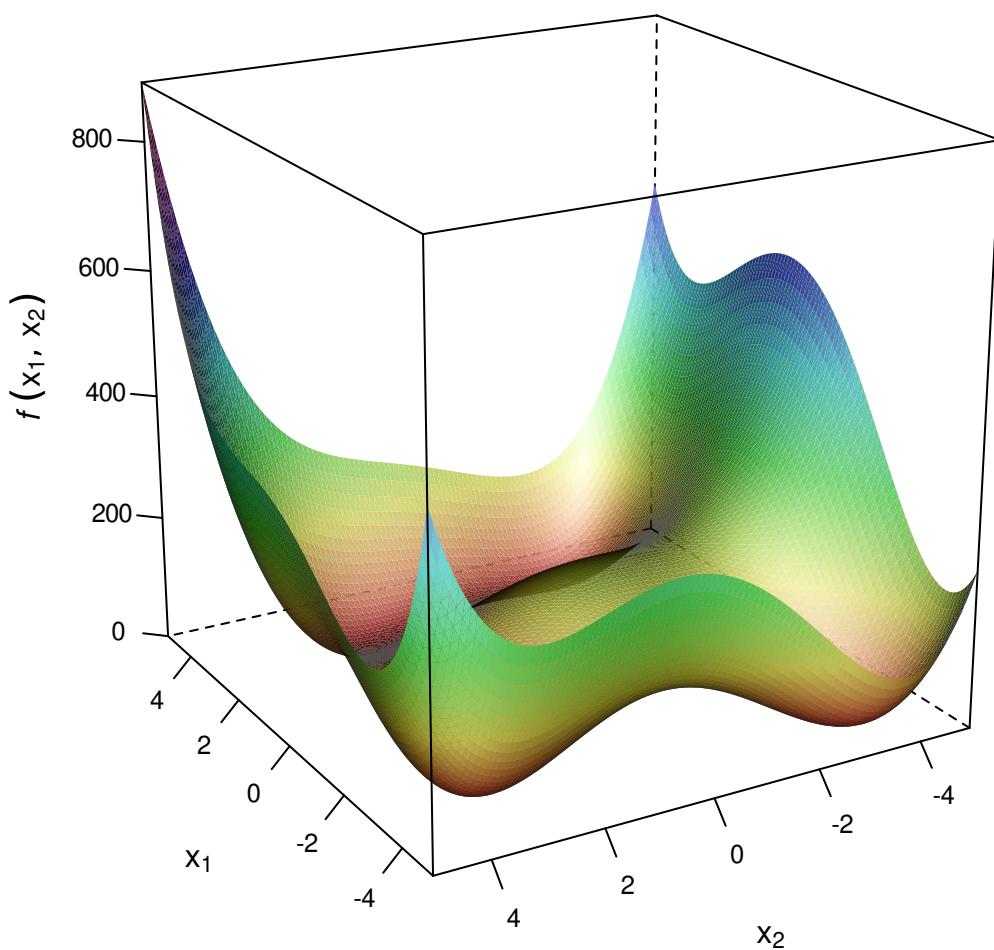


图 13.4: Himmelblau 函数图像



```
##  
## $counts  
## function gradient  
##      42      15  
##  
## $convergence  
## [1] 0  
##  
## $message  
## NULL
```

### 13.4.2.2 Peaks 函数

测试函数

$$f(x, y) = 3 * (1 - x) * e^{-x^2 - (y+1)^2} - 10 * \left(\frac{x}{5} - x^3 - y^5\right) * e^{-x^2 - y^2} - \frac{1}{3} * e^{-(x+1)^2 - y^2}$$

```
peaks <- expression(3*(1-x)*exp^(-x^2 - (y+1)^2) - 10*(x/5 - x^3 - y^5)*exp^(-x^2-y^2) - 1/3*exp^(-(x+1)^2 - y^2))

D(peaks, "x")

## -(3 * (1 - x) * (exp^(-x^2 - (y + 1)^2) * (log(exp) * (2 * x))) +
##   3 * exp^(-x^2 - (y + 1)^2) + (10 * (1/5 - 3 * x^2) * exp^(-x^2 -
##   y^2) - 10 * (x/5 - x^3 - y^5) * (exp^(-x^2 - y^2) * (log(exp) *
##   (2 * x))) - 1/3 * (exp^(-(x + 1)^2 - y^2) * (log(exp) *
##   (2 * (x + 1)))))

D(peaks, "y")

## -(3 * (1 - x) * (exp^(-x^2 - (y + 1)^2) * (log(exp) * (2 * (y +
##   1)))) - (10 * (x/5 - x^3 - y^5) * (exp^(-x^2 - y^2) * (log(exp) *
##   (2 * y))) + 10 * (5 * y^4) * exp^(-x^2 - y^2)) - 1/3 * (exp^(-(x +
##   1)^2 - y^2) * (log(exp) * (2 * y)))

library(Deriv)

Simplify(D(peaks, "x"))

## -(10 * ((0.2 - 3 * x^2)/exp^(x^2 + y^2)) + 3/exp^((1 + y)^2 +
##   x^2) + log(exp) * (x * (6 * ((1 - x)/exp^((1 + y)^2 + x^2)) -
##   20 * ((x * (0.2 - x^2) - y^5)/exp^(x^2 + y^2))) - 0.6666666666666667 *
##   ((1 + x)/exp^((1 + x)^2 + y^2)))

Simplify(D(peaks, "y"))

## -((6 * ((1 - x) * (1 + y)/exp^((1 + y)^2 + x^2)) - 0.6666666666666667 *
##   (y/exp^((1 + x)^2 + y^2))) * log(exp) - y * (20 * (log(exp) *
##   (x * (0.2 - x^2) - y^5)/exp^(x^2 + y^2)) + 50 * (y^3/exp^(x^2 +
##   y^2))))
```



```
fn <- function(x) {  
  3 * (1 - x[1])^2 * exp(-x[1]^2 - (x[2] + 1)^2) -  
  10 * (x[1] / 5 - x[1]^3 - x[2]^5) * exp(-x[1]^2 - x[2]^2) -  
  1 / 3 * exp(-(x[1] + 1)^2 - x[2]^2)  
}  
# 梯度函数  
gr <- function(x) {  
  numDeriv:::grad(fn, c(x[1], x[2]))  
}  
  
optim(par = c(-1.2, 1), fn = fn, gr = gr, method = "BFGS")
```

```
## $par  
## [1] -1.3473958  0.2045192  
##  
## $value  
## [1] -3.049849  
##  
## $counts  
## function gradient  
##       28       10  
##  
## $convergence  
## [1] 0  
##  
## $message  
## NULL
```

在 (-1.3473958, 0.2045192) 处取得极小值

```
df <- expand.grid(  
  x = seq(-3, 3, length = 101),  
  y = seq(-3, 3, length = 101)  
)  
  
df$fnxy = apply(df, 1, fn)  
  
library(lattice)  
wireframe(  
  data = df, fnxy ~ x * y,  
  shade = TRUE, drape = FALSE,  
  xlab = expression(x[1]),  
  ylab = expression(x[2]),  
  zlab = list(expression(italic(f) ~ group("(", list(x[1], x[2]), ")")), rot = 90),  
  scales = list(arrows = FALSE, col = "black"),  
  par.settings = list(axis.line = list(col = "transparent"))),
```

```
screen = list(z = -240, x = -70, y = 0)
)
```

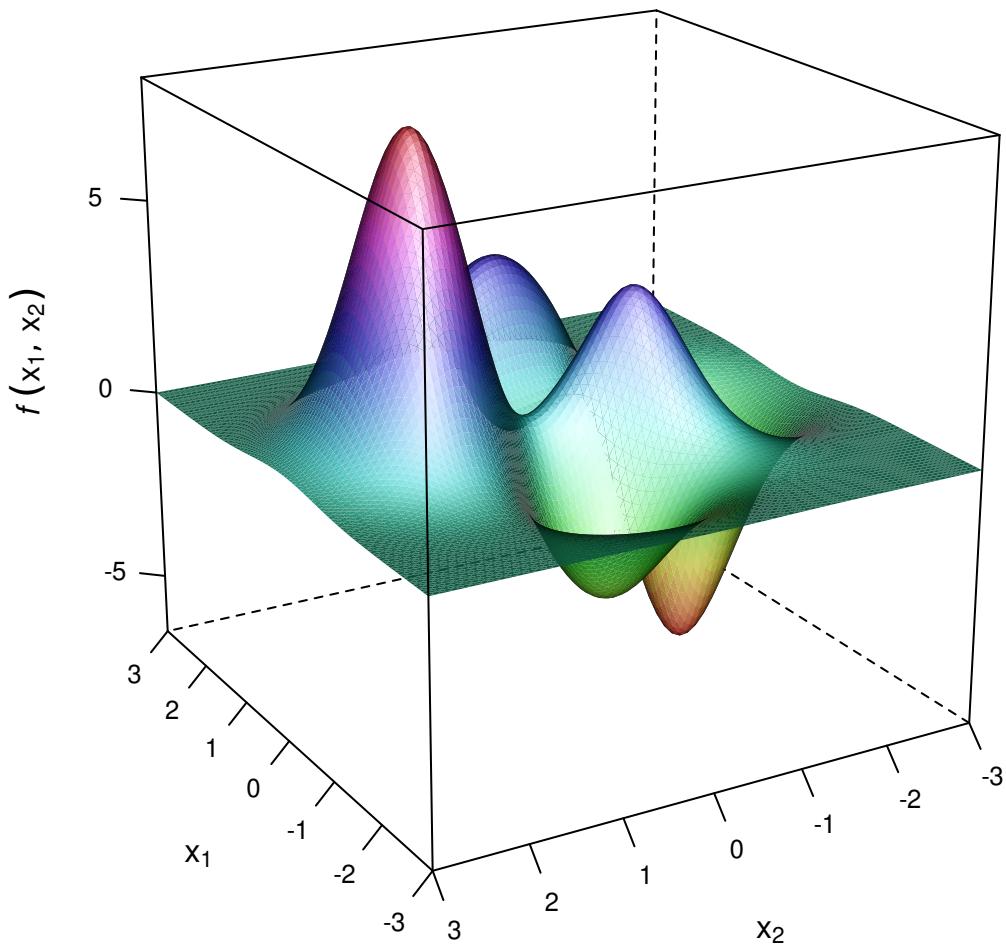


图 13.5: Peaks 多峰图像

函数来自 Octave 内置的 `peaks()` 函数, 它有很多的局部极大值和极小值, 可在 [Octave Online](#) 上输入命令 `help peaks` 查看其帮助文档。

#### 13.4.2.3 Rosenbrock 函数

香蕉函数 定义如下:

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

```
fn <- function(x) {
  (100 * (x[2] - x[1]^2)^2 + (1 - x[1])^2)
}

df <- expand.grid(
  x = seq(-2.5, 2.5, length = 101),
  y = seq(-2.5, 2.5, length = 101)
```

```
)  
df$fnxy = apply(df, 1, fn)  
  
wireframe(  
  data = df, fnxy ~ x * y,  
  shade = TRUE, drape = FALSE,  
  xlab = expression(x[1]),  
  ylab = expression(x[2]),  
  zlab = list(expression(italic(f) ~ group("(", list(x[1], x[2]), ")"))), rot = 90),  
  scales = list(arrows = FALSE, col = "black"),  
  par.settings = list(axis.line = list(col = "transparent")),  
  screen = list(z = 120, x = -70, y = 0)  
)
```

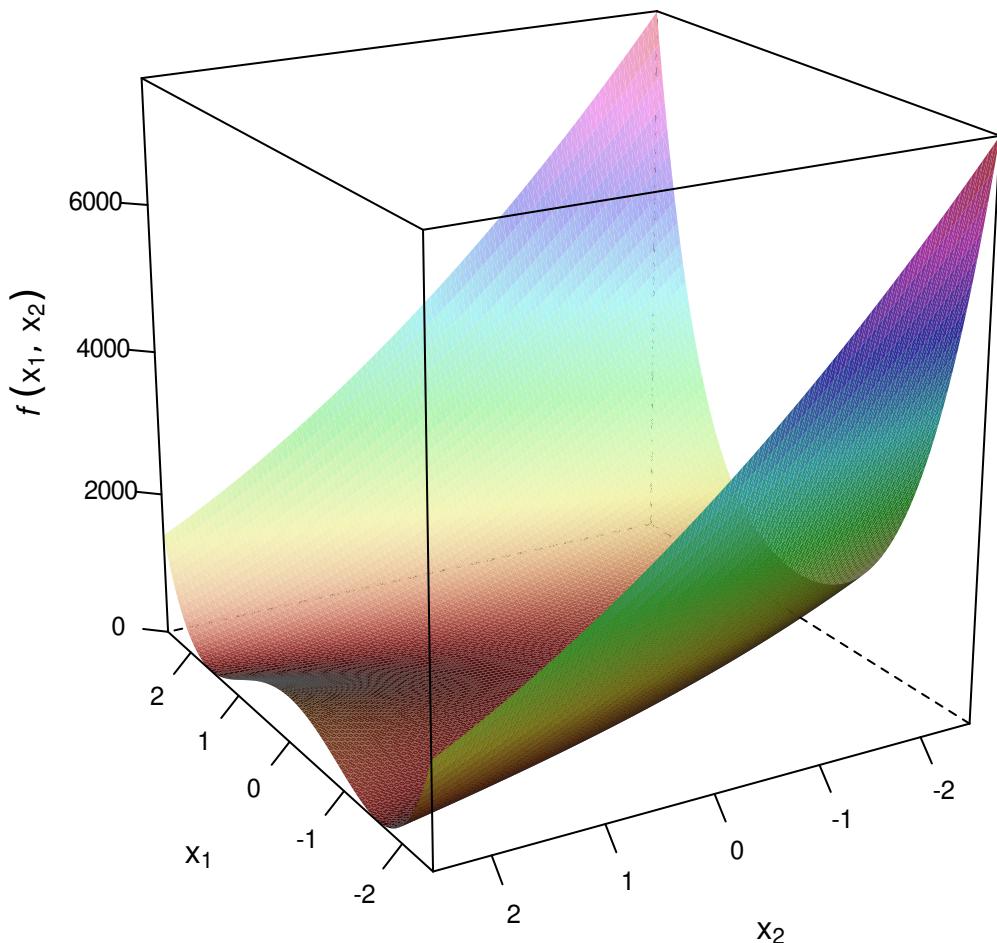


图 13.6: 香蕉函数图像

```
r <- raster::rasterFromXYZ(df, crs = CRS("+proj=longlat +datum=WGS84"))  
rasterVis::vectorplot(r, par.settings = RdBuTheme())
```

```
# 梯度函数  
gr <- function(x) {  
  numDeriv::grad(fn, c(x[1], x[2]))
```

```
}

optim(par = c(-1.2, 1), fn = fn, gr = gr, method = "BFGS")

## $par
## [1] 1 1
##
## $value
## [1] 9.595012e-18
##
## $counts
## function gradient
##      110      43
##
## $convergence
## [1] 0
##
## $message
## NULL

op <- OP(
  objective = F_objective(fn, n = 2L, G = gr),
  bounds = V_bound(ld = -3, ud = 3, nobj = 2L)
)
nlp <- ROI_solve(op, solver = "nloptr.lbfgs", start = c(-1.2, 1))
nlp$objval

## [1] 1.364878e-17

nlp$solution

## [1] 1 1
```

#### 13.4.2.4 Ackley 函数

Ackley 函数是一个非凸函数，有大量局部极小值点，获取全局极小值点是一个比较有挑战的事。它的  $n$  维形式如下：

$$f(\mathbf{x}) = -ae^{-b\sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}} - e^{\frac{1}{n} \sum_{i=1}^n \cos(cx_i)} + a + e$$

其中， $a = 20, b = 0.2, c = 2\pi$ ，对  $\forall i = 1, 2, \dots, n, x_i \in [-10, 10]$ ， $f(\mathbf{x})$  在  $\mathbf{x}^* = (0, 0, \dots, 0)$  取得全局最小值  $f(\mathbf{x}^*) = 0$ ，二维图像如图 13.7。

```
fn <- function(x, a = 20, b = 0.2, c = 2 * pi) {
  mean1 <- mean(x^2)
  mean2 <- mean(cos(c * x))
  -a * exp(-b * sqrt(mean1)) - exp(mean2) + a + exp(1)
}

df <- expand.grid(
```

④

```
x = seq(-10, 10, length.out = 201),
y = seq(-10, 10, length.out = 201)
)
df$fnxy = apply(df, 1, fn)

wireframe(
  data = df, fnxy ~ x * y,
  shade = TRUE, drape = FALSE,
  xlab = expression(x[1]),
  ylab = expression(x[2]),
  zlab = list(expression(italic(f) ~ group("(", list(x[1], x[2]), ")"))), rot = 90),
  scales = list(arrows = FALSE, col = "black"),
  par.settings = list(axis.line = list(col = "transparent")),
  screen = list(z = 120, x = -70, y = 0)
)
```

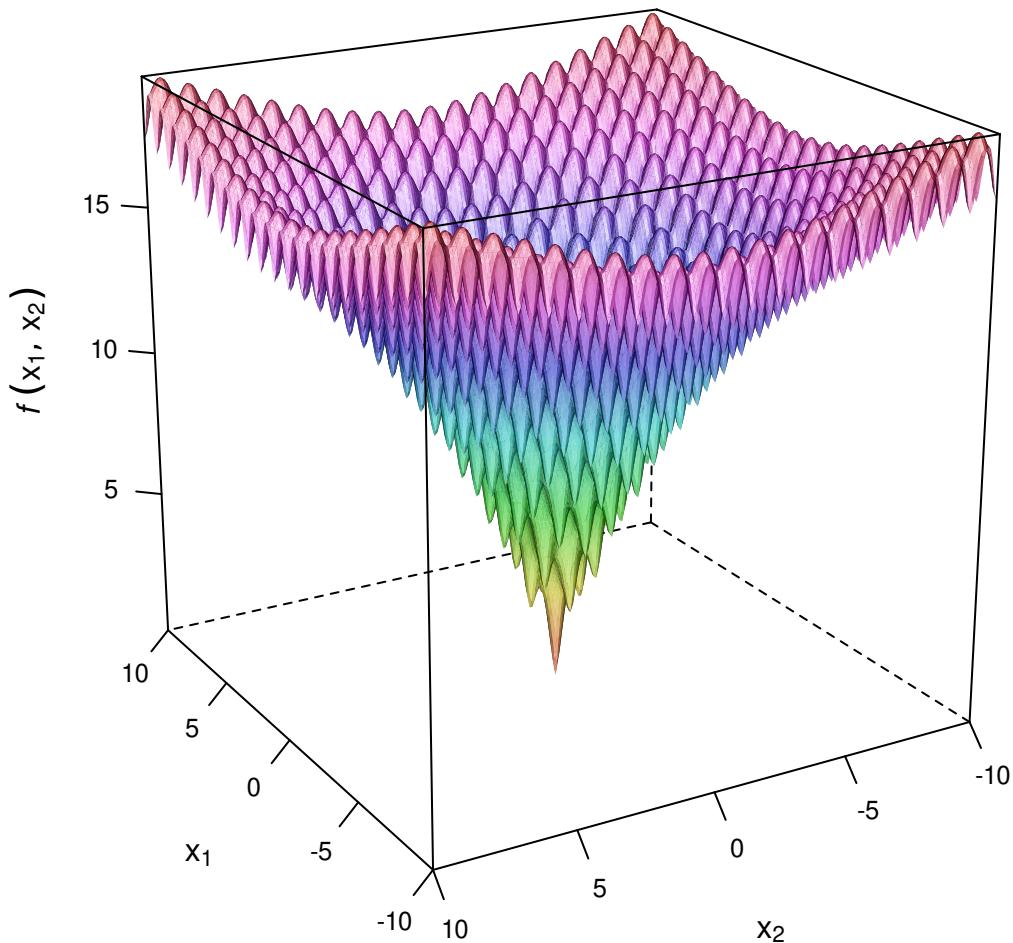


图 13.7: 二维 Ackley 函数图像

以 10 维的 Ackley 函数为例, 先试一下普通的局部优化算法 — Nelder-Mead 算法, 选择初值  $(2, 2, \dots, 2)$ , 看下效果, 再与全局优化算法比较。



```
op <- OP(  
  objective = F_objective(fn, n = 10L),  
  bounds = V_bound(ld = -10, ud = 10, nobj = 10L)  
)  
  
nlp <- ROI_solve(op, solver = "nloptr.neldermead", start = rep(2, 10))  
nlp$solution  
  
## [1] 2 2 2 2 2 2 2 2 2 2  
nlp$objval  
  
## [1] 6.593599
```

可以说完全没有优化效果，已经陷入局部极小值。根据[nloptr 全局优化算法](#)的介绍，这里采用 directL 算法，因为是全局优化，不用选择初值。

```
# 调全局优化器  
nlp <- ROI_solve(op, solver = "nloptr.directL")  
nlp$solution  
  
## [1] 0 0 0 0 0 0 0 0 0 0  
nlp$objval  
  
## [1] 4.440892e-16  
fn(x = c(2, 2))  
  
## [1] 6.593599  
fn(x = rep(2, 10))  
  
## [1] 6.593599
```

#### 13.4.2.5 Radistrigin 函数

这里，还有另外一个例子，Radistrigin 函数也是多模函数

$$f(\mathbf{x}) = \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i) + 10)$$

```
fn <- function(x) {  
  sum(x^2 - 10 * cos(2 * pi * x) + 10)  
}  
  
df <- expand.grid(  
  x = seq(-4, 4, length.out = 201),  
  y = seq(-4, 4, length.out = 201)  
)
```

```
df$fnxy = apply(df, 1, fn)

wireframe(
  data = df, fnxy ~ x * y,
  shade = TRUE, drape = FALSE,
  xlab = expression(x[1]),
  ylab = expression(x[2]),
  zlab = list(expression(italic(f) ~ group("(", list(x[1], x[2]), ")"))),
  scales = list(arrows = FALSE, col = "black"),
  par.settings = list(axis.line = list(col = "transparent")),
  screen = list(z = 120, x = -65, y = 0)
)
```

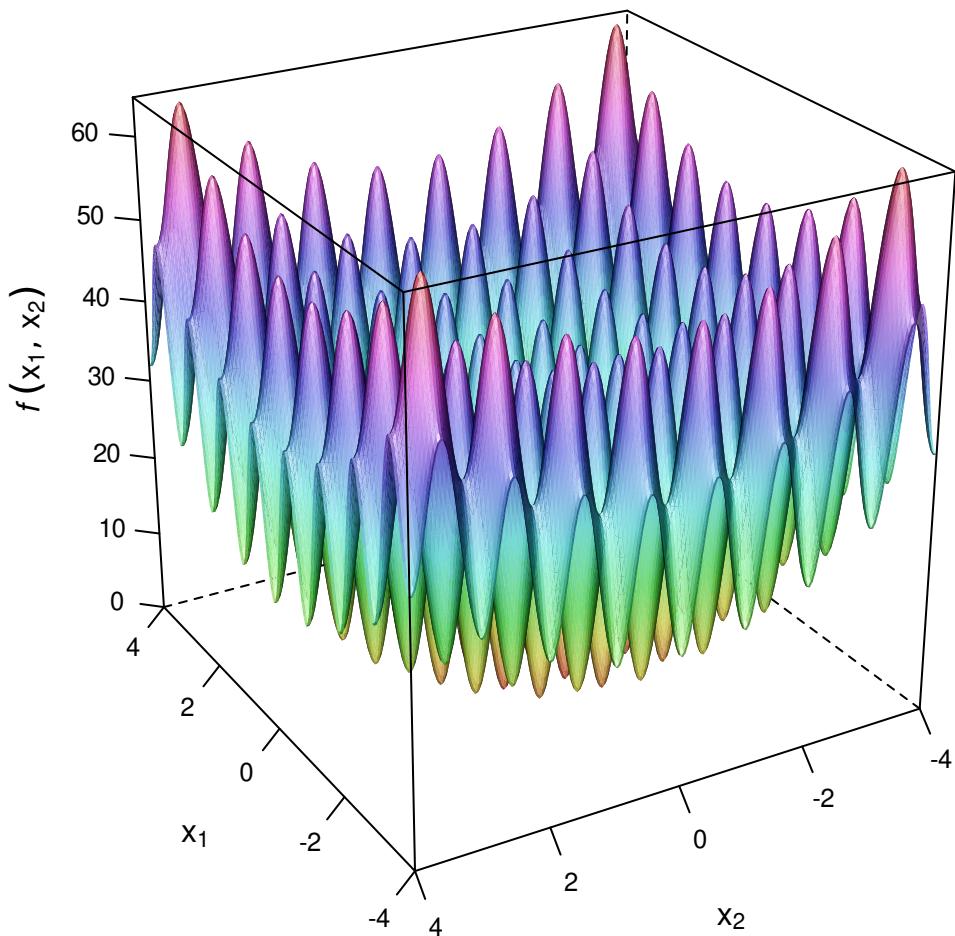


图 13.8: Radistrigin 函数

设置 10 维的优化

```
op <- OP(
  objective = F_objective(fn, n = 10L),
  bounds = V_bound(lb = -50, ub = 50, nobj = 10L)
)
```



调全局优化器求解非凸优化问题

```
nlp <- ROI_solve(op, solver = "nloptr.directL")
nlp$solution

## [1] 0 0 0 0 0 0 0 0 0 0

nlp$objval

## [1] 0
```

#### 13.4.2.6 Schaffer 函数

$$f(x_1, x_2) = 0.5 + \frac{\sin^2(x_1^2 - x_2^2) - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}$$

在  $\mathbf{x}^* = (0, 0)$  处取得全局最小值  $f(\mathbf{x}^*) = 0$

```
fn <- function(x) {
  0.5 + ((sin(x[1]^2 - x[2]^2))^2 - 0.5) / (1 + 0.001*(x[1]^2 + x[2]^2))^2
}

df <- expand.grid(
  x = seq(-50, 50, length = 201),
  y = seq(-50, 50, length = 201)
)
df$fnxy = apply(df, 1, fn)

wireframe(
  data = df, fnxy ~ x * y,
  shade = TRUE, drape = FALSE,
  xlab = expression(x[1]),
  ylab = expression(x[2]),
  zlab = list(expression(italic(f) ~ group("(", list(x[1], x[2]), ")"))),
  rot = 90,
  scales = list(arrows = FALSE, col = "black"),
  par.settings = list(axis.line = list(col = "transparent")),
  screen = list(z = 120, x = -70, y = 0)
)

df <- expand.grid(
  x = seq(-2, 2, length = 101),
  y = seq(-2, 2, length = 101)
)
df$fnxy = apply(df, 1, fn)

wireframe(
  data = df, fnxy ~ x * y,
  shade = TRUE, drape = FALSE,
  xlab = expression(x[1]),
```

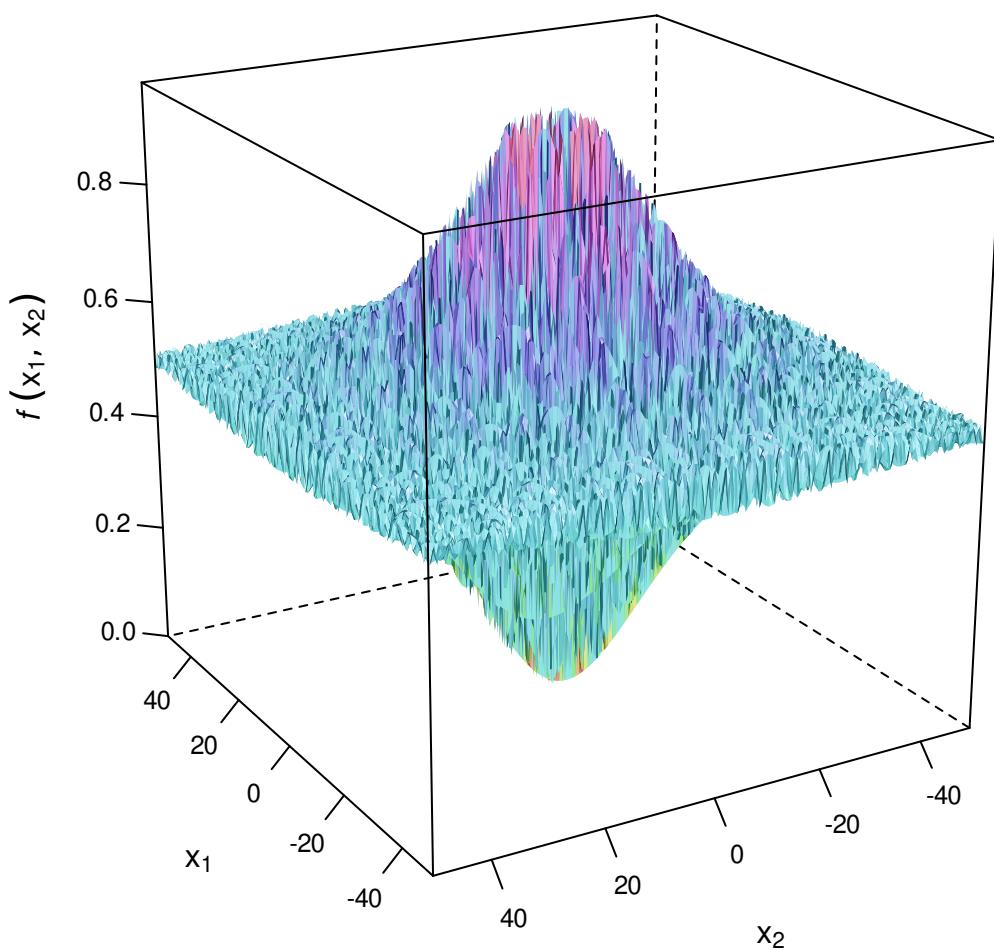


图 13.9: Schaffer 函数

```
ylab = expression(x[2]),
zlab = list(expression(italic(f) ~ group("(", list(x[1], x[2]), ")")), rot = 90),
scales = list(arrows = FALSE, col = "black"),
par.settings = list(axis.line = list(col = "transparent")),
screen = list(z = 120, x = -70, y = 0)
)
```

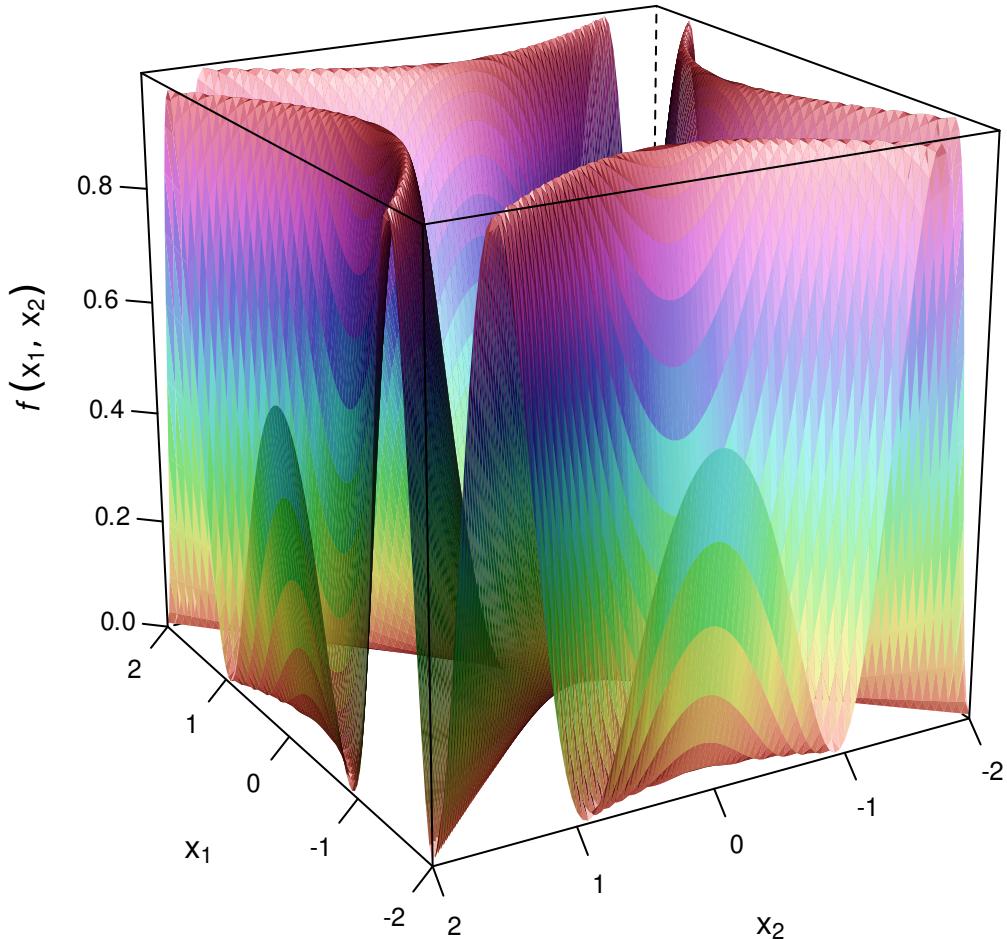


图 13.10: Schaffer 函数

## 13.4.2.7 Hölder 函数

Hölder 桌面函数

$$f(x_1, x_2) = -|\sin(x_1) \cos(x_2) \exp\left(|1 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi}|\right)|$$

在  $(8.05502, 9.66459)$ 、 $(-8.05502, 9.66459)$ 、 $(8.05502, -9.66459)$ 、 $(-8.05502, -9.66459)$  同时取得最小值  $-19.2085$ 。

```
fn <- function(x) {
  -abs(sin(x[1]) * cos(x[2])) * exp(abs(1 - sqrt(x[1]^2 + x[2]^2) / pi))
}
```

```
df <- expand.grid(
  x = seq(-10, 10, length = 101),
  y = seq(-10, 10, length = 101)
)
df$fnxy = apply(df, 1, fn)

wireframe(
  data = df, fnxy ~ x * y,
  shade = TRUE, drape = FALSE,
  xlab = expression(x[1]),
  ylab = expression(x[2]),
  zlab = list(expression(italic(f) ~ group("(", list(x[1], x[2]), ")"))), rot = 90),
  scales = list(arrows = FALSE, col = "black"),
  par.settings = list(axis.line = list(col = "transparent")),
  screen = list(z = 120, x = -60, y = 0)
)
```

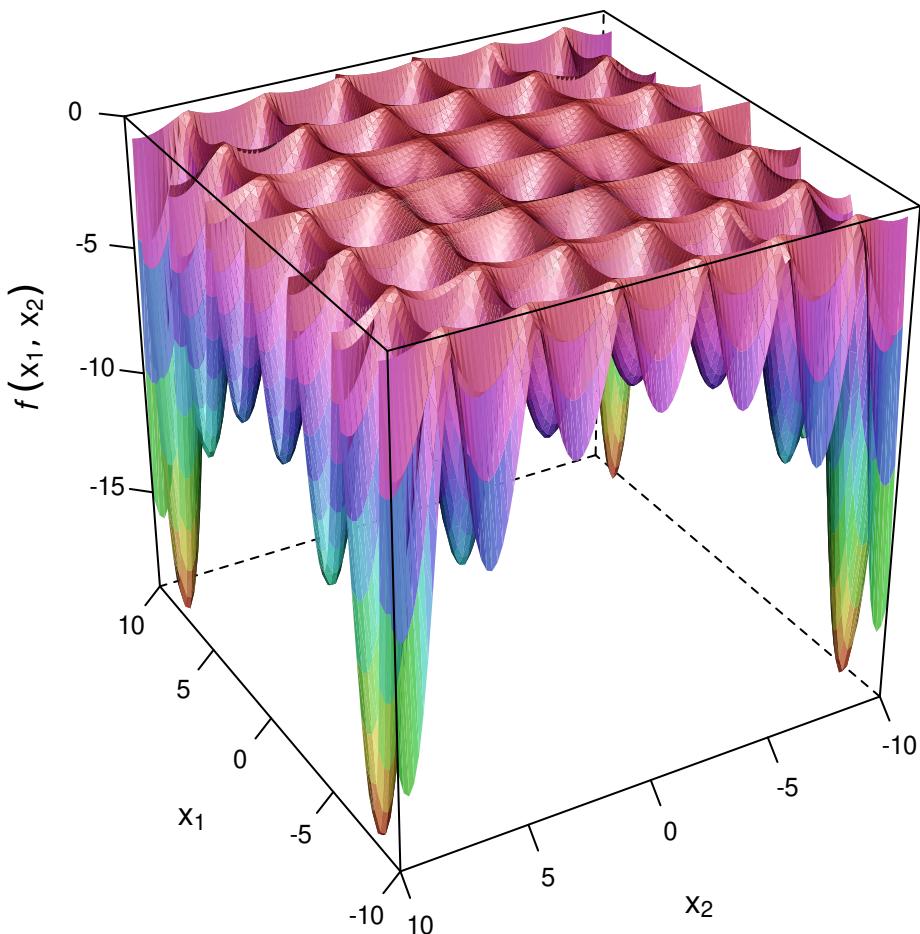


图 13.11: Hölder 函数



### 13.4.2.8 Trid 函数

$n \geq 2$  维 Trid 函数

$$f(x) = \sum_{i=1}^n (x_i - 1)^2 - \sum_{i=2}^n x_i x_{i-1}$$

$\forall i = 1, 2, \dots, n$ ,  $f(x)$  在  $x_i = i(n+1-i)$  处取得全局极小值  $f(\mathbf{x}^*) = -n(n+4)(n-1)/6$ , 取值区间  $x \in [-n^2, n^2]$ ,  $\forall i = 1, 2, \dots, n$

```
fn <- function(x) {
  n <- length(x)
  sum((x - 1)^2) - sum(x[-1] * x[-n])
}

df <- expand.grid(
  x = seq(-4, 4, length = 101),
  y = seq(-4, 4, length = 101)
)
df$fnxy = apply(df, 1, fn)

wireframe(
  data = df, fnxy ~ x * y,
  shade = TRUE, drape = FALSE,
  xlab = expression(x[1]),
  ylab = expression(x[2]),
  zlab = list(expression(italic(f) ~ group("(", list(x[1], x[2]), ")"))),
  rot = 90,
  scales = list(arrows = FALSE, col = "black"),
  par.settings = list(axis.line = list(col = "transparent")),
  screen = list(z = -60, x = -70, y = 0)
)
```

### 13.4.2.9 超级复杂函数

有如下复杂的目标函数

$$\begin{aligned} \min_x \quad & \cos(x_1) \cos(x_2) - \sum_{i=1}^5 \left( (-1)^i \cdot i \cdot 2 \cdot \exp \left( -500 \cdot ((x_1 - i \cdot 2)^2 + (x_2 - i \cdot 2)^2) \right) \right) \\ \text{s.t.} \quad & -50 \leq x_1, x_2 \leq 50 \end{aligned}$$

```
subfun <- function(i, m) {
  (-1)^i * i * 2 * exp(-500 * ((m[1] - i * 2)^2 + (m[2] - i * 2)^2))
}

fn <- function(x) {
  cos(x[1]) * cos(x[2]) -
  sum(mapply(FUN = subfun, i = 1:5, MoreArgs = list(m = x)))
}
```

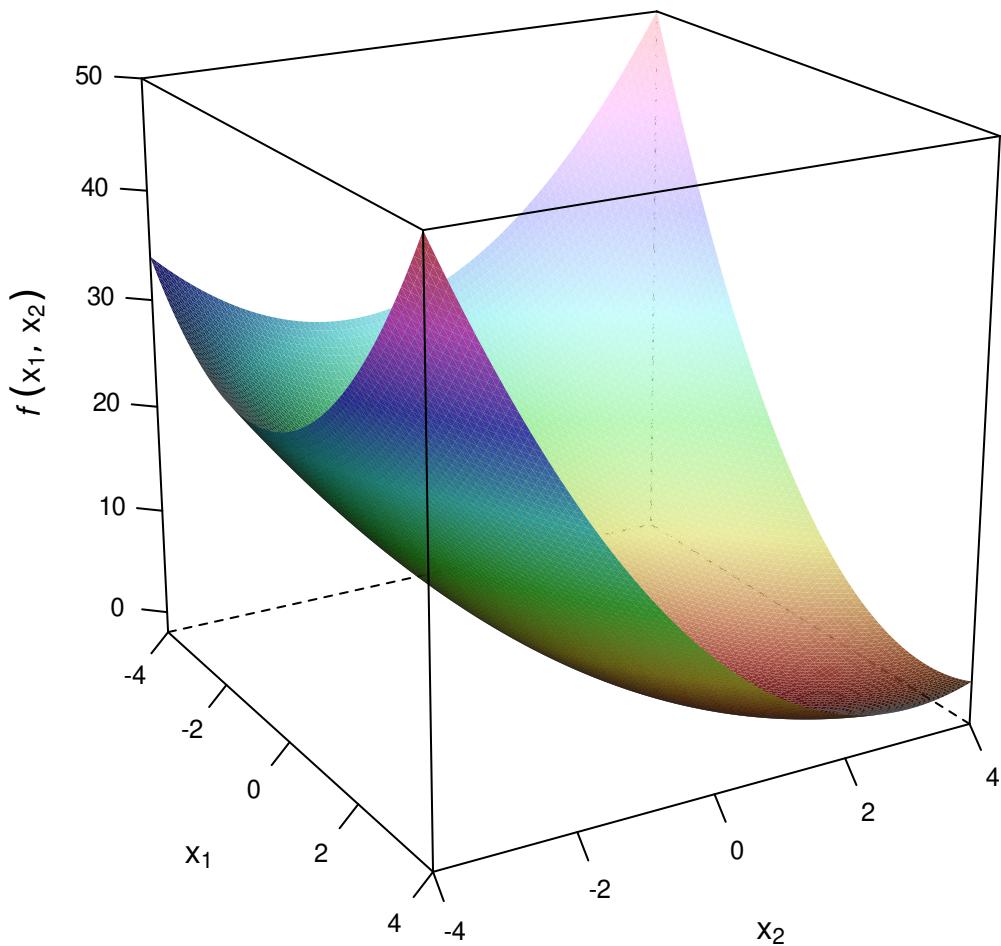


图 13.12: Trid 函数

目标函数的图像见图 13.13, 搜索区域  $[-50, 50] \times [-50, 50]$  内几乎没有变化的梯度, 给寻优过程带来很大困难。

```
df <- expand.grid(
  x = seq(-50, 50, length.out = 101),
  y = seq(-50, 50, length.out = 101)
)

df$fnxy = apply(df, 1, fn)

wireframe(
  data = df, fnxy ~ x * y,
  shade = TRUE, drape = FALSE,
  xlab = expression(x[1]),
  ylab = expression(x[2]),
  zlab = list(expression(italic(f) ~ group("(", list(x[1], x[2]), ")"))),
  rot = 90),
  scales = list(arrows = FALSE, col = "black"),
  par.settings = list(axis.line = list(col = "transparent")),
  screen = list(z = 120, x = -65, y = 0)
)
```

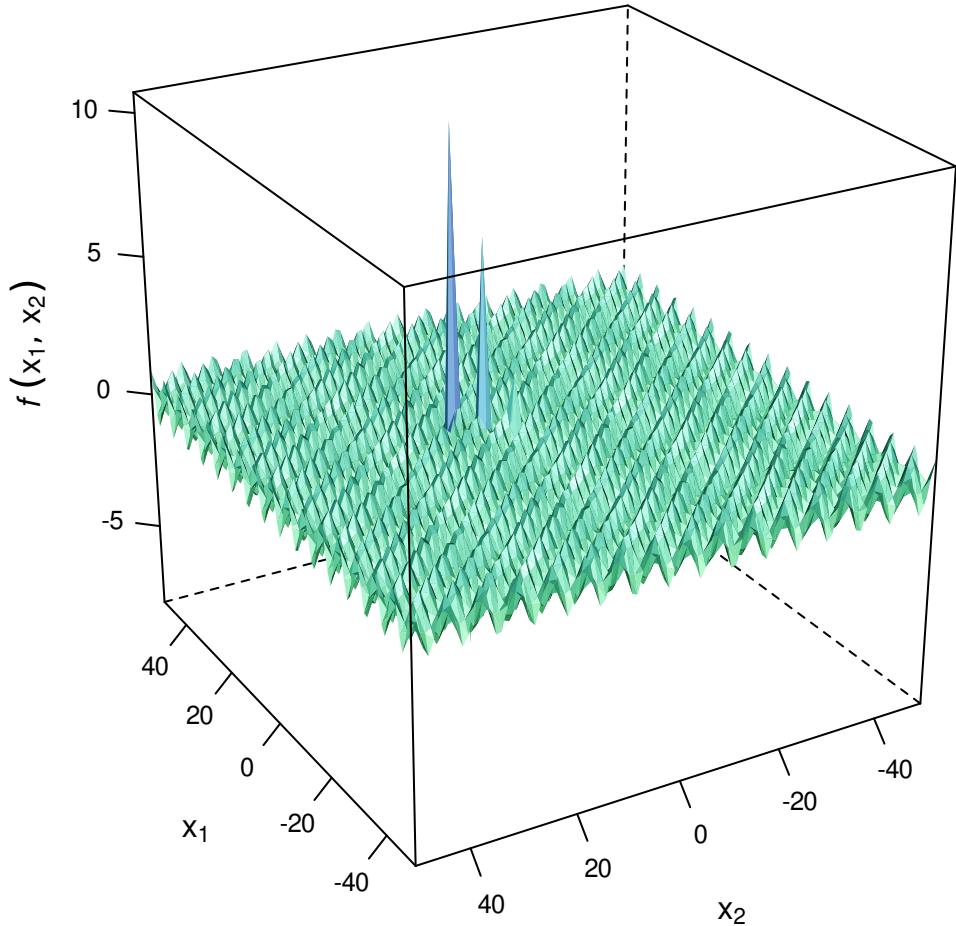


图 13.13: 函数图像

将区域  $[0, 12] \times [0, 12]$  的图像绘制出来，不难发现，有不少局部陷阱。

```
df <- expand.grid(
  x = seq(0, 12, length.out = 201),
  y = seq(0, 12, length.out = 201)
)
df$fnxy = apply(df, 1, fn)

wireframe(
  data = df, fnxy ~ x * y,
  shade = TRUE, drape = FALSE,
  xlab = expression(x[1]),
  ylab = expression(x[2]),
  zlab = list(expression(italic(f) ~ group("(", list(x[1], x[2]), ")"))), rot = 90),
  scales = list(arrows = FALSE, col = "black"),
  par.settings = list(axis.line = list(col = "transparent")),
  screen = list(z = 120, x = -65, y = 0)
)
```

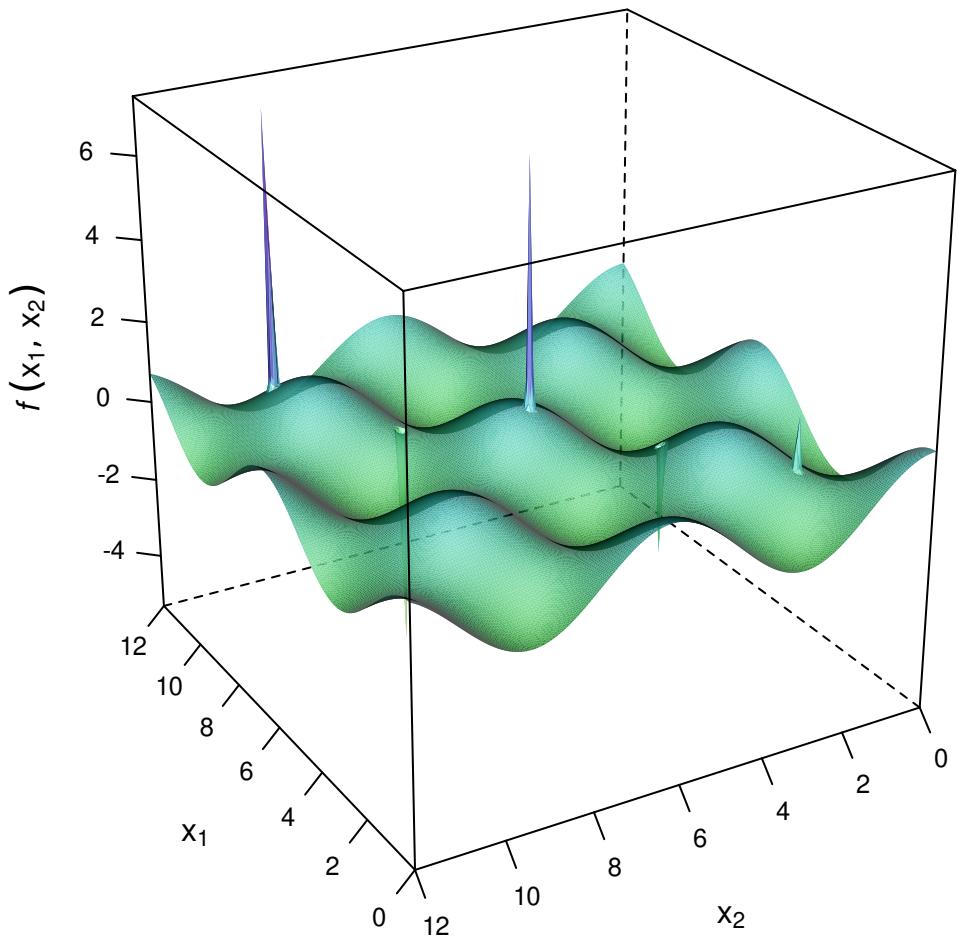


图 13.14: 局部放大函数图像

最优解在  $(7.999982, 7.999982)$  取得, 目标函数值为 -7.978832。

```
fn(x = c(7.999982, 7.999982))
```

```
## [1] -7.978832
```

面对如此复杂的函数, 调用全局优化器

```
op <- OP(  
  objective = F_objective(fn, n = 2L),  
  bounds = V_bound(lb = -50, ub = 50, nobj = 2L)  
)  
nlp <- ROI_solve(op, solver = "nloptr.directL")  
nlp$solution
```

```
## [1] 22.22222 0.00000
```

```
nlp$objval
```

```
## [1] -0.9734211
```

实际上, 还是陷入局部最优解。

```
SETS:  
P/1..5/;  
Endsets  
Min=@cos(x1) * @cos(x2) - @Sum(P(j): (-1)^j * j * 2 * @exp(-500 * ((x1 - j * 2)^2 + (x2 - j * 2)^2)));  
@Bnd(-50, x1, 50);  
@Bnd(-50, x2, 50);
```

Lingo 18.0 启用全局优化求解器后, 在  $(x_1 = 7.999982, x_2 = 7.999982)$  取得最小值 -7.978832。而默认未启用全局优化求解器的情况下, 在  $(x_1 = 18.84956, x_2 = -40.84070)$  取得局部极小值 -1.000000。

### 13.4.3 多元非线性约束优化

R 自带的函数 `nlminb()` 可求解无约束、箱式约束优化问题, `constrOptim()` 还可求解线性不等式约束优化, 其中包括带线性约束的二次规划。`optim()` 提供一大类优化算法, 且包含随机优化算法—模拟退火算法, 可求解无约束、箱式约束优化问题。

#### 13.4.3.1 普通箱式约束

有如下箱式约束优化问题, 目标函数和[香蕉函数](#)有些相似。

$$\begin{aligned} \min_x \quad & (x_1 - 1)^2 + 4 \sum_{i=1}^{n-1} (x_{i+1} - x_i^2)^2 \\ \text{s.t.} \quad & 2 \leq x_1, x_2, \dots, x_n \leq 4 \end{aligned}$$

```
fn <- function(x) {  
  n <- length(x)  
  sum(c(1, rep(4, n - 1)) * (x - c(1, x[-n])^2)^2)  
}
```



$n$  维目标函数是非线性的, 给定初值  $(3, 3, \dots, 3)$ , 下面求解 25 维的箱式约束,

```
nlminb(start = rep(3, 25), objective = fn, lower = rep(2, 25), upper = rep(4, 25))

## $par
## [1] 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000
## [9] 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000
## [17] 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.109093
## [25] 4.000000
##
## $objective
## [1] 368.1059
##
## $convergence
## [1] 0
##
## $iterations
## [1] 6
##
## $evaluations
## function gradient
##      10      177
##
## $message
## [1] "relative convergence (4)"
```

`nlminb()` 出于历史兼容性的原因尚且存在, 最优解的第 24 个分量没有在可行域的边界上。使用 `constrOptim()` 函数求解, 默认求极小, 需将箱式或线性不等式约束写成矩阵形式, 即  $Ax \geq b$  的形式, 参数 `ui` 是  $k \times n$  的约束矩阵  $A$ , `ci` 是右侧  $k$  维约束向量  $b$ 。以上面的优化问题为例, 将箱式约束  $2 \leq x_1, x_2 \leq 4$  转化为矩阵形式, 约束矩阵和向量分别为:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}, \quad b = (2, 2, -4, -4)$$

```
constrOptim(
  theta = rep(3, 25), # 初始值
  f = fn, # 目标函数
  method = "Nelder-Mead", # 没有提供梯度, 则必须用 Nelder-Mead 方法
  ui = rbind(diag(rep(1, 25)), diag(rep(-1, 25))),
  ci = c(rep(2, 25), rep(-4, 25))
)

## $par
## [1] 2.006142 2.002260 2.003971 2.003967 2.004143 2.004255 2.001178 2.002990
## [9] 2.003883 2.006029 2.017345 2.009236 2.000949 2.007793 2.025831 2.007896
## [17] 2.004514 2.004381 2.008771 2.015695 2.005803 2.009127 2.017988 2.257782
```

```
## [25] 3.999846
##
## $value
## [1] 378.4208
##
## $counts
## function gradient
##      12048      NA
##
## $convergence
## [1] 1
##
## $message
## NULL
##
## $outer.iterations
## [1] 25
##
## $barrier.value
## [1] -0.003278963
```

从求解的结果来看, convergence = 1 意味着迭代次数到达默认的极限 maxit = 500, 结合 `nlminb()` 函数的求解结果来看, 实际上还没有收敛。如果没有提供梯度, 则必须用 Nelder-Mead 方法, 下面增加迭代次数到 1000。

```
constrOptim(
  theta = rep(3, 25), # 初始值
  f = fn, # 目标函数
  method = "Nelder-Mead",
  control = list(maxit = 1000),
  ui = rbind(diag(rep(1, 25)), diag(rep(-1, 25))),
  ci = c(rep(2, 25), rep(-4, 25))
)

## $par
##  [1] 2.000081 2.000142 2.001919 2.000584 2.000007 2.000003 2.001097 2.001600
##  [9] 2.000207 2.000042 2.000250 2.000295 2.000580 2.002165 2.000453 2.000932
## [17] 2.000456 2.000363 2.000418 2.000474 2.009483 2.001156 2.003173 2.241046
## [25] 3.990754
##
## $value
## [1] 370.8601
##
## $counts
## function gradient
##      18036      NA
##
```



```
## $convergence
## [1] 1
##
## $message
## NULL
##
## $outer.iterations
## [1] 19
##
## $barrier.value
## [1] -0.003366467
```

还是没有收敛，可见 Nelder-Mead 方法在这个优化问题上收敛速度比较慢。下面考虑调用基于梯度的优化算法 — BFGS 方法。

```
# 输入 n 维向量, 输出 n 维向量
gr <- function(x) {
  n <- length(x)
  c(2 * (x[1] - 2), rep(0, n - 1))
  +8 * c(0, x[-1] - x[-n]^2)
  -16 * c(x[-n], 0) * c(x[-1] - x[-n]^2, 0)
}

constrOptim(
  theta = rep(3, 25), # 初始值
  f = fn, # 目标函数
  grad = gr,
  method = "BFGS",
  control = list(maxit = 1000),
  ui = rbind(diag(rep(1, 25)), diag(rep(-1, 25))),
  ci = c(rep(2, 25), rep(-4, 25))
)

## $par
## [1] 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000
## [9] 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000
## [17] 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000001
## [25] 3.000000
##
## $value
## [1] 373
##
## $counts
## function gradient
##      3721      464
##
## $convergence
```

```
## [1] 0
##
## $message
## NULL
##
## $outer.iterations
## [1] 3
##
## $barrier.value
## [1] -0.003327104
```

相比于 Nelder-Mead 方法，目标值 373 更大，可见已陷入局部最优解，下面通过 ROI 包，分别调用求解器 L-BFGS 和 directL，发现前者同样陷入局部最优解，而后者可以获得与 `nlmnb()` 函数一致的结果。

```
# 调用改进的 BFGS 算法
op <- OP(
  objective = F_objective(fn, n = 25L, G = gr),
  bounds = V_bound(ld = 2, ud = 4, nobj = 25L)
)
nlp <- ROI_solve(op, solver = "nloptr.lbfgs", start = rep(3, 25))
nlp$objval

## [1] 373

nlp$solution

## [1] 2 3

# 调全局优化算法
nlp <- ROI_solve(op, solver = "nloptr.directL")
nlp$objval

## [1] 368.1059

nlp$solution

## [1] 2.00000 2.00000 2.00000 2.00000 2.00000 2.00000 2.00000 2.00000 2.00000
## [10] 2.00000 2.00000 2.00000 2.00000 2.00000 2.00000 2.00000 2.00000 2.00000
## [19] 2.00000 2.00000 2.00000 2.00000 2.00000 2.10913 4.00000

下面再与函数 optim() 提供的 L-BFGS-B 算法比较

optim(
  par = rep(3, 25), fn = fn, gr = NULL, method = "L-BFGS-B",
  lower = rep(2, 25), upper = rep(4, 25)
)

## $par
## [1] 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000
## [9] 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000
## [17] 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.109093
```



```
## [25] 4.000000
##
## $value
## [1] 368.1059
##
## $counts
## function gradient
##       6       6
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

值得注意的是，当提供梯度信息的时候，虽然求解速度提升了，但是最优解变差了。

```
optim(
  par = rep(3, 25), fn = fn, gr = gr, method = "L-BFGS-B",
  lower = rep(2, 25), upper = rep(4, 25)
)
```

```
## $par
## [1] 2 3
##
## $value
## [1] 373
##
## $counts
## function gradient
##       2       2
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: NORM OF PROJECTED GRADIENT <= PGTOL"
```

### 13.4.3.2 非线性严格不等式约束

第一个例子，目标函数是非线性的，约束条件也是非线性的，非线性不等式约束不包含等号。

$$\begin{aligned} \min_x \quad & (x_1 + 3x_2 + x_3)^2 + 4(x_1 - x_2)^2 \\ \text{s.t.} \quad & \begin{cases} x_1 + x_2 + x_3 = 1 \\ 6x_2 + 4x_3 - x_1^3 > 3 \\ x_1, x_2, x_3 > 0 \end{cases} \end{aligned}$$



```
# 目标函数
fn <- function(x) (x[1] + 3 * x[2] + x[3])^2 + 4 * (x[1] - x[2])^2

# 目标函数的梯度
gr <- function(x) {
  c(
    2 * (x[1] + 3 * x[2] + x[3]) + 8 * (x[1] - x[2]), # 对 x[1] 求偏导
    6 * (x[1] + 3 * x[2] + x[3]) - 8 * (x[1] - x[2]), # 对 x[2] 求偏导
    2 * (x[1] + 3 * x[2] + x[3]) # 对 x[3] 求偏导
  )
}

# 等式约束
heq <- function(x) {
  x[1] + x[2] + x[3] - 1
}

# 等式约束的雅可比矩阵
# 这里只有一个等式约束, 所以雅可比矩阵行数为 1
heq.jac <- function(x) {
  matrix(c(1, 1, 1), ncol = 3, byrow = TRUE)
}

# 不等式约束
# 要求必须是严格不等式, 不能带等号, 方向是 x > 0
hin <- function(x) {
  c(6 * x[2] + 4 * x[3] - x[1]^3 - 3, x[1], x[2], x[3])
}

# 不等式约束的雅可比矩阵
# 其实是有 4 个不等式约束, 3 个目标变量约束, 雅可比矩阵行数是 4
hin.jac <- function(x) {
  matrix(c(
    -3 * x[1]^2, 6, 4,
    1, 0, 0,
    0, 1, 0,
    0, 0, 1
  ), ncol = 3, byrow = TRUE)
}
```

调用 **alabama** 包的求解器

```
set.seed(12)
# 初始值
p0 <- runif(3)
# 求目标函数的极小值
ans <- alabama::constrOptim.nl(
  par = p0,
  # 目标函数
  fn = fn,
  gr = gr,
```

```
# 等式约束
heq = heq,
heq.jac = heq.jac,
# 不等式约束
hin = hin,
hin.jac = hin.jac,
# 不显示迭代过程
control.outer = list(trace = FALSE)
)
ans

## $par
## [1] 7.390292e-04 4.497160e-12 9.992610e-01
##
## $value
## [1] 1.000002
##
## $counts
## function gradient
##      1230      163
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##      [,1]      [,2]      [,3]
## [1,] 120517098 120517087 120517091
## [2,] 120517087 120517115 120517095
## [3,] 120517091 120517095 120517091
##
## $outer.iterations
## [1] 13
##
## $lambda
## [1] 4.481599
##
## $sigma
## [1] 120517089
##
## $barrier.value
## [1] 0.003472071
##
```



```
## $K
## [1] 4.269112e-08
```

ans 是 `constrOptim.nl()` 返回的一个 list, `convergence = 0` 表示迭代成功收敛, `value` 表示目标函数在迭代终止时的取值, `par` 表示满足约束条件, 成功收敛的情况下, 目标函数的参数值, `counts` 表示迭代过程中目标函数及其梯度计算的次数。

# 不提供梯度函数, 照样可以求解

```
ans <- alabama::constrOptim.nl(par = p0, fn = fn, heq = heq, hin = hin)
```

注意

等式和不等式约束的雅可比矩阵必须以 `matrix` 数据类型存储, 而不能以 `vector` 类型存储。要注意和后面 ROI 包的调用形式区别。

实际上, 可以用 ROI 调用 alabama 求解器的方式, 这种方式可以简化目标函数梯度和约束条件的表示

```
# 目标函数
fn <- function(x) (x[1] + 3 * x[2] + x[3])^2 + 4 * (x[1] - x[2])^2
# 目标函数的梯度
gr <- function(x) {
  c(
    2 * (x[1] + 3 * x[2] + x[3]) + 8 * (x[1] - x[2]),
    6 * (x[1] + 3 * x[2] + x[3]) - 8 * (x[1] - x[2]),
    2 * (x[1] + 3 * x[2] + x[3])
  )
}
heq <- function(x) {
  x[1] + x[2] + x[3]
}
heq.jac <- function(x) {
  c(1, 1, 1)
}
hin <- function(x) {
  6 * x[2] + 4 * x[3] - x[1]^3
}
hin.jac <- function(x) {
  c(-3 * x[1]^2, 6, 4)
}
```

通过 ROI 调用 alabama 求解器

```
set.seed(2020)
# 初始值
p0 <- runif(3)
# 定义目标规划
op <- OP(
  objective = F_objective(F = fn, n = 3L, G = gr), # 4 个目标变量
  constraints = F_constraint(
```



```
F = list(heq = heq, hin = hin),
dir = c("==", ">"),
rhs = c(1, 3),
# 等式和不等式约束的雅可比
J = list(heq.jac = heq.jac, hin.jac = hin.jac)
),
bounds = V_bound(lb = 0, ub = Inf, nobj = 3L),
maximum = FALSE # 求最小
)
nlp <- ROI_solve(op, solver = "alabama", start = p0)
nlp$solution
```

```
## [1] 1.674812e-06 9.994336e-08 9.999982e-01
```

```
nlp$objval
```

```
## [1] 1
```

### 13.4.3.3 非线性和箱式约束

与上面的例子不同，下面这个例子的不等式约束包含等号，还有箱式约束，优化问题来源于[Ipopt 官网](#)，提供的初始值为  $x_0 = (1, 5, 5, 1)$ ，最优解为  $x_* = (1.00000000, 4.74299963, 3.82114998, 1.37940829)$ 。优化问题的具体内容如下：

$$\begin{aligned} \min_x \quad & x_1 x_4 (x_1 + x_2 + x_3) + x_3 \\ \text{s.t.} \quad & \begin{cases} x_1^2 + x_2^2 + x_3^2 + x_4^2 = 40 \\ x_1 x_2 x_3 x_4 \geq 25 \\ 1 \leq x_1, x_2, x_3, x_4 \leq 5 \end{cases} \end{aligned}$$

考虑用 ROI 调 nloptr 实现，看结果是否和例子一致，nloptr 支持不等式约束包含等号，支持箱式约束。

```
# 一个 4 维的目标函数
fn <- function(x) {
  x[1] * x[4] * (x[1] + x[2] + x[3]) + x[3]
}

# 目标函数的梯度
gr <- function(x) {
  c(
    x[4] * (2 * x[1] + x[2] + x[3]), x[1] * x[4],
    x[1] * x[4] + 1, x[1] * (x[1] + x[2] + x[3])
  )
}

# 等式约束
heq <- function(x) {
  sum(x^2)
}

# 等式约束的雅可比
```

```
heq.jac <- function(x) {
  2 * c(x[1], x[2], x[3], x[4])
}

# 不等式约束
hin <- function(x) {
  prod(x)
}

# 不等式约束的雅可比
hin.jac <- function(x) {
  c(prod(x[-1]), prod(x[-2]), prod(x[-3]), prod(x[-4]))
}

# 定义目标规划
op <- OP(
  objective = F_objective(F = fn, n = 4L, G = gr), # 4 个目标变量
  constraints = F_constraint(
    F = list(heq = heq, hin = hin),
    dir = c("==", ">="),
    rhs = c(40, 25),
    # 等式和不等式约束的雅可比
    J = list(heq.jac = heq.jac, hin.jac = hin.jac)
  ),
  bounds = V_bound(ld = 1, ud = 5, nobj = 4L),
  maximum = FALSE # 求最小
)
```

```
# 目标函数初始值
fn(c(1, 5, 5, 1))

## [1] 16

# 目标函数最优值
fn(c(1.0000000, 4.74299963, 3.82114998, 1.37940829))

## [1] 17.01402
```

求解一般的非线性约束问题，求解器 nloptr.mma / nloptr.cobyla 仅支持非线性不等式约束，不支持等式约束，而 nlminb 只支持等式约束，因此，下面分别调用 nloptr.auglag、nloptr.slsqp 和 nloptr.isres 来求解上述优化问题。

```
nlp <- ROI_solve(op, solver = "nloptr.auglag", start = c(1, 5, 5, 1))
nlp$solution

## [1] 1.000000 4.743174 3.820922 1.379440

nlp$objval

## [1] 17.01402

nlp <- ROI_solve(op, solver = "nloptr.slsqp", start = c(1, 5, 5, 1))
nlp$solution
```



```
## [1] 1.000000 4.742996 3.821155 1.379408
nlp$objval
## [1] 17.01402
nlp <- ROI_solve(op, solver = "nloptr.isres", start = c(1, 5, 5, 1))
nlp$solution
## [1] 1.146838 4.487311 4.140554 1.191561
nlp$objval
## [1] 17.49795
```

可以看出，nloptr 提供的优化能力可以覆盖Iopt 求解器，推荐使用 nloptr.slsqp 求解器。

#### 13.4.3.4 非线性混合整数约束

$$\begin{aligned} \max_x \quad & 1.5(x_1 - \sin(x_1 - x_2))^2 + 0.5x_2^2 + x_3^2 - x_1x_2 - 2x_1 + x_2x_3 \\ \text{s.t.} \quad & \begin{cases} -20 < x_1 < 20 \\ -20 < x_2 < 20 \\ -10 < x_3 < 10 \\ x_1, x_2 \in \mathbb{R}, \quad x_3 \in \mathbb{Z} \end{cases} \end{aligned}$$

```
fn <- function(x) {
  1.5 * (x[1] - sin(x[1] - x[2]))^2 + 0.5 * x[2]^2 + x[3]^2
  -x[1] * x[2] - 2 * x[1] + x[2] * x[3]
}

gr <- function(x) {
  c(
    3 * (x[1] - sin(x[1] - x[2])) * (1 - cos(x[1] - x[2])) - x[2] - 2,
    3 * (x[1] - sin(x[1] - x[2])) * cos(x[1] - x[2]) - x[2] - x[1] + x[3],
    2 * x[3] + x[2]
  )
}
```

目前 ROI 还解不了

```
# 初始值
p0 <- c(2.1, 5.1, 5)
# 定义目标规划
op <- OP(
  objective = F_objective(F = fn, n = 3L, G = gr), # 3 个目标变量
  types = c("C", "C", "I"), # 目标变量的类型
  bounds = V_bound(lb = c(-20, -20, -10), ub = c(20, 20, 10), nobj = 3L),
  maximum = FALSE # 求最小
)
nlp <- ROI_solve(op, solver = "auto", start = p0)
nlp$solution
```



目标函数在  $(4.49712, 9.147501, -4)$  取得最小值 -86.72165

```
fn(x = c(4.49712, 9.147501, -4))  
## [1] -86.72165
```

### 13.4.3.5 含复杂目标函数

下面这个目标函数比较复杂，约束条件也是非线性的

$$\begin{aligned} \max_x \quad & \frac{(\sin(2\pi x_1))^3 \sin(2\pi x_2)}{x_1^3(x_1+x_2)} \\ s.t. \quad & \begin{cases} x_1^2 - x_2 + 1 \leq 0 \\ 1 - x_1 + (x_2 - 4)^2 \geq 0 \\ 0 \leq x_1, x_2 \leq 10 \end{cases} \end{aligned}$$

```
# 目标函数  
fn <- function(x) (sin(2*pi*x[1]))^3 * sin(2*pi*x[2])/(x[1]^3*(x[1] + x[2]))  
# 目标函数的梯度  
gr <- function(x) {  
  numDeriv::grad(fn, c(x[1], x[2]))  
}  
  
hin <- function(x) {  
  c(  
    x[1]^2 - x[2] + 1,  
    1 - x[1] + (x[2] - 4)^2  
  )  
}  
  
hin.jac <- function(x) {  
  matrix(c(  
    2 * x[1], -1,  
    -1, 2 * x[2]  
,  
    ncol = 2, byrow = TRUE  
)  
}  
  
# 初始值  
p0 <- c(2, 5)  
# 定义目标规划  
op <- OP(  
  objective = F_objective(F = fn, n = 2L, G = gr), # 2 个目标变量  
  constraints = F_constraint(  
    F = list(hin = hin),
```



```
dir = c("<=", "<="),
rhs = c(0, 0),
# 不等式约束的雅可比
J = list(hin.jac = hin.jac)
),
bounds = V_bound(ld = 0, ud = 10, nobj = 2L),
maximum = TRUE # 求最大
)
nlp <- ROI_solve(op, solver = "nloptr.isres", start = p0)
nlp$solution
```

```
## [1] 1.227975 4.245365
```

```
nlp$objval
```

```
## [1] 0.09582504
```

下面再给一个来自 [Octave 优化文档](#) 的示例，该优化问题包含多个非线性的等式约束。

$$\begin{aligned} \min_x \quad & e^{\prod_{i=1}^5 x_i} - \frac{1}{2}(x_1^3 + x_2^3 + 1)^2 \\ \text{s.t.} \quad & \begin{cases} \sum_{i=1}^5 x_i^2 - 10 = 0 \\ x_2 x_3 - 5x_4 x_5 = 0 \\ x_1^3 + x_2^3 + 1 = 0 \end{cases} \end{aligned}$$

```
# 一个 5 维的目标函数
fn <- function(x) {
  exp(prod(x)) - 0.5 * (x[1]^3 + x[2]^3 + 1)^2
}
# 目标函数的梯度
gr <- function(x) {
  c(
    exp(prod(x))*prod(x[-1]) - 3*(x[1]^3 + x[2]^3 + 1)*x[1]^2,
    exp(prod(x))*prod(x[-2]) - 3*(x[1]^3 + x[2]^3 + 1)*x[2]^2,
    exp(prod(x))*prod(x[-3]),
    exp(prod(x))*prod(x[-4]),
    exp(prod(x))*prod(x[-5])
  )
}
# 等式约束
heq <- function(x) {
  c(
    sum(x^2) - 10,
    x[2] * x[3] - 5 * x[4] * x[5],
    x[1]^3 + x[2]^3 + 1
  )
}
# 等式约束的雅可比
```



```
heq.jac <- function(x) {
  matrix(c(2 * x[1], 2 * x[2], 2 * x[3], 2 * x[4], 2 * x[5],
  0, x[3], x[2], -5 * x[5], -5 * x[4],
  3 * x[1]^2, 3 * x[2]^2, 0, 0, 0),
  ncol = 5, byrow = TRUE
)
}

# 定义目标规划
op <- OP(
  objective = F_objective(F = fn, n = 5L, G = gr), # 5 个目标变量
  constraints = F_constraint(
    F = list(heq = heq),
    dir = "==" ,
    rhs = 0,
    # 等式的雅可比
    J = list(heq.jac = heq.jac)
  ),
  bounds = V_bound(ld = -Inf, ud = Inf, nobj = 5L),
  maximum = FALSE # 求最小
)
```

调用 SQP (序列二次规划) 求解器

```
nlp <- ROI_solve(op, solver = "nloptr.slsqp", start = c(-1.8, 1.7, 1.9, -0.8, -0.8))
nlp$solution
```

```
## [1] -1.7171435 1.5957096 1.8272458 -0.7636431 -0.7636431
```

计算结果和 Octave 的示例一致。

#### 13.4.3.6 含复杂约束条件

$$\begin{aligned} \min_x \quad & \exp(\sin(50 \cdot x)) + \sin(60 \cdot \exp(y)) + \sin(70 \cdot \sin(x)) \\ & + \sin(\sin(80 \cdot y)) - \sin(10 \cdot (x + y)) + \frac{(x^2 + y^2)^{\sin(y)}}{4} \\ s.t. \quad & \begin{cases} x - ((\cos(y))^x - x)^y = 0 \\ -50 \leq x_1, x_2 \leq 50 \end{cases} \end{aligned}$$

Lingo 代码如下：

```
Min = @exp(@sin(50 * x)) + @sin(60 * @exp(y)) + @sin(70 * @sin(x))
      + @sin(@sin(80 * y)) - @sin(10 * (x + y)) + (x^2 + y^2)^{@sin(y)} / 4;

x - (( @cos(y) )^x - x)^y = 0;

@bnd(-50, x, 50);
@bnd(-50, y, 50);
```



启用全局优化求解器, 求解 14 分钟, 在 (0.08256372, 24.56510) 取得极小值 -2.863497。不启用全局优化器就无法解, Lingo 会报错, 找不到最优解, 勉强找到一个可行解 (0.06082750, 44.12793), 目标值为 -1.29816。

```
fn <- function(x) {
  exp(sin(50 * x[1])) + sin(60 * exp(x[2])) +
  sin(70 * sin(x[1])) + sin(sin(80 * x[2])) -
  sin(10 * (x[1] + x[2])) + (x[1]^2 + x[2]^2)^(sin(x[2])) / 4
}

gr <- function(x){
  numDeriv::grad(fn, c(x[1], x[2]))
}

heq <- function(x){
  x[1] - ( (cos(x[2]))^x[1] - x[1] )^x[2]
}

heq.jac <- function(x){
  numDeriv::grad(heq, c(x[1], x[2]))
}

fn(x = c(0.06082750, 44.12793))

## [1] -1.29816
fn(x = c(1, 0))

## [1] 1.966877
heq(x = c(0.06082750, 44.12793))

## [1] 1.923673e-08
heq(x = c(1, 0))

## [1] 0

# 定义目标规划
op <- OP(
  objective = F_objective(F = fn, n = 2L, G = gr), # 2 个目标变量
  constraints = F_constraint(
    F = list(heq = heq),
    dir = "==" ,
    rhs = 0,
    J = list(heq.jac = heq.jac)
  ),
  bounds = V_bound(lb = -50, ub = 50, nobj = 2L),
  maximum = FALSE # 求最小
)
```

nloptr.auglag 无法求解此优化问题

```
nlp <- ROI_solve(op, solver = "nloptr.auglag", start = c(1, 0))
nlp$solution
```



调 nloptr.isres 求解器，每次执行都会得到不同的局部最优解

```
nlp <- ROI_solve(op, solver = "nloptr.isres", start = c(1, 0))
nlp$solution
```

```
## [1] 40.296583 5.091255
```

```
nlp$objval
```

```
## [1] -3.186307
```

比如下面三组

```
fn(x = c(40.95941, 41.52914))
```

```
## [1] -1.025926
```

```
heq(x = c(40.95941, 41.52914))
```

```
## [1] NaN
```

```
fn(x = c(-21.88091, 28.96994))
```

```
## [1] -1.467513
```

```
heq(x = c(-21.88091, 28.96994))
```

```
## [1] NaN
```

```
fn(x = c(-49.921967437, 4.8499336803))
```

```
## [1] -3.466596
```

```
heq(x = c(-49.921967437, 4.8499336803))
```

```
## [1] -8.515447e+208
```

## 13.5 非线性方程

### 13.5.1 一元非线性方程

牛顿-拉弗森方法

```
library(rootSolve)
```

### 13.5.2 非线性方程组

```
library(BB)
```

二项混合泊松分布的参数最大似然估计

```
poissmix.loglik <- function(p, y) {
  # Log-likelihood for a binary Poisson mixture distribution
  i <- 0:(length(y) - 1)
```

```
loglik <- y * log(p[1] * exp(-p[2]) * p[2]^i / exp(lgamma(i + 1)) +
  (1 - p[1]) * exp(-p[3]) * p[3]^i / exp(lgamma(i + 1)))

  sum(loglik)
}

# Data from Hasselblad (JASA 1969)
# 介绍应用场景
poissmix.dat <- data.frame(death = 0:9,
                           freq = c(162, 267, 271, 185, 111, 61, 27, 8, 3, 1))
lo <- c(0, 0, 0) # lower limits for parameters
hi <- c(1, Inf, Inf) # upper limits for parameters
p0 <- runif(3, c(0.2, 1, 1), c(0.8, 5, 8))
# a randomly generated vector of length 3
y <- c(162, 267, 271, 185, 111, 61, 27, 8, 3, 1)

ans1 <- spg(
  par = p0, fn = poissmix.loglik, y = y, lower = lo, upper = hi,
  control = list(maximize = TRUE, trace = FALSE)
)
ans2 <- BBoptim(
  par = p0, fn = poissmix.loglik, y = y,
  lower = lo, upper = hi, control = list(maximize = TRUE)
)

## iter: 0 f-value: -2136.431 pgrad: 236.9752
## iter: 10 f-value: -1995.89 pgrad: 2.961353
## iter: 20 f-value: -2041.139 pgrad: 2.57697
## iter: 30 f-value: -1989.974 pgrad: 0.4742151
## iter: 40 f-value: -1989.949 pgrad: 0.2614752
## iter: 50 f-value: -1989.946 pgrad: 0.01959506
## iter: 60 f-value: -1989.946 pgrad: 0.002494289
## Successful convergence.

ans2

## $par
## [1] 0.3598829 1.2560906 2.6634011
##
## $value
## [1] -1989.946
##
## $gradient
## [1] 0.0001000444
##
## $fn.reduction
## [1] -146.4848
```

```

## 
## $iter
## [1] 68
## 
## $feval
## [1] 170
## 
## $convergence
## [1] 0
## 
## $message
## [1] "Successful convergence"
## 
## $cpar
## method      M
##      2      50

```

计算最大似然处的黑塞矩阵以及参数的标准差

```

hess <- numDeriv:::hessian(x = ans2$par, func = poissmix.loglik, y = y)
# Note that we have to supplied data vector 'y'
hess

##          [,1]      [,2]      [,3]
## [1,] -907.1105  270.2287  341.2543
## [2,]  270.2287 -113.4794 -61.6819
## [3,]  341.2543 -61.6819 -192.7822

se <- sqrt(diag(solve(-hess)))
se

## [1] 0.1946836 0.3500308 0.2504769

```

从不同初始值出发尝试寻找全局最大值，实际找的是一系列局部最大值

```

# 3 randomly generated starting values
p0 <- matrix(runif(30, c(0.2, 1, 1), c(0.8, 8, 8)), 10, 3, byrow = TRUE)
ans <- multiStart(
  par = p0, fn = poissmix.loglik, action = "optimize",
  y = y, lower = lo, upper = hi, control = list(maximize = TRUE)
)

## Parameter set : 1 ...
## iter: 0  f-value: -2076.377  pgrad: 266.5811
## iter: 10  f-value: -1991.788  pgrad: 3.394882
## iter: 20  f-value: -1990.932  pgrad: 8.266675
## iter: 30  f-value: -1989.958  pgrad: 0.2441652
## iter: 40  f-value: -1989.946  pgrad: 0.001411991
##   Successful convergence.

## Parameter set : 2 ...

```



```
## iter:  0  f-value:  -3999.343  pgrad:  6.350898
## iter:  10  f-value:  -2015.457  pgrad:  2.400803
##   Successful convergence.

## Parameter set :  3 ...
## iter:  0  f-value:  -2526.385  pgrad:  3.959104
## iter:  10  f-value:  -1997.785  pgrad:  4.651176
## iter:  20  f-value:  -2041.124  pgrad:  130.6335
## iter:  30  f-value:  -1989.979  pgrad:  0.4133676
## iter:  40  f-value:  -1989.953  pgrad:  0.2001525
## iter:  50  f-value:  -1989.946  pgrad:  0.02953584
##   Successful convergence.

## Parameter set :  4 ...
## iter:  0  f-value:  -4036.966  pgrad:  7.725057
## iter:  10  f-value:  -1993.146  pgrad:  3.356279
## iter:  20  f-value:  -1992.445  pgrad:  3.162911
## iter:  30  f-value:  -1999.964  pgrad:  3.124857
## iter:  40  f-value:  -1990.201  pgrad:  0.9762675
## iter:  50  f-value:  -1989.962  pgrad:  0.3950169
## iter:  60  f-value:  -1989.946  pgrad:  0.0507498
## iter:  70  f-value:  -1989.946  pgrad:  0.0001978151
##   Successful convergence.

## Parameter set :  5 ...
## iter:  0  f-value:  -2048.809  pgrad:  2.862445
## iter:  10  f-value:  -1992.344  pgrad:  2.68979
## iter:  20  f-value:  -1990.604  pgrad:  7.2791
## iter:  30  f-value:  -1989.978  pgrad:  0.3772993
## iter:  40  f-value:  -1989.946  pgrad:  0.004172307
## iter:  50  f-value:  -1989.946  pgrad:  0.004260983
##   Successful convergence.

## Parameter set :  6 ...
## iter:  0  f-value:  -4777.283  pgrad:  7.596832
## iter:  10  f-value:  -1991.838  pgrad:  11.02078
## iter:  20  f-value:  -1990.272  pgrad:  0.5307333
## iter:  30  f-value:  -1989.963  pgrad:  2.230793
## iter:  40  f-value:  -1989.946  pgrad:  0.008421921
## iter:  50  f-value:  -1989.946  pgrad:  0.0001841727
##   Successful convergence.

## Parameter set :  7 ...
## iter:  0  f-value:  -2019.928  pgrad:  3.485709
## iter:  10  f-value:  -1990.626  pgrad:  1.833378
## iter:  20  f-value:  -1989.999  pgrad:  1.098717
## iter:  30  f-value:  -1989.947  pgrad:  0.3092782
## iter:  40  f-value:  -1989.946  pgrad:  0.007039489
##   Successful convergence.

## Parameter set :  8 ...
```



```
## iter:  0  f-value:  -2764.625  pgrad:  4.891128
## iter:  10  f-value:  -2001.398  pgrad:  2.273737e-06
##   Successful convergence.
## Parameter set :  9 ...
## iter:  0  f-value:  -2167.165  pgrad:  195.5499
## iter:  10  f-value:  -2001.54  pgrad:  2.194864
## iter:  20  f-value:  -2000.825  pgrad:  0.6559458
## iter:  30  f-value:  -1992.777  pgrad:  7.064828
## iter:  40  f-value:  -1991.747  pgrad:  3.357115
## iter:  50  f-value:  -1989.983  pgrad:  2.772795
## iter:  60  f-value:  -1989.946  pgrad:  0.03392643
## iter:  70  f-value:  -1989.946  pgrad:  0.0003728928
##   Successful convergence.
## Parameter set :  10 ...
## iter:  0  f-value:  -2100.94  pgrad:  317.5313
## iter:  10  f-value:  -1991.327  pgrad:  2.7843
## iter:  20  f-value:  -1990.415  pgrad:  1.435174
## iter:  30  f-value:  -1990.046  pgrad:  3.248585
## iter:  40  f-value:  -1989.946  pgrad:  0.06813025
## iter:  50  f-value:  -1989.946  pgrad:  0.001450644
##   Successful convergence.

# selecting only converged solutions
pmat <- round(cbind(ans$fvalue[ans$conv], ans$par[ans$conv, ]), 4)
dimnames(pmat) <- list(NULL, c("fvalue", "parameter 1", "parameter 2", "parameter 3"))
pmat[!duplicated(pmat), ]
```

	fvalue	parameter 1	parameter 2	parameter 3
## [1,]	-1989.946	0.6401	2.6634	1.2561
## [2,]	-1997.263	0.4922	2.4559	1.8567
## [3,]	-1989.946	0.3599	1.2561	2.6634
## [4,]	-2000.039	0.7931	2.0681	2.4778
## [5,]	-1989.946	0.3599	1.2560	2.6634

用一个具体的参数估计问题，求极大似然点，混合正态分布隐函数方程组求解非线性方程组 [Varadhan and Gilbert, 2009]

## 13.6 多目标规划

多目标规划的基本想法是将多目标问题转化为单目标问题，常见方法有理想点法、线性加权法、非劣解集法、极大极小法。理想点法是先在给定约束条件下分别求解单个目标的最优值，构造新的单目标函数。线性加权法是给每个目标函数赋予权重系数，各个权重系数之和等于1。非劣解集法是先求解其中一个单目标函数的最优值，然后将其设为等式约束，将其最优值从最小值开始递增，然后求解另一个目标函数的最小值。极大极小法是采用标准的简面体爬山法和通用全局优化法求解多目标优化问题。

R 环境中，**GPareto** 主要用来求解多目标规划问题。[试验设计和过程优化与 R 语言](#) 的 [约束优化](#) 章节，[优化](#)



和解方程。另外,《Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques》[Deb, 2005] 多目标优化方法

$$\begin{aligned} \min_x \quad & \begin{cases} f_1(x) = 0.5x_1 + 0.6x_2 + 0.7 \exp\left(\frac{x_1+x_3}{10}\right) \\ f_2(x) = (x_1 - 2x_2)^2 + (2x_2 - 3x_3)^2 + (5x_3 - x_1)^2 \end{cases} \\ \text{s.t.} \quad & x_1 \in [10, 80], x_2 \in [20, 90], x_3 \in [15, 100] \end{aligned}$$

```
library(DiceKriging)
library(emoa)
library(GPareto)
library(DiceDesign)

library(Ternary)
TernaryPlot(
  atip = "Top", btip = "Bottom", ctip = "Right",
  axis.col = "red", col = rgb(0.8, 0.8, 0.8)
)
HorizontalGrid(grid.lines = 2, grid.col = "blue", grid.lty = 1)
```

## 13.7 经典优化问题

旅行商问题、背包问题、指派问题、选址问题、网络流量问题

规划快递员送餐的路线: 从快递员出发地到各个取餐地, 再到顾客家里, 如何规划路线使得每个顾客下单到拿到餐的时间间隔小于 50 分钟, 完成送餐, 快递员的总时间最少?

## 13.8 回归与优化

简单线性回归

`nlsr`

是否能给大家提供一些思路?

Lasso [Tibshirani, 1996]

Least Angle Regression [Efron et al., 2004]

为了解决 Lasso 的有偏估计问题, 自适应 Lasso、松弛 Lasso SCAD (Smoothly Clipped Absolute Deviation)[Kim et al., 2008] MCP (Minimax Concave Penalty)[Zhang, 2010]

由于缺少高效的求解算法, Lasso 在高维小样本特征选择研究中没有广泛流行, 最小角回归 (Least Angle Regression, LAR) 算法 [Efron et al., 2004] 的出现有力促进了 Lasso 在高维小样本数据中的应用

`bestsubset` 最优子集回归

经典的普通最小二乘、广义最小二乘、岭回归、逐步回归、Lasso 回归、最优子集回归都可转化为优化问题, 一般形式如下

$$\hat{\theta}_{\lambda_n} \in \arg \min_{\theta \in \Omega} \left\{ \underbrace{\mathcal{L}(\theta; Z_1^n)}_{\text{损失函数}} + \lambda_n \underbrace{\mathcal{R}(\theta)}_{\text{正则化项}} \right\}.$$

下面尝试以 nloptr 包的优化器来展示求解过程，并与 Base R、glmnet 和 MASS 实现的回归模型比较。

$$\arg \min_{\beta, \lambda} \frac{1}{2} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \|\beta\|_1$$

其中,  $X \in \mathbb{R}^{m \times n}$ ,  $y \in \mathbb{R}^m$ ,  $\beta \in \mathbb{R}^n$ ,  $0 < \lambda \in \mathbb{R}$

$$y = X\beta + \epsilon$$

$$\hat{\beta} = (X^\top X)^{-1} X^\top y, \quad \hat{Y} = X(X^\top X)^{-1} X^\top y$$

```
set.seed(123)
n <- 200
p <- 50
x <- matrix(rnorm(n * p), n)
y <- rnorm(n)
lm(y ~ x + 0)
# y 的估计
# 教科书版
fit_base = function(x, y) {
  x %*% solve(t(x) %*% x) %*% t(x) %*% y
}
# 先向量计算, 然后矩阵计算
fit_vector = function(x, y) {
  x %*% (solve(t(x) %*% x) %*% (t(x) %*% y))
}
#  $X'X$  是对称的, 防止求逆
fit_inv = function(x, y) {
  x %*% solve(crossprod(x), crossprod(x, y))
}
```

QR 分解  $X_{n \times p} = Q_{n \times p} R_{p \times p}$ ,  $n > p$ ,  $Q^\top Q = I$ ,  $R$  是上三角矩阵,  $\hat{Y} = X(X^\top X)^{-1} X^\top y = Q Q^\top y$

```
fit_qr <- function(x, y) {
  decomp <- qr(x)
  qr.qy(decomp, qr.qty(decomp, y))
}
lm.fit(x, y)
```

若  $A = X^\top X$  是正定矩阵, 则  $A = LL^\top$ ,  $L$  是下三角矩阵

```
fit_chol <- function(x, y) {
  decomp <- chol(crossprod(x))
  lxy <- backsolve(decomp, crossprod(x, y), transpose = TRUE)
  b <- backsolve(decomp, lxy)
```



```

    x %*% b
}

## Using C/C++
system.time(RcppEigen::fastLmPure(x, y, method = 1)) ## QR
system.time(RcppEigen::fastLmPure(x, y, method = 2)) ## Cholesky
system.time(RcppArmadillo::fastLmPure(x, y, method = 1)) ## QR
system.time(RcppArmadillo::fastLmPure(x, y, method = 2)) ## Cholesky

```

## 13.9 对数似然

随机变量  $X$  服从参数为  $\lambda > 0$  的指数分布, 密度函数  $p(x)$  为

$$p(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

其中,  $\lambda > 0$ , 下面给定一系列模拟样本观察值  $x_1, x_2, \dots, x_n$ , 估计参数  $\lambda$ 。对数似然函数  $\ell(\lambda) = \log \prod_{i=1}^n f(x_i) = n \log \lambda - \lambda \sum_{i=1}^n x_i$ 。解此方程即可得到  $\lambda$  的极大似然估计  $\lambda_{mle} = \frac{1}{\bar{X}}$ , 极大值  $\ell(\lambda_{mle}) = -n(1 + \log \bar{X})$ 。

根据上述样本, 计算样本均值  $(\mu - 1.5 * \sigma / \sqrt{n}, \mu + 1.5 * \sigma / \sqrt{n})$  和方差  $(0.8\sigma, 1.5\sigma)$ 。已知正态分布  $f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$  的对数似然形式  $\ell(\mu, \sigma^2) = \log \prod_{i=1}^n f(x_i) = \sum_{i=1}^n \log f(x_i)$ 。正态分布的密度函数的对数可用 `dnorm(..., log = TRUE)` 计算。

生成服从指数分布的样本, 计算样本的均值和方差, 依据均值和方差构造区间, 然后将区间网格化, 在此网格上绘制正态分布的对数似然函数。绕那么大一个圈子, 其实就是绘制正态分布的对数似然函数。

```

set.seed(2021)
n <- 20 # 随机数的个数
x <- rexp(n, rate = 5) # 服从指数分布的随机数
m <- 40 # 网格数

mu <- seq(
  mean(x) - 1.5 * sd(x) / sqrt(n),
  mean(x) + 1.5 * sd(x) / sqrt(n),
  length.out = m
)
sigma <- seq(0.8 * sd(x), 1.5 * sd(x), length.out = m)
df <- expand.grid(x = mu, y = sigma)
# 正态分布的对数似然
loglik <- function(b, x0) -sum(dnorm(x0, b[1], b[2], log = TRUE))

df$fnxy = apply(df, 1, loglik, x0 = x)

wireframe(
  data = df, fnxy ~ x * y,

```

```
shade = TRUE, drape = FALSE,
xlab = expression(mu),
ylab = expression(sigma),
zlab = list(expression(-loglik(mu, sigma)), rot = 90),
scales = list(arrows = FALSE, col = "black"),
par.settings = list(axis.line = list(col = "transparent")),
screen = list(z = 120, x = -70, y = 0)
)
```

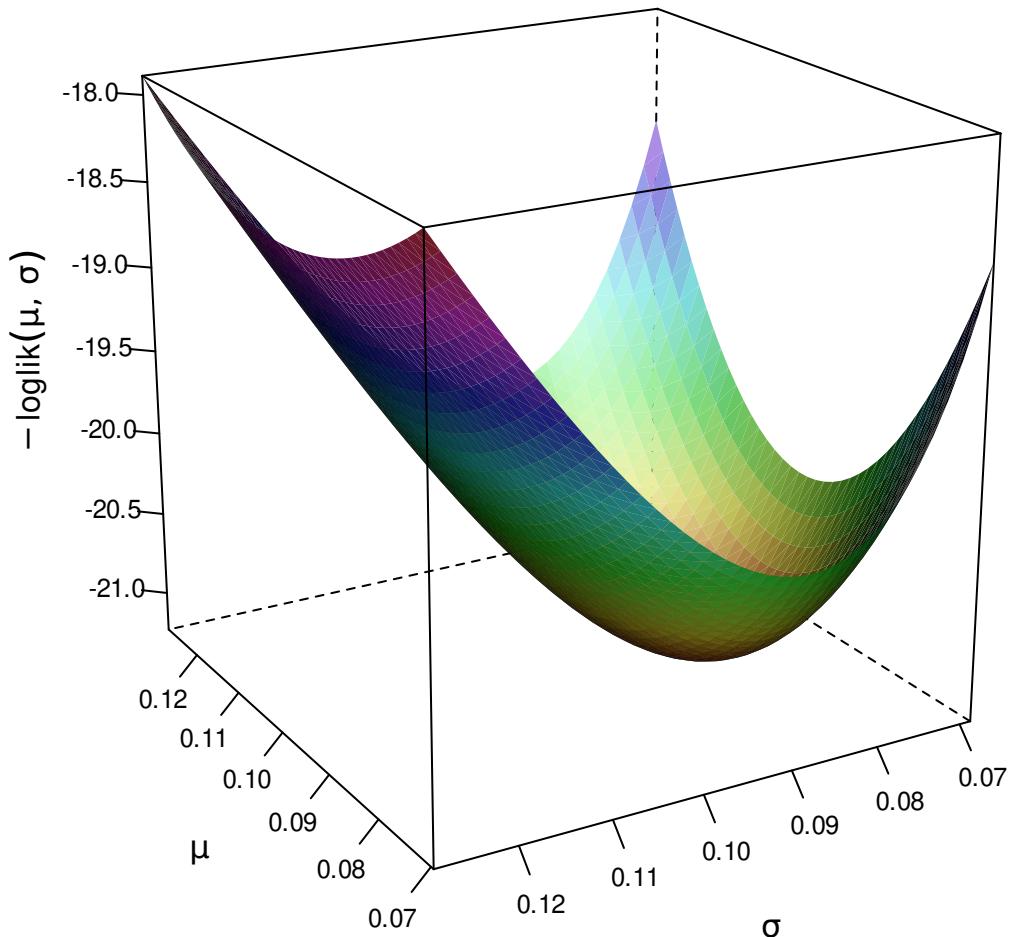


图 13.15: 正态分布参数的负对数似然函数

## 13.10 微分方程

### ode45 求解偏微分方程

**pracma** 实现了 `ode23`, `ode23s`, `ode45` 等几个自适应的 Runge-Kutta 求解器, **deSolve** 包求解 ODE (常微分方程), DAE (微分代数方程), DDE (延迟微分方程, 包含刚性和非刚性方程) 和 PDE (偏微分方程), **bvpSolve** 包求解 DAE/ODE 方程的边值问题。 **ReacTran** [Soetaert and Meysman, 2012] 可将偏微分方程转为常微分方程组, 解决反应运输问题, 在笛卡尔、极坐标、圆柱形和球形网格上离散偏微分方程。 **sundials** 提供一系列非线性方程、常微分方程、微分代数方程求解器, Satyaprakash Nayak 开发了相应的 **sundialr**



包。

### 13.10.1 常微分方程

洛伦兹系统是一个常微分方程组，系统参数的默认值为  $(\sigma = 10, \rho = 28, \beta = 8/3)$ ，初值为  $(-13, -14, 47)$ 。

$$\begin{cases} \frac{\partial x}{\partial t} = \sigma(y - x) \\ \frac{\partial y}{\partial t} = x(\rho - z) - y \\ \frac{\partial z}{\partial t} = xy - \beta z \end{cases}$$

```
library(deSolve)
# 参数
pars <- c(a = -8 / 3, b = -10, c = 28)
# 初值
state <- c(X = 1, Y = 1, Z = 1)
# 时间间隔
times <- seq(0, 100, by = 0.01)
# 定义方程组
lorenz_fun <- function(t, state, parameters) {
  with(as.list(c(state, parameters)), {
    dX <- a * X + Y * Z
    dY <- b * (Y - Z)
    dZ <- -X * Y + c * Y - Z
    list(c(dX, dY, dZ))
  })
}
out <- ode(
  y = state, times = times,
  func = lorenz_fun, parms = pars
)
```

调用 **scatterplot3d** 绘制三维曲线图，如图13.16 所示

```
library(scatterplot3d)

scatterplot3d(
  x = out[, "X"], y = out[, "Y"], z = out[, "Z"],
  col.axis = "black", type = "l", color = "gray",
  xlab = expression(x), ylab = expression(y), zlab = expression(z),
  col.grid = "gray", main = "Lorenz"
)
```

### 13.10.2 偏微分方程

ReacTran 的几个关键函数介绍

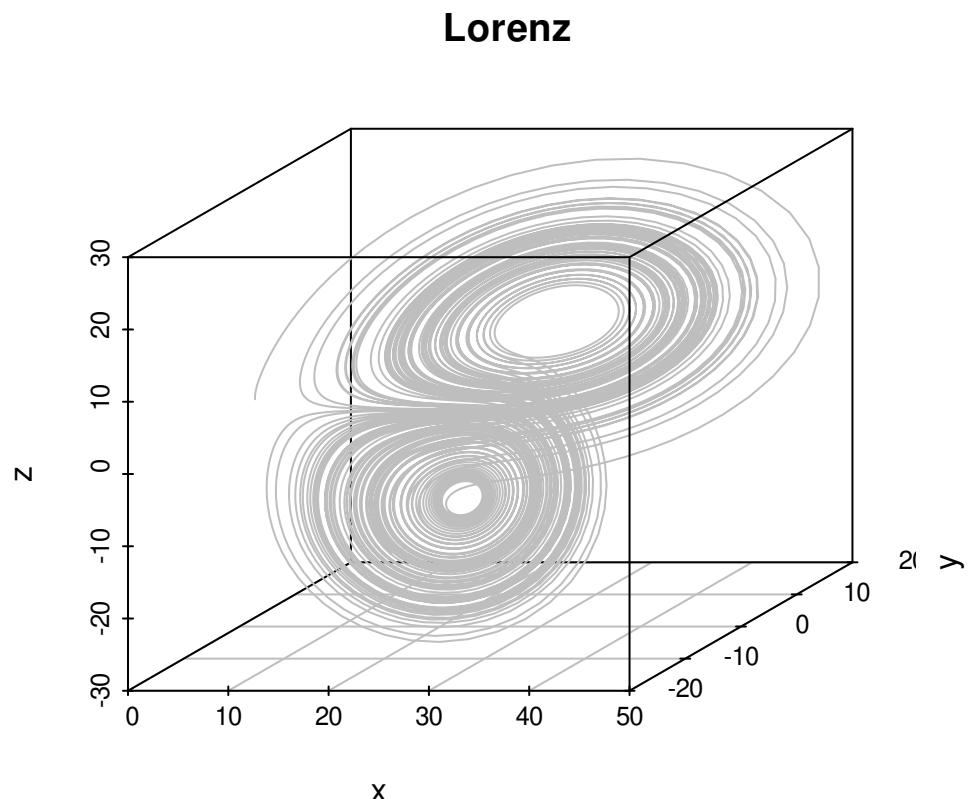


图 13.16: 洛伦兹曲线



## 一维热传导方程

$$\left\{ \begin{array}{l} \frac{\partial y}{\partial t} = D \frac{\partial^2 y}{\partial x^2} \end{array} \right.$$

参数  $D = 0.01$ , 边界条件  $y_{t,x=0} = 0, y_{t,x=1} = 1$ , 初始条件  $y_{t=0,x} = \sin(\pi x)$ 。

```
(C) library(ReacTran)

N <- 100
xgrid <- setup.grid.1D(x.up = 0, x.down = 1, N = N)
x <- xgrid$x.mid
D.coeff <- 0.01
Diffusion <- function(t, Y, parms) {
  tran <- tran.1D(
    C = Y, C.up = 0, C.down = 1,
    D = D.coeff, dx = xgrid
  )
  list(
    dY = tran$dc,
    flux.up = tran$flux.up,
    flux.down = tran$flux.down
  )
}
yini <- sin(pi * x)
times <- seq(from = 0, to = 5, by = 0.01)
out <- ode.1D(
  y = yini, times = times, func = Diffusion,
  parms = NULL, dimens = N
)

image(out,
  grid = xgrid$x.mid, xlab = "times",
  ylab = "Distance", main = "PDE", add.contour = TRUE
)
```

## 二维拉普拉斯方程

$$\left\{ \begin{array}{l} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \end{array} \right.$$

边界条件

$$\left\{ \begin{array}{l} u_{x=0,y} = u_{x=1,y} = 0 \\ \frac{\partial u_{x,y=0}}{\partial y} = 0 \\ \frac{\partial u_{x,y=1}}{\partial y} = \pi \sinh(\pi) \sin(\pi x) \end{array} \right.$$

它有解析解

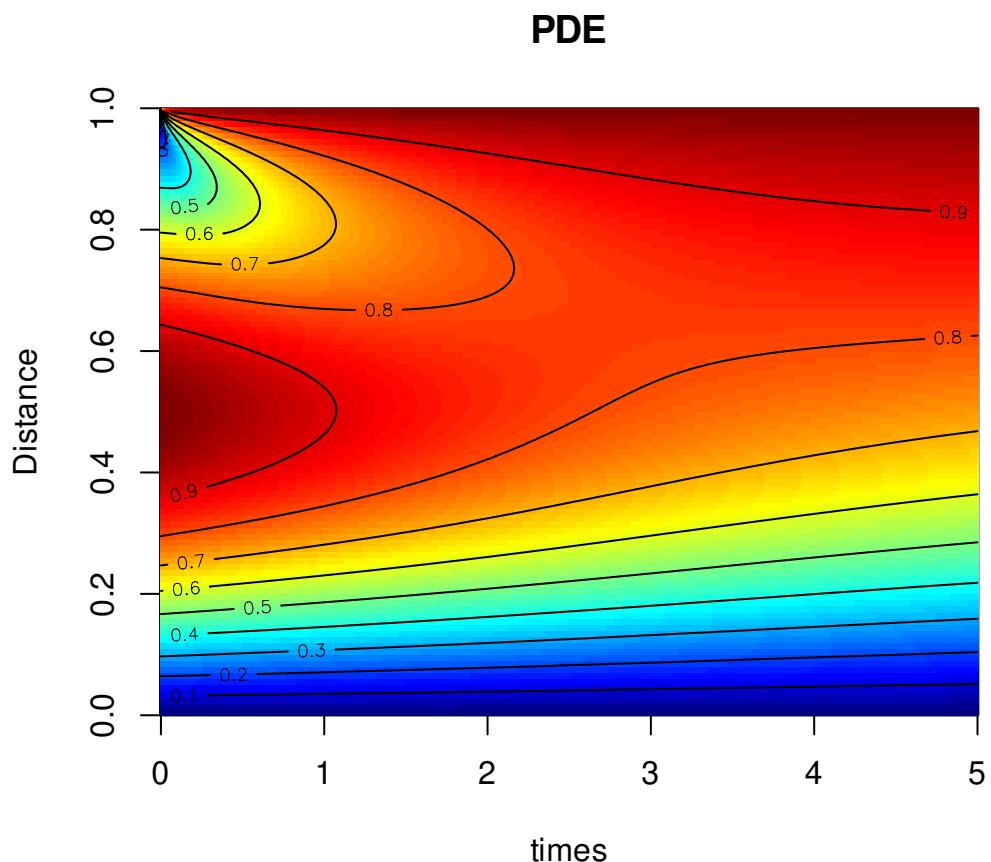


图 13.17: 一维热传导方程的数值解热力图

$$u(x, y) = \sin(\pi x) \cosh(\pi y)$$

其中  $x \in [0, 1], y \in [0, 1]$

④

```
fn <- function(x, y) {
  sin(pi * x) * cosh(pi * y)
}
x <- seq(0, 1, length.out = 101)
y <- seq(0, 1, length.out = 101)
z <- outer(x, y, fn)

image(z, col = terrain.colors(20))
contour(z, method = "flat", add = TRUE, lty = 1)
```

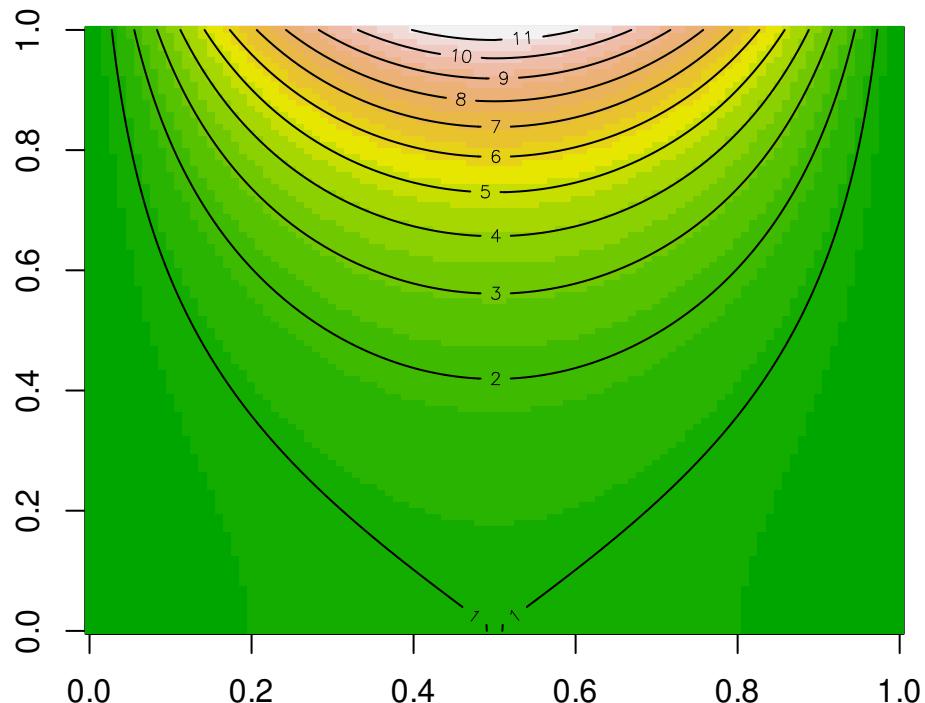


图 13.18: 解析解的二维图像

```
persp(z,
  theta = 30, phi = 20,
  r = 50, d = 0.1, expand = 0.5, ltheta = 90, lphi = 180,
  shade = 0.1, ticktype = "detailed", nticks = 5, box = TRUE,
  col = drapocol(z, col = terrain.colors(20)),
  border = "transparent",
```

```
  xlab = "X", ylab = "Y", zlab = "Z",
  main = ""
)
```

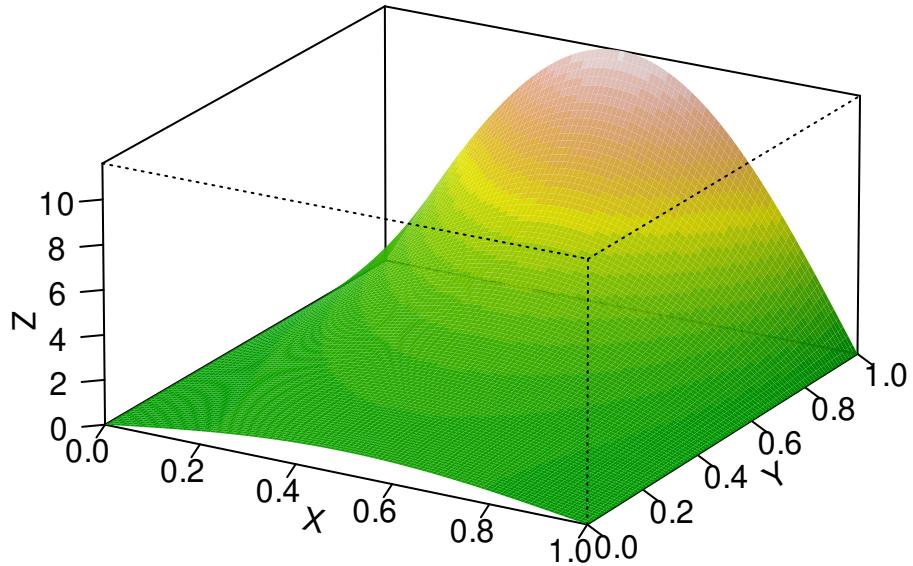


图 13.19: 解析解的三维透視图像

求解 PDE

```
dx <- 0.2
xgrid <- setup.grid.1D(-100, 100, dx.1 = dx)
x <- xgrid$x.mid
N <- xgrid$N

uini <- exp(-0.05 * x^2)
vini <- rep(0, N)
yini <- c(uini, vini)
times <- seq(from = 0, to = 50, by = 1)

wave <- function(t, y, parms) {
  u1 <- y[1:N]
  u2 <- y[-(1:N)]
  du1 <- u2
```

```
du2 <- tran.1D(C = u1, C.up = 0, C.down = 0, D = 1, dx = xgrid)$dC
return(list(c(du1, du2)))
}

out <- ode.1D(
  func = wave, y = yini, times = times, parms = NULL,
  nspec = 2, method = "ode45", dimens = N, names = c("u", "v")
)
```

### 13.10.3 延迟微分方程

```
library(PBSddesolve) # DAE 延迟微分方程
```

**PBSddesolve** [Couture-Beil et al., 2019] PBSmodelling PBSmapping

**nlmeODE** 通过微分方程整合用于混合效应模型的 odesolve 和 nlme 包。

### 13.10.4 随机微分方程

**Sim.DiffProc**

随机微分方程入门：基于 R 语言的模拟和推断

```
# library(Sim.DiffProc)
```

**nlmixr** 借助 **RxODE** 求解基于常微分方程的非线性混合效应模型

## 13.11 运行环境

```
sessionInfo()

## R version 4.2.0 (2022-04-22)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.4 LTS
##
## Matrix products: default
## BLAS:    /usr/lib/x86_64-linux-gnublas/libblas.so.3.9.0
## LAPACK:  /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0
##
## locale:
## [1] LC_CTYPE=en_US.UTF-8      LC_NUMERIC=C
## [3] LC_TIME=en_US.UTF-8       LC_COLLATE=en_US.UTF-8
## [5] LC_MONETARY=en_US.UTF-8   LC_MESSAGES=en_US.UTF-8
## [7] LC_PAPER=en_US.UTF-8      LC_NAME=C
## [9] LC_ADDRESS=C              LC_TELEPHONE=C
```



```
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats      graphics   grDevices  utils      datasets   methods    base
##
## other attached packages:
## [1] Deriv_4.1.3           quadprog_1.5-8
## [3] kableExtra_1.3.4       tibble_3.1.7
## [5] nlme_3.1-157          PBSddesolve_1.12.6
## [7] ReacTran_1.4.3.1       shape_1.4.6
## [9] scatterplot3d_0.3-41   deSolve_1.32
## [11] BB_2019.10-1          rootSolve_1.8.2.3
## [13] kernlab_0.9-30        lattice_0.20-45
## [15] ROI.plugin.scs_1.1-1   ROI.plugin.quadprog_1.0-0
## [17] ROI.plugin.lpsolve_1.0-1 ROI.plugin.nloptr_1.0-0
## [19] ROI.plugin.alabama_1.0-0 ROI_1.0-0
## [21] lpSolve_5.6.15
##
## loaded via a namespace (and not attached):
## [1] svglite_2.1.0           sysfonts_0.8.8      digest_0.6.29
## [4] utf8_1.2.2              slam_0.1-50          R6_2.5.1
## [7] alabama_2022.4-1        evaluate_0.15        httr_1.4.3
## [10] pillar_1.7.0            rlang_1.0.2          curl_4.3.2
## [13] rstudioapi_0.13         nloptr_2.0.1          rmarkdown_2.14
## [16] webshot_0.5.3           stringr_1.4.0         munsell_0.5.0
## [19] compiler_4.2.0          numDeriv_2016.8-1.1   xfun_0.31
## [22] pkgconfig_2.0.3          systemfonts_1.0.4     htmltools_0.5.2
## [25] bookdown_0.26            viridisLite_0.4.0     fansi_1.0.3
## [28] crayon_1.5.1            grid_4.2.0            lifecycle_1.0.1
## [31] registry_0.5-1          magrittr_2.0.3         scales_1.2.0
## [34] cli_3.3.0               stringi_1.7.6         xml2_1.3.3
## [37] ellipsis_0.3.2          vctrs_0.4.1           lpSolveAPI_5.5.2.0-17.7
## [40] tools_4.2.0              glue_1.6.2            fastmap_1.1.0
## [43] yaml_2.3.5              colorspace_2.0-3      scs_3.0-0
## [46] rvest_1.0.2              knitr_1.39
```

## 附录 A 命令行操作

Bash 文件查找、查看（内容、大小）、移动（重命名）、删除、创建、修改权限

Linux 命令行工具是非常强大的，命令行中的数据科学 <https://www.datascienceatthecommandline.com/>，Linux 命令行 <https://github.com/jaywcjlove/linux-command>

`optparse`、`docopt`、`littler` 包提供了很多便捷的命令行工具，`sys`、`fs` 在 R 中运行操作系统命令

如表A.1所示，总结了 R 和 Shell 命令的等价表示，下面以 `list.files()` 和 `ls` 为例，介绍其等价的内容

表 A.1: R 和 Shell 命令的等价表示<sup>1</sup>

	R	Shell
查看文件	<code>list.files()</code>	<code>ls</code>
查看目录	<code>list.dirs()</code>	<code>dir</code>
目录层次	<code>fs::dir_tree()</code>	<code>tree</code>

### A.1 查看文件

`ls`/`mkdir`/`mv`/`du`

查看文件

```
```bash
ls -a
```

```

列出目录下所有文件

```
```bash
ls -1
```

```

一行显示一个文件或文件夹

<sup>1</sup>CentOS 系统默认没有安装 `tree` 软件，需要先安装才能使用此命令 `sudo dnf install -y tree`

```
```bash
ls -l
```
```

按从 aA-zZ 的顺序列出所有文件以及所属权限

```
```bash
ls -rl
```
```

相比于 `ls -l` 文件是逆序排列

```
```bash
ls -lh
```
```

列出文件或文件夹（不包含子文件夹）的大小

```
```bash
ls -ld
```
```

列出当前目录本身，而不是其所包含的内容

## A.2 创建文件夹

```
```bash
mkdir images
```
```

创建文件用 `touch` 如 `touch .Rprofile`

```
```bash
# 删除文件夹及子文件夹，递归删除
rm -rf images/
# 删除文件
rm .Rprofile
```
```

## A.3 移动文件

在当前目录下



```
```bash
# 移动文件夹 images 下的所有文件到 figures 文件夹下
mv images/* figures/
# images 文件夹移动到 figures 文件夹下
mv images/ figures/
# 移动特定的文件
mv images/*.png figures/
````
```

同一目录下有两个文件 `R-3.5.1.tar.gz` 未下载完整 和 `R-3.5.1.tar.gz.1` 完全下载

```
```bash
# 删除 R-3.5.1.tar.gz
rm R-3.5.1.tar.gz
# 重命名 R-3.5.1.tar.gz.1
mv R-3.5.1.tar.gz.1 R-3.5.1.tar.gz
````
```

## A.4 查看文件大小

当前目录下各文件夹的大小，`-h` 表示人类（相对于机器来说）可读的方式显示，如 Kb、Mb、Gb，`-d` 表示目录深度`

```
```bash
du -h -d 1 ./
````

```bash
# 对当前目录下的文件/夹 按大小排序
du -sh * | sort -nr
````
```

## A.5 终端模拟器

oh-my-zsh 是 Z Shell 扩展，开发在 Github 上 <https://github.com/ohmyzsh/ohmyzsh>。

zsh 相比于 bash，在语法高亮、自动补全等方面有优势

```
sudo dnf install -y zsh
sh -c "$(curl -fsSL https://raw.github.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

RStudio 集成的终端支持 Zsh，操作路径 Tools -> Global Options -> Terminal，见图 A.1

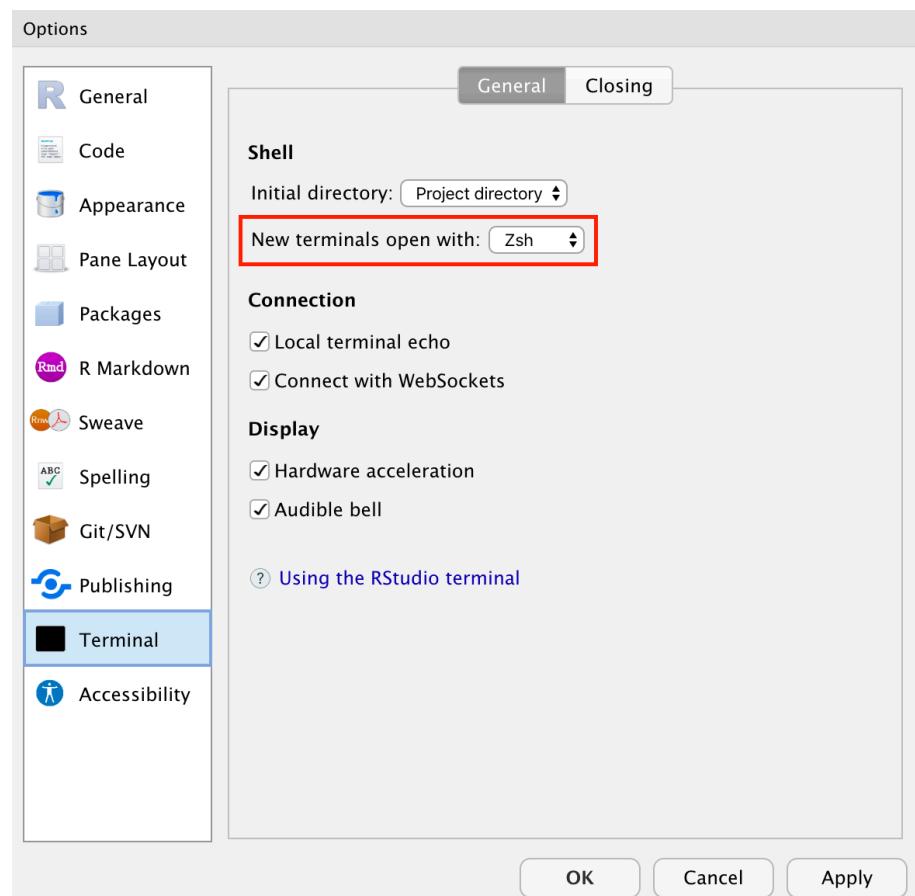


图 A.1: RStudio IDE 集成的 Zsh 终端模拟器



## A.6 压缩和解压缩

最常见的压缩文件格式有 .tar、.tar.gz、.tar.bz2、.zip 和 .rar，分别对应于 Tar <https://www.gnu.org/software/tar/>、Gzip <https://www.gzip.org/>、Bzip2 <https://www.bzip.org/>、UnZip/Zip <http://www.infozip.org> 和 WinRAR <https://www.rarlab.com/>。Tar 提供了基本的打包和解包工具，Gzip 和 Bzip2 在 Tar 打包的基础上提供了压缩功能，UnZip/Zip 是兼容 Windows 原生压缩/解压缩功能的程序，WinRAR 是广泛流行于 Windows 系统的压缩/解压缩收费软件，除了 WinRAR，其它都是免费甚至开源软件。下面以 .tar.gz 和 .tar.bz2 两种格式的压缩文件为例，介绍文件压缩和解压缩的操作，其它文件格式的操作类似<sup>2</sup>。WinRAR <https://www.rarlab.com/> 是收费的压缩和解压缩工具，也支持 Linux 和 macOS 系统，鉴于它是收费软件，这里就不多展开介绍了，详情请见官网。

```
sudo dnf install -y tar gzip zip unzip
# 将目录 ~/tmp 压缩成文件 filename.tar.gz
tar -czf filename.tar.gz ~/tmp
# 将文件 filename.tar.gz 解压到目录 ~/tmp
tar -xzf filename.tar.gz -C ~/tmp
```

解压不带 tar 的.gz 文件，比如 `tex.eps.gz` 解压后变成 `tex.eps`

```
gzip filename.gz -d ~/tmp
```

```
sudo dnf install -y bzip2
# 将目录 ~/tmp 压缩成文件 filename.tar.bz2
tar -cjf filename.tar.bz2 ~/tmp
# 将文件 filename.tar.bz2 解压到目录 ~/tmp
tar -xjf filename.tar.bz2 -C ~/tmp
```

## A.7 从仓库安装 R

### A.7.1 Ubuntu

安装 openssh zsh 和 Git

```
sudo apt-get install zsh openssh-server
sudo add-apt-repository -y ppa:git-core/ppa
sudo apt update && sudo apt install git
sh -c "$(curl -fsSL https://raw.githubusercontent.com/robbryussell/oh-my-zsh/master/tools/install.sh)"
```

只考虑最新的 Ubuntu 18.04 因为本书写成的时候，该版本应该已经大规模使用了，默认版本的安装和之前版本的安装就不再展示了。安装最新版 R-3.5.x，无论安装哪个版本，都要先导入密钥

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv E084DAB9
```

- Ubuntu 14.04.5 提供的默认版本 R 3.0.2，安装 R 3.5.x 系列之前的版本，如 R 3.4.4

```
sudo apt-add-repository -y "deb http://cran.rstudio.com/bin/linux/ubuntu `lsb_release -cs`/"
sudo apt-get install r-base-dev
```

添加完仓库后，都需要更新源 `sudo apt-get update`，安装 R 3.5.x 系列最新版

```
sudo apt-add-repository -y "deb https://mirrors.tuna.tsinghua.edu.cn/CRAN/bin/linux/ubuntu trusty-cra
```

- Ubuntu 16.04.5 提供的默认版本 R 3.4.4，这是 R 3.4.x 系列的最新版，安装目前最新的 R 3.5.x 版本需要

<sup>2</sup>zip 格式的文件需要额外安装 zip 和 unzip 两款软件实现压缩和解压缩。

```
sudo apt-add-repository -y "deb https://mirrors.tuna.tsinghua.edu.cn/CRAN/bin/linux/ubuntu xenial"
```

- Ubuntu 18.04.1 提供的默认版本 R 3.4.4，安装目前的最新版本需要

```
sudo apt-add-repository -y "deb https://mirrors.tuna.tsinghua.edu.cn/CRAN/bin/linux/ubuntu bionic"
```

接下来安装 R，详细安装指导见 [CRAN 官网](#)。

```
sudo apt-get install -y r-base-dev
```

Michael Rutter 维护了编译好的二进制版本 <https://launchpad.net/~marutter>，比如 rstan 包可以通过安装 r-cran-rstan 完成

```
# R packages for Ubuntu LTS. Based on CRAN Task Views.
```

```
sudo add-apt-repository -y ppa:marutter/c2d4u3.5
```

```
sudo apt-get install r-cran-rstan
```

## A.7.2 CentOS

同样适用于 Fedora

安装指导<sup>3</sup>

# A.8 源码安装

## A.8.1 Ubuntu

- 首先启用源码仓库，否则执行 sudo apt-get build-dep r-base 会报如下错误

```
E: You must put some 'source' URIs in your sources.list
```

```
sudo sed -i -- 's/#deb-src/deb-src/g' /etc/apt/sources.list && sudo sed -i -- 's/# deb-src/deb-src/g' /etc/apt/sources.list.d/rstudio-server.list
```

- 安装编译 R 所需的系统依赖

```
sudo apt-get build-dep r-base-dev
```

- 编译安装 R

```
./configure  
make && make install
```

- 自定义编译选项

```
./configure --help
```

## A.8.2 CentOS

基于 CentOS 7 和 GCC 4.8.5，参考 R-admin 手册

<sup>3</sup>在 CentOS 7 上打造 R 语言编程环境



- 下载源码包

最新发布的稳定版

```
curl -fLo ./R-latest.tar.gz https://mirrors.tuna.tsinghua.edu.cn/CRAN/src/base/R-latest.tar.gz
```

| % Total  | % Received | % Xferd | Average Speed | Time   | Time  | Time    | Current |              |
|----------|------------|---------|---------------|--------|-------|---------|---------|--------------|
|          |            |         | Dload         | Upload | Total | Spent   | Left    | Speed        |
| 10 28.7M | 10 3232k   | 0       | 0             | 107k   | 0     | 0:04:34 | 0:00:30 | 0:04:04 118k |

- 安装依赖

```
sudo yum install -y yum-utils epel-release && sudo yum-builddep R-devel  
sudo dnf update && sudo dnf builddep R-devel # Fedora 30
```

- 解压配置

```
mkdir R-latest && tar -xzf ./R-latest.tar.gz -C ./R-latest && cd R-3.5.2
```

```
./configure --enable-R-shlib --enable-byte-compiled-packages \  
--enable-BLAS-shlib --enable-memory-profiling
```

```
R is now configured for x86_64-pc-linux-gnu
```

```
Source directory: .
Installation directory: /usr/local

C compiler: gcc -std=gnu99 -g -O2
Fortran 77 compiler: gfortran -g -O2

Default C++ compiler: g++ -g -O2
C++98 compiler: g++ -std=gnu++98 -g -O2
C++11 compiler: g++ -std=gnu++11 -g -O2
C++14 compiler:
C++17 compiler:
Fortran 90/95 compiler: gfortran -g -O2
Obj-C compiler: gcc -g -O2 -fobjc-exceptions

Interfaces supported: X11, tcltk
External libraries: readline, curl
Additional capabilities: PNG, JPEG, TIFF, NLS, cairo, ICU
Options enabled: shared R library, shared BLAS, R profiling, memory profiling

Capabilities skipped:
Options not enabled:

Recommended packages: yes
```

- 编译安装

```
make -j 2 all  
sudo make install
```

- BLAS 加持 (可选)

BLAS对于加快矩阵计算至关重要,编译R带 [BLAS 支持](#),添加OpenBLAS 支持`--with-blas="-lopenblas"`或 ATLAS 支持`--with-blas="-L/usr/lib64/atlas -lsatlas"`

```
sudo yum install -y openblas openblas-threads openblas-openmp
```

```
./configure --enable-R-shlib --enable-byte-compiled-packages \  
--enable-BLAS-shlib --enable-memory-profiling \  
--with-blas="-lopenblas"
```

```
R is now configured for x86_64-pc-linux-gnu
```

```
Source directory: .  
Installation directory: /usr/local  
  
C compiler: gcc -std=gnu99 -g -O2  
Fortran 77 compiler: gfortran -g -O2  
  
Default C++ compiler: g++ -g -O2  
C++98 compiler: g++ -std=gnu++98 -g -O2  
C++11 compiler: g++ -std=gnu++11 -g -O2  
C++14 compiler:  
C++17 compiler:  
Fortran 90/95 compiler: gfortran -g -O2  
Obj-C compiler: gcc -g -O2 -fobjc-exceptions  
  
Interfaces supported: X11, tcltk  
External libraries: readline, **BLAS(OpenBLAS)**, curl  
Additional capabilities: PNG, JPEG, TIFF, NLS, cairo, ICU  
Options enabled: shared R library, shared BLAS, R profiling, memory profiling  
  
Capabilities skipped:  
Options not enabled:  
  
Recommended packages: yes
```

配置成功的标志,如 OpenBLAS

```
checking for dgemm_ in -lopenblas... yes  
checking whether double complex BLAS can be used... yes  
checking whether the BLAS is complete... yes
```

ATLAS 加持



```
sudo yum install -y atlas
./configure --enable-R-shlib --enable-byte-compiled-packages \
--enable-BLAS-shlib --enable-memory-profiling \
--with-blas="-L/usr/lib64/atlas -lsatlas"

R is now configured for x86_64-pc-linux-gnu

Source directory: .
Installation directory: /usr/local

C compiler: gcc -std=gnu99 -g -O2
Fortran 77 compiler: gfortran -g -O2

Default C++ compiler: g++ -g -O2
C++98 compiler: g++ -std=gnu++98 -g -O2
C++11 compiler: g++ -std=gnu++11 -g -O2
C++14 compiler:
C++17 compiler:
Fortran 90/95 compiler: gfortran -g -O2
Obj-C compiler: gcc -g -O2 -fobjc-exceptions

Interfaces supported: X11, tcltk
External libraries: readline, **BLAS(generic)**, curl
Additional capabilities: PNG, JPEG, TIFF, NLS, cairo, ICU
Options enabled: shared R library, shared BLAS, R profiling, memory profiling

Capabilities skipped:
Options not enabled:

Recommended packages: yes
```

ATLAS 配置成功

```
checking for dgemm_ in -L/usr/lib64/atlas -lsatlas... yes
checking whether double complex BLAS can be used... yes
checking whether the BLAS is complete... yes
```

后续步骤同上

## A.9 忍者安装

从源码自定义安装：加速 Intel MKL 和大文件支持

<https://software.intel.com/en-us/articles/using-intel-mkl-with-r>

## A.10 配置

### A.10.1 初始会话 .Rprofile

.Rprofile 文件位于 ~/ 目录下或者 R 项目的根目录下

查看帮助 ? .Rprofile

更多配置设置 [startup](#)

### A.10.2 环境变量 .Renvironment

.Renvironment 文件位于 ~/ 目录下

### A.10.3 编译选项 Makevars

Makevars 文件位于 ~/.R/ 目录下

## A.11 命令行参数

commandArgs 从终端命令行中传递参数

- [rdoc](#) 高亮 R 帮助文档中的 R 函数、关键字 NULL。启用需要在 R 控制台中执行 `rdoc::use_rdmc()`
- [radian](#) 代码自动补全和语法高亮，进入 R 控制台，终端中输入 `radian`
- [docopt](#) 提供 R 命令行工具，如 [littler](#) 包，[getopt](#) 从终端命令行接受参数
- [optparse](#) 命令行选项参数的解析器

安装完 R-littler R-littler-examples (centos) 或 littler r-cran-littler (ubuntu) 后，执行

```
# centos
sudo ln -s /usr/lib64/R/library/littler/examples/install.r /usr/bin/install.r
sudo ln -s /usr/lib64/R/library/littler/examples/install2.r /usr/bin/install2.r
sudo ln -s /usr/lib64/R/library/littler/examples/installGithub.r /usr/bin/installGithub.r
sudo ln -s /usr/lib64/R/library/littler/examples/testInstalled.r /usr/bin/testInstalled.r

# ubuntu
sudo ln -s /usr/lib/R/site-library/littler/examples/install.r /usr/bin/install.r
sudo ln -s /usr/lib/R/site-library/littler/examples/install2.r /usr/bin/install2.r
sudo ln -s /usr/lib/R/site-library/littler/examples/installGithub.r /usr/bin/installGithub.r
sudo ln -s /usr/lib/R/site-library/littler/examples/testInstalled.r /usr/bin/testInstalled.r
```

这样可以在终端中安装 R 包了

```
install.r docopt

#!/usr/bin/env Rscript
# 安装 optparse 提供更加灵活的传参方式
# 也可参考 littler https://github.com/eddelbuettel/littler
# if("optparse" %in% .packages(TRUE)) install.packages('optparse', repos = "https://cran.rstudio.com")
```



```
# https://cran.r-project.org/doc/manuals/R-intro.html#Invoking-R-from-the-command-line
# http://www.cureffi.org/2014/01/15/running-r-batch-mode-linux/
args = commandArgs(trailingOnly=TRUE)

# 函数功能: 在浏览器中同时打开多个 PDF 文档
open_pdf <- function(pdf_path = "./figures/", n = 1) {
  # pdf_path:      PDF文件所在目录
  # n:             默认打开1个PDF文档
  # PDF文档目录
  pdfs <- list.files(pdf_path, pattern = '\\.pdf$')
  # PDF 文档路径
  path_to_pdfs <- paste(pdf_path, pdfs, sep = .Platform$file.sep)
  # 打开 PDF 文档
  invisible(lapply(head(path_to_pdfs, n), browseURL))
}

open_pdf(pdf_path, n = args[1])

# 使用: Rscript --vanilla code/batch-open-pdf.R 20
```

## A.12 从源码安装 R

从源码编译 R 的需求大概有以下几点:

1. 爱折腾的极客: 玩配置, 学习 make 相关工具和 Linux 世界的依赖
2. 追求性能: 如 [LFS 支持](#) 和 [Intel MKL 加速](#)
3. 环境限制: CentOS 或者红帽系统, 自带的 R 版本比较落后

```
./configure --prefix=/opt/R/R-devel \
--enable-R-shlib --enable-byte-compiled-packages \
--enable-BLAS-shlib --enable-memory-profiling --with-blas="-lopenblas"
```

```
R is now configured for x86_64-pc-linux-gnu
```

```
Source directory: .
Installation directory: /opt/R/R-devel

C compiler:           gcc -g -O2
Fortran fixed-form compiler: gfortran -fno-optimize-sibling-calls -g -O2

Default C++ compiler: g++ -std=gnu++11 -g -O2
C++14 compiler:      g++ -std=gnu++14 -g -O2
C++17 compiler:      g++ -std=gnu++17 -g -O2
C++20 compiler:      g++ -std=gnu++2a -g -O2
Fortran free-form compiler: gfortran -fno-optimize-sibling-calls -g -O2
Obj-C compiler:
```



```
Interfaces supported:      X11, tcltk
External libraries:        pcre2, readline, BLAS(OpenBLAS), curl
Additional capabilities:  PNG, JPEG, TIFF, NLS, cairo, ICU
Options enabled:          shared R library, shared BLAS, R profiling, memory profiling

Capabilities skipped:
Options not enabled:

Recommended packages:     yes
```

配置好了以后，可以编译安装了

```
make && sudo make install
```

**flexiblas** 支持多种 BLAS 库自由切换

## A.13 安装软件

本书在后续章节中陆续用到新的 R 包，其安装过程不会在正文中呈现，下面以在 CentOS 8 上安装 **sf** 包为例介绍。首先需要安装一些系统依赖，具体安装哪些依赖参见 **sf** 包开发站点 <https://github.com/r-spatial/sf>。

```
sudo dnf config-manager --set-disabled PowerTools # openblas-devel
sudo dnf install -y sqlite-devel gdal-devel \
proj-devel geos-devel udunits2-devel
```

然后，在 R 命令行窗口中，执行安装命令：

```
install.packages('sf')
```

至此，安装完成。如遇本地未安装的新 R 包，可从其官方文档中找寻安装方式。如果你完全不知道自己应该安装哪些，考虑把下面的依赖都安装上

```
sudo dnf install -y \
# magick
ImageMagick-c++-devel \
# pdftools
poppler-cpp-devel \
# gifski
cargo
```

软件包管理器架构图，各个命令分别担负什么样的功能，每个命令学习的一般路径是什么，而不是详细介绍每个命令、每个参数的使用，只需给出一个命令的完整使用即可，其余给出一个查询命令帮助手册

```
dnf copr
dnf config-manager
```



## A.14 安装 R 包

Iñaki Ucar 开发的 [cran2copr](#) 项目实现在 Fedora 上安装预编译好的二进制 R 包，项目的类似 Debian 平台上的 [cran2deb](#)

1. [devtools](#) 是开发 R 包的常用工具，同时具有很重的依赖，请看

```
tools:::package_dependencies('devtools', recursive = TRUE)
```

```
## $devtools
## [1] "usethis"      "callr"        "cli"          "desc"         "ellipsis"
## [6] "fs"           "httr"         "lifecycle"    "memoise"      "pkgbuild"
## [11] "pkgload"       "rcmdcheck"    "remotes"      "rlang"        "roxygen2"
## [16] "rstudioapi"   "rversions"    "sessioninfo"  "stats"        "testthat"
## [21] "tools"         "utils"        "withr"        "processx"    "R6"
## [26] "glue"          "rprojroot"   "methods"      "curl"         "jsonlite"
## [31] "mime"          "openssl"      "cachem"       "crayon"      "prettyunits"
## [36] "digest"        "xopen"        "brew"         "commonmark"  "knitr"
## [41] "purrr"         "stringi"      "stringr"      "xml2"        "cpp11"
## [46] "brio"          "evaluate"    "magrittr"     "praise"      "ps"
## [51] "waldo"         "clipr"        "gert"         "gh"          "rappdirs"
## [56] "whisker"       "yaml"         "graphics"    "grDevices"   "fastmap"
## [61] "askpass"       "credentials" "sys"         "zip"         "gitcreds"
## [66] "ini"           "highr"        "xfun"        "diffobj"    "fansi"
## [71] "rematch2"      "tibble"       "pillar"      "pkgconfig"   "vctrs"
## [76] "utf8"
```

其中，依赖关系见表 A.2

表 A.2: devtools 的系统依赖

|        | curl                     | git2r         | openssl       |
|--------|--------------------------|---------------|---------------|
| Ubuntu | libcurl-dev <sup>4</sup> | libgit2-dev   | libssl-dev    |
| CentOS | libcurl-devel            | libgit2-devel | openssl-devel |

1. [sf](#) 是处理空间数据的常用工具

```
tools:::package_dependencies('sf', recursive = TRUE)
```

```
## $sf
## [1] "methods"      "classInt"     "DBI"          "graphics"     "grDevices"
## [6] "grid"          "magrittr"     "Rcpp"         "s2"           "stats"
## [11] "tools"         "units"        "utils"        "e1071"        "class"
## [16] "KernSmooth"   "wk"           "MASS"        "proxy"
```

其主要的系统依赖分别是 GEOS 3.5.1, GDAL 2.2.2, PROJ 4.9.2

<sup>4</sup>libcurl-dev 是一个虚包 virtual package，由 libcurl4-openssl-dev 或 libcurl4-nss-dev 或 libcurl4-gnults-dev 实际提供，选择其中一个安装即可。



```
sudo add-apt-repository -y ppa:ubuntugis/ubuntugis-unstable
sudo apt-get update
sudo apt-get install -y libudunits2-dev libgdal-dev libgeos-dev libproj-dev
```

这样也同时解决了 `udunits2`、`rgdal` 和 `rgeos` 等 3 个 R 包的系统依赖，其中 `udunits2` 使用如下命令安装

```
install.packages('udunits2', configure.args = '--with-udunits2-include=/usr/include/udunits2')
```

## 2. 图形设备支持 cairo png jpeg tiff

```
sudo apt-get install -y libcairo2-dev libjpeg-dev libpng-dev libtiff-dev
```

## 3. 图像处理 imager 和 magick

```
sudo yum install fftw-devel # CentOS
sudo apt-get install libfftw3-dev # Ubuntu
```

在 Ubuntu 系统上安装最新的 `libmagick++-dev` 库

```
sudo add-apt-repository -y ppa:opencpu/imagemagick
sudo apt-get update
sudo apt-get install -y libmagick++-dev
```

在 CentOS 系统上

```
sudo yum install -y ImageMagick-c++-devel
```

然后安装 R 包 `install.packages(c('imager', 'magick'))`

## 4. `rgl` 是绘制真三维图形的重量级 R 包

```
sudo apt-get install libcgal-dev libglu1-mesa-dev libxi-dev # Ubuntu
sudo yum install mesa-libGLU mesa-libGLU-devel # CentOS
```

然后安装 R 包

```
install.packages('rgl')
```

在 Ubuntu 系统上还可以这样安装

```
sudo add-apt-repository ppa:marutter/rrutter3.5
sudo apt-get update
sudo apt-get install r-cran-rgl
```

## 5. `rJava` 是 Java 语言和 R 语言之间实现通信交流的桥梁

```
sudo apt-get install -y default-jdk
sudo R CMD javareconf
```

然后安装 `rJava` 包 `install.packages('rJava')`

## 6. `igraph` 是网络数据分析的必备 R 包，为了发挥其最大性能，需要安装三个系统依赖

```
sudo apt-get install -y libgmp-dev libxml2-dev libglpk-dev
```

然后安装 R 包



```
install.packages('igraph')
```

7. `gpuR` 是基于 GPU 进行矩阵计算的扩展包，依赖 `RcppEigen` 确保安装 `OpenCL` 和 `RViennaCL` 或者安装 Nvidia 驱动和 CUDA，使用 `gpuRcuda` 和 `gputools` 扩展包，下面安装指导来自其 Wiki

```
# Install OpenCL headers
sudo apt-get install opencl-headers opencv-dev

# Install NVIDIA Drivers and CUDA
sudo add-apt-repository -y ppa:xorg-edgers/ppa
sudo apt-get update
sudo apt-get install nvidia-346 nvidia-settings
```

8. `nloptr` 是 `Nlopt` 的 R 语言接口，首先安装 `Nlopt` 程序库 `sudo apt-get install libnlopt-dev` 然后安装 R 包 `install.packages('nloptr')`，`nloptr` 被 700+ R 包依赖，如 `lme4`, `spaMM`, `glmmTMB`, `rstanarm` 等。

9. `Rmpfr`

```
sudo apt-get install libmpfr-dev

install.packages('Rmpfr')
```

10. `geojson`

```
sudo yum install jq-devel protobuf-devel

install.packages(c('geojson', 'geojsonio', 'jqr', 'protolite'))
```

11. `lgcp`

```
sudo yum install bwidget

install.packages(c('rpanel', 'lgcp'))
```

12. `ijtiff`

```
sudo yum install jbigkit-devel

install.packages('ijtiff')
```

13. `webshot` 包用于截图

```
sudo apt install phantomjs

install.packages('webshot')
```

14. `gifski` 包合成 GIF 动图

```
sudo apt-get install cargo

install.packages('gifski')
```

## A.15 软件包管理器

### A.15.1 dnf

#### 1. 清理升级后的 CentOS 8 系统内核

查找系统安装的内核

```
rpm -qa | sort | grep kernel
```

```
kernel-4.18.0-147.8.1.el8_1.x86_64
kernel-4.18.0-193.6.3.el8_2.x86_64
kernel-core-4.18.0-147.8.1.el8_1.x86_64
kernel-core-4.18.0-193.6.3.el8_2.x86_64
kernel-headers-4.18.0-193.6.3.el8_2.x86_64
kernel-modules-4.18.0-147.8.1.el8_1.x86_64
kernel-modules-4.18.0-193.6.3.el8_2.x86_64
kernel-tools-4.18.0-193.6.3.el8_2.x86_64
kernel-tools-libs-4.18.0-193.6.3.el8_2.x86_64
```

仅保留一个版本的内核，其它旧的内核都删除掉

```
sudo dnf remove $(dnf repoquery --installonly --latest-limit=1 -q)
```

模块依赖问题

问题 1: conflicting requests

- nothing provides module/perl:5.26 needed by module perl-DBD-MySQL:4.046:8010020191114030811:073fa5

问题 2: conflicting requests

- nothing provides module/perl:5.26 needed by module perl-DBI:1.641:8010020191113222731:16b3ab4d-0.x8

问题 3: conflicting requests

- nothing provides module/perl:5.26 needed by module perl-YAML:1.24:8010020191114031501:a5949e2e-0.x8

依赖关系解决。

```
=====
软件包          架构      版本          仓库          大小
=====
移除:
kernel          x86_64    4.18.0-147.8.1.el8_1    @BaseOS      0
kernel-core      x86_64    4.18.0-147.8.1.el8_1    @BaseOS      58 M
kernel-modules   x86_64    4.18.0-147.8.1.el8_1    @BaseOS      20 M
```

事务概要

```
=====
移除 3 软件包
```

将会释放空间: 78 M

确定吗? [y/N]: y

运行事务检查



事务检查成功。

运行事务测试

事务测试成功。

运行事务

准备中 :

|                                                  |     |
|--------------------------------------------------|-----|
| 删除 : kernel-4.18.0-147.8.1.el8_1.x86_64          | 1/1 |
| 运行脚本: kernel-4.18.0-147.8.1.el8_1.x86_64         | 1/3 |
| 删除 : kernel-modules-4.18.0-147.8.1.el8_1.x86_64  | 2/3 |
| 运行脚本: kernel-modules-4.18.0-147.8.1.el8_1.x86_64 | 2/3 |
| 运行脚本: kernel-core-4.18.0-147.8.1.el8_1.x86_64    | 3/3 |
| 删除 : kernel-core-4.18.0-147.8.1.el8_1.x86_64     | 3/3 |
| 运行脚本: kernel-core-4.18.0-147.8.1.el8_1.x86_64    | 3/3 |
| 验证 : kernel-4.18.0-147.8.1.el8_1.x86_64          | 1/3 |
| 验证 : kernel-core-4.18.0-147.8.1.el8_1.x86_64     | 2/3 |
| 验证 : kernel-modules-4.18.0-147.8.1.el8_1.x86_64  | 3/3 |

已移除:

|                                            |                                         |
|--------------------------------------------|-----------------------------------------|
| kernel-4.18.0-147.8.1.el8_1.x86_64         | kernel-core-4.18.0-147.8.1.el8_1.x86_64 |
| kernel-modules-4.18.0-147.8.1.el8_1.x86_64 |                                         |

完毕!

[解决上述模块依赖问题的办法](#) 是重置三个 Perl 模块

```
sudo dnf module reset perl-DBD-MySQL perl-YAML perl-DBI
```

依赖关系解决。

| 软件包 | 架构 | 版本 | 仓库 | 大小 |
|-----|----|----|----|----|
|-----|----|----|----|----|

重置模块:

```
perl-DBD-MySQL
perl-DBI
perl-YAML
```

事务概要

确定吗 ? [y/N]: y

完毕!

## A.15.2 apt

添加或删除 PPA (Personal Package Archive), 比如在 Ubuntu 20.04 及之前的版本上安装[新版 Inkscape](#)

```
sudo add-apt-repository ppa:inkscape.dev/stable
sudo add-apt-repository --remove ppa:inkscape.dev/stable
```

```
sudo apt-get install build-essential # 修复依赖问题
sudo apt update # 更新资源列表
sudo apt-get upgrade # 更新软件包
sudo apt-get autoclean # 删除已卸的软件的备份
sudo apt-get clean # 删除已装或已卸的软件的备份
sudo apt-get autoremove --purge * # 推荐卸载软件的方式
apt-get list --upgradable # 列出可升级的包
```

找到并删除旧的内核

```
dpkg --list | grep linux-image
sudo apt-get purge linux-image-3.19.0-{18,20,21,25}
sudo update-grub2
```

```
# 搜索
apt-cache search octave | grep octave
# 查询
apt show octave
# 安装
sudo apt install octave
```

```
sudo apt-get install lsb-core
lsb_release -a

adduser cloud2016 # 添加用户
passwd cloud2016 # 用户密码设为 cloud
whereis sudoers # 查找文件位置
chmod -v u+w /etc/sudoers # 给文件 sudoers 添加写权限
vim /etc/sudoers # 添加 cloud2016 管理员权限
chmod -v u-w /etc/sudoers # 收回权限
```

安装确认 openssh-server 服务

```
sudo apt install openssh-server
sudo /etc/init.d/ssh start
ps -aux | grep ssh
```

## 索引

bookdown, 5

Octave, 2

Pandoc, 5

Python, 2

## 参考文献

- JJ Allaire, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. *rmarkdown: Dynamic Documents for R*, 2021. URL <https://CRAN.R-project.org/package=rmarkdown>. R package version 2.9.
- Mickaël Binois and Victor Picheny. GPareto: An R package for gaussian-process-based multi-objective optimization and analysis. *Journal of Statistical Software*, 89(8):1–30, 2019. doi: 10.18637/jss.v089.i08.
- Hans W. Borchers. *pracma: Practical Numerical Math Functions*, 2021. URL <https://CRAN.R-project.org/package=pracma>. R package version 2.3.3.
- John M. Chambers. S, R, and Data Science. *The R Journal*, 12(1):462–476, 2020. doi: 10.32614/RJ-2020-028. URL <https://doi.org/10.32614/RJ-2020-028>.
- Winston Chang, Alexej Kryukov, and Paul Murrell. *fontcm: Computer Modern font for use with extrafont package*, 2014. URL <https://github.com/wch/fontcm>. R package version 1.1.
- Alex Couture-Beil, Jon T. Schnute, Rowan Haigh, Simon N. Wood, and Benjamin J. Cairns. *PBSddesolve: Solver for Delay Differential Equations*, 2019. URL <https://CRAN.R-project.org/package=PBSddesolve>. R package version 1.12.6.
- Kalyanmoy Deb. *Multi-Objective Optimization*, pages 273–316. Springer US, Boston, MA, 2005. ISBN 978-0-387-28356-2. doi: 10.1007/0-387-28356-0\_10. URL [https://doi.org/10.1007/0-387-28356-0\\_10](https://doi.org/10.1007/0-387-28356-0_10).
- Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004. doi: 10.1214/009053604000000067.
- Anqi Fu, Balasubramanian Narasimhan, and Stephen Boyd. CVXR: An R package for disciplined convex optimization. *Journal of Statistical Software*, 94(14):1–34, 2020. doi: 10.18637/jss.v094.i14.
- Manfred Gilli, Dietmar Maringer, and Enrico Schumann. *Numerical Methods and Optimization in Finance*. Elsevier/Academic Press, Waltham, MA, USA, second edition, 2019. URL <http://www.enricoschumann.net/NMOF/>. ISBN 978-0128150658.
- Gabor Grothendieck and Thomas Petzoldt. R Help Desk: Date and time classes in R. *R News*, 4(1):29–32, June 2004. URL [https://www.r-project.org/doc/Rnews/Rnews\\_2004-1.pdf](https://www.r-project.org/doc/Rnews/Rnews_2004-1.pdf).
- Zuguang Gu, Roland Eils, and Matthias Schlesner. Complex heatmaps reveal patterns and correlations in multidimensional genomic data. *Bioinformatics*, 2016.
- Kurt Hornik. R FAQ: Frequently asked questions on R, 2020. URL <https://CRAN.R-project.org/doc/FAQ/R-FAQ.html>.
- Peter Kampstra. beanplot: A boxplot alternative for visual comparison of distributions. *Journal of Statistical Software*, 28(1):1–9, 2008. URL <http://www.jstatsoft.org/v28/c01/>.



Yongdai Kim, Hosik Choi, and Hee-Seok Oh. Smoothly clipped absolute deviation on high dimensions. *Journal of the American Statistical Association*, 103(484):1665–1673, 2008. doi: 10.1198/016214508000001066.

Paul Murrell. Integrating grid graphics output with base graphics output. *R News*, 3(2):7–12, 2003.

Paul Murrell and Ross Ihaka. An approach to providing mathematical annotation in plots. *Journal of Computational and Graphical Statistics*, 9(3):582–599, 2000.

John C. Nash. On best practice optimization methods in r. *Journal of Statistical Software*, 60(2):1–14, 2014. doi: 10.18637/jss.v060.i02. URL <https://www.jstatsoft.org/v060/i02>.

Erich Neuwirth. *RColorBrewer: ColorBrewer Palettes*, 2014. URL <https://CRAN.R-project.org/package=RColorBrewer>. R package version 1.1-2.

Yixuan Qiu. showtext: Using system fonts in R graphics. *The R Journal*, 7(1):99–108, jun 2015. doi: 10.32614/RJ-2015-008.

Brian D. Ripley. Statistical methods need software: A view of statistical computing, 9 2002. URL <https://www.stats.ox.ac.uk/~ripley/RSS2002.pdf>.

Brian D. Ripley and Kurt Hornik. Date-time classes. *R News*, 1(2):8–11, June 2001. URL [https://cran.r-project.org/doc/Rnews/Rnews\\_2001-2.pdf](https://cran.r-project.org/doc/Rnews/Rnews_2001-2.pdf).

Luca Scrucca. GA: A package for genetic algorithms in R. *Journal of Statistical Software*, 53(4):1–37, 2013. URL <https://www.jstatsoft.org/v53/i04/>.

Luca Scrucca. On some extensions to GA package: hybrid optimisation, parallelisation and islands evolution. *The R Journal*, 9(1):187–206, 2017. URL <https://journal.r-project.org/archive/2017/RJ-2017-008/>.

Karline Soetaert and Filip Meysman. Reactive transport in aquatic ecosystems: Rapid model prototyping in the open source software R. *Environmental Modelling & Software*, 32:49–60, 2012.

Reto Stauffer, Georg J. Mayr, Markus Dabernig, and Achim Zeileis. Somewhere over the rainbow: How to make effective use of colors in meteorological visualizations. *Bulletin of the American Meteorological Society*, 96(2):203–216, 2009. doi: 10.1175/BAMS-D-13-00155.1.

Yuan Tang. autoplotly: An r package for automatic generation of interactive visualizations for statistical results. *Journal of Open Source Software*, 3, 2018. URL <https://doi.org/10.21105/joss.00657>.

Yuan Tang, Masaaki Horikoshi, and Wenxuan Li. ggfortify: Unified interface to visualize statistical results of popular r packages. *The R Journal*, 8(2):474–485, 2016. doi: 10.32614/RJ-2016-060. URL <https://journal.r-project.org/archive/2016/RJ-2016-060/RJ-2016-060.pdf>.

Stefan Theußl, Florian Schwendinger, and Kurt Hornik. ROI: An extensible R optimization infrastructure. *Journal of Statistical Software*, 94(15):1–64, 2020. doi: 10.18637/jss.v094.i15.

Robert Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. URL <http://www.jstor.org/stable/2346178>.

Emilio Torres-Manzanera. *xkcd: Plotting ggplot2 Graphics in an XKCD Style*, 2018. R package version 0.0.6.

Michail Tsagris and Manos Papadakis. Taking r to its limits: 70+ tips. *PeerJ Preprints*, 6:e26605v1, 2018.



- ISSN 2167-9843. doi: 10.7287/peerj.preprints.26605v1. URL <https://doi.org/10.7287/peerj.preprints.26605v1>.
- Berwin A. Turlach. *quadprog: Functions to Solve Quadratic Programming Problems.*, 2019. URL <https://CRAN.R-project.org/package=quadprog>. R package version 1.5-8.
- M.P.J. van der Loo. The stringdist package for approximate string matching. *The R Journal*, 6:111–122, 2014. URL <https://CRAN.R-project.org/package=stringdist>.
- Ravi Varadhan and Paul Gilbert. BB: An R package for solving a large system of nonlinear equations and for optimizing a high-dimensional nonlinear objective function. *Journal of Statistical Software*, 32(4):1–26, 2009. URL <https://www.jstatsoft.org/v32/i04/>.
- Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag, New York, 2nd edition, 2016. URL <https://ggplot2-book.org/>. ISBN 978-3319242774.
- Yihui Xie. animation: An R package for creating animations and demonstrating statistical methods. *Journal of Statistical Software*, 53(1):1–27, 2013. URL <http://www.jstatsoft.org/v53/i01/>.
- Yihui Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition, 2015. URL <https://yihui.org/knitr/>. ISBN 978-1498716963.
- Yihui Xie. *bookdown: Authoring Books and Technical Documents with R Markdown*. Chapman and Hall/CRC, Boca Raton, Florida, 2016. URL <https://github.com/rstudio/bookdown>. ISBN 978-1138700109.
- Yihui Xie. TinyTeX: A lightweight, cross-platform, and easy-to-maintain latex distribution based on TeX Live. *TUGboat*, (1):30–32, 2019. URL <https://tug.org/TUGboat/Contents/contents40-1.html>.
- Jelmer Ypma. *R Interface to NLOpt*, 2020. URL <https://github.com/jyypma/nloptr>. R package version 1.2.2.2.
- Achim Zeileis, Kurt Hornik, and Paul Murrell. Escaping RGBland: Selecting colors for statistical graphics. *Computational Statistics & Data Analysis*, 53(9):3259–3270, 2009. doi: 10.1016/j.csda.2008.11.033.
- Achim Zeileis, Jason C. Fisher, Kurt Hornik, Ross Ihaka, Claire D. McWhite, Paul Murrell, Reto Stauffer, and Claus O. Wilke. colorspace: A toolbox for manipulating and assessing colors and palettes. arXiv 1903.06490, arXiv.org E-Print Archive, March 2019. URL <http://arxiv.org/abs/1903.06490>.
- Cun-Hui Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2):894 – 942, 2010. doi: 10.1214/09-AOS729.