© 黄湘云

R 语言数据分析实战

黄湘云

2024 年 1 月 23 日

© 黄湘云

目录

插图目录 xiii

列表目录 xxi

欢迎 1

前言 3
- 为什么是 R 语言? 3
- 为什么写这本书? 4
- 本书是怎么写的? 4
- 写作理念是什么? 5
- 目标读者是哪些? 5
- 本书有哪些内容? 5
- 公开数据从哪找? 6
- 学会有效地提问? 6

介绍 7
- 数据探索和分析 8
- 数据展示和交流 13

第一部分 数据准备 14

第一章 数据对象 15
- 1.1 数据类型 15
  - 1.1.1 整型 15
  - 1.1.2 逻辑型 15
  - 1.1.3 字符型 15
  - 1.1.4 日期型 15
  - 1.1.5 数值型 16
- 1.2 数据结构 16
  - 1.2.1 向量 16
  - 1.2.2 矩阵 16
  - 1.2.3 数组 16
  - 1.2.4 列表 16
  - 1.2.5 因子 16
  - 1.2.6 数据框 16
  - 1.2.7 ts 16

第二章 数据获取 19
- 2.1 从本地文件读取 19
  - 2.1.1 csv 文件 19
  - 2.1.2 xlsx 文件 19
  - 2.1.3 arrow 文件 20
- 2.2 从数据库中导入 20
  - 2.2.1 RSQLite 20
  - 2.2.2 odbc 20
  - 2.2.3 RJDBC 20
- 2.3 从各类网页中抓取 20
  - 2.3.1 豆瓣排行榜 20
  - 2.3.2 链家二手房 20
- 2.4 从数据接口中获取 20
  - 2.4.1 中国地震台网 20
  - 2.4.2 美国地质调查局 20
  - 2.4.3 美国人口调查局 21
  - 2.4.4 世界银行 21

第三章 数据清洗 22
- 3.1 正则表达式 22
  - 3.1.1 量词 22
  - 3.1.2 级联 22
  - 3.1.3 断言 22
  - 3.1.4 反向引用 22
  - 3.1.5 命名捕捉 22

© 黄湘云

3.2 字符串操作 22
3.2.1 查找 22
3.2.2 替换 22
3.2.3 提取 23

第四章 数据操作 24
4.1 操作工具 24
4.1.1 Base R 24
4.1.2 data.table 24
4.1.3 dplyr 25
4.1.4 SQL 25
4.2 Base R 操作 27
4.2.1 筛选 27
4.2.2 变换 27
4.2.3 排序 28
4.2.4 聚合 28
4.2.5 合并 28
4.2.6 重塑 29
4.3 data.table 操作 30
4.3.1 筛选 31
4.3.2 变换 31
4.3.3 排序 31
4.3.4 聚合 31
4.3.5 合并 32
4.3.6 重塑 33

第五章 数据处理 34
5.1 缺失值处理 34
5.1.1 查找 34
5.1.2 汇总 34
5.1.3 替换 34
5.1.4 插补 34
5.2 异常值处理 34
5.2.1 检测 35
5.2.2 识别 35
5.2.3 处理 35
5.3 离群值处理 35
5.3.1 检测 35
5.3.2 识别 35
5.3.3 处理 35

第二部分 数据探索 36

第六章 ggplot2 入门 37
6.1 图层 38
6.2 标签 39
6.3 刻度 39
6.4 配色 45
6.5 图例 47
6.6 主题 49
6.7 注释 55
6.8 分面 58
6.9 动画 61
6.10 组合 61
6.11 艺术 65

第七章 基础图形 68
7.1 描述趋势 68
7.1.1 折线图 69
7.1.2 瀑布图 73
7.1.3 曲线图 75
7.1.4 流线图 82
7.1.5 曲面图 83
7.1.6 热力图 85
7.1.7 日历图 87
7.1.8 棋盘图 89
7.1.9 时间线图 92
7.2 描述对比 96
7.2.1 柱形图 96
7.2.2 条形图 98
7.2.3 点线图 99
7.2.4 词云图 100
7.3 描述占比 102
7.3.1 简单饼图 102
7.3.2 环形饼图 106
7.3.3 扇形饼图 107
7.3.4 帕累托图 108
7.3.5 马赛克图 111
7.3.6 矩阵树图 113
7.3.7 量表图 114
7.4 习题 118

© 高等教育出版社

第八章 统计图形 120
8.1 描述分布 120
8.1.1 箱线图 120
8.1.2 提琴图 121
8.1.3 直方图 121
8.1.4 密度图 124
8.1.5 岭线图 130
8.1.6 抖动图 130
8.2 描述关系 135
8.2.1 散点图 135
8.2.2 气泡图 136
8.2.3 凹凸图 137
8.2.4 韦恩图 139
8.2.5 网络图 140
8.3 描述不确定性 142
8.3.1 置信区间 142
8.3.2 假设检验 144
8.3.3 模型预测 148
8.3.4 模型诊断 149
8.3.5 边际效应 151

第九章 lattice 入门 153
9.1 分组散点图 153
9.2 图形参数 155
9.3 常见图形 162
9.3.1 分组柱形图 162
9.3.2 分组箱线图 163
9.3.3 经验分布图 164
9.3.4 回归曲线图 165
9.3.5 置信区间图 168
9.3.6 置信椭圆图 168
9.3.7 切片水平图 169
9.3.8 三维散点图 171
9.3.9 三维透视图 175
9.3.10 地形轮廓图 178
9.3.11 地区分布图 180
9.4 总结 182

第十章 graphics 入门 185
10.1 绘图基础 185
10.1.1 `plot()` 185
10.1.2 标签 187
10.1.3 字体 188
10.1.4 分组 188
10.1.5 配色 190
10.1.6 注释 191
10.1.7 图例 192
10.1.8 统计 195
10.2 绘图进阶 197
10.2.1 组合图形 197
10.2.2 多图布局 199
10.3 图形选择 200
10.3.1 颜色图 201
10.3.2 透视图 201
10.3.3 等值线图 202
10.3.4 填充等值线图 203
10.4 总结 204
10.4.1 plot2 包 204
10.4.2 plot3D 包 207

第十一章 TikZ 入门 211
11.1 standalone 宏包 211
11.2 PGF 宏包 213
11.3 三维图 215
11.4 网络图 218
11.5 思维导图 219
11.6 SmartArt 图 222
11.7 TikZ 与 R 224

第十二章 探索实践 228
12.1 分析老忠实间歇泉喷发规律 228
12.2 中国地区级男女性别比分布 228
12.3 美国历年各年龄死亡率变化 231
12.4 美国弗吉尼亚州城乡死亡率 231
12.5 如何用图表示累积概率分布 235
12.6 解释二项总体参数的置信带 236
12.7 解释置信区间及其覆盖概率 238
12.8 习题 240

© 黄湘云

第三部分 数据交流 242

第十三章 交互图形 243
13.1 基础元素 243
13.1.1 图层 243
13.1.2 配色 244
13.1.3 刻度 245
13.1.4 标签 245
13.1.5 主题 245
13.1.6 字体 246
13.1.7 图例 246
13.2 常用图形 246
13.2.1 散点图 246
13.2.2 柱形图 247
13.2.3 曲线图 247
13.2.4 直方图 248
13.2.5 箱线图 250
13.2.6 热力图 251
13.2.7 面量图 252
13.2.8 动态图 252
13.3 常用技巧 254
13.3.1 数学公式 254
13.3.2 动静转化 256
13.3.3 坐标系统 257
13.3.4 添加水印 258
13.3.5 多图布局 259
13.3.6 图表联动 260

第十四章 交互表格 261
14.1 基础功能 261
14.1.1 创建表格 261
14.1.2 添加标题 261
14.1.3 添加注释 261
14.1.4 水平滚动 261
14.1.5 垂直滚动 261
14.1.6 数据分页 261
14.1.7 适应宽度 261
14.1.8 行列分组 261
14.1.9 列格式化 261
14.1.10 数据配色 261
14.2 扩展功能 264
14.2.1 汉化表格 264
14.2.2 下载数据 264
14.3 其它工具 264

第十五章 交互应用 265
15.1 简单示例 265
15.1.1 UI 前端 266
15.1.2 Server 后端 266
15.2 Shiny 组件 266
15.2.1 筛选器 266
15.2.2 输入框 266
15.2.3 动作按钮 266
15.2.4 书签 266
15.3 Shiny 扩展 267
15.3.1 页面布局 267
15.3.2 交互表格 267
15.3.3 交互图形 269
15.4 Shiny 仪表盘 269
15.4.1 shinydashboard 包 270
15.4.2 shinydashboardPlus 包 271
15.4.3 bs4Dash 包 272
15.4.4 miniUI 包 273
15.5 Shiny 主题 275
15.5.1 bslib 包 275
15.5.2 shinymaterial 包 275
15.6 Shiny 优化 276
15.7 Shiny 部署 276
15.7.1 promises 并发 276
15.8 Shiny 替代品 277
15.9 Shiny 案例 277
15.10 总结 277

第十六章 HTML 文档 280
16.1 文档元素 280
16.1.1 样式 280
16.1.2 图片 280
16.1.3 表格 282
16.1.4 列表 283
16.1.5 引用 283

© 黄湘云

16.1.6 脚注 284
16.1.7 公式 284
16.2 制作报告 285
16.2.1 SQL 查询 285
16.3 制作演示 286
16.4 编写书籍 286

第十七章 PDF 文档 287
17.1 LaTeX 基础 287
17.1.1 中英字体 288
17.1.2 数学公式 289
17.1.3 代码抄录 290
17.1.4 插入图表 292
17.1.5 交叉引用 293
17.1.6 PDF-A/X 293
17.2 R Markdown 基础 294
17.2.1 中英字体 295
17.2.2 数学公式 295
17.2.3 代码抄录 295
17.2.4 插入图表 295
17.2.5 交叉引用 295
17.2.6 布局排版 295
17.3 Quarto 基础 297
17.3.1 中英字体 298
17.3.2 数学公式 298
17.3.3 代码抄录 298
17.3.4 插入图表 298
17.3.5 交叉引用 299
17.4 Quarto beamer 299

第十八章 Office 文档 303
18.1 Word 文档 303
18.1.1 Markdown 制作 Word 文档 . 303
18.1.2 R Markdown 制作 Word 文档 303
18.1.3 自定义 Word 模版 303
18.2 PowerPoint 演示 304
18.3 电子邮件 304
18.3.1 curl 包 304
18.3.2 blastula 包 304

第四部分 统计分析 307

第十九章 常见的统计检验 308
19.1 单样本检验 309
19.1.1 正态总体均值检验 309
19.1.2 正态总体方差检验 312
19.1.3 总体未知均值检验 313
19.1.4 总体未知方差检验 314
19.2 两样本检验 314
19.2.1 正态总体均值检验 314
19.2.2 正态总体方差检验 321
19.2.3 总体未知均值检验 322
19.2.4 总体未知方差检验 323
19.3 多样本检验 324
19.3.1 正态总体均值检验 327
19.3.2 正态总体方差检验 329
19.3.3 总体未知均值检验 330
19.3.4 总体未知方差检验 332
19.4 配对样本检验 332
19.4.1 配对 t 检验 333
19.4.2 配对 Wilcoxon 检验 335
19.5 多重假设检验 335
19.5.1 多重 t 检验 335
19.5.2 多重比例检验 336
19.5.3 Wilcoxon 检验 337
19.5.4 Dunn 检验 337
19.6 总体分布的检验 338
19.6.1 正态性检验 338
19.6.2 同分布检验 339
19.6.3 相关性检验 340
19.6.4 独立性检验 340
19.6.5 平稳性检验 341
19.7 多元分布情形 341
19.7.1 Hotelling T^2 检验 341
19.7.2 Mauchly 球形检验 341
19.8 假设检验的一些注记 343
19.8.1 假设检验和多重比较的关系 . 345
19.8.2 假设检验和方差分析的关系 . 345
19.8.3 假设检验与区间估计的关系 . 350
19.8.4 常见的统计检验是线性模型 . 351

© 黄湘云

19.8.5 假设检验的工业应用 352
19.9 习题 353

第二十章 回归与相关分析 356
20.1 子代身高与亲代身高的关系 356
20.2 预期寿命与人均收入的关系 359
20.3 分析影响入院等待时间的因素 364
20.4 习题 365

第二十一章 分类数据的分析 366
21.1 比例检验 366
21.1.1 单样本检验 366
21.1.2 两样本检验 368
21.1.3 多样本检验 369
21.2 泊松检验 370
21.2.1 单样本 370
21.2.2 两样本 370
21.3 列联表描述 370
21.3.1 行列分组表格 371
21.3.2 百分比堆积图 372
21.3.3 桑基图 373
21.3.4 马赛克图 374
21.4 列联表分析 376
21.4.1 相互独立性 377
21.4.2 边际独立性 380
21.4.3 对称性 382
21.4.4 条件独立性 382
21.5 加州伯克利分校的录取情况 384
21.6 分析泰坦尼克号乘客生存率 389

第二十二章 统计检验的功效 391
22.1 三大检验方法 391
22.1.1 Wald 检验 391
22.1.2 Wilks 检验 391
22.1.3 Rao 检验 391
22.2 t 检验的功效 391
22.3 比例检验的功效 394
22.4 方差分析的功效 396

第五部分 数据建模 398

第二十三章 网络分析 399
23.1 R 语言社区的规模 399
23.2 R 语言社区的组织 401
23.3 R 语言社区的开发者 407
23.3.1 最高产的开发者 407
23.3.2 开发者协作关系 409
23.3.3 节点出入度分布 412
23.3.4 可视化协作网络 412
23.4 扩展阅读 418
23.5 习题 420

第二十四章 文本分析 421
24.1 基本概念 421
24.1.1 中文分词 421
24.1.2 英文分词 422
24.1.3 停止词 422
24.1.4 分词与词性 422
24.1.5 词频统计 423
24.1.6 TF-IDF 424
24.2 益辉的日志 424
24.2.1 整体概况 424
24.2.2 数据清洗 424
24.2.3 主题建模 428
24.2.4 日志相似性 430
24.3 习题 430

第二十五章 生存分析 431
25.1 问题背景 431
25.2 模型拟合 432
25.2.1 survival 432
25.2.2 ggsurvfit 435
25.2.3 glmnet 436
25.2.4 INLA 436
25.3 Tobit 回归 437

第二十六章 时序分析 439
26.1 数据获取 439
26.2 数据探索 441
26.2.1 zoo 441

© 黄湘云

26.2.2 xts 443
26.2.3 ggfortify 444
26.2.4 dygraphs 445
26.3 平稳性诊断 447
26.3.1 自相关图 447
26.3.2 偏自相关图 447
26.3.3 延迟算子 448
26.3.4 差分算子 449
26.3.5 单位根检验 450
26.3.6 格兰杰因果检验 450
26.4 指数平滑模型 450
26.4.1 指数平滑 450
26.4.2 函数 `filter()` 450
26.4.3 简单指数平滑 453
26.4.4 Holt 指数平滑 455
26.4.5 Holt-Winters 指数平滑 456
26.5 时间序列分解 459
26.5.1 函数 `decompose()` 459
26.5.2 函数 `stl()` 461
26.6 经典时间序列模型 462
26.6.1 自回归模型 462
26.6.2 移动平均模型 463
26.6.3 自回归移动平均模型 463
26.7 总结 464

第二十七章 空间点模式分析 465
27.1 数据操作 465
27.1.1 类型转化 465
27.1.2 坐标转化 466
27.1.3 凸包操作 468
27.2 数据探索 469
27.2.1 核密度估计 469
27.2.2 绘制热力图 470

第二十八章 空间点参考数据分析 472
28.1 数据说明 472
28.2 数据探索 475
28.3 数据建模 475
28.3.1 广义线性模型 475
28.3.2 空间线性混合效应模型 479
28.3.3 空间广义线性混合效应模型 . 486
28.4 模型预测 487
28.4.1 海岸线数据 487
28.4.2 边界处理 488
28.4.3 构造网格 490
28.4.4 整理数据 492
28.4.5 展示结果 493

第二十九章 空间区域数据分析 497
29.1 苏格兰唇癌数据分析 497
29.2 美国各州犯罪率分析 499

第六部分 优化建模 503

第三十章 统计计算 504
30.1 回归问题与优化问题 504
30.2 对数似然与损失函数 504
30.2.1 Logistic 分布 504
30.2.2 逻辑回归 505
30.3 数值优化问题求解器 507
30.3.1 `optim()` 507
30.3.2 `glm()` 511
30.3.3 glmnet 包 512
30.4 评估模型的分类效果 514
30.4.1 ROC 曲线和 AUC 值 514
30.4.2 Wilcoxon 检验 515

第三十一章 数值优化 517
31.1 线性优化 518
31.2 凸二次优化 520
31.3 凸锥优化 524
31.3.1 锥与凸锥 524
31.3.2 零锥 525
31.3.3 线性锥 525
31.3.4 二阶锥 527
31.3.5 指数锥 529
31.3.6 幂锥 530
31.3.7 半正定锥 532
31.4 非线性优化 533
31.4.1 一元非线性优化 534

© 黄湘云

31.4.2 多元隐函数优化 535
31.4.3 多元无约束优化 539
31.4.4 多元箱式约束优化 543
31.4.5 多元线性约束优化 551
31.4.6 多元非线性约束优化 552
31.5 整数优化 557
31.5.1 纯整数线性优化 558
31.5.2 0-1 整数线性优化 559
31.5.3 混合整数线性优化 560
31.5.4 混合整数二次优化 561
31.5.5 混合整数非线性优化 564
31.6 总结 566
31.7 习题 567

第三十二章 优化问题 569
32.1 旅行商问题 569
32.2 投资组合问题 571
32.3 高斯过程回归 576
32.4 泊松混合分布 581
32.5 极大似然估计 585
32.6 习题 587

第七部分 贝叶斯建模 589

第三十三章 概率推理框架 590
33.1 Stan 概览 590
33.2 Stan 入门 592
33.2.1 Stan 的基础语法 592
33.2.2 Stan 的变量类型 598
33.2.3 Stan 的代码结构 598
33.2.4 Stan 的函数使用 599
33.3 先验分布 601
33.3.1 正态先验 601
33.3.2 Lasso 先验 603
33.3.3 Horseshoe 先验 606
33.3.4 SpikeSlab 先验 610
33.4 推理算法 611
33.4.1 惩罚极大似然算法 612
33.4.2 变分近似推断算法 612
33.4.3 拉普拉斯近似算法 613
33.4.4 探路者变分算法 614
33.5 习题 615

第三十四章 广义线性模型 619
34.1 生成模拟数据 619
34.2 拟合泊松模型 619
34.3 参数后验分布 623
34.4 模型评估指标 632
34.5 可选替代实现 633
34.6 案例：吸烟喝酒和食道癌的关系 . . . 634
34.6.1 描述分析 635
34.6.2 拟合模型 636
34.6.3 与 brms 比较 638
34.7 案例：哥本哈根住房状况调查 639
34.8 习题 640

第三十五章 分层正态模型 642
35.1 rstan 包 643
35.1.1 拟合模型 643
35.1.2 模型输出 644
35.1.3 操作数据 646
35.1.4 采样诊断 648
35.1.5 后验分布 650
35.2 其它 R 包 652
35.2.1 nlme 652
35.2.2 lme4 654
35.2.3 blme 655
35.2.4 MCMCglmm 656
35.2.5 cmdstanr 657
35.3 案例：rats 数据 660
35.4 频率派方法 661
35.4.1 nlme 661
35.4.2 lavaan 667
35.4.3 lme4 671
35.4.4 glmmTMB 672
35.4.5 MASS 673
35.4.6 spaMM 674
35.4.7 hglm 675
35.4.8 mgcv 677
35.5 贝叶斯方法 680

© 黄湘云

35.5.1 rstan 680
35.5.2 cmdstanr 686
35.5.3 brms 687
35.5.4 rstanarm 688
35.5.5 blme 689
35.5.6 rjags 690
35.5.7 MCMCglmm 692
35.5.8 INLA 695
35.6 总结 696
35.7 习题 697

第三十六章 混合效应模型 704
36.1 线性混合效应模型 705
36.1.1 nlme 708
36.1.2 MASS 711
36.1.3 lme4 712
36.1.4 blme 713
36.1.5 brms 714
36.1.6 MCMCglmm 715
36.1.7 INLA 716
36.2 广义线性混合效应模型 717
36.2.1 MASS 719
36.2.2 GLMMadaptive 720
36.2.3 glmmTMB 721
36.2.4 lme4 722
36.2.5 mgcv 723
36.2.6 blme 724
36.2.7 brms 725
36.2.8 MCMCglmm 726
36.2.9 INLA 730
36.3 非线性混合效应模型 731
36.3.1 nlme 732
36.3.2 lme4 735
36.3.3 brms 736
36.4 模拟实验比较（补充） 737
36.4.1 lme4 738
36.4.2 GLMMadaptive 739
36.4.3 glmmTMB 740
36.4.4 hglm 741
36.4.5 glmmML 743
36.4.6 glmm 743
36.4.7 gee 745
36.4.8 geepack 746
36.4.9 blme 747
36.4.10 brms 748
36.4.11 MCMCglmm 748
36.4.12 INLA 751
36.5 总结 751
36.6 习题 752

第三十七章 广义可加模型 754
37.1 案例：模拟摩托车事故 754
37.1.1 mgcv 754
37.1.2 cmdstanr 758
37.1.3 rstanarm 758
37.1.4 brms 760
37.1.5 GINLA 762
37.1.6 INLA 763
37.2 案例：朗格拉普岛核污染 763
37.2.1 mgcv 763
37.2.2 cmdstanr 766
37.2.3 GINLA 767
37.2.4 INLA 769
37.3 案例：城市土壤重金属污染 773
37.3.1 mgcv 779
37.3.2 INLA 779

第三十八章 高斯过程回归 780
38.1 多元正态分布 780
38.1.1 多元正态分布模拟 780
38.1.2 多元正态分布拟合 786
38.2 二维高斯过程 789
38.2.1 二维高斯过程模拟 789
38.2.2 二维高斯过程拟合 793
38.3 高斯过程回归 796
38.3.1 模型介绍 796
38.3.2 观测数据 796
38.3.3 预测数据 797
38.3.4 模型编码 798
38.3.5 预测分布 802

© 高等教育出版社

38.4 总结 809
38.5 习题 812

第三十九章 时间序列回归 816
39.1 随机波动率模型 816
39.1.1 Stan 框架 819
39.1.2 fGarch 包 821
39.2 贝叶斯可加模型 825
39.2.1 Stan 框架 827
39.2.2 INLA 框架 827
39.3 一些非参数模型 831
39.3.1 mgcv 包 831
39.3.2 tensorflow 框架 836
39.4 习题 840

第八部分 机器学习 842

第四十章 分类问题 843
40.1 多项回归模型 844
40.2 线性判别分析 847
40.3 二次判别分析 848
40.4 朴素贝叶斯 848
40.5 支持向量机 850
40.6 K 最近邻 851
40.7 神经网络 852
40.8 决策树 852
40.9 随机森林 854
40.10 集成学习 855
40.11 总结 857
40.12 习题 858

第四十一章 聚类问题 859
41.1 层次聚类 859
41.2 快速聚类 859
41.3 模糊聚类 859
41.4 基于模型的聚类 859
41.5 基于密度的聚类 859

第四十二章 回归问题 860
42.1 线性回归 861
42.1.1 最小二乘回归 862
42.1.2 逐步回归 863
42.1.3 偏最小二乘回归 864
42.1.4 主成分回归 865
42.2 惩罚回归 866
42.2.1 岭回归 866
42.2.2 Lasso 回归 868
42.2.3 弹性网络 869
42.2.4 自适应 Lasso 870
42.2.5 松弛 Lasso 872
42.2.6 MCP 874
42.2.7 SCAD 877
42.2.8 最小角回归 880
42.2.9 最优子集回归 881
42.3 支持向量机 883
42.4 神经网络 884
42.5 决策树 884
42.6 随机森林 885
42.7 集成学习 886

参考文献 888

附录 901

附录 A 数学符号 901

附录 B 矩阵运算 904
B.1 基础运算 904
B.1.1 加、减、乘 904
B.1.2 对数、指数与幂 905
B.1.3 迹、秩、条件数 906
B.1.4 求逆与广义逆 907
B.1.5 行列式与伴随 907
B.1.6 外积、直积与交叉积 908
B.1.7 Hadamard 积 909
B.1.8 矩阵范数 910
B.1.9 转置与旋转 910
B.1.10 正交与投影 911
B.1.11 Givens 变换 (*) 911
B.1.12 Householder 变换 (*) 911
B.1.13 单位矩阵 912

© 黄湘云

B.1.14 对角矩阵 913
B.1.15 稀疏矩阵 913
B.1.16 上、下三角矩阵 913
B.2 矩阵分解 914
B.2.1 LU 分解 914
B.2.2 Schur 分解 915
B.2.3 QR 分解 915
B.2.4 Cholesky 分解 916
B.2.5 特征值分解 917
B.2.6 SVD 分解 917
B.3 Eigen 库 919
B.4 应用 920

附录 C Git 和 Github 923

C.1 安装配置 923
C.1.1 创建账户 923
C.1.2 安装 Git 926
C.1.3 配置密钥 926
C.1.4 (*) 账户共存 927
C.2 基本操作 928
C.2.1 初始化仓库 928
C.2.2 添加文件 928
C.2.3 记录修改 928
C.2.4 推送修改 929
C.2.5 克隆项目 929
C.3 分支操作 929
C.3.1 创建分支 929
C.3.2 分支切换 929
C.3.3 修改 PR 929
C.3.4 (*) 创建 gh-pages 分支 930
C.4 R 与 Git 交互 930
C.4.1 从 R 操作 Git 930
C.4.2 分析 Git 记录 931
C.5 (*) 辅助工具 931
C.5.1 语法高亮 932
C.5.2 文本接口 932
C.5.3 大文件存储 932

© 黄湘云

插图目录

1 影响植物生长的因素 3
a 箱线图 3
b 脊柱图 3
2 数据可视化为何如此重要 11
3 数据可视化为何如此重要 12
a 第一组数据 12
b 第二组数据 12
c 第三组数据 12
d 第四组数据 12

6.1 ggplot2 绘图三要素 40
a 只有数据 40
b 只有数据和坐标映射 40
c 数据、坐标映射和点图层 . . 40
d 数据、坐标映射、点图层和视觉映射（可选） 40
6.2 添加标签 41
a 默认设置 41
b 自定义标签 41
6.3 人均 GDP 做对数变换 42
6.4 刻度标签随数据变换调整 42
6.5 设置数据展示范围 43
6.6 设置数据展示范围 44
6.7 添加次刻度线，提供更多参考 45
6.8 使用 RColorBrewer 包提供的 Set1 调色板 46
6.9 手动挨个指定分类变量的颜色 47
6.10 在两个图例中保留一个 48
6.11 修改图例刻度标签 49
6.12 ggplot2 内置的经典主题风格 51
6.13 图例置于图形下方 52
6.14 微调图例位置 53
6.15 ggthemes 的极简主题 Tufte 54
6.16 添加文本注释 56
6.17 缓解文本注释相互覆盖的问题 58
6.18 按收入水平变量分面 59
6.19 按区域变量分面 60
6.20 制作动画 62
6.21 patchwork 组合多幅子图 64
6.22 R 语言与生成艺术 67
a 函数 `geom_regon()` 67
b 函数 `geom_spiro()` 67
c 函数 `geom_diagonal()` 67
d 函数 `geom_spoke()` 67

7.1 过去 25 年代码提交次数的变化情况 71
7.2 提交代码的时段分布 72
7.3 提交代码的月份分布 73
7.4 25 年代码逐年提交量的变化趋势 . . 74
7.5 矩形图层构造瀑布图 75
7.6 过去 25 年代码提交次数的变化情况 76
7.7 过去 25 年代码提交次数的变化情况 77
7.8 过去 25 年代码提交次数的变化情况 78
7.9 过去 25 年代码提交次数的变化情况 81
a 自定义样条插值 spline 81
b 广义可加模型样条拟合 81
c 自由度为 3 的正交多项式拟合 81
7.10 各开发者的提交量趋势 83
7.11 25 年代码提交量变化趋势图 84
7.12 25 年代码提交量变化趋势图 85

© 黄湘云

7.13 25 年代码提交量变化热力图 86
7.14 25 年代码提交量变化热力图 87
7.15 最近 5 年休息和工作日打码活跃度 . 89
7.16 25 年 R 软件发版情况 91
7.17 2020-2022 年 R 软件发版情况 94
7.18 25 年里 R 软件重大及主要版本发布情况 95
7.19 分年龄段比较城市、镇和乡村的性别比数据 97
7.20 分年龄段比较城市、镇和乡村的性别比数据 98
7.21 分年龄段比较城市、镇和乡村的性别比数据 99
7.22 分年龄段比较城市、镇和乡村的性别比数据 100
7.23 词云图 101
7.24 开发者提交量排行榜 102
7.25 维护者提交量占比 103
7.26 维护者提交量占比 104
7.27 维护者提交量占比 105
7.28 维护者提交量占比 107
7.29 维护者提交量分布 108
7.30 代码提交量的比例趋势 109
7.31 代码提交量的比例趋势 110
7.32 代码提交量的比例分布 111
7.33 加州伯克利分校院系录取情况 112
7.34 G20 主要经济体的 GDP 占比 114
7.35 你喜欢数学吗 117
7.36 Likert 图 118
7.37 Github 打卡日历图 119

8.1 箱线图的几种绘制形式 122
a ggplot2 包 122
b lvplot 包 122
c Base R 包 122
8.2 提琴图 123
a 函数 geom_violin() 123
b vioplot 包 123
c beanplot 包 123
8.3 直方图 124
a 函数 geom_histogram() . . . 124
b 函数 geom_freqpoly() 124
8.4 密度图 124
8.5 累积分布密度图 125
a 堆积密度图 after_stat(density) 125
b 堆积密度图 after_stat(density * n) 125
8.6 堆积密度图 126
a 函数 after_stat(density) . 126
b 函数 after_stat(density * n) 126
8.7 二维联合密度图 127
8.8 描述边际分布 128
8.9 ggplot2 包绘制二维填充密度图 . . . 129
8.10 ggdensity 包绘制二维填充密度图 . . 130
8.11 描述数据分布 131
a 岭线图 131
b 岭线图和抖动图组合 131
c 岭线图和轴须图组合 131
8.12 散点图 133
a 函数 geom_point() 133
b 函数 stripchart() 133
c 函数 geom_jitter() 133
8.13 加强版的抖动图 134
a ggforce 包的函数 geom_sina() 134
b 函数 geom_sina() 叠加函数 geom_violin() 134
c ggbeeswarm 包的函数 geom_quasirandom() 134
d 函数 geom_quasirandom() 叠加函数 geom_violin() . . 134
8.14 文盲率与抚养比的关系 136
8.15 文盲率和抚养比数据 137
8.16 凹凸图 139
8.17 A、B、C 三个集合的交叉关系 . . . 140
8.18 高中男生间关系的变化 142
8.19 Clopper-Pearson 置信区间 144
8.20 植物生长 145
8.21 展示假设检验的结果 147
8.22 趋势拟合、预测和推断 149

© 黄湘云

8.23 线性模型的诊断图 150
8.24 边际效应 152

9.1 分组散点图 154
9.2 调整图例位置 156
a 图例位于图右侧 156
b 图例位于内内部 156
9.3 常用图形参数 158
9.4 图形参数效果预览 159
9.5 调整点、线、图例元素 160
9.6 latticeExtra 内置的两个主题 161
a ggplot2 包默认的绘图主题 . . 161
b 《经济学人》杂志的绘图主题 161
9.7 分组柱形图 163
9.8 分组箱线图 164
9.9 经验分布图 165
9.10 调色板 165
9.11 回归曲线图 167
a 线性回归 167
b 局部多项式回归 167
c 自然样条回归 167
d 广义可加回归 167
9.12 置信区间图 168
9.13 分组置信椭圆图 169
9.14 分面水平图 171
9.15 三维散点图 172
9.16 添加三维网格参考线和透视曲线 . . . 175
9.17 三维透视图 177
9.18 奥克兰火山地形图 178
9.19 奥克兰火山的地形轮廓图 179
9.20 奥克兰火山的地形轮廓图 180
9.21 共和党在各州的得票率 182
9.22 对比 Base R 和 lattice 制作的分组散点图 183
a Base R 图形 183
b lattice 图形 183

10.1 三种鸢尾花 185
a versicolor 杂色鸢尾 185
b setosa 山鸢尾 185
c virginica 弗吉尼亚鸢尾 185
10.2 快速作图函数 `plot()` 186
a 公式语法绘制散点图 186
b 带背景参考线的散点图 186
10.3 标签 187
10.4 字体 188
10.5 分组 189
10.6 分组 190
10.7 默认调色板 191
10.8 配色 191
10.9 注释 192
10.10 图例 193
10.11 图例 194
10.12 图例 195
10.13 分组线性回归 197
10.14 正态总体下两样本均值之差的检验 . 199
10.15 数据可视化很重要 200
10.16 颜色图 201
10.17 透视图 202
10.18 等值线图 203
10.19 填充等值线图 204
10.20 plot2 包绘制分组散点图 205
10.21 plot2 包调整图例位置 206
10.22 plot2 包绘制密度曲线图 207
10.23 奥克兰火山地形图 209
a 函数 `image2D()` 二维颜色图 . 209
b 函数 `persp3D()` 三维透视图 . 209
10.24 matplotlib 绘制三维透视图 210

11.1 TikZ 绘图 212
11.2 PSTricks 绘图 213
11.3 PGF 绘制曲线图 214
11.4 PGF 绘制曲线图 214
11.5 PGF 绘制曲线图 215
11.6 TikZ 绘制三维图 viridis 调色板 . . . 216
11.7 TikZ 绘制三维图 jet 调色板 217
11.8 TikZ 绘制三维图 cool 调色板 217
11.9 柯尼斯堡七桥 219
11.10 TikZ 绘制思维导图 221
11.11 气泡图 223

© 黄湘云

11.12 描述图 224
11.13 简单线性模型 225
11.14 贝塞尔函数图像 227
a 区间 $(10^{-8}, 10^2)$ 上的贝塞尔函数 227
b 区间 $(0, 4)$ 上的贝塞尔函数 . 227

12.1 二维核密度估计 229
a faithful 数据集的散点图 . . . 229
b 点的亮暗表示核密度估计值的大小 229
c 等高线表示核密度估计值 . . 229
d 等高线表示核密度估计值 . . 229
12.2 2020 年中国地区级男女性别比分布 . 230
a 原始性别比数据 230
b 对数变换后的性别比数据 . . 230
12.3 1933-2020 年美国男性死亡率曲线 . . 232
12.4 1933-2020 年美国男性死亡率热力图 233
12.5 弗吉尼亚州各年龄段死亡率的对比 . 234
12.6 弗吉尼亚州城乡死亡率的对比 235
12.7 不同切工的钻石随价格的分布 236
12.8 二项分布参数的置信带 239
12.9 二项分布参数的几种区间估计：覆盖概率随成功概率的变化 241

13.1 默认风格的简单散点图 244
13.2 普通散点图 247
13.3 地震震级的频数分布图 248
13.4 地震震级的频率分布图 249
13.5 地震震级的频率分布图 250
13.6 空间点数据的核密度估计 251
13.7 1974 年美国各州的人均收入 253
13.8 设置数学公式 255
13.9 ggplot2 绘制的静态图形 256
13.10 空间点数据图 258

15.1 Shiny 生态系统 278

16.1 三种鸢尾花 281
a versicolor 杂色鸢尾 281
b setosa 山鸢尾 281
c virginica 弗吉尼亚鸢尾 281
16.2 流程图 281
16.3 一幅简单的 ggplot2 图形 282

17.1 newtxmath 包渲染的公式效果 290
17.2 verbatim 抄录环境 291
17.3 lstlisting 抄录环境 291
17.4 图片的标题 292
17.5 Peaks 函数图像 299

19.1 单样本检验 310
19.2 两样本检验 314
19.3 两样本均值之差的检验 315
19.4 学生睡眠数据的分布 319
19.5 办公软件 Numbers 的两样本 t 检验 . 320
19.6 多样本检验 325
19.7 植物干重 326
19.8 1879 年迈克尔逊光速实验数据 . . . 332
19.9 最重要的统计学家及其学术传承关系 345
19.10 OJ 和 VC 的交互作用 348
19.11 鸢尾花萼片长度的分布 349
19.12 不同喂食方式对小鸡的影响 354
19.13 不同喂食方式对小鸡的影响（续） . 355

20.1 子代身高与亲代身高的关系 357
20.2 子代身高与亲代身高的关系 358
20.3 二维核密度估计与二元正态分布 . . . 359
20.4 预期寿命与人均收入的关系图 361
20.5 分地域预期寿命与人均收入的气泡图 362
20.6 1977 年美国各州预期寿命与人均收入的关系：回归分析 363

21.1 百分比堆积柱形图展示多维分类数据 373
21.2 桑基图展示多维分类数据 374
21.3 马赛克图展示多维分类数据 375
21.4 马赛克图展示多维分类数据 376
21.5 加州伯克利分校院系录取情况 385
21.6 加州伯克利分校各院系录取情况 . . . 386

22.1 t 检验的功效 392

23.1 CRAN 上 R 包的更新情况 400

© 黄湘云

23.2 CRAN 上的维护者活跃情况 401
23.3 高产的 R 包开发者 407
23.4 开发者数量的分布 408
a 直方图 408
b 直方图（对数尺度） 408
23.5 节点的入度和出度的分布 413
a 入度的分布 413
b 出度的分布 413
23.6 开发者的协作关系网络 414
23.7 探测协作关系网络中的社区 417
23.8 开发者的影响力网络 418
23.9 开发者的影响力网络（visNetwork） 419

24.1 益辉每年发布的日志数量 425
24.2 词云可视化词频结果 427
24.3 主题分布 429

25.1 急性粒细胞白血病 432
25.2 急性粒细胞白血病生存数据 434
25.3 比例风险回归 436

26.1 美团在香港上市以来的股价走势 . . . 441
26.2 美团调整的股价逐年走势 442
26.3 美团在香港上市以来的股价走势 . . . 443
26.4 美团 2021 年的股价走势 444
26.5 美团股价走势 445
26.6 美团股价变化趋势 446
26.7 乘客数量自相关图 447
26.8 乘客数量偏自相关图 448
26.9 简单指数平滑模型 454
26.10 简单指数平滑模型预测 455
26.11 holt 指数平滑模型 456
26.12 可加 Holt-Winters 平滑模型拟合 . . 458
26.13 可乘 Holt-Winters 平滑模型拟合 . . 459
26.14 变化趋势的分解 460
26.15 季节性调整 461
26.16 变化趋势的分解 462

27.1 斐济地震的空间分布 469
a 坐标参考系 4326（默认） . . 469
b 坐标参考系 3460 469
27.2 凸包 470
a 凸包（base R） 470
b 凸包（ggplot2） 470
27.3 热力图 471
a 核密度估计 471
b 核密度估计（原始数据） . . 471

28.1 朗格拉普环礁和朗格拉普岛 473
28.2 采样点在岛上的分布 474
a 采样分布 474
b 采样顺序 474
28.3 岛上各采样点的核辐射强度 475
28.4 岛上各采样点的辐射强度 476
28.5 残差的空间分布 477
28.6 残差分布图 478
a 残差与编号的关系 478
b 残差与横坐标的关系 478
c 残差与纵坐标的关系 478
28.7 梅隆型自相关函数曲线 480
28.8 理论半变差函数图 484
28.9 残差的经验半变差图 485
28.10 标准化残差的经验半变差图 486
28.11 朗格拉普岛海岸线的表示 489
a 点数据 489
b 多边形数据 489
28.12 朗格拉普岛海岸线及其缓冲区 . . . 490
28.13 朗格拉普岛规则化网格操作 491
28.14 朗格拉普岛规则网格划分结果 . . . 492
28.15 朗格拉普岛核辐射强度的分布 . . . 494
28.16 核辐射强度预测方差的分布 495
28.17 朗格拉普岛核辐射强度的分布 . . . 496

29.1 苏格兰各地区唇癌病例数分布 499
29.2 因袭击被逮捕的人数分布 500
29.3 逮捕人数比例与城市化率的关系 . . . 501

30.1 逻辑斯谛分布 505
a 概率密度函数 505
b 概率分布函数 505
30.2 二维情形下的逻辑回归模型的负对数似然函数曲面 508

© 黄湘云

30.3 回归系数的迭代路径 512
30.4 惩罚系数的迭代路径 513
30.5 ROC 曲线 515

31.1 数值优化扩展包、插件包和 ROI 包的关系图 518
31.2 对比无约束和有约束条件下的解 . . . 523
31.3 常见的三维凸锥 524
31.4 锥 526
31.5 一维函数图像 534
31.6 隐函数图像 537
31.7 二维 Rastrigin 函数图像 540
31.8 局部放大前后的函数图像 542
a 区域 $[-50,50]\times[-50,50]$ 内的函数图像 542
b 区域 $[0,12]\times[0,12]$ 内的函数图像 542
31.9 类香蕉函数的曲面图 544
31.10 目标函数的曲面图 568

32.1 10 个城市的分布图 570
32.2 10 个城市的路线图 572
32.3 对数似然函数的曲面图 579
32.4 对数似然函数的等高线图 580
32.5 伽马分布的概率密度函数 586

33.1 Stan、CmdStan 和 CmdStanR 等的依赖关系图 591
33.2 lp___ 的后验分布 595
33.3 几个常用的概率分布 602
33.4 黄石公园老忠实间歇泉 617

34.1 β_1 和 β_2 的联合分布 623
34.2 β_1 和 β_2 的联合分布 624
34.3 β_1 和 β_2 的热力图 625
34.4 各个参数的轨迹图 626
34.5 各个参数的分布图（密度图） 627
34.6 各个参数的分布图（岭线图） 628
34.7 各个参数的分布图（岭线图） 629
34.8 NUTS 能量诊断图 630
34.9 后验预测诊断图（密度图） 631
34.10 后验预测诊断图（区间图） 632
34.11 食道癌发病率与年龄组、酒精量的关系 636
34.12 航天飞机 O 型环在不同温度下失效的条件密度图 641

35.1 Base R 绘制参数 μ 的迭代轨迹 . . . 649
35.2 rstan 绘制参数 μ 的迭代轨迹 650
35.3 rstan 包绘制后验分布图 651
35.4 bayesplot 包绘制后验分布图 652
35.5 参数 $\mu,\log(\tau)$ 的联合分布 660
35.6 30 只小鼠 5 次测量的数据 662
a 小鼠重量的分布 662
b 小鼠重量的变化 662
35.7 小鼠重量变化曲线 663
35.8 参数 $\boldsymbol{\alpha}$ 的后验分布 684
35.9 参数 $\boldsymbol{\beta}$ 的后验分布 685

36.1 sleepstudy 数据集 707
36.2 分面展示 sleepstudy 数据集 708
36.3 边际效应图 711
36.4 感染比例随变量 herd 和 period 的变化 719
36.5 火炬松树的高度（英尺）随时间（年）的变化 732
36.6 方差协方差参数的后验分布 750
36.7 Puromycin 反应速率变化趋势 753

37.1 mcycle 数据集 755
37.2 mcycle 数据集 757
37.3 ggplot2 平滑 758
37.4 后验预测分布检查 762
a 样条平滑效应 762
b 后验预测分布 762
37.5 岛上各采样点的辐射强度 764
37.6 核辐射强度的预测分布 766
37.7 截距项的边际后验概率密度分布 . . . 768
37.8 核辐射强度的预测分布 773
37.9 某城市的地形 774
37.10 某城市边界线 776
37.11 某城市地形图 777

© 黄湘云

37.12 重金属砷 As 和镉 Cd 的浓度分布 . 778
a 重金属砷 As 778
b 重金属镉 Cd 778
37.13 重金属汞 Hg 的浓度分布 779

38.1 采样序列的轨迹和分布 783
38.2 采样序列的轨迹和分布 784
38.3 生成二元正态分布的随机数 785
38.4 三元正态分布 789
38.5 位置 1 和 2 处的迭代轨迹 792
38.6 二维高斯过程 793
38.7 朗格拉普岛 798
38.8 σ 和 ϕ 的迭代轨迹 801
38.9 σ 和 ϕ 的后验分布 802
38.10 后验预测诊断图（密度图） 807
38.11 朗格拉普岛核辐射强度的分布 . . . 809
38.12 朗格拉普岛核辐射强度的分布 . . . 810

39.1 美团股价走势 817
39.2 美团股价对数收益率的情况 818
a 对数收益率的变动 818
b 对数收益率的分布 818
39.3 对数收益率的自相关图 818
39.4 对数收益率的残差平方 821
a 自相关图 821
b 偏自相关图 821
39.5 非线性回归 826
39.6 AirPassengers 的时序图 828
a 对数尺度 828
b 一阶差分 828
39.7 AirPassengers 的拟合图 831
a 随机游走模型 831
b 季节效应模型 831
39.8 年和月的趋势变化 832
39.9 趋势拟合效果 833
39.10 交互作用 834
39.11 趋势拟合效果 835
39.12 趋势拟合效果 836
39.13 单层感知机预测 838
39.14 多层感知机预测 840
39.15 预测月粒度太阳黑子数量 841

40.1 迭代路径 846
a 回归系数 setosa 的迭代路径 . 846
b 回归系数 versicolor 的迭代路径 846
c 回归系数 virginica 的迭代路径846
d 惩罚系数的迭代路径 846
40.2 分类回归树 853
40.3 随机森林 854
40.4 变量重要性 855

42.1 RMSE 随成分数量的变化 865
42.2 岭回归 867
a 回归系数的迭代路径 867
b 惩罚系数的迭代路径 867
42.3 Lasso 回归 868
a 回归系数的迭代路径 868
b 惩罚系数的迭代路径 868
42.4 弹性网络 870
a 回归系数的迭代路径 870
b 惩罚系数的迭代路径 870
42.5 自适应 Lasso 回归模型的超参数选择 871
a 岭回归 871
b 自适应 Lasso 回归 871
42.6 回归系数的迭代路径 873
42.7 回归系数的迭代路径 875
42.8 惩罚系数的迭代路径 877
42.9 回归系数的迭代路径 878
42.10 惩罚系数的迭代路径 880
42.11 Lasso 和最小角回归系数的迭代路径 881
a Lasso 回归 881
b 最小角回归 881
42.12 交叉验证均方误差的变化 882
a Lasso 回归 882
b 最小角回归 882
42.13 最优子集回归 883
a 惩罚系数的迭代路径 883
b 交叉验证筛选变量个数 883
42.14 分类回归树 885

© 黄湘云

C.1 点击注册 923
C.2 输入邮箱 924
C.3 输入用户名 924
C.4 回答问题 925
C.5 输入验证码 925
C.6 验证账户 926
C.7 创建代码仓库 927
C.8 Git 分支操作 929

© 黄湘云

列表目录

1 datasaurus_dozen 数据集的一些描述性统计量和线性回归结果 8
1 datasaurus_dozen 数据集的一些描述性统计量和线性回归结果 9
2 anscombe 数据集 9

6.1 gapminder 数据集（部分） 37
6.2 **ggplot2** 包可以绘制丰富的统计图形 38

7.1 数学公式与 R 语言表示的计算公式 . 81
7.2 R 软件发版数据集（部分） 92
7.3 中国各年龄段的性别比数据（部分） 96
7.4 SVN 日志中的贡献者（部分） . . . 105
7.4 SVN 日志中的贡献者（部分） . . . 106
7.5 G20 国家经济水平：GDP 总量、人类发展指数等 113
7.6 您觉得在本页面，找想看的消息方便吗？ 115
7.7 你对数学感到焦虑吗？ 115

8.1 各省、直辖市和自治区分区域的性别比数据（部分） 120
8.2 2020 年各省、直辖市、自治区，总抚养比和文盲率相关数据（部分） . 135
8.3 二项分布的分布列 143

9.1 lattice 和 latticeExtra 包的部分函数 183

12.1 1940 年美国弗吉尼亚州死亡率 . . . 234

13.1 **plotly** 包可以绘制丰富的统计图形 . 243
13.2 plotly 包支持绘制的散点图类型 . . . 246

14.1 数据配色 263
14.2 Base R 包内置的数据集 264

16.2 鸢尾花数据集 282
16.3 鸢尾花数据集 282
16.4 几种列表 283

17.1 LaTeX 公式排版环境 289
17.3 表格的标题 293
17.4 制作表格的管道语法 299

19.1 t 分布的分位数表 312
19.2 χ^2 分布的分位数表 313
19.3 检验方法分类 323
19.4 线性回归的输出 327
19.5 配对样本检验 333
19.6 多重假设检验 335
19.7 对假设检验理论有重要贡献的学者 . 343

20.1 高尔顿收集的 205 对夫妇及其子女的身高数据（部分） 356
20.2 子女身高向中亲平均身高回归 359
20.3 1977 年美国人口调查局发布的各州统计数据（部分） 360

21.1 泰坦尼克号乘客生存死亡统计数据 . 372
21.2 佛罗里达州的凶杀案件统计数据 . . . 377
21.3 卡方独立性检验 377
21.4 加州伯克利分校的录取情况 384

22.1 函数 `power.t.test()` 的参数及其含义 392

© 黄湘云

23.1 CRAN 团队开发维护 R 包数量情况 405
a 表 405
b 续表 405
23.2 Brian Ripley 维护的 R 包 405
23.3 RStudio 团队开发维护 R 包数量情况（部分） 406
a 表 406
b 续表 406

28.1 朗格拉普岛核辐射检测数据及海岸线坐标数据 473
a 核辐射检测数据 473
b 海岸线坐标数据 473

29.1 数据集 USArrests（部分） 499
29.1 数据集 USArrests（部分） 500

31.1 **ROI** 包可以表示的目标函数和约束条件 520
31.2 R 软件内置的非线性优化函数 534
31.3 常用的非线性优化求解器 553

32.1 死亡人数的统计 582
32.2 一些互联网公司及股票代码 588

33.1 Stan 变量类型和 R 语言中的对应 . . 598

34.1 食道癌研究数据（部分） 634
34.2 广义线性模型各个参数的估计结果 . 637
34.2 广义线性模型各个参数的估计结果 . 638

35.1 小鼠重量数据（部分） 661
35.2 频率派方法比较 696
35.3 贝叶斯方法比较 697
35.4 实验数据 698

36.1 响应变量的分布 718
36.2 Loblolly 数据集 731
36.3 混合效应模型及相关 R 包拟合函数 . 752

38.1 不同模型与参数估计方法的比较 . . . 811

42.1 惩罚回归的 R 包实现 866

A.1 数学符号表 901
A.2 统计术语的英文缩写 902

©黄湘云

欢迎

> **警告**
>
> Book in early development. Planned release in 2024.

本书初稿是在 RStudio IDE 内使用 Quarto 编辑的，Quarto 是继R Markdown之后，一个新的开源的科学和技术发布系统，它基于 Pandoc支持输出多种格式的书稿，比如 HTML 网页、EPUB 电子书、DOCX 文档和 PDF 便携式文档等。Quarto 吸收了过去 10 年 R Markdown 取得的经验和教训，在书籍写作、创建博客、制作简历和幻灯片等系列场景中支持更加统一的使用语法，一份源文档输出多种格式，使文档内容在不同场景中的迁移成本更低。了解更多 Quarto 特性，请访问 https://quarto.org/。

书中的代码字体采用美观的 Source Code Pro 字体，为方便跨操作系统编译书籍电子版，正文的中文字体采用开源的 fandol 字体。此外，考虑到美观性，本书图形使用了 Noto 系列中英文字体，它们来自 Google Fonts 字体库，分别是 Noto Sans 无衬线英文字体和 Noto Serif SC 宋体中文字体。

书中 R 包名以粗体表示，如 **knitr** 包，函数名以等宽体表示，如 `plot()`，函数的参数名同理。代码块内注释用 `#` 表示，运行结果每一行开头以 `#>` 标记。本书写作过程中，依赖 **knitr** (Xie 2015)、**ggplot2** (H. Wickham 2016) 和 **lattice** (Sarkar 2008) 等众多 R 包。考虑到要同时支持 DOCX、EPUB、PDF 和 HTML 四种书籍格式，书中使用 **knitr** 包和 **gt** 包制作静态的表格。

为方便测试贡献者提供的 PR，本书托管在 Github 上，同时启用 Github Action 服务，为书籍自定义了一个可复现全书内容的运行环境，包括 R 软件、扩展包和系统软件依赖，详见仓库中的 DESCRIPTION 文件。你现在看到的是在线编译版本，使用 Quarto 1.4.547，最新一次编译时间是 2024-01-23 17:24:32。

```
xfun::session_info(packages = c(
  "ggplot2", "gganimate", "ggrepel", "ggraph",
  "ggridges", "ggsignif", "ggforce", "ggdensity",
  "ggbeeswarm", "ggeffects", "ggnewscale",
  "patchwork", "shiny", "plotly", "lattice",
  "dplyr", "purrr", "tidyr", "data.table",
  "rsconnect", "knitr", "rmarkdown", "gt", "DT",
  "mgcv", "glmnet", "lme4", "xgboost", "spaMM",
  "sf", "stars", "terra", "INLA", "cmdstanr",
  "showtext", "gifski", "tinytex", "magick"
```

内部资料 ©

```
), dependencies = FALSE)
#> R version 4.3.2 (2023-10-31)
#> Platform: x86_64-apple-darwin20 (64-bit)
#> Running under: macOS Ventura 13.6.3
#>
#> Locale: en_US.UTF-8 / en_US.UTF-8 / en_US.UTF-8 / C / en_US.UTF-8 / en_US.UTF-8
#>
#> Package version:
#>   cmdstanr_0.7.1     data.table_1.14.10 dplyr_1.1.4        DT_0.31
#>   gganimate_1.0.8    ggbeeswarm_0.7.2   ggdensity_1.0.0    ggeffects_1.3.4
#>   ggforce_0.4.1      ggnewscale_0.4.9   ggplot2_3.5.0      ggraph_2.1.0.9000
#>   ggrepel_0.9.5      ggridges_0.5.5     ggsignif_0.6.4     gifski_1.12.0.2
#>   glmnet_4.1.8       gt_0.10.1          INLA_23.9.9        knitr_1.45
#>   lattice_0.21.9     lme4_1.1.35.1      magick_2.8.2       mgcv_1.9.0
#>   patchwork_1.2.0    plotly_4.10.4      purrr_1.0.2        rmarkdown_2.25
#>   rsconnect_1.2.0    sf_1.0.15          shiny_1.8.0        showtext_0.9.6
#>   spaMM_4.4.16       stars_0.6.4        terra_1.7.65       tidyr_1.3.0
#>   tinytex_0.49       xgboost_1.7.6.1
#>
#> Pandoc version: 3.1.11
#>
#> LaTeX version used:
#>   TeX Live 2023 (TinyTeX) with tlmgr 2024-01-13
```

© 黄湘云

前言

为什么是 R 语言？

R 语言在统计图形方面不仅走得早还走得远，当然，Python 语言也不错，近年来新起的 Julia 语言也很好。R 语言在统计图形方面的沉淀是非常深厚的，近年来，我发现越是简洁的越是优美，灵活的东西使用起来还非常简单，以 R 包 **datasets** 内的数据集 PlantGrowth 为例，一般地，展示数据的分布会想到箱线图、直方图、密度图等，R 函数的泛型设计可以根据数据对象和变量的类型自动选择合适的图形，图 19.7 是泛型函数 `plot()` 调用普通函数 `boxplot()` 和 `spineplot()` 绘制的。

(a) 箱线图　　(b) 脊柱图

图 1: 影响植物生长的因素

所以，直接调用相应的绘图函数也是可以的，如下：

```
boxplot(weight ~ group, data = PlantGrowth,
        ylab = "植物干重", xlab = "组")
spineplot(cut(weight, 2) ~ group, data = PlantGrowth,
          ylab = "植物干重", xlab = "组")
```

非常清晰 ©

脊柱图是马赛克图的一种特殊情况，也可以看做是堆积条形图的推广形式或者直方图的扩展。上面 `cut()` 函数的作用是将数值型变量 weight 分桶，对照组（control，简写 ctrl）和两个不同的实验组（treatment，简写 trt）都按同样的划分方式分作两桶。

```
dat <- transform(PlantGrowth, weight_bucket = cut(weight, 2))
aggregate(data = dat, weight ~ weight_bucket + group, FUN = length)
```

```
#>   weight_bucket group weight
#> 1   (3.59,4.95]  ctrl      4
#> 2   (4.95,6.31]  ctrl      6
#> 3   (3.59,4.95]  trt1      8
#> 4   (4.95,6.31]  trt1      2
#> 5   (3.59,4.95]  trt2      1
#> 6   (4.95,6.31]  trt2      9
```

为什么写这本书？

近年来，数字经济成为热门词汇，企业数字化转型离不开数据，精细化运营更离不开数据分析，数据分析受到越来越多的关注。在数据分析领域，R 语言越来越流行，一本以 R 语言为依托，以实战为导向的数据分析书，市面上还不多。

1. 提供完整可复现的书籍源码，书中示例可以在 R 语言环境下复现。
2. 数据可视化部分，以一个真实数据串联绘图的基本要素，从图形的用途出发，将图形分类，结合真实数据介绍图形。
3. 展现数据分析的完整工作流，数据获取、操作、处理、可视化探索和分析、展示交流、建模分析、解释。
4. 将工作流应用于特定领域的数据分析，覆盖网络数据、文本数据、时序数据、空间数据等四大常见且重要的场景。

本书是怎么写的？

本书在写作风格上借鉴了以下书籍

- 《现代统计图形》(赵鹏, 谢益辉, 和黄湘云 2021) 讲清楚统计图形的来龙去脉，提供丰富的实战案例。
- 《R in Action》(Kabacoff 2022) 根据入门、进阶和高阶将书籍内容分出层次。
- 《R for Data Science》(H. Wickham, Çetinkaya-Rundel, 和 Grolemund 2023) 根据数据分析的整个工作流拆分各个部分、章节。

本书的写作素材来源非常广泛，比如

- 大量的原始论文、书籍，回顾经典理论、数据案例，追根溯源

© 黄湘云

- 大量的 R 包帮助文档，配合真实数据提供软件工具的使用说明
- 一些国内外政府网站发布的权威数据，提供大量的实际案例数据
- 从国内外论坛、书店搜集数据操作、展示和交流等方面的高频问题

写作理念是什么？

1. 以真实的数据为基础，介绍数据分析所用到的软件工具、统计方法和算法模型，对经典的数据分析案例，力求还原历史，讲清楚故事背景，数据处理的过程，不单单是分析方法和结果。

2. 尽可能选用来自社会、经济、文化、历史等方面的真实的、最新的或经典的数据，在讲数据分析技术的同时，也了解一点我们所处的社会，希望给读者一些启发，勾起读者的兴趣，主动探寻有趣的问题，收集整理所需的数据，做自己的研究，找到问题的答案，享受数据探索分析的过程，摸索出适合自己的分析方法和分析工具。

3. 结合多年使用 R 语言的经验以及最近几年在互联网行业工作的体会，形成数据分析师的技能栈，梳理知识体系，沉淀一套数据分析的方法。

目标读者是哪些？

1. 想通过编程实现数据分析的完整过程，使得整个过程可以复现，可以重复利用。

2. 对数据分析的实战有兴趣，想将数据分析技能应用于解决实际问题。

本书有哪些内容？

1. 入门部分：介绍软件 R、RStudio 和 VS Code 的安装配置过程，常见的基本数据结构和类型，循环、判断、函数等基本的编程知识。

2. 数据部分：从本地文件、远程数据库、网页爬取等数据获取方式，筛选、变换、重塑、排序等基础的数据操作，离群值、异常值检测，缺失值处理等基础的数据处理

3. 展示部分：ggplot2 基础、统计图形、实战应用、经验总结

4. 交流部分：交互的图形、表格和应用，动态的 HTML 网页、PDF 文档和办公文档。

5. 建模部分：线性模型、广义线性模型、混合效应模型、数据挖掘算法和神经网络模型

6. 应用部分：网络数据、文本数据、时序数据、空间数据的分析

7. 其它部分：参数估计、假设检验和抽样分布等基础的统计推断，L-BFGS 算法、EM 算法等统计计算，自助法、重抽样等统计模拟。

© 黄湘云

公开数据从哪找？

- 各国、各级政府的统计局，比如美国人口调查局、中国国家统计局等。
- 国际、国内各类组织机构，比如世界银行、美国疾病预防控制中心等。
- 各类网站提供的数据集，比如 GitHub 开放数据集列表 awesome-public-datasets，kaggle 网站提供大量数据分析竞赛及相应的数据集。
- R 包内置数据集，已整理得很好，比如 **spData** 包收集整理了很多空间统计方面的数据集。Rdatasets 更是收集约 1900 个数据集，全部来自 CRAN 上发布的 R 包。
- 一些 R 包封装数据下载的接口，比如**tidyBdE**包可以下载西班牙银行开放的数据，**WDI** 可以下载世界银行开放的数据。

学会有效地提问？

- 想清楚自己的问题是什么？尽力做好拆解和界定。
- 去掉枝叶，保留主干，提供最小的可重复的示例。
- 有耐心地等待社区的回应，积极地与社区沟通。
- 为社区提供力所能及的帮助，提升自己的影响力。

© 黄湘云

介绍

> 提示
>
> 其它小节完成后再写本节。

本书围绕数据分析实战工作流分四大部分：

(1) 数据收集和整理。
(2) 数据探索和分析。
(3) 数据建模和解释。
(4) 数据交流和应用。

对分析师来说，相比于整理、探索、建模和交流，收集、分析、解释和应用是更难的事。始终围绕 R 语言、数据分析、实战来介绍每一部分、每一章的内容，让每一部分、每一章的内容都在丰富和解释 R 语言、数据分析、实战主题。

《Design Principles for Data Analysis》在数据分析的实践中，提炼出设计思维，在解决问题的过程中，理解解决方案为谁而设计。不同的数据分析师（数据分析的生产者）在分析方法、工具和工作流等方面的选择，不仅影响数据分析产品本身，而且影响数据分析的消费者的体验。生产者的角色可以看作围绕一套设计原则设计数据分析，基于这套原则去量化。

2015 年《科学》杂志发表重量级文章《Estimating the reproducibility of psychological science》，讨论了目前心理学的可重复性问题。可重复性受到越来越多的关注，数据分析是科学研究中非常重要的一环，可重复性数据分析的需求越来越大，在真实数据的基础上，本书试图通过 R 语言展现数据分析的技术栈，包括数据获取、数据清洗、数据整理和数据操作，数据探索和分析，数据建模和结果解释，以及数据展示和交流。

近年来，R 语言社区在数据分析领域频频发力，特别是 RStudio 公司及其打造的产品矩阵。数据获取、探索和分析方面，有 ggplot2、dplyr、tidyr、purrr 及整个 tidyverse 家族。数据交流和应用方面，有 Shiny、Quarto、flexdashboard、R Markdown。数据建模和解释方面，有 tensorflow、keras、vetiver、plumber 及整个 tidymodels 家族。数据工作流集成方面，提供许多接口 R 包，支持大量数据库的连接驱动，sparklyr 与 Apache Spark 连接实现大规模数据处理，**renv** 实现系统软件依赖和 R 包版本管理，**reticulate** 包实现与 Python 连接，支持导入许多 Python 社区的模块。还有一系列遵循 tidyverse 设计原则的数据分析框架，比如 DrWhy、mlr3verse、easystats、tidymodels、fastverse、healthyverse、fable 等。

© 统计出版社

RStudio 公司在解决数据分析技术栈，提供通用解决方案，而在特定领域内，也逐渐走向整合，形成生态。2005 年 Rigby R. A. 和 Stasinopoulos D. M. **gamlss** 包 https://www.gamlss.com/ 试图将一系列统计模型，如线性模型 LM、广义线性模型 GLM、广义可加模型 GAM、线性混合效应模型 LMM、广义线性混合效应模型 GLMM、广义可加混合效应模型 GAMM 纳入统一的广义可加模型框架下 (Rigby 和 Stasinopoulos 2005)。2009 年 Håvard Rue 开发**INLA** 包，类似 **gamlss** 包，同样是整合一系列统计模型，纳入一个新的基于贝叶斯视角的集成嵌套拉普拉斯框架下，主要应用于空间统计领域，更多详见 https://www.r-inla.org/。

数据探索和分析

数据可视化是数据探索和分析的一个手段，数据可视化的主要目的有两个：其一是探索 Explore，其二是解释 Explain。

探索是面向数据分析师自己，而展示是面向数据分析的消费者。面对不同的角色，可视化的目的是不一样的，探索是了解数据，展示是传递信息。了解数据的分布、隐藏的模式、缺失情况、异常情况，步步深入地挖掘数据的潜在规律。展示是传递数据分析的结论和洞见，强调美观、效率、效果，除了数据分析师本人几乎没人想看探索数据过程中产生的数以十计的中间图形。

数据可视化是通过计算机程序绘制图形来展示数据，有时是在图上展示原始数据，比如散点图，有时展示汇总数据，比如直方图，有时借助一些数据变换，比如对数变换，甚至更为复杂的统计变换。数据可视化主要是描述、提炼和汇总原始数据，从数据中获取信息。

除了选择合适的工具（Base R / grid / lattice / ggplot2）绘制图形（提供 R 代码实现），选择图形（30+ 多种常见图形）和解释图形（真实数据背景）往往比想的更加困难，本书试图去回答这些问题。

大多教科书侧重理论和方法，计算机强调编程，数值计算是精确的，图形是粗燥的。然而，只有模型和方法，缺乏数据探索的分析和建模，计算的结果和分析的结论可能是不正确的，数据可能在欺骗你 (Anscombe 1973)。

datasauRus 包 (Davies, Locke, 和 D'Agostino McGowan 2022) 内置了一个数据集 datasaurus_dozen，它整合了 13 个子数据集，它们在均值、标准差等描述性统计量方面十分接近，见下表格 1 。其中 $\bar{x}, \sigma_x$ 分别代表预测变量 X 的均值和标准差，$\bar{y}, \sigma_y$ 代表响应变量 Y 的均值和标准差，β_0, β_1 代表回归方程方程式 1 的截距和斜率，R^2 代表模型拟合数据的程度。

$$y = \beta_0 + \beta_1 x + \epsilon \tag{1}$$

表格 1: datasaurus_dozen 数据集的一些描述性统计量和线性回归结果

子数据集	$\bar{x}$	σ_x	$\bar{y}$	σ_y	β_0	β_1	R^2
dino	54.263	16.765	47.832	26.935	53.453	-0.104	0.004
away	54.266	16.770	47.835	26.940	53.425	-0.103	0.004
h_lines	54.261	16.766	47.830	26.940	53.211	-0.099	0.004

© 王建湘

表格 1: datasaurus_dozen 数据集的一些描述性统计量和线性回归结果

子数据集	$\bar{x}$	σ_x	$\bar{y}$	σ_y	β_0	β_1	R^2
v_lines	54.270	16.770	47.837	26.938	53.891	-0.112	0.005
x_shape	54.260	16.770	47.840	26.930	53.554	-0.105	0.004
star	54.267	16.769	47.840	26.930	53.327	-0.101	0.004
high_lines	54.269	16.767	47.835	26.940	53.809	-0.110	0.005
dots	54.260	16.768	47.840	26.930	53.098	-0.097	0.004
circle	54.267	16.760	47.838	26.930	53.797	-0.110	0.005
bullseye	54.269	16.769	47.831	26.936	53.809	-0.110	0.005
slant_up	54.266	16.769	47.831	26.939	53.813	-0.110	0.005
slant_down	54.268	16.767	47.836	26.936	53.850	-0.111	0.005
wide_lines	54.267	16.770	47.832	26.938	53.635	-0.107	0.004

诸多统计量都难以发现它们的差异，透过数据可视化这面照妖镜，却可以使数据的本来面目无所遁形，如图 2 所示。可见，单个统计量就好比管窥蠡测，稍有不慎，我们就成了盲人摸象。

数据可视化的重要性在于探索数据的真实分布，为数据建模提供假设和依据，也为验证、评估模型的效果。结合图 2 也解释了为什么线性回归模型在解释数据方面的无能为力，即 R^2 介于 0.004 至 0.005 之间，数据根本不符合线性模型的条件。

有时候是有的数据符合模型假设，而有的不符合，我们没有上帝之眼，看不到哪些符合哪些不符合。在数据集不多的情况下，可以全部展示出来，数据集很多的时候，可以抽样一部分，再展示。下面再举一个例子，anscombe 数据集来自 R 软件内置的 R 包 **datasets**，它包含四组数据 $(x_i, y_i), i = 1, 2, 3, 4$，如表格 2 所示。

表格 2: anscombe 数据集

第 1 组		第 2 组		第 3 组		第 4 组	
x1	y1	x2	y2	x3	y3	x4	y4
10	8.04	10	9.14	10	7.46	8	6.58
8	6.95	8	8.14	8	6.77	8	5.76
13	7.58	13	8.74	13	12.74	8	7.71
9	8.81	9	8.77	9	7.11	8	8.84
11	8.33	11	9.26	11	7.81	8	8.47
14	9.96	14	8.10	14	8.84	8	7.04
6	7.24	6	6.13	6	6.08	8	5.25
4	4.26	4	3.10	4	5.39	19	12.50
12	10.84	12	9.13	12	8.15	8	5.56
7	4.82	7	7.26	7	6.42	8	7.91

5	5.68	5	4.74	5	5.73	8	6.89

© 基础设施

© 黄湘云

图 2: 数据可视化为何如此重要

© 基础统计

用统计的方法发现四组数据的样本均值、方差、相关系数和回归系数几乎是相同的，实际上，借助散点图 10.15 分别描述各组数据的关系时，却发现四组数据之间有极大的差异，且只有第一组数据看起来符合线性模型的条件 (Anscombe 1973)。

(a) 第一组数据

(b) 第二组数据

(c) 第三组数据

(d) 第四组数据

图 3: 数据可视化为何如此重要

图形还告诉我们第二组数据的更适合二次非线性回归，第三组数据受到离群点的重大影响，第四组数据自变量只有两个取值，像是两个分布按不同比例混合的结果。

© 黄湘云

数据展示和交流

无论是数据表格还是交互图形，首先都承担着数据展示的基础作用，通过趋势、对比继而传递更加明确的信息和洞见，采用合适的表达方式可以高效准确地传递信息，促进交流，获取反馈，从而改善已有的分析方法和结论。

数据展示和交流主要分两大部分：其一是用户可与之交互的图形、表格和应用，其二是文档内容可重复的 HTML 动态网页文档、PDF 便携式文档、Office 办公文档。涵盖完整数据分析过程的网页文档，用于毕业的学位论文、投稿的期刊论文、出版的书籍初稿、交流的演示文稿，无论是 LaTeX 编译的 PDF 格式文档还是 DOCX 文档，R 语言社区都有非常先进的工具满足需求。

章节 十三 首先介绍 **plotly** 包绘图的基础语法以及与 **ggplot2** 包绘图的关系，其次介绍制作常用的交互图形，如条形图、直方图、箱线图、曲线图等，最后介绍一些常用的技巧，如导出静态图片、添加水印徽标等。

章节 十四 首先介绍 **DT** 包制作交互表格的基础语法，其次介绍常用的功能，如列分层分组、按列配色、列格式化、搜索排序、数据导出等，最后介绍一些基础的 CSS 和 JavaScript 知识，支持一些中高级的表格定制功能。

章节 十五 首先介绍 **shiny** 包制作交互应用的整体概览，如前端布局、后端计算、筛选器、模块交互等，其次从易到难介绍一个完整的数据应用，最后介绍生产级的 Shiny 应用开发的技术栈。

章节 十六 首先回顾 R 语言社区陆续出现的 R Sweave、R Markdown 和 Quarto 三套创作工具，其次介绍 Quarto 的基础用法，如 Markdown 基础和 Pandoc 的基础，接着根据使用场景分别介绍 HTML、PDF 和 Office 文档的特性。

© 黄湘云

第一部分

数据准备

©黄湘云

第一章　数据对象

数据类型有整型（32 位或 64 位）、逻辑型、字符型、日期型、数值型（单、双精度浮点型）等。数据结构有向量、矩阵、数组、列表和数据框等。

1.1　数据类型

1.1.1　整型

```
c(1L, 2L)
```

```
[1] 1 2
```

1.1.2　逻辑型

```
c(TRUE, FALSE)
```

```
[1]  TRUE FALSE
```

1.1.3　字符型

```
c("A", "B")
```

```
[1] "A" "B"
```

1.1.4　日期型

```
c(as.Date("2022-01-01"), as.Date("2022-01-02"))
```

```
[1] "2022-01-01" "2022-01-02"
```

©黄文清

1.1.5 数值型

```
c(1,1.2)
```

```
[1] 1.0 1.2
```

1.2 数据结构

1.2.1 向量

所有元素都是同一类型

1.2.2 矩阵

所有元素都是同一类型

1.2.3 数组

所有元素都是同一类型

1.2.4 列表

元素可以属于不同类型

1.2.5 因子

1.2.6 数据框

同列的元素类型必须一致，不同列的元素类型可以不同。

1.2.7 ts

ts 类型用于表示时间序列数据，是继承自数组类型的。给定数据、采样初始时间、采样频率的情况下，利用内置的函数 ts() 构造一个 ts 类型的分钟级的时间序列对象。

```
x <- ts(
  data = rnorm(100),
  start = c(2017, 1),
  frequency = 365.25 * 24 * 60,
```

© 基础统计


```
  class = "ts", names = "Time_Series"
)
```

ts() 函数的 start 和 frequency 参数很关键，前者指定了时间单位是天，后者指定每个时间单位下的数据点的数量。其中 365.25 是因为每隔 4 年有 366 天，平均下来，每年算 365.25 天。每隔 1 / (24 * 60) 天（即 1 分钟）采样一个点。如果初始时间不是从一年的第 1 分钟开始，而是从此时此刻 2023-01-31 10:43:30 CST 开始，则可以换算成今年的第 30 * 24 * 60 + 9 * 60 + 43 = 43783 分钟，则 Start = c(2023, 43783)。

以数据集 x 为例，它是一个 ts 类型的时间序列数据对象。时间序列对象有很多方法，如函数 class()、mode() 和 str() 分别可以查看其数据类型、存储类型和数据结构。

```
# 数据类型
class(x)
```

```
[1] "ts"
```

```
# 存储类型
mode(x)
```

```
[1] "numeric"
```

```
# 数据结构
str(x)
```

```
 Time-Series [1:100] from 2017 to 2017: -0.646 1.195 0.319 -0.528 0.136 ...
```

函数 start() 和 end() 查看开始和结束的时间点。

```
c(start(x), end(x))
```

```
[1] 2017    1 2017  100
```

函数 time() 可以查看在以上时间区间的划分。

```
time(x)
```

```
Time Series:
Start = c(2017, 1)
End = c(2017, 100)
Frequency = 525960
  [1] 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017
 [16] 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017
 [31] 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017
 [46] 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017
 [61] 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017
 [76] 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017
 [91] 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017
```

函数 tsp() 可以查看其期初、期末和周期。

```
tsp(x)
```

```
[1]   2017   2017 525960
```

内部稿 ©

© 黄湘云

第二章　数据获取

数据获取包含两层意思，其一是数据收集，其二是数据搜集。数据收集，往往意味着自己做实验设计，执行实验，回收数据，掌握第一手资料。而数据搜集，往往意味着自己从各个地方搜罗数据，再清洗整理校验，得到可靠的二手或三手数据。从前，统计学家下到试验田，在不同 NPK（氮肥、磷肥和钾肥）配比的情况下，收集小麦的产量数据，以确定最佳配比。如今，许多互联网公司都有自己的 App，通过 App 收集大量用户及其行为数据，再以一定的数据模型加工整理成可用的二维表格。此外，许多政府和非政府的组织机构网站也发布大量的数据，比如各个国家的国家统计局和地方统计局，世界银行，国际货币基金组织等。这其中，有的以图片形式发布，有的以二维表格形式发布，有的以数据 API 服务形式发布，比起散落在各个公告中要好多了。

数据收集的方式有线下发放问卷、从网络爬取、网络调查问卷、线下市场调查、走访、有奖征集、埋点等。在真实的数据分析中，有时候需要借助 SQL 或浏览器开发者工具，从不同的数据源获取数据，清洗整理，再将数据导入 R 环境。

2.1　从本地文件读取

利用 Base R 提供的基础函数从各类文件导入数据

2.1.1　csv 文件

小的 csv 文件，可用 Base R 提供的 `read.csv()` 函数读取。大型 csv 文件，可用 **data.table** 的 `fread()` 函数读取。

2.1.2　xlsx 文件

readxl 读 xls 和 xlsx 文件，**writexl** 写 xlsx。

openxlsx 读/写 xlsx 文件

内部使用 ©

2.1.3 arrow 文件

Apache Arrow 的 R 语言接口 arrow 超出内存的大规模数据操作。比如在时空数据处理场景，数据文件往往比较大，需要在远程服务器上处理超出本地计算机内存的数据，geoarrow包和sfarrow包都是应对此类需求。

2.2 从数据库中导入

从各类数据库导入数据，比如 RSQLite 等

2.2.1 RSQLite

2.2.2 odbc

2.2.3 RJDBC

很多数据库都有 Java 接口驱动

2.3 从各类网页中抓取

rvest 包从网页、网站抓取数据，再用 xml2 和 httr2 解析处理网页数据。

2.3.1 豆瓣排行榜

2.3.2 链家二手房

2.4 从数据接口中获取

2.4.1 中国地震台网

中国地震台网 可以想象后台有一个数据库，在页面的小窗口中输入查询条件，转化为某种 SQL 语句，传递给数据库管理系统，执行查询语句，返回查询结果，即数据。

2.4.2 美国地质调查局

美国地质调查局提供一些选项窗口，可供选择数据范围，直接下载 CSV 或 XLS 文件。

© 基础

2.4.3 美国人口调查局

美国人口调查局

tidycensus 需要注册账号，获取使用 API 接口的访问令牌，可以想象后台不仅有一个数据库，在此之上，还有一层数据鉴权。

2.4.4 世界银行

世界银行和国际货币基金组织

wbstats 包封装世界银行提供的数据接口 REST API

© 黄湘云

第三章　数据清洗

从非结构的、半结构的数据中抽取有用的信息，常常需要一番数据清洗操作，最重要的工具之一是正则表达式。R 语言内置一系列函数，组成一套工具，详见 `?regex` 。

3.1　正则表达式

3.1.1　量词

3.1.2　级联

3.1.3　断言

正向查找 / 反向查找

3.1.4　反向引用

3.1.5　命名捕捉

3.2　字符串操作

3.2.1　查找

`grep()` / `grepl()` 返回是否匹配的结果

3.2.2　替换

`sub()` / `gsub()` 替换一次和多次

© 电子工业出版社

3.2.3 提取

`regexpr()` / `gregexpr()`

`regexec()` / `gregexec()`

© 王建国

第四章　数据操作

目前，R 语言在数据操作方面陆续出现三套工具，最早的是 Base R（1997 年 4 月），之后是 **data.table**（2006 年 4 月）和 **dplyr**（2014 年 1 月）。下面将从世界银行下载的原始数据开始，以各种数据操作及其组合串联起来介绍，完成数据探查的工作。

4.1　操作工具

本节所用数据来自世界银行，介绍 Base R、**data.table**、**dplyr** 的简介、特点、对比

4.1.1　Base R

在 data.frame 的基础上，提供一系列辅助函数实现各类数据操作。

```
aggregate(iris, Sepal.Length ~ Species, FUN = length)
```

```
     Species Sepal.Length
1     setosa           50
2 versicolor           50
3  virginica           50
```

4.1.2　data.table

data.table 包在 Base R 的基础上，扩展和加强了原有函数的功能，提供一套完整的链式操作语法。

```
library(data.table)
iris_dt <- as.data.table(iris)
iris_dt[ ,.(cnt = length(Sepal.Length)) , by = "Species"]
```

```
      Species cnt
1:     setosa  50
2: versicolor  50
3:  virginica  50
```

©黄湘云

4.1.3 dplyr

dplyr 包提供一套全新的数据操作语法，与 **purrr** 包和 **tidyr** 包一起形成完备的数据操作功能。在 R 环境下，**dplyr** 包提供一套等价的表示，代码如下：

```
iris |>
  dplyr::group_by(Species) |>
  dplyr::count()
```

```
# A tibble: 3 x 2
# Groups:   Species [3]
  Species        n
  <fct>      <int>
1 setosa        50
2 versicolor    50
3 virginica     50
```

4.1.4 SQL

实际工作中，SQL （结构化查询语言）是必不可少的基础性工具，比如 SQLite、Hive 和 Spark 等都提供基于 SQL 的数据查询引擎，没有重点介绍 SQL 操作是因为本书以 R 语言为数据分析的主要工具，而不是它不重要。以 **dplyr** 来说吧，它的诸多语义动词就是对标 SQL 的。

```
library(DBI)
conn <- DBI::dbConnect(RSQLite::SQLite(),
  dbname = system.file("db", "datasets.sqlite", package = "RSQLite")
)
```

按 Species 分组统计数据条数，SQL 查询语句如下：

```
SELECT COUNT(1) AS cnt, Species
FROM iris
GROUP BY Species;
```

SQL 代码执行的结果如下：

```
iris_preview
```

```
  cnt    Species
1  50     setosa
2  50 versicolor
3  50  virginica
```

dplyr 包能连接数据库，以上 SQL 代码也可以翻译成等价的 **dplyr** 语句。

©清华大学出版社

```
dplyr::tbl(conn, "iris") |>
  dplyr::group_by(Species) |>
  dplyr::count()
```

```
# Source:   SQL [3 x 2]
# Database: sqlite 3.45.0 [/Users/runner/work/_temp/Library/RSQLite/db/datasets.sqlite]
# Groups:   Species
  Species        n
  <chr>      <int>
1 setosa        50
2 versicolor    50
3 virginica     50
```

dplyr 包的函数 `show_query()` 可以将 **dplyr** 语句转化为查询语句，这有助于排错。

```
dplyr::tbl(conn, "iris") |>
  dplyr::group_by(Species) |>
  dplyr::count() |>
  dplyr::show_query()
```

```
<SQL>
SELECT `Species`, COUNT(*) AS `n`
FROM `iris`
GROUP BY `Species`
```

glue 包可以使用 R 环境中的变量，相比于 `sprintf()` 函数，可以组合更大型的 SQL 语句，这在生产环境中广泛使用。

```
# R 环境中的变量
group <- "Species"
# 组合 SQL
query <- glue::glue("
  SELECT COUNT(1) AS cnt, Species
  FROM iris
  GROUP BY ({group})
")
# 将 SQL 语句传递给数据库，执行 SQL 语句
DBI::dbGetQuery(conn, query)
```

```
  cnt    Species
1  50     setosa
2  50 versicolor
3  50  virginica
```

用完后，关闭连接通道。

©黄湘云


```
dbDisconnect(conn = conn)
```

更多关于 SQL 语句的使用介绍见书籍《Become a SELECT star》。

4.2 Base R 操作

介绍最核心的 Base R 数据操作，如筛选、排序、变换、聚合、重塑等

4.2.1 筛选

筛选操作可以用函数 subset() 或 [实现

```
subset(iris, subset = Species == "setosa" & Sepal.Length > 5.5, select = c("Sepal.Length", "Sepal.W
```

```
   Sepal.Length Sepal.Width
15          5.8         4.0
16          5.7         4.4
19          5.7         3.8
```

```
iris[iris$Species == "setosa" & iris$Sepal.Length > 5.5, c("Sepal.Length", "Sepal.Width")]
```

```
   Sepal.Length Sepal.Width
15          5.8         4.0
16          5.7         4.4
19          5.7         3.8
```

4.2.2 变换

变换操作可以用函数 within()/transform() 实现。最常见的变换操作是类型转化，比如从字符串型转为因子型、整型或日期型等。

```
# iris2 <- transform(iris, Species_N = as.integer(Species))[1:3, ]
iris2 <- within(iris, {
  Species_N <- as.integer(Species)
})
str(iris2)
```

```
'data.frame':   150 obs. of  6 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 ...
```

©黄海广

```
 $ Species_N   : int  1 1 1 1 1 1 1 1 1 1 ...
```

4.2.3 排序

排序操作可以用函数 order() 实现

```
iris[order(iris$Sepal.Length, decreasing = FALSE)[1:3], ]
```

```
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
14          4.3         3.0          1.1         0.1  setosa
9           4.4         2.9          1.4         0.2  setosa
39          4.4         3.0          1.3         0.2  setosa
```

4.2.4 聚合

聚合操作可以用函数 aggregate() 实现

```
aggregate(iris, Sepal.Length ~ Species, mean)
```

```
     Species Sepal.Length
1     setosa        5.006
2 versicolor        5.936
3  virginica        6.588
```

4.2.5 合并

两个数据框的合并操作可以用函数 merge() 实现

```
df1 <- data.frame(a1 = c(1, 2, 3), a2 = c("A", "B", "C"))
df2 <- data.frame(b1 = c(2, 3, 4), b2 = c("A", "B", "D"))
# LEFT JOIN
merge(x = df1, y = df2, by.x = "a2", by.y = "b2", all.x = TRUE)
```

```
  a2 a1 b1
1  A  1  2
2  B  2  3
3  C  3 NA
```

```
# RIGHT JOIN
merge(x = df1, y = df2, by.x = "a2", by.y = "b2", all.y = TRUE)
```

```
  a2 a1 b1
1  A  1  2
2  B  2  3
```

```
3  D NA  4
```

```r
# INNER JOIN
merge(x = df1, y = df2, by.x = "a2", by.y = "b2", all = FALSE)
```

```
  a2 a1 b1
1  A  1  2
2  B  2  3
```

```r
# FULL JOIN
merge(x = df1, y = df2, by.x = "a2", by.y = "b2", all = TRUE)
```

```
  a2 a1 b1
1  A  1  2
2  B  2  3
3  C  3 NA
4  D NA  4
```

4.2.6 重塑

将数据集从宽格式转为长格式，可以用函数 reshape() 实现，反之，亦然。

```r
# 长格式
df3 <- data.frame(
  extra = c(0.7, -1.6, -0.2, -1.2, -0.1, 3.4),
  group = c("A", "A", "A", "B", "B", "B"),
  id = c(1, 2, 3, 1, 2, 3)
)
# 长转宽
reshape(df3, direction = "wide", timevar = "group", idvar = "id")
```

```
  id extra.A extra.B
1  1     0.7    -1.2
2  2    -1.6    -0.1
3  3    -0.2     3.4
```

```r
# 也可以指定组合变量的列名
reshape(df3, direction = "wide", timevar = "group", idvar = "id",
        v.names = "extra", sep = "_")
```

```
  id extra_A extra_B
1  1     0.7    -1.2
2  2    -1.6    -0.1
3  3    -0.2     3.4
```

提取并整理分组线性回归系数。函数 `split()` 将数据集 iris 按分类变量 Species 拆分成列表，函数 `lapply()` 将线性回归操作 `lm()` 应用于列表的每一个元素上，再次用函数 `lapply()` 将函数 `coef()` 应用于线性回归后的列表上，提取回归系数，用函数 `do.call()` 将系数合并成矩阵，最后，用函数 `as.data.frame()` 转化成数据框。

```
s1 <- split(iris, ~Species)
s2 <- lapply(s1, lm, formula = Sepal.Length ~ Sepal.Width)
s3 <- lapply(s2, coef)
s4 <- do.call("rbind", s3)
s5 <- as.data.frame(s4)
s5
```

```
           (Intercept) Sepal.Width
setosa        2.639001   0.6904897
versicolor    3.539735   0.8650777
virginica     3.906836   0.9015345
```

```
do.call(
  "rbind",
  lapply(
    lapply(
      split(iris, ~Species), lm,
      formula = Sepal.Length ~ Sepal.Width
    ),
    coef
  )
)
```

```
           (Intercept) Sepal.Width
setosa        2.639001   0.6904897
versicolor    3.539735   0.8650777
virginica     3.906836   0.9015345
```

4.3 data.table 操作

掌握此等基础性的工具，再去了解新工具也不难，更重要的是，只要将一种工具掌握的足够好，也就足以应付绝大多数的情况。

1. 介绍 **data.table** 基础语法，对标 Base R，介绍基础操作，同时给出等价的 **dplyr** 实现，但不运行代码。
2. **data.table** 扩展 Base R 数据操作，介绍常用的操作 8 个，讲清楚出现的具体场景，同时给出等价的 dplyr 实现，但不运行代码。

©黄海广

3. **data.table** 特有的高级数据操作 on、.SD 、.I 、.J 等。

4.3.1 筛选

data.table 扩展了函数 [功能，简化 iris$Species == "setosa" 代码 Species == "setosa"

```
iris_dt[Species == "setosa" & Sepal.Length > 5.5, c("Sepal.Length", "Sepal.Width")]
```

```
   Sepal.Length Sepal.Width
1:          5.8         4.0
2:          5.7         4.4
3:          5.7         3.8
```

4.3.2 变换

变换操作可以用函数 :=

```
iris_dt[, Species_N := as.integer(Species)]
str(iris_dt)
```

```
Classes 'data.table' and 'data.frame':  150 obs. of  6 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 ...
 $ Species_N   : int  1 1 1 1 1 1 1 1 1 1 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

4.3.3 排序

排序操作可以用函数 order()

```
iris_dt[order(Sepal.Length, decreasing = FALSE)[1:3], ]
```

```
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species Species_N
1:          4.3         3.0          1.1         0.1  setosa         1
2:          4.4         2.9          1.4         0.2  setosa         1
3:          4.4         3.0          1.3         0.2  setosa         1
```

4.3.4 聚合

聚合操作用函数 .() 和 by 组合

```
iris_dt[, .(mean = mean(Sepal.Length)), by = "Species"]
```

```
      Species  mean
1:     setosa 5.006
2: versicolor 5.936
3:  virginica 6.588
```

4.3.5 合并

合并操作也是用函数 merge() 来实现。

```
dt1 <- data.table(a1 = c(1, 2, 3), a2 = c("A", "B", "C"))
dt2 <- data.table(b1 = c(2, 3, 4), b2 = c("A", "B", "D"))
# LEFT JOIN
merge(x = dt1, y = dt2, by.x = "a2", by.y = "b2", all.x = TRUE)
```

```
   a2 a1 b1
1:  A  1  2
2:  B  2  3
3:  C  3 NA
```

```
# RIGHT JOIN
merge(x = dt1, y = dt2, by.x = "a2", by.y = "b2", all.y = TRUE)
```

```
   a2 a1 b1
1:  A  1  2
2:  B  2  3
3:  D NA  4
```

```
# INNER JOIN
merge(x = dt1, y = dt2, by.x = "a2", by.y = "b2", all = FALSE)
```

```
   a2 a1 b1
1:  A  1  2
2:  B  2  3
```

```
# FULL JOIN
merge(x = dt1, y = dt2, by.x = "a2", by.y = "b2", all = TRUE)
```

```
   a2 a1 b1
1:  A  1  2
2:  B  2  3
3:  C  3 NA
4:  D NA  4
```

© 统计学习

4.3.6 重塑

将数据集从宽格式转为长格式，可以用函数 `dcast()` 实现，反之，可以用函数 `melt()` 实现。

```
# 长格式
dt3 <- data.table(
  extra = c(0.7, -1.6, -0.2, -1.2, -0.1, 3.4),
  group = c("A", "A", "A", "B", "B", "B"),
  id = c(1, 2, 3, 1, 2, 3)
)
# 长转宽
dcast(dt3, id ~ group, value.var = "extra")
```

```
   id    A    B
1:  1  0.7 -1.2
2:  2 -1.6 -0.1
3:  3 -0.2  3.4
```

类似 Base R，也用 **data.table** 来实现 iris 分组线性回归

```
iris_dt[, as.list(coef(lm(Sepal.Length ~ Sepal.Width))), by = "Species"]
```

```
      Species (Intercept) Sepal.Width
1:     setosa    2.639001   0.6904897
2: versicolor    3.539735   0.8650777
3:  virginica    3.906836   0.9015345
```

© 吉林大学

第五章　数据处理

5.1　缺失值处理

缺失是一种非常常见的数据问题。

5.1.1　查找

缺失值在数据框中的位置

5.1.2　汇总

缺失值的占比、分布情况，可视化获得缺失的结构 VIM

5.1.3　替换

替换数据框中的缺失值

5.1.4　插补

mice Multivariate Imputation by Chained Equations 缺失值插补

5.2　异常值处理

提及异常，一般会联想到数据本身出问题了，比如数据错误。比较常见的情况是业务有异动，导致数据异常波动，需要及时捕捉到这种异常波动，找到异常的原因，进而采取措施。

© 黄湘云

5.2.1 检测

5.2.2 识别

5.2.3 处理

5.3 离群值处理

离群，并不是数据本身出问题，而是数据隐藏着特殊信息，与平时不一样的情况，与大家伙不一样的情况。比如情人节鲜花和蛋糕的需求量激增，端午节粽子的需求激增，这和平时很不一样。需求数据本身没有问题，如实反应了现实情况。因此，需要根据现实情况，调整预测模型，做出更加准确的需求预测，提前安排供给。

5.3.1 检测

5.3.2 识别

5.3.3 处理

© 黄湘云

第二部分

数据探索

©黄湘云

第六章　ggplot2 入门

2006 年 Hans Rosling（汉斯 · 罗琳）在 TED 做了一场精彩的演讲 — The best stats you've ever seen。演讲中展示了一系列生动形象的动画，用数据记录的事实帮助大家理解世界的变化，可谓是动态图形领域的惊世之作。时至今日，已经超过 1500 万人观看，产生了十分广泛的影响。下面从数据源头 — 世界银行获取数据，整理后取名 `gapminder`。本节将基于 `gapminder` 数据集介绍 **ggplot2** 绘图的基础知识，包括图层、标签、刻度、配色、图例、主题、文本、分面、字体、动画和组合等 11 个方面，理解这些有助于绘制和加工各种各样的统计图形，可以覆盖日常所需。gapminder 数据集以数据框的形式存储在 R 软件运行环境中，一共 4950 行，7 列。篇幅所限，下表格 6.1 展示该数据集的部分内容，表中人均 GDP 和预期寿命两列四舍五入保留一位小数。

表格 6.1: gapminder 数据集（部分）

年份	国家或地区	区域划分	收入水平	人均 GDP	预期寿命	人口总数
1991	阿鲁巴	拉丁美洲与加勒比海地区	高收入	13494.7	73.5	64623
1992	阿鲁巴	拉丁美洲与加勒比海地区	高收入	14048.3	73.5	68240
1993	阿鲁巴	拉丁美洲与加勒比海地区	高收入	14942.3	73.6	72495
1994	阿鲁巴	拉丁美洲与加勒比海地区	高收入	16241.6	73.6	76705
1995	阿鲁巴	拉丁美洲与加勒比海地区	高收入	16441.8	73.6	80324
1996	阿鲁巴	拉丁美洲与加勒比海地区	高收入	16583.0	73.6	83211

在 R 环境中，加载 gapminder 数据集后，可以用 `str()` 函数查看数据集 gapminder 各个列的数据类型和部分属性值。

```
# 查看数据
str(gapminder)
```

```
#> 'data.frame':    4950 obs. of  7 variables:
#>  $ year        : num  1991 1992 1993 1994 1995 ...
#>  $ country     : chr  "阿鲁巴" "阿鲁巴" "阿鲁巴" "阿鲁巴" ...
#> $ region    : Factor w/ 7 levels "北美","拉丁美洲与加勒比海地区",..: 2 2 2 2 2 2 2 2 2 2 ...
#> $ income_level: Ord.factor w/ 4 levels "低收入"<"中低等收入"<..: 4 4 4 4 4 4 4 4 4 4 ...
#>  $ gdpPercap   : num  13495 14048 14942 16242 16442 ...
#>  $ lifeExp     : num  73.5 73.5 73.6 73.6 73.6 ...
```

```
#>  $ pop         : num  64623 68240 72495 76705 80324 ...
```

其中，country（国家或地区）是字符型变量，region （区域）是因子型变量，income_level（收入水平）是有序的因子型变量，year （年份）、pop （人口总数）、lifeExp （出生时的预期寿命，单位：岁）和 gdpPercap （人均 GDP，单位：美元）是数值型变量。

6.1 图层

ggplot2 绘图必须包含以下三个要素，缺少任何一个，图形都是不完整的。

1. 数据，前面已经重点介绍和准备了；
2. 映射，数据中的变量与几何元素的对应关系；
3. 图层，至少需要一个图层用来渲染观察值。

下面逐一说明三个要素的作用，为简单起见，从数据集 gapminder 中选取 2007 年的数据。

```
library(ggplot2)
gapminder_2007 <- gapminder[gapminder$year == 2007, ]
ggplot(data = gapminder_2007)
ggplot(data = gapminder_2007, aes(x = gdpPercap, y = lifeExp))
ggplot(data = gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
  geom_point()
ggplot(data = gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(aes(size = pop))
```

图 6.1a 仅提供数据，只渲染出来一个绘图区域。图 6.1b 仅提供数据和映射，将变量 gdpPercap 映射给横轴，变量 lifeExp 映射给纵轴，继续渲染出来横、纵坐标轴及标签。图 6.1c 提供了数据、映射和图层三要素，观察值根据几何图层 `geom_point()` 将几何元素「点」渲染在绘图区域上，形成散点图。函数 `ggplot()` 和函数 `geom_point()` 之间是以加号 `+` 连接的。无论最终产出的图形如何复杂，这个模式贯穿 ggplot2 绘图。

10 多年来，**ggplot2** 包陆续添加了很多几何图层，目前支持的有 53 个，如下：

表格 6.2: **ggplot2** 包可以绘制丰富的统计图形

geom_abline	geom_dotplot	geom_qq_line
geom_area	geom_errorbar	geom_quantile
geom_bar	geom_errorbarh	geom_raster
geom_bin_2d	geom_freqpoly	geom_rect
geom_bin2d	geom_function	geom_ribbon
geom_blank	geom_hex	geom_rug
geom_boxplot	geom_histogram	geom_segment
geom_col	geom_hline	geom_sf

© 基础统计

geom_contour	geom_jitter	geom_sf_label
geom_contour_filled	geom_label	geom_sf_text
geom_count	geom_line	geom_smooth
geom_crossbar	geom_linerange	geom_spoke
geom_curve	geom_map	geom_step
geom_density	geom_path	geom_text
geom_density_2d	geom_point	geom_tile
geom_density_2d_filled	geom_pointrange	geom_violin
geom_density2d	geom_polygon	geom_vline
geom_density2d_filled	geom_qq	

也正因这些丰富多彩的图层，ggplot2 可以非常便捷地做各种数据探索和展示工作。从时间序列数据、网络社交数据到文本数据、空间数据，乃至时空数据都有它大显身手的地方。

6.2 标签

用函数 `labs()` 可以添加横轴、纵轴、图例的标题，整个图片的标题和副标题等。下图图 6.2a 是默认设置下显示的标签内容，而图 6.2b 是用户指定标签内容后的显示效果。

```
ggplot(data = gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(aes(color = region))
ggplot(data = gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(aes(color = region)) +
  labs(x = "人均 GDP", y = "预期寿命", tag = "标签",
       title = "这里是标题", caption = "这是图形说明",
       subtitle = "这里是副标题", color = "图例标题")
```

6.3 刻度

有时候图 6.1c 看起来不太好，收入低的国家太多，聚集在一起，重叠覆盖比较严重。而高收入国家相对较少，分布稀疏，距离低收入比较远，数据整体的分布很不平衡。此时，可以考虑对横轴标度做一些变换，常用的有以 10 为底的对数变换，如图 6.3 。

```
library(scales)
ggplot(data = gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
  geom_point() +
  scale_x_log10() +
  labs(x = "人均 GDP", y = "预期寿命")
```

© 黄湘云

(a) 只有数据

(b) 只有数据和坐标映射

(c) 数据、坐标映射和点图层

(d) 数据、坐标映射、点图层和视觉映射（可选）

图 6.1: ggplot2 绘图三要素

© 黄湘云

(a) 默认设置

(b) 自定义标签

图 6.2: 添加标签

© 黄湘云

图 6.3: 人均 GDP 做对数变换

为了更加醒目地展示横轴做了对数变换，需要添加对应的刻度标签。**scales** 包 (H. Wickham 和 Seidel 2022) 提供很多刻度标签支持，比如函数 `label_log()` 默认提供以 10 为底的刻度标签，如图 6.4 。

```
ggplot(data = gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
  geom_point() +
  scale_x_log10(labels = label_log()) +
  labs(x = "人均 GDP", y = "预期寿命")
```

图 6.4: 刻度标签随数据变换调整

这其实还不够，有的刻度标签含义不够显然，且看图 6.4 的横轴第一个刻度标签 $10^{2.48}$ 是用来替换图 6.3 的横轴第一个刻度标签 300。10 的 2.48 次方可不容易看出是 300 的意思，实际上它等于 302。因此，结

© 黄湘云

合人均 GDP 的实际范围，有必要适当调整横轴显示范围，这可以在函数 `scale_x_log10()` 中设置参数 `limits`，横轴刻度标签会随之适当调整，调整后的效果如图 6.5 。

```
ggplot(data = gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
  geom_point() +
  scale_x_log10(labels = label_log(), limits = c(100, 110000)) +
  labs(x = "人均 GDP", y = "预期寿命")
```

图 6.5: 设置数据展示范围

根据横轴所代表的人均 GDP （单位：美元）的实际含义，其实，可以进一步，添加更多的信息，即刻度标签带上数量单位，此处是美元符号。**scales** 包提供的函数 `label_dollar()` 可以实现，效果如图 6.6 。

```
ggplot(data = gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
  geom_point() +
  scale_x_log10(labels = label_dollar(), limits = c(100, 110000)) +
  labs(x = "人均 GDP", y = "预期寿命")
```

©黄湘云

图 6.6: 设置数据展示范围

最后，有必要添加次刻度线作为辅助参考线。图中点与点之间的横向距离代表人均 GDP 差距，以 10 为底的对数变换不是线性变化的，肉眼识别起来有点困难。从 100 美元到 100000 美元，在 100 美元、1000 美元、10000 美元和 100000 美元之间均添加 10 条次刻度线，每个区间内相邻的两条次刻度线之差保持恒定。下面构造刻度线的位置，了解原值和对数变换后的对应关系。

```
# 刻度线位置
mb <- unique(as.numeric(1:10 %o% 10^(1:4)))
# 对数变换后
log10(mb)
```

```
#>  [1] 1.000000 1.301030 1.477121 1.602060 1.698970 1.778151 1.845098 1.903090
#>  [9] 1.954243 2.000000 2.301030 2.477121 2.602060 2.698970 2.778151 2.845098
#> [17] 2.903090 2.954243 3.000000 3.301030 3.477121 3.602060 3.698970 3.778151
#> [25] 3.845098 3.903090 3.954243 4.000000 4.301030 4.477121 4.602060 4.698970
#> [33] 4.778151 4.845098 4.903090 4.954243 5.000000
```

```
# 刻度线位置
format(mb, big.mark = ",", scientific = 999)
```

```
#>  [1] "     10" "     20" "     30" "     40" "     50" "     60" "     70"
#>  [8] "     80" "     90" "    100" "    200" "    300" "    400" "    500"
#> [15] "    600" "    700" "    800" "    900" "  1,000" "  2,000" "  3,000"
#> [22] "  4,000" "  5,000" "  6,000" "  7,000" "  8,000" "  9,000" " 10,000"
#> [29] " 20,000" " 30,000" " 40,000" " 50,000" " 60,000" " 70,000" " 80,000"
#> [36] " 90,000" "100,000"
```

函数 `scale_x_log10()` 提供参数 `minor_breaks` 设定刻度线的位置。最终效果如图 6.7 。

©黄湘云


```
ggplot(data = gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
  geom_point() +
  scale_x_log10(
    labels = label_dollar(), minor_breaks = mb, limits = c(100, 110000)
  ) +
  labs(x = "人均 GDP", y = "预期寿命")
```

图 6.7: 添加次刻度线，提供更多参考

6.4 配色

好的配色可以让图形产生眼前一亮的效果，R 语言社区在统计图形领域深耕 20 多年，陆续涌现很多专门调色的 R 包，常见的有：

- **RColorBrewer** (Neuwirth 2022) (https://github.com/axismaps/colorbrewer/)
- **munsell** (C. Wickham 2018) (https://github.com/cwickham/munsell/)
- **colorspace** (Zeileis 等 2020) (https://colorspace.r-forge.r-project.org/)
- **paletteer** (Hvitfeldt 2021) (https://github.com/EmilHvitfeldt/paletteer)
- **scico** (Pedersen 和 Crameri 2022) (https://github.com/thomasp85/scico)
- **viridis** (Garnier 等 2021) (https://github.com/sjmgarnier/viridis/)
- **viridisLite** (Garnier 等 2021) (https://github.com/sjmgarnier/viridisLite/)
- **colormap** (Karambelkar 2016) (https://github.com/bhaskarvk/colormap)

ggplot2 提供多种方式给图形配色，最常见的要数函数 `scale_color_brewer()`，它调用 RColorBrewer 包制作离散型的调色板，根据离散型变量的具体情况，可分为发散型 qualitative、对撞型 Diverging、有序型 Sequential。在图图 6.7 的基础上，将分类型的区域变量映射给散点的颜色，即得到图 6.8 。

```
ggplot(data = gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
```

© 黄湘云


```
  geom_point(aes(color = region)) +
  scale_color_brewer(palette = "Set1") +
  scale_x_log10(
    labels = label_dollar(), minor_breaks = mb, limits = c(100, 110000)
  ) +
  labs(x = "人均 GDP", y = "预期寿命", color = "区域")
```

图 6.8: 使用 RColorBrewer 包提供的 Set1 调色板

另一种方式是调用函数 `scale_color_manual()`，需要用户给分类变量值逐个指定颜色，即提供一个命名的向量，效果如图 6.9 。

```
ggplot(data = gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(aes(color = region)) +
  scale_color_manual(values = c(
    `拉丁美洲与加勒比海地区` = "#E41A1C", `撒哈拉以南非洲地区` = "#377EB8",
    `欧洲与中亚地区` = "#4DAF4A", `中东与北非地区` = "#984EA3",
    `东亚与太平洋地区` = "#FF7F00", `南亚` = "#FFFF33", `北美` = "#A65628"
  )) +
  scale_x_log10(
    labels = label_dollar(), minor_breaks = mb, limits = c(100, 110000)
  ) +
  labs(x = "人均 GDP", y = "预期寿命", color = "区域")
```

© 黄湘云

图 6.9: 手动挨个指定分类变量的颜色

6.5 图例

在图 6.8 的基础上，继续将每个国家的人口总数映射给点的大小，绘制气泡图。此时有两个视觉映射变量 — 离散型的变量 country（国家）和连续型的变量 pop（人口总数）。不仅仅是图层函数 `geom_point()`，所有的几何图层都提供参数 `show.legend` 来控制图例的显示或隐藏。传递命名逻辑向量还可以在多个图例中选择性保留。图 6.10 在两个图例中保留一个，即人口总数。

```
ggplot(data = gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(aes(color = region, size = pop),
    show.legend = c(color = FALSE, size = TRUE)
  ) +
  scale_color_manual(values = c(
    `拉丁美洲与加勒比海地区` = "#E41A1C", `撒哈拉以南非洲地区` = "#377EB8",
    `欧洲与中亚地区` = "#4DAF4A", `中东与北非地区` = "#984EA3",
    `东亚与太平洋地区` = "#FF7F00", `南亚` = "#FFFF33", `北美` = "#A65628"
  )) +
  scale_size(range = c(2, 12)) +
  scale_x_log10(
    labels = label_dollar(), minor_breaks = mb, limits = c(100, 110000)
  ) +
  labs(x = "人均 GDP", y = "预期寿命", size = "人口总数")
```

© 黄湘云

图 6.10: 在两个图例中保留一个

全世界各个国家的人口总数从百万级横跨到十亿级，根据此实际情况，适当调整图例刻度标签是很有必要的，可以让图例内容更具可读性。图 6.11 是修改图例刻度标签后的效果，其中 M 表示 Million（百万），B 表示 Billion （十亿）。

```
ggplot(data = gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(aes(color = region, size = pop),
    show.legend = c(color = FALSE, size = TRUE)
  ) +
  scale_color_manual(values = c(
    `拉丁美洲与加勒比海地区` = "#E41A1C", `撒哈拉以南非洲地区` = "#377EB8",
    `欧洲与中亚地区` = "#4DAF4A", `中东与北非地区` = "#984EA3",
    `东亚与太平洋地区` = "#FF7F00", `南亚` = "#FFFF33", `北美` = "#A65628"
  )) +
  scale_size(range = c(2, 12), labels = label_number(scale_cut = cut_short_scale())) +
  scale_x_log10(
    labels = label_dollar(), minor_breaks = mb, limits = c(100, 110000)
  ) +
  labs(x = "人均 GDP", y = "预期寿命", size = "人口总数")
```

© 黄湘云

图 6.11: 修改图例刻度标签

6.6 主题

主题就是一系列风格样式的集合，提前设定标题、文本、坐标轴、图例等元素的默认参数，供后续调用。10 年来，R 语言社区陆续出现很多主题包。

- **ggthemes** (Arnold 2021) 收集了网站(如 Fivethirtyeight)、杂志(如《经济学家》)、软件(如 Stata)等的配色主题，打包成可供 **ggplot2** 绘图的主题，更多内容见 (https://github.com/jrnold/ggthemes)
- **ggsci** (Xiao 2018) 包收集了多份期刊杂志的图形配色，将其融入 **ggplot2** 绘图主题中，更多内容见 (https://github.com/road2stat/ggsci)。
- **ggpubr** (Kassambara 2022) 包在 **ggplot2** 之上封装一套更加易用的函数，可以快速绘制出版级的统计图形 (https://github.com/kassambara/ggpubr)。
- **ggcharts** (Neitmann 2020) 包类似 **ggpubr** 包，也提供一套更加快捷的函数接口，缩短数据可视化的想法与实际图形的距离，更多内容见 (https://github.com/thomas-neitmann/ggcharts)。
- **ggthemr** (C. Tobin 2020) 是比较早的 **ggplot2** 主题包，上游依赖少，更多内容见 (https://github.com/Mikata-Project/ggthemr)。
- **ggtech** (Bion 2018) 包收集了许多科技公司的设计风格，将其制作成可供 ggplot2 绘图使用的主题，更多内容见 (https://github.com/ricardo-bion/ggtech)。
- **bbplot** (Stylianou 等 2022) 为 BBC 新闻定制的一套主题，更多内容见 (https://github.com/bbc/bbplot)。
- **pilot** (Hawkins 2022) 包提供一套简洁的 **ggplot2** 主题，特别是适合展示分类、离散型数据，更多内容见 (https://github.com/olihawkins/pilot)。
- **ggthemeassist** (Gross 和 Ottolinger 2016) 包提供 RStudio IDE 插件，帮助用户以鼠标点击的交互

© 草稿

方式设置 **ggplot2** 图形的主题样式，更多内容见 (https://github.com/calligross/ggthemeassist)。

在图 6.11 的基础上，以 **ggplot2** 包内置的主题 `theme_classic()` 替换默认的主题，效果如下图 6.12 ，这是一套非常经典的主题，它去掉所有的背景色和参考系，显得非常简洁。

```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(
    data = function(x) subset(x, year == 2007),
    aes(fill = region, size = pop), shape = 21, col = "white",
    show.legend = c(fill = TRUE, size = FALSE)
  ) +
  scale_fill_manual(values = c(
    `拉丁美洲与加勒比海地区` = "#E41A1C", `撒哈拉以南非洲地区` = "#377EB8",
    `欧洲与中亚地区` = "#4DAF4A", `中东与北非地区` = "#984EA3",
    `东亚与太平洋地区` = "#FF7F00", `南亚` = "#FFFF33", `北美` = "#A65628"
  )) +
  scale_size(range = c(2, 12)) +
  scale_x_log10(
    labels = label_dollar(), minor_breaks = mb, limits = c(100, 110000)
  ) +
  theme_classic() +
  labs(x = "人均 GDP", y = "预期寿命", fill = "区域")
```

©黄湘云

图 6.12: ggplot2 内置的经典主题风格

在已有主题的基础上，还可以进一步细微调整，比如，将图例移动至绘图区域的下方，见图 6.13 。

```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(
    data = function(x) subset(x, year == 2007),
    aes(fill = region, size = pop), shape = 21, col = "white",
    show.legend = c(fill = TRUE, size = FALSE)
  ) +
  scale_fill_manual(values = c(
    `拉丁美洲与加勒比海地区` = "#E41A1C", `撒哈拉以南非洲地区` = "#377EB8",
    `欧洲与中亚地区` = "#4DAF4A", `中东与北非地区` = "#984EA3",
    `东亚与太平洋地区` = "#FF7F00", `南亚` = "#FFFF33", `北美` = "#A65628"
  )) +
  scale_size(range = c(2, 12)) +
  scale_x_log10(
    labels = label_dollar(), minor_breaks = mb, limits = c(100, 110000)
  ) +
  theme_classic() +
```

©黄湘云

```
  theme(legend.position = "bottom") +
  labs(x = "人均 GDP", y = "预期寿命", fill = "区域")
```

图 6.13: 图例置于图形下方

或者用户觉得合适的任意位置。

```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(
    data = function(x) subset(x, year == 2007),
    aes(fill = region, size = pop), shape = 21, col = "white",
    show.legend = c(fill = TRUE, size = FALSE)
  ) +
  scale_fill_manual(values = c(
    `拉丁美洲与加勒比海地区` = "#E41A1C", `撒哈拉以南非洲地区` = "#377EB8",
    `欧洲与中亚地区` = "#4DAF4A", `中东与北非地区` = "#984EA3",
    `东亚与太平洋地区` = "#FF7F00", `南亚` = "#FFFF33", `北美` = "#A65628"
  )) +
  scale_size(range = c(2, 12)) +
  scale_x_log10(
    labels = label_dollar(), minor_breaks = mb, limits = c(100, 110000)
```

©黄湘云


```
  ) +
  theme_classic() +
  theme(legend.position = "inside", legend.position.inside = c(0.875, 0.3)) +
  labs(x = "人均 GDP", y = "预期寿命", fill = "区域")
```

图 6.14: 微调图例位置

或者更换其它主题，比如 **ggthemes** 包内置极简主题 `theme_tufte()`，它仅保留主刻度线，更加凸显数据。

```
library(ggthemes)
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(
    data = function(x) subset(x, year == 2007),
    aes(fill = region, size = pop),
    show.legend = c(fill = TRUE, size = FALSE),
    shape = 21, col = "white"
  ) +
  scale_fill_manual(values = c(
    `拉丁美洲与加勒比海地区` = "#E41A1C", `撒哈拉以南非洲地区` = "#377EB8",
    `欧洲与中亚地区` = "#4DAF4A", `中东与北非地区` = "#984EA3",
```

© 黄湘云

```
  `东亚与太平洋地区` = "#FF7F00", `南亚` = "#FFFF33", `北美` = "#A65628"
)) +
scale_size(range = c(2, 12)) +
scale_x_log10(
  labels = label_dollar(), minor_breaks = mb, limits = c(100, 110000)
) +
theme_tufte(base_family = "sans") +
theme(
  legend.position = "inside",
  legend.position.inside =  c(0.875, 0.3),
  legend.title = element_text(family = "Noto Sans CJK SC"),
  legend.text = element_text(family = "Noto Sans CJK SC"),
  axis.title = element_text(family = "Noto Sans CJK SC")) +
labs(x = "人均 GDP", y = "预期寿命", fill = "区域")
```

图 6.15: ggthemes 的极简主题 Tufte

©黄湘云

6.7 注释

注释可以是普通文本，数学公式，还可以是图形照片、表情包。注释功能非常强大，但也是非常灵活，往往使用起来颇费功夫，需要结合数据情况，从图形所要传递的信息出发，适当添加。R 语言社区陆续出现一些扩展包，让用户使用起来更方便些。

- **ggrepel** (Slowikowski 2021) 包可以通过添加一定距离的扰动，可以缓解文本重叠的问题，更多内容见 (https://github.com/slowkow/ggrepel)。
- **ggtext** (Wilke 2020) 包支持以 Markdown 语法添加丰富的文本内容，更多内容见 (https://github.com/wilkelab/ggtext)。
- **string2path** (Yutani 2022) 包字体轮廓生成路径，注释文本随路径变化，更多内容见 (https://github.com/yutannihilation/string2path)。
- **ggimage** (Yu 2022) 包提供图像图层，实现以图片代替散点的效果，图片还可以是表情包，更多内容见 (https://github.com/GuangchuangYu/ggimage)。

在图 6.15 的基础上，给人口总数大于 2 亿的国家添加文本注释。这可以用 **ggplot2** 包提供的文本图层函数 `geom_text()` 实现，效果如图 6.16 。

```
library(ggrepel)
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(
    data = function(x) subset(x, year == 2007),
    aes(fill = region, size = pop),
    show.legend = c(fill = TRUE, size = FALSE),
    shape = 21, col = "white"
  ) +
  scale_fill_manual(values = c(
    `拉丁美洲与加勒比海地区` = "#E41A1C", `撒哈拉以南非洲地区` = "#377EB8",
    `欧洲与中亚地区` = "#4DAF4A", `中东与北非地区` = "#984EA3",
    `东亚与太平洋地区` = "#FF7F00", `南亚` = "#FFFF33", `北美` = "#A65628"
  )) +
  scale_x_log10(
    labels = label_dollar(), minor_breaks = mb, limits = c(100, 110000)
  ) +
  geom_text(
    data = function(x) subset(x, year == 2007 & pop >= 20 * 10^7),
    aes(label = country), show.legend = FALSE
  ) +
  scale_size(range = c(2, 12)) +
  theme_tufte(base_family = "sans") +
  theme(
```

©黄湘云

```
    legend.position = "inside",
    legend.position.inside =  c(0.9, 0.3),
    legend.title = element_text(family = "Noto Sans CJK SC"),
    legend.text = element_text(family = "Noto Sans CJK SC"),
    axis.title = element_text(family = "Noto Sans CJK SC")) +
  labs(x = "人均 GDP", y = "预期寿命", fill = "区域")
```

图 6.16: 添加文本注释

当需要给许多点添加文本注释时，就难以避免地遇到注释文本重叠的问题。比如给人口总数大于 5000 万的国家添加文本注释，此时，适合使用 **ggrepel** 包，调用函数 `geom_text_repel()` — 这是一个新的文本图层，通过添加适当的位移缓解文本重叠问题。

```
library(ggrepel)
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(data = function(x) subset(x, year == 2007),
             aes(fill = region, size = pop),
    show.legend = c(fill = TRUE, size = FALSE),
    shape = 21, col = "white"
  ) +
  scale_fill_manual(values = c(
```

© 统计出版社


```
    `拉丁美洲与加勒比海地区` = "#E41A1C", `撒哈拉以南非洲地区` = "#377EB8",
    `欧洲与中亚地区` = "#4DAF4A", `中东与北非地区` = "#984EA3",
    `东亚与太平洋地区` = "#FF7F00", `南亚` = "#FFFF33", `北美` = "#A65628"
  )) +
  scale_x_log10(
    labels = label_dollar(), minor_breaks = mb, limits = c(100, 110000)
  ) +
  geom_text_repel(
    data = function(x) subset(x, year == 2007 & pop >= 5 * 10^7),
    aes(label = country), size = 3, max.overlaps = 50,
    segment.colour = "gray", seed = 2022, show.legend = FALSE
  ) +
  scale_size(range = c(2, 12)) +
  theme_tufte(base_family = "sans") +
  theme(
    legend.position = "inside",
    legend.position.inside =  c(0.9, 0.3),
    legend.title = element_text(family = "Noto Sans CJK SC"),
    legend.text = element_text(family = "Noto Sans CJK SC"),
    axis.title = element_text(family = "Noto Sans CJK SC")) +
  labs(x = "人均 GDP", y = "预期寿命", fill = "区域")
```

©黄湘云

图 6.17: 缓解文本注释相互覆盖的问题

6.8 分面

ggplot2 包有两个函数 `facet_wrap()` 和 `facet_grid()` 都可以用来实现分面操作，分面的目的是将数据切分，一块一块地展示。下面在图 6.15 的基础上，按收入水平变量分面，即将各个国家或地区按收入水平分开，效果如图 6.18 所示。`facet_grid()` 与 `facet_wrap()` 的效果是类似的，就不再赘述了。

```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(data = function(x) subset(x, year == 2007),
             aes(fill = region, size = pop),
    show.legend = c(fill = TRUE, size = FALSE),
    shape = 21, col = "white"
  ) +
  scale_fill_manual(values = c(
    `拉丁美洲与加勒比海地区` = "#E41A1C", `撒哈拉以南非洲地区` = "#377EB8",
    `欧洲与中亚地区` = "#4DAF4A", `中东与北非地区` = "#984EA3",
    `东亚与太平洋地区` = "#FF7F00", `南亚` = "#FFFF33", `北美` = "#A65628"
  )) +
  scale_size(range = c(2, 12)) +
```

©黄湘云


```
  scale_x_log10(labels = label_log(), limits = c(100, 110000)) +
  facet_wrap(facets = ~income_level, ncol = 2) +
  theme_classic() +
  labs(x = "人均 GDP", y = "预期寿命", fill = "区域")
```

图 6.18: 按收入水平变量分面

在函数 `facet_wrap()` 内设置不同的参数值，会有不同的排列效果。设置 `ncol = 3`，意味着排成 3 列，而分类变量 continent 总共有 5 种不同的类别，因此将会是 3 列 2 行的布局，效果如下图 6.19 。

```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(data = function(x) subset(x, year == 2007),
             aes(fill = region, size = pop),
    show.legend = c(fill = TRUE, size = FALSE),
    shape = 21, col = "white"
  ) +
  scale_fill_manual(values = c(
    `拉丁美洲与加勒比海地区` = "#E41A1C", `撒哈拉以南非洲地区` = "#377EB8",
    `欧洲与中亚地区` = "#4DAF4A", `中东与北非地区` = "#984EA3",
```

© 王建平

```
  `东亚与太平洋地区` = "#FF7F00", `南亚` = "#FFFF33", `北美` = "#A65628"
)) +
scale_size(range = c(2, 12)) +
scale_x_log10(labels = label_log(), limits = c(100, 110000)) +
facet_wrap(facets = ~income_level, ncol = 3) +
theme_classic() +
theme(legend.position = "inside", legend.position.inside = c(0.9, 0.2)) +
labs(x = "人均 GDP", y = "预期寿命", fill = "区域")
```

图 6.19: 按区域变量分面

© 基础出版社

6.9 动画

从 1991 年至 2020 年，gapminder 数据集一共是 30 年的数据。根据 2007 年的数据绘制了图 6.20 ，每年的数据绘制一幅图像，30 年总共可获得 30 帧图像，再以每秒播放 6 帧图像的速度将 30 帧图像合成 GIF 动画。因此，设置这个动画总共 30 帧，每秒播放的图像数为 6。

```
options(gganimate.nframes = 30, gganimate.fps = 6)
```

gganimate 包提供一套代码风格类似 **ggplot2** 包的动态图形语法，可以非常顺滑地与之连接。在了解了 ggplot2 绘制图形的过程后，用 **gganimate** 包制作动画是非常容易的。**gganimate** 包会调用 **gifski** (https://github.com/r-rust/gifski) 包来合成动画，因此，除了安装 **gganimate** 包，还需要安装 **gifski** 包。接着，在已有的 **ggplot2** 绘图代码基础上，再追加一个转场图层函数 `transition_time()`，这里是按年逐帧展示图像，因此，其转场的时间变量为 gapminder 数据集中的变量 `year`。

```
library(gganimate)
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(aes(fill = region, size = pop),
    show.legend = c(fill = TRUE, size = FALSE),
    alpha = 0.65, shape = 21, col = "white"
  ) +
  scale_fill_manual(values = c(
    `拉丁美洲与加勒比海地区` = "#E41A1C", `撒哈拉以南非洲地区` = "#377EB8",
    `欧洲与中亚地区` = "#4DAF4A", `中东与北非地区` = "#984EA3",
    `东亚与太平洋地区` = "#FF7F00", `南亚` = "#FFFF33", `北美` = "#A65628"
  )) +
  scale_size(range = c(2, 12), labels = label_number(scale_cut = cut_short_scale())) +
  scale_x_log10(labels = label_log(), limits = c(10, 130000)) +
  facet_wrap(facets = ~income_level) +
  theme_classic() +
  labs(
    title = "{frame_time} 年", x = "人均 GDP",
    y = "预期寿命", size = "人口总数", fill = "区域"
  ) +
  transition_time(time = year)
```

6.10 组合

将多幅小图组合起来构成一幅大图也是常见的需求，常见于出版级、产品级的作品中。组合涉及到布局，布局涉及到层次。有的组合图是从不同角度呈现数据，有的组合图是从传递信息的主次出发，等等。**patchwork** 包是非常流行的一个基于 **ggplot2** 的用于图形组合的 R 包，下面基于 faithful 数据展示绘制组合图形的过程。

© 黄湘云

图 6.20: 制作动画

©黄湘云

首先根据喷发时间将 faithful 数据分成两组。

```
# 根据喷发时间将数据分成两组
faithful <- transform(faithful, group = ifelse(eruptions > 3, "A", "B"))
```

绘制分组散点图，叠加二维核密度曲线。

```
# 绘制分组散点图
scatterplot <- ggplot(faithful, aes(eruptions, waiting, color = group)) +
  geom_point() +
  geom_density_2d() +
  theme_classic() +
  theme(axis.text = element_blank(), axis.title = element_blank())
```

将上图中的图例单独抽取出来，作为一个子图。

```
# https://stackoverflow.com/questions/46079033/
# Extract legend from ggplot object
extract_legend <- function(gg) {
  grobs <- ggplot_gtable(ggplot_build(gg))
  foo <- which(sapply(grobs$grobs, function(x) x$name) == "guide-box")
  grobs$grobs[[foo]]
}
legend <- extract_legend(scatterplot)
```

获得图例后，原图中不需要图例了。

```
scatterplot <- scatterplot + theme(legend.position = "none")
```

准备两个箱线图分别描述 faithful 数据集中的等待时间 waiting 和喷发时间 eruptions 。

```
boxplot_left <- ggplot(faithful, aes(group, waiting, fill = group)) +
  geom_boxplot() +
  theme_classic() +
  theme(
    legend.position = "none", axis.ticks.x = element_blank(),
    axis.text.x = element_blank(), axis.title.x = element_blank()
  )

boxplot_bottom <- ggplot(faithful, aes(group, eruptions, fill = group)) +
  geom_boxplot() +
  theme_classic() +
  theme(
    legend.position = "none", axis.ticks.y = element_blank(),
    axis.text.y = element_blank(), axis.title.y = element_blank()
```

```
  ) +
  coord_flip()
```

加载 **patchwork** 包，使用函数 `wrap_plots()` 组合 `boxplot_left` 、`scatterplot` 、`legend` 和 `boxplot_bottom` 四个子图，最终效果见下图。

```
library(patchwork)
top <- wrap_plots(boxplot_left, scatterplot, ncol = 2, widths = c(0.2, 0.8))
bottom <- wrap_plots(legend, boxplot_bottom, ncol = 2, widths = c(0.22, 0.8))
final <- wrap_plots(top, bottom, nrow = 2, heights = c(0.8, 0.2))
final
```

图 6.21: patchwork 组合多幅子图

主图是占据着最大篇幅的叠加二维密度曲线的散点图，展示数据的二维分布，两个箱线图辅助展示等待时间 waiting 和喷发时间 eruptions 的分布，而左下角的图例是次要的说明。

©黄湘云

6.11 艺术

Georgios Karamanis 基于 R 语言和扩展包 **ggforce** 制作了一系列生成艺术（Generative Arts）作品。下图是 **ggforce** 包的 4 个图层函数 `geom_regon()`、`geom_spiro()`、`geom_diagonal()` 和 `geom_spoke()` 分别生成的四幅图片。

```
library(ggforce)
s <- 900
ggplot() +
  geom_regon(aes(
    x0 = cos((1:s) / 57), y0 = sin((1:s) / 57),
    sides = 6, r = cos((1:s) / 24),
    angle = cos((1:s) / 23), color = 1:s %% 15
  ),
  linewidth = 0.2, fill = NA, linetype = "twodash"
  ) +
  scale_color_viridis_c(option = 15, guide = "none") +
  coord_fixed() +
  theme_void()

r <- seq(1, 11, 0.1)
ggplot() +
  geom_spiro(aes(r = r, R = r * 20, d = r^2, outer = T, color = r %% 10), linewidth = 3) +
  scale_color_viridis_c(option = "turbo") +
  coord_fixed() +
  theme_void() +
  theme(legend.position = "none")

s <- 1200
ggplot() +
  geom_diagonal(aes(
    x = cos(seq(0, pi, length.out = s)),
    y = sin(seq(0, pi, length.out = s)),
    xend = cos(seq(0, 360 * pi, length.out = s)),
    yend = sin(seq(0, 360 * pi, length.out = s))
  ),
  linewidth = 0.1, strength = 1
  ) +
  coord_fixed() +
  theme_void()
```

```
e <- 1e-3
s <- 1e4
t <- pi / 2 * cumsum(seq(e, -e, length.out = s))^3
ggplot() +
  geom_spoke(aes(
    x = cumsum(cos(t)), y = cumsum(sin(t)),
    angle = t, color = t, radius = 1:s %% 500
  ), alpha = 0.5) +
  scale_color_distiller(palette = 15, guide = "none") +
  coord_fixed() +
  theme_void()
```

需要充满想象，或借助数学、物理方程，或借助算法、数据生成。好看，但没什么用的生成艺术作品。

- https://art-from-code.netlify.app/
- https://clauswilke.com/art/project/before-after
- https://clauswilke.com/art/
- https://art.djnavarro.net/
- https://www.data-imaginist.com/art

© 黄湘云

(a) 函数 geom_regon()

(b) 函数 geom_spiro()

(c) 函数 geom_diagonal()

(d) 函数 geom_spoke()

图 6.22: R 语言与生成艺术

© 黄湘云

第七章　基础图形

本章按照图形作用分类介绍各种各样的统计图形，每个小节包含 5-8 个常用的图形，每个图形会结合数据说明其作用、绘制代码。希望借助真实的数据引发读者兴趣，提出问题，探案数据，讲故事，将其应用于其它场景，举一反三。除了数据获取、清理和预处理的工作外，从原始数据出发，还将穿插介绍图形绘制相关的数据操作，比如适当的分组计算、数据重塑等操作。当然，不可能逐行给出代码的说明和使用，因为这会显得非常累赘。探索数据和绘制图形的过程中，会有很多的中间代码，这些也不再展示了，仅给出最终展示图，但会做适当说明。

- 章节 7.1 探索、展示数据中隐含的趋势信息，具体有折线图、瀑布图、曲线图、曲面图、热力图、日历图、棋盘图和时间线图。
- 章节 7.2 以图形展示数据对比，达到更加突出、显著的效果，让差异给人留下印象，具体有柱形图、条形图、点线图（也叫克利夫兰点图）、雷达图和词云图。
- 章节 7.3 探索、展示数据中隐含的比例信息，以突出重点，具体有简单饼图、环形饼图、扇形饼图、帕累托图、马赛克图和矩阵树图。

7.1　描述趋势

GNU R 是一个自由的统计计算和统计绘图环境，最初由新西兰奥克兰大学统计系的 Ross Ihaka 和 Robert Gentleman 共同开发。1997 年之后，成立了一个 R Core Team（R 语言核心团队），他们在版本控制系统 Apache Subversion上一起协作开发至今。25 年—四分之一个世纪过去了，下面分析他们留下的一份开发日志，了解一段不轻易为人所知的故事。

首先，下载 1997 年至今约 25 年的原始代码提交日志数据。下载数据的代码如下，它是一行 Shell 命令，可在 MacOS 或 Ubuntu 等 Linux 系统的终端里运行，借助 Apache Subversion 软件，将提交日志导出为 XML 格式 的数据文件，保存在目录 `data-raw/` 下，文件名为 `svn_trunk_log_2022.xml`，本书网页版随附。

```
svn log --xml --verbose -r 6:83528 \
  https://svn.r-project.org/R/trunk > data-raw/svn_trunk_log_2022.xml
```

去掉没什么信息的前 5 次代码提交记录：初始化仓库，上传原始的 R 软件源码等。从 Ross Ihaka 在 1997-09-18 提交第 1 次代码改动开始，下载所有的提交日志。截至 2022-12-31，代码最新版本号为 83528，意味着代码仓库已存在 8 万多次提交。

©黄湘云

下载数据后，借助 **xml2** 包预处理这份 XML 格式数据，提取最重要的信息，谁在什么时间做了什么改动。经过一番操作后，将清洗干净的数据保存到目录 `data/` 下，以 R 软件特有的文件格式保存为 `svn-trunk-log-2022.rds`，同样与书随附。这样下来，原 XML 格式保存的 35 M 文件减少为 1 M 多，极大地减少存储空间，方便后续的数据探索和可视化。下面是这份日志数据最初的两行：

```
svn_trunk_log <- readRDS(file = "data/svn-trunk-log-2022.rds")
head(svn_trunk_log, 2)
```

```
#>   revision author               stamp                                msg
#> 1        6  ihaka 1997-09-18 04:41:25 New predict.lm from Peter Dalgaard
#> 2        7  ihaka 1997-09-18 04:42:42             Updated release number
```

一共是四个字段，分别是代码提交时记录的版本号 revision，提交代码的人 author，提交代码的时间 stamp 和提交代码时伴随的说明 msg。接下来，带着问题一起探索开源自由的统计软件 R 过去 25 年波澜壮阔的历史！

7.1.1 折线图

> **提示**
>
> 不再介绍每个函数、每个参数和每行代码的作用，而是重点阐述折线图的作用，以及如何解读数据，阐述解读的思路和方向，建立起数据分析的思维。将重点放在这些方面，有助于书籍存在的长远意义，又结合了最真实的背景和原始数据，相信对实际工作的帮助会更大。而对于使用到统计方法的函数，则详加介绍，展示背后的实现细节，而不是调用函数做调包侠。

折线图的意义是什么？在表达趋势变化，趋势的解读很重要。先来了解一下总体趋势，即过去 25 年里代码提交次数的变化情况。数据集 `svn_trunk_log` 没有年份字段，但时间字段 stamp 隐含了年份信息，因此，新生成一个字段 year 将年份信息从 stamp 提取出来。

```
svn_trunk_log <- within(svn_trunk_log, {
  # 提取日期、月份、年份、星期、第几周、第几天等时间成分
  year <- as.integer(format(stamp, "%Y"))
  date <- format(stamp, format = "%Y-%m-%d", tz = "UTC")
  month <- format(stamp, format = "%m", tz = "UTC")
  hour <- format(stamp, format = "%H", tz = "UTC")
  week <- format(stamp, format = "%U", tz = "UTC")
  wday <- format(stamp, format = "%a", tz = "UTC")
  nday <- format(stamp, format = "%j", tz = "UTC")
})
# 代码维护者 ID 和姓名对应
ctb_map <- c(
  "bates" = "Douglas Bates", "deepayan" = "Deepayan Sarkar",
```

```
  "duncan" = "Duncan Temple Lang", "falcon" = "Seth Falcon",
  "guido" = "Guido Masarotto", "hornik" = "Kurt Hornik",
  "iacus" = "Stefano M. Iacus", "ihaka" = "Ross Ihaka",
  "jmc" = "John Chambers", "kalibera" = "Tomas Kalibera",
  "lawrence" = "Michael Lawrence", "leisch" = "Friedrich Leisch",
  "ligges" = "Uwe Ligges", "luke" = "Luke Tierney",
  "lyndon" = "Others", "maechler" = "Martin Maechler",
  "mike" = "Others", "morgan" = "Martin Morgan",
  "murdoch" = "Duncan Murdoch", "murrell" = "Paul Murrell",
  "pd" = "Peter Dalgaard", "plummer" = "Martyn Plummer",
  "rgentlem" = "Robert Gentleman", "ripley" = "Brian Ripley",
  "smeyer" = "Sebastian Meyer", "system" = "Others",
  "tlumley" = "Thomas Lumley", "urbaneks" = "Simon Urbanek"
)
svn_trunk_log$author <- ctb_map[svn_trunk_log$author]
```

接着，调用分组聚合函数 aggregate() 统计各年的代码提交量。

```
trunk_year <- aggregate(data = svn_trunk_log, revision ~ year, FUN = length)
```

然后，将数据集 trunk_year 以折线图展示，如图 7.1 所示。

```
library(ggplot2)
ggplot(data = trunk_year, aes(x = year, y = revision)) +
  geom_point() +
  geom_line() +
  theme_classic() +
  theme(panel.grid.major.y = element_line(colour = "gray90")) +
  labs(x = "年份", y = "提交量")
```

© 黄湘云

图 7.1: 过去 25 年代码提交次数的变化情况

为什么呈现这样的变化趋势？我最初想到的是先逐步增加，然后下降一会儿，再趋于平稳。这比较符合软件从快速迭代开发期，过渡到成熟稳定期的生命周期。接着，从小时趋势图观察代码提交量的变化，发现有高峰有低谷，上午高峰，晚上低峰，但也并不是所有年份都一致，这是因为开发者来自世界各地，位于不同的时区。

```
aggregate(data = svn_trunk_log, revision ~ year + hour, length) |>
  ggplot(aes(x = hour, y = revision, group = year)) +
  geom_line() +
  geom_line(data = function(x) subset(x, year < 2006),
            aes(color = as.character(year))) +
  theme_classic() +
  labs(x = "时段", y = "提交量", color = "年份")
```

© 黄湘云

图 7.2: 提交代码的时段分布

最后，观察代码提交量的月趋势图，12 月和次年 1 月、7-8 月份提交量迎来小高峰，应该是教授们放寒暑假。

```
aggregate(data = svn_trunk_log, revision ~ year + month, length) |>
  transform(date = as.Date(paste(year, month, "01", sep = "-"))) |>
  ggplot(aes(x = date, y = revision)) +
  geom_point(aes(color = factor(year)), show.legend = F, size = 0.75) +
  geom_line(aes(color = factor(year)), show.legend = F) +
  scale_x_date(date_minor_breaks = "1 year") +
  theme_classic() +
  theme(panel.grid.minor.x = element_line()) +
  labs(x = "时间（月粒度）", y = "提交量")
```

©黄湘云

图 7.3: 提交代码的月份分布

7.1.2 瀑布图

相比于折线图，瀑布图将变化趋势和增减量都展示了，如图 7.4 所示，每年的提交量就好像瀑布上的水，图中每一段水柱表示当期相对于上一期的增减量。瀑布图是用矩形图层 `geom_rect()` 构造的，数据点作为矩形对角点，对撞型的颜色表示增减。

```
trunk_year <- trunk_year[order(trunk_year$year), ]

trunk_year_tmp <- data.frame(
  xmin = trunk_year$year[-length(trunk_year$year)],
  ymin = trunk_year$revision[-length(trunk_year$revision)],
  xmax = trunk_year$year[-1],
  ymax = trunk_year$revision[-1],
  fill = trunk_year$revision[-1] - trunk_year$revision[-length(trunk_year$revision)] > 0
)

ggplot() +
```

©梁海文

```
  geom_rect(
    data = trunk_year_tmp,
    aes(xmin = xmin, ymin = ymin, xmax = xmax, ymax = ymax, fill = fill
),
    show.legend = FALSE
  ) +
  geom_point(
    data = trunk_year, aes(x = year, y = revision), size = 0.75
  ) +
  scale_fill_brewer(palette = "Set2") +
  theme_classic() +
  theme(panel.grid.major.y = element_line(colour = "gray90")) +
  labs(x = "年份", y = "提交量")
```

图 7.4: 25 年代码逐年提交量的变化趋势

ggTimeSeries 包 (Kothari 2022) (https://github.com/thecomeonman/ggTimeSeries) 提供统计图层 `stat_waterfall()` 实现类似的瀑布图，如图 7.5 所示。

```
library(ggTimeSeries)
ggplot(data = trunk_year, aes(x = year, y = revision)) +
  stat_waterfall() +
  scale_fill_brewer(palette = "Set2") +
  theme_classic() +
  theme(panel.grid.major.y = element_line(colour = "gray90")) +
```

© 黄湘云


```
  labs(x = "年份", y = "提交量")
```

图 7.5: 矩形图层构造瀑布图

7.1.3 曲线图

将散点以线段逐个连接起来，形成折线图，刻画原始的变化，而曲线图的目标是刻画潜在趋势。有两种画法，其一从代数的角度出发，做插值平滑，在相邻两点之间以一条平滑的曲线连接起来；其二从统计的角度出发，做趋势拟合，通过线性或非线性回归，获得变化趋势，以图呈现，使得散点之中隐藏的趋势更加清晰。

ggplot2 (H. Wickham 2016) 包提供函数 `geom_smooth()` 拟合散点图中隐含的趋势，通过查看函数 `geom_smooth()` 的帮助文档，可以了解其内部调用的统计方法。默认情况下，采用局部多项式回归拟合方法，内部调用了函数 `loess()` 来拟合趋势，如图 7.6 所示。

```
ggplot(data = trunk_year, aes(x = year, y = revision)) +
  geom_point() +
  geom_smooth(data = subset(trunk_year, year != 1997)) +
  theme_classic() +
  theme(panel.grid.major.y = element_line(colour = "gray90")) +
  labs(x = "年份", y = "提交量")

#> `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

© 基湘石

图 7.6: 过去 25 年代码提交次数的变化情况

类似大家熟悉的线性回归拟合函数 `lm()`，函数 `loess()` 也是基于类似的使用语法。下面继续以此数据为例，了解该函数的使用，继而了解 ggplot2 绘制平滑曲线图背后的统计方法。1997 年是不完整的，不参与模型参数的估计。

```
trunk_year_loess <- loess(revision ~ year,
  data = subset(trunk_year, year != 1997),
  span = 0.75, degree = 2, method = "loess",
  family = "symmetric",
  control = loess.control(surface = "direct", iterations = 4)
)
```

下面通过设定函数 `geom_smooth()` 的参数，可以达到一样的效果，见下图 7.7

```
ggplot(data = trunk_year, aes(x = year, y = revision)) +
  geom_point() +
  geom_smooth(method = "loess", formula = "y~x",
    method.args = list(
      span = 0.75, degree = 2, family = "symmetric",
      control = loess.control(surface = "direct", iterations = 4)
    ), data = subset(trunk_year, year != 1997)) +
  theme_classic() +
  theme(panel.grid.major.y = element_line(colour = "gray90")) +
  labs(x = "年份", y = "提交量")
```

©黄湘云

图 7.7: 过去 25 年代码提交次数的变化情况

`method = "loess"` 意味着调用了一种非参数的回归方法，即局部估计散点平滑（locally estimated scatterplot smoothing），另一个与之类似的回归方法是局部加权散点平滑（locally weighted scatterplot smoothing），简称 lowess 。1991 年 Jerome Friedman 提出多元适应性回归样条（**M**ultivariate **A**daptive **R**egression **S**plines），R 语言社区对应功能的扩展包是 **earth** 。

除了 `method = "loess"`，函数 `geom_smooth()` 支持的统计方法还有很多，比如非线性回归拟合 `nls()`

```
trunk_year_nls <- nls(revision ~ a * (year - 1996)^2 + b,
  data = subset(trunk_year, year != 1997),
  start = list(a = -0.1, b = 1000)
)
```

采用一元二次非线性回归拟合方法，效果如图 7.8 所示。

```
ggplot(data = trunk_year, aes(x = year, y = revision)) +
  geom_point() +
  geom_smooth(
    method = "nls",
    formula = "y ~ a * (x - 1996)^2 + b",
    method.args = list(
      start = list(a = -0.1, b = 1000)
    ), se = FALSE,
    data = subset(trunk_year, year != 1997),
  ) +
```

©黄天宇

```
  theme_classic() +
  theme(panel.grid.major.y = element_line(colour = "gray90")) +
  labs(x = "年份", y = "提交量")
```

图 7.8: 过去 25 年代码提交次数的变化情况

> **注意**
>
> 在函数 `geom_smooth()` 内调用非线性回归拟合方法时，暂不支持提供置信区间。

即便在不清楚统计原理的情况下，也不难看出图 7.7 和图 7.8 的差异，局部多项式回归捕捉到了更多的信息，特别是起步阶段的上升趋势，以及 2000-2005 年的高峰特点。

```
summary(trunk_year_loess)
```

```
#> Call:
#> loess(formula = revision ~ year, data = subset(trunk_year, year !=
#>     1997), span = 0.75, degree = 2, family = "symmetric", method = "loess",
#>     control = loess.control(surface = "direct", iterations = 4))
#>
#> Number of Observations: 25
#> Equivalent Number of Parameters: 4.53
#> Residual Scale Estimate: 308.4
#> Trace of smoother matrix: 4.97  (exact)
#>
#> Control settings:
```

©黄湘云


```
#>   span     :  0.75
#>   degree   :  2
#>   family   :  symmetric      iterations = 4
#>   surface  :  direct
#>   normalize:  TRUE
#>  parametric:  FALSE
#> drop.square:  FALSE
```

```
summary(trunk_year_nls)
```

```
#>
#> Formula: revision ~ a * (year - 1996)^2 + b
#>
#> Parameters:
#>    Estimate Std. Error t value Pr(>|t|)
#> a   -2.9625     0.4555  -6.504 1.23e-06 ***
#> b 3070.0890   147.1920  20.858  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 471.8 on 23 degrees of freedom
#>
#> Number of iterations to convergence: 1
#> Achieved convergence tolerance: 2.808e-08
```

非线性回归模型带有 2 个参数，一共 26 个观察值，因此，自由度为 26 - 2 = 24。RSE 残差平方和的标准差为

```
# 非线性回归的残差平方和的标准差
sqrt(sum(residuals(trunk_year_nls)^2)/24)
```

```
#> [1] 461.8963
```

以平滑曲线连接相邻的散点，可以构造一个插值方法给函数 `geom_smooth()`，如下示例基于样条插值函数 `spline()`。样条源于德国宝马工程师，车辆外壳弧线，那些拥有非常漂亮的弧线，越光滑，与空气的摩擦阻力越小，车辆的气动外形更加符合流体力学的要求，加工打磨更加困难，往往价值不菲。美感是相通的，即使不懂车标，通过气动外形，也能识别出车辆的档次。

ggplot2 包支持的平滑方法有很多，如借助函数 `splinefun()` 构造样条插值获得平滑曲线，调用 **mgcv** 包的函数 `gam()` ，调用 **ggalt** 包的函数 `geom_xspline()` 。

```
xxspline <- function(formula, data, ...) {
  dat <- model.frame(formula, data)
  res <- splinefun(dat[[2]], dat[[1]])
  class(res) <- "xxspline"
```

内部资料 ©

```
  res
}

predict.xxspline <- function(object, newdata, ...) {
  object(newdata[[1]])
}

ggplot(data = trunk_year, aes(x = year, y = revision)) +
  geom_point() +
  geom_smooth(
    formula = "y~x",
    method = xxspline, se = FALSE,
    data = subset(trunk_year, year != 1997)
  ) +
  theme_classic() +
  theme(panel.grid.major.y = element_line(colour = "gray90")) +
  labs(x = "年份", y = "提交量")

ggplot(data = trunk_year, aes(x = year, y = revision)) +
  geom_point() +
  geom_smooth(
    formula = y ~ s(x, k = 12),
    method = "gam", se = FALSE,
    data = subset(trunk_year, year != 1997)
  ) +
  theme_classic() +
  theme(panel.grid.major.y = element_line(colour = "gray90")) +
  labs(x = "年份", y = "提交量")

ggplot(data = trunk_year, aes(x = year, y = revision)) +
  geom_point() +
  geom_smooth(
    method = "lm",
    formula = "y ~ poly((x - 1996), 3)",
    se = FALSE,
    data = subset(trunk_year, year != 1997),
  ) +
  theme_classic() +
  theme(panel.grid.major.y = element_line(colour = "gray90")) +
  labs(x = "年份", y = "提交量")
```

© 黄湘云

(a) 自定义样条插值 spline

(b) 广义可加模型样条拟合

(c) 自由度为 3 的正交多项式拟合

图 7.9: 过去 25 年代码提交次数的变化情况

数学公式表达的统计模型与 R 语言表达的计算公式的对应关系见下表格 7.1 ，更多详情见帮助文档 `?formula`。

表格 7.1: 数学公式与 R 语言表示的计算公式

数学公式	R 语言计算公式
$y = \beta_0$	`y ~ 1`
$y = \beta_0 + \beta_1 x_1$	`y ~ 1 + x1` 或 `y ~ x1` 或 `y ~ x1 + x1^2`
$y = \beta_1 x_1$	`y ~ 0 + x1` 或 `y ~ -1 + x1`

数学公式	R 语言计算公式
$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$	`y ~ x1 + x2`
$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$	`y ~ x1 * x2`
$y = \beta_0 + \beta_1 x_1 x_2$	`y ~ x1:x2`
$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$	`y ~ x1 + x2 + x1:x2`
$y = \beta_0 + \sum_{i=1}^{999} \beta_i x_i$	`y ~ .`
$y = \beta_0 + \beta_1 x + \beta_2 x^5$	`y ~ x + I(x^5)`
$y = \beta_0 + \beta_1 x + \beta_2 x^2$	`y ~ x + I(x^2)`
$y = \beta_0 + \beta_1 x + \beta_2 x^2$	`y ~ poly(x, degree = 2, raw = TRUE)`

7.1.4 流线图

流线图（Stream Graph）是堆积面积图（Stacked Area Graph）的一种变体，适合描述时间序列数据的趋势。**ggplot2** 扩展包 ggstream 可以制作流线图，如下图所示。

```
library(ggstream)
trunk_year_author <- aggregate(data = svn_trunk_log, revision ~ year + author, FUN = length)
ggplot(trunk_year_author, aes(x = year, y = revision, fill = author)) +
  geom_stream() +
  theme_classic() +
  theme(legend.position = "bottom") +
  labs(x = "年份", y = "提交量", fill = "贡献者")
```

© 黄湘云

图 7.10: 各开发者的提交量趋势

7.1.5 曲面图

ggplot2 包暂不支持绘制三维曲面图，而 **lattice** 包支持，但也是非常有限的支持。**lattice 包**和 **ggplot2** 包都是基于图形语法的，层层叠加就必然会出现覆盖，只有在绘制函数型数据的图像时是合适的，因为覆盖少，即使覆盖也不妨碍趋势的表达。根据不同的使用场景有两个更好的选择，基于 OpenGL 的真三维图形可以用 rayrender 和 rayshader 包绘制，而基于 JavaScripts 的交互式三维图形可以用 rgl 或 plotly 包绘制。

下图 7.11 是用 **lattice** 包的 `wireframe()` 函数绘制的，这是一个三维曲面透视图，三维图形有时候并不能很好地表达数据，或者数据并不适合用三维图形表示。数据本身并没有那么明显的趋势规律，同样也会体现不出三维图形的表达能力。大部分情况下，我们应当避免使用静态的三维图形，但函数型数据是适合用三维图形来表达的。

每周的代码提交量受影响因素多，不确定性多，波动表现尖锐高频，上图反而对整体趋势的表达不够简

© 黄湘云

图 7.11: 25 年代码提交量变化趋势图

©黄湘云

洁清晰。按年、月统计提交量平均掉了每日的波动，反而可以体现更大的周期性和趋势性。下面绘制三维柱形图，三维图形天然给人有更加直观的感觉，毕竟立体。**latticeExtra** 包提供三维柱形图图层 `panel.3dbars()`，如图 7.12 所示。

图 7.12: 25 年代码提交量变化趋势图

7.1.6 热力图

图 7.13 提交量变化趋势

```
ggplot(data = trunk_year_week, aes(x = as.integer(week) , y = year, fill = revision)) +
  geom_tile(linewidth = 0.4) +
  scale_fill_viridis_c(option = "C") +
  scale_x_continuous(expand = c(0, 0)) +
  scale_y_continuous(expand = c(0, 0)) +
  theme_classic() +
```

© 基湘工


```
  labs(x = "第几周", y = "年份", fill = "提交量")
```

图 7.13: 25 年代码提交量变化热力图

图层 scale_x_continuous() 中设置 expand = c(0, 0) 可以去掉数据与 x 轴之间的空隙。或者添加坐标参考系图层 coord_cartesian()，设置参数 expand = FALSE 同时去掉横纵轴与数据之间的空隙。

```
aggregate(data = svn_trunk_log, revision ~ year + month, length) |>
  ggplot(aes(x = month, y = year, fill = revision)) +
  geom_tile(linewidth = 0.4) +
  scale_fill_viridis_c(option = "C") +
  coord_cartesian(expand = FALSE) +
  theme_classic() +
  labs(x = "月份", y = "年份", fill = "提交量")
```

©黄湘云

图 7.14: 25 年代码提交量变化热力图

7.1.7 日历图

更加直观地展示出节假日、休息工作日、寒暑假，比如描述学生学习规律、需求的季节性变化、周期性变化。

```
# 星期、月份缩写
week.abb <- c("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat")
month.abb <- c(
  "Jan", "Feb", "Mar", "Apr", "May", "Jun",
  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
)
# 按年、星期、第几周聚合统计提交量数据
svn_trunk_year <- aggregate(
  revision ~ year + wday + week, FUN = length,
  data = svn_trunk_log, subset = year %in% 2018:2022
)
# 第几周转为整型数据
# 周几转为因子型数据
svn_trunk_year <- within(svn_trunk_year, {
   week = as.integer(week)
```

内部稿 ©

```
  wday = factor(wday, labels = week.abb)
})
```

```
ggplot(data = svn_trunk_year, aes(
  x = week, y = wday, fill = cut(revision, breaks = 5 * 0:5)
)) +
  geom_tile(color = "white", linewidth = 0.5) +
  scale_fill_brewer(palette = "Greens") +
  scale_x_continuous(
    expand = c(0, 0), breaks = seq(1, 52, length = 12), labels = month.abb
  ) +
  facet_wrap(~year, ncol = 1) +
  theme_minimal() +
  labs(x = "月份", y = "星期", fill = "提交量")
```

©黄湘云

图 7.15: 最近 5 年休息和工作日打码活跃度

经过了解 svn_trunk_year 2018 - 2022 年每天提交量的范围是 0 次到 21 次，0 次表示当天没有提交代码，SVN 上也不会有日志记录。因此，将提交量划分为 5 档

7.1.8 棋盘图

棋盘图一般可以放所有时间节点的聚合信息，格点处为落的子

该数据集的存储结构很简单，是一个两列的数据框，它的一些属性如下：

```
str(rversion)
```

```
#> 'data.frame':    140 obs. of  2 variables:
#>  $ version: chr  "0.49" "0.50-a1" "0.50-a4" "0.60.0" ...
```

```
#>  $ date   : chr  "1997-04-23" "1997-07-22" "1997-09-10" "1997-12-04" ...
```

做一点数据处理，将 date 字段转为日期类型，并从日期中提取年、月信息。

```
rversion$date <- as.Date(rversion$date, format = "%Y-%m-%d", tz = "UTC")
rversion$year <- format(rversion$date, "%Y")
rversion$month <- format(rversion$date, "%m")
```

统计过去 25 年里每月的发版次数，如图图 7.16

```
aggregate(data = rversion, version ~ year + month, length) |>
  ggplot(aes(x = month, y = year)) +
  geom_label(aes(label = version, fill = version),
    show.legend = F, color = "white") +
  scale_fill_viridis_c(option = "D", begin = 0.2, end = 0.8) +
  theme_classic() +
  theme(panel.grid.major.y = element_line(colour = "gray95")) +
  labs(x = "月份", y = "年份")
```

© 黄湘云

图 7.16: 25 年 R 软件发版情况

©谢益辉

7.1.9 时间线图

时间线图非常适合回顾过去，展望未来，讲故事

时间线图展示信息的层次和密度一般由时间跨度决定。时间跨度大时，展示重点节点信息，时间跨度小时，重点和次重点信息都可以放。从更加宏观的视角，厘清发展脉络，比如近两年的 R 软件发版情况。

本节用到一个数据集 rversion，记录了历次 R 软件发版时间及版本号，见表格 7.2

表格 7.2: R 软件发版数据集（部分）

版本号	发版日期	发版年份	发版月份
0.49	1997-04-23	1997	04
0.50-a1	1997-07-22	1997	07
0.50-a4	1997-09-10	1997	09
0.60.0	1997-12-04	1997	12
0.60.1	1997-12-07	1997	12
0.61.0	1997-12-22	1997	12

```
rversion_tl <- within(rversion, {
  # 版本号为 x.0.0 为重大版本 big
  # 版本号为 x.1.0 x.12.0 x.20.0 为主要版本 major
  # 版本号为 x.0.1 为次要版本 minor
  status <- ifelse(grepl(pattern = "*\\.0\\.0", x = version), "big", version)
  status <- ifelse(grepl(pattern = "*\\.[1-9]{1,2}\\.0$", x = status), "major", status)
  status <- ifelse(!status %in% c("big", "major"), "minor", status)
})
positions <- c(0.5, -0.5, 1.0, -1.0, 1.5, -1.5)
directions <- c(1, -1)
# 位置
rversion_pos <- data.frame(
  # 只要不是同一天发布的版本，方向相对
  date = unique(rversion_tl$date),
  position = rep_len(positions, length.out = length(unique(rversion_tl$date))),
  direction = rep_len(directions, length.out = length(unique(rversion_tl$date)))
)
# 原始数据上添加方向和位置信息
rversion_df <- merge(x = rversion_tl, y = rversion_pos, by = "date", all = TRUE)
# 最重要的状态放在最后绘制到图上
rversion_df <- rversion_df[with(rversion_df, order(date, status)), ]
```

选取一小段时间内的发版情况，比如最近的三年 — 2020 - 2022 年

```r
# 选取 2020 - 2022 年的数据
sub_rversion_df<- rversion_df[rversion_df$year %in% 2020:2022, ]
# 月份注释
month_dat <- data.frame(
  date = seq(from = as.Date('2020-01-01'), to = as.Date('2022-12-31'), by = "3 month")
)
month_dat <- within(month_dat, {
  month = format(date, "%b")
})
# 年份注释
year_dat <- data.frame(
  date = seq(from = as.Date('2020-01-01'), to = as.Date('2022-12-31'), by = "1 year")
)
year_dat <- within(year_dat, {
  year = format(date, "%Y")
})
```

图 7.17 展示 2020-2022 年 R 软件发版情况

```r
ggplot(data = sub_rversion_df) +
  geom_segment(aes(x = date, y = 0, xend = date, yend = position)) +
  geom_hline(yintercept = 0, color = "black", linewidth = 1) +
  geom_label(
    aes(x = date, y = position, label = version, color = status),
    show.legend = FALSE
  ) +
  geom_point(aes(x = date, y = 0, color = status),
    size = 3, show.legend = FALSE
  ) +
  geom_text(
    data = month_dat, aes(x = date, y = 0, label = month), vjust = 1.5
  ) +
  geom_text(
    data = year_dat, aes(x = date, y = 0, label = year), vjust = -0.5
  ) +
  theme_void()
```

© 基湘 江

图 7.17: 2020-2022 年 R 软件发版情况

图中红色标注的是里程碑式的重大版本，绿色标注的是主要版本，蓝色标注的次要版本，小修小补，小版本更新。

当时间跨度非常大时，比如过去 25 年，那就只能放重大版本和主要版本信息了，时间上月份信息就不能用名称简写，而用数字更加合适。而且还得竖着放，同时添加那个版本最有影响力的改动。相比于，棋盘图，这是时间线图的优势。

```
sub_rversion_df2 <- rversion_df[rversion_df$status %in% c("big", "major"), ]
ggplot(data = sub_rversion_df2) +
  geom_segment(aes(x = 0, y = date, xend = position, yend = date, color = status),
    show.legend = F
  ) +
  geom_vline(xintercept = 0, color = "black", linewidth = 1) +
  geom_label(
    aes(x = position, y = date, label = version, color = status),
    show.legend = FALSE
  ) +
  geom_point(aes(x = 0, y = date, color = status), size = 3, show.legend = FALSE) +
  geom_text(
    aes(x = 0, y = as.Date(format(date, "%Y-01-01")), label = year),
    hjust = -0.1
  ) +
  theme_void()
```

© 黄湘云

图 7.18: 25 年里 R 软件重大及主要版本发布情况

在 R 语言诞生的前 5 年里，每年发布 3 个主要版本，这 5 年是 R 软件活跃开发的时期。而 2003-2012

年的这 10 年，基本上每年发布 2 个主要版本。2013-2022 年的这 10 年，基本上每年发布 1 个主要版本。

timevis 包基于 JavaScript 库 Vis 的 vis-timeline 模块，可以创建交互式的时间线图，支持与 Shiny 应用集成。

7.2 描述对比

数据来自中国国家统计局发布的 2021 年统计年鉴，

表格 7.3: 中国各年龄段的性别比数据（部分）

年龄	人口数/男	人口数/女	性别比（女 =100）	区域
0-4	16078524	14523013	110.71	城市
5-9	17172999	15087731	113.82	城市
10-14	14619691	12727731	114.86	城市
15-19	17249362	15404683	111.97	城市
20-24	19776472	18481665	107.01	城市
25-29	22937131	21478748	106.79	城市

对比的是什么？城市、镇和乡村的性别分布，是否失衡？在哪个年龄段表现很失衡？

7.2.1 柱形图

分年龄段比较城市、镇和乡村的性别比数据

```
ggplot(data = china_age_sex, aes(x = `年龄`, y = `性别比（女=100）`, fill = `区域`)) +
  geom_hline(yintercept = 100, color = "gray", lty = 2, linewidth = 1) +
  geom_col(position = "dodge2", width = 0.75) +
  theme_bw()
```

© 黄湘云

图 7.19: 分年龄段比较城市、镇和乡村的性别比数据

考虑到数据本身的含义，一般来说，性别比不可能从 0 开始，除非现实中出现了《西游记》里的女儿国。因此，将纵轴的范围，稍加限制，从性别比为 70 开始，目的是突出城市、镇和乡村的差异。

```
ggplot(data = china_age_sex, aes(x = `年龄`, y = `性别比（女=100）`, fill = `区域`)) +
  geom_hline(yintercept = 100, color = "gray", lty = 2, linewidth = 1) +
  geom_col(position = "dodge2", width = 0.75) +
  coord_cartesian(ylim = c(70, 130)) +
  theme_bw()
```

© 黄湘云

图 7.20: 分年龄段比较城市、镇和乡村的性别比数据

7.2.2 条形图

将柱形图横过来即可得到条形图，横过来的好处主要体现在分类很多的时候，留足空间给年龄分组的分类标签，从左到右，从上往下也十分符合大众的阅读习惯

```
ggplot(data = china_age_sex, aes(x = `性别比（女=100）`, y = `年龄`, fill = `区域`)) +
  geom_vline(xintercept = 100, color = "gray", lty = 2, linewidth = 1) +
  geom_col(position = "dodge2", width = 0.75) +
  coord_cartesian(xlim = c(70, 130)) +
  theme_bw()
```

© 黄湘云

图 7.21: 分年龄段比较城市、镇和乡村的性别比数据

7.2.3 点线图

克利夫兰点图 dotchart() 在条形图的基础上，省略了条形图的宽度，可以容纳更多的数据点。

```
ggplot(data = china_age_sex, aes(x = `性别比（女=100）`, y = `年龄`, color = `区域`)) +
  geom_vline(xintercept = 100, color = "lightgray", lty = 2, linewidth = 1) +
  geom_point() +
  theme_bw()
```

© 黄湘云

图 7.22: 分年龄段比较城市、镇和乡村的性别比数据

7.2.4 词云图

ggwordcloud 包提供词云图层 `geom_text_wordcloud()` 根据代码提交的说明制作词云图。

```
library(ggwordcloud)
aggregate(data = svn_trunk_log, revision ~ author, FUN = length) |>
  ggplot(aes(label = author, size = revision)) +
  geom_text_wordcloud(seed = 2022, grid_size = 10, max_grid_size = 24) +
  scale_size_area(max_size = 20)
```

© 统计出版社

图 7.23: 词云图

词云图也可以是条形图或柱形图的一种替代，词云图不用担心数目多少，而条形图不适合太多的分类情形。

```
aggregate(data = svn_trunk_log, revision ~ author, FUN = length) |>
  subset(subset = revision >= 100) |>
  ggplot(aes(x = revision, y = reorder(author, revision))) +
  geom_col() +
  theme_classic() +
  coord_cartesian(expand = FALSE) +
  labs(x = "提交量", y = "维护者")
```

©黄湘云

图 7.24: 开发者提交量排行榜

7.3 描述占比

7.3.1 简单饼图

提交量小于 2000 次的贡献者合并为一类 Others，按贡献者分组统计提交量及其占比，如图 7.25 所示。

```
aggregate(data = svn_trunk_log, revision ~ author, FUN = length) |>
  transform(author2 = ifelse(revision < 2000, "Others", author)) |>
  aggregate(revision ~ author2, FUN = sum) |>
  transform(label = paste0(round(revision / sum(revision), digits = 4) * 100, "%")) |>
  ggplot(aes(x = 1, fill = reorder(author2, revision), y = revision)) +
  geom_col(position = "fill", show.legend = FALSE, color = "white") +
  scale_y_continuous(labels = scales::label_percent()) +
  coord_polar(theta = "y") +
  geom_text(aes(x = 1.2, label = author2),
    position = position_fill(vjust = 0.5), color = "black"
  ) +
  geom_text(aes(x = 1.65, label = label),
    position = position_fill(vjust = 0.5), color = "black"
  ) +
```

© 黄湘云


```
  theme_void() +
  labs(x = NULL, y = NULL)
```

图 7.25: 维护者提交量占比

当把提交量小于 1000 次的贡献者合并为 Others，则分类较多，占比小的也有一席之地，饼图上显得十分拥挤。

```
aggregate(data = svn_trunk_log, revision ~ author, FUN = length) |>
  transform(author2 = ifelse(revision < 1000, "Others", author)) |>
  aggregate(revision ~ author2, FUN = sum) |>
  transform(label = paste0(round(revision / sum(revision), digits = 4) * 100, "%")) |>
  ggplot(aes(x = 1, fill = reorder(author2, revision)  , y = revision)) +
  geom_col(position = "fill", show.legend = FALSE, color = "white") +
  scale_y_continuous(labels = scales::label_percent()) +
  coord_polar(theta = "y") +
  geom_text(aes(x = 1.2, label = author2),
    position = position_fill(vjust = 0.5), color = "black"
  ) +
  geom_text(aes(x = 1.6, label = label),
    position = position_fill(vjust = 0.5), color = "black"
  ) +
  theme_void() +
  labs(x = NULL, y = NULL)
```

© 基础图形

图 7.26: 维护者提交量占比

一种缓解拥挤的办法是通过 ggrepel 包在扇形区域旁边添加注释

```
library(ggrepel)
dat1 <- aggregate(data = svn_trunk_log, revision ~ author, FUN = length) |>
  transform(author2 = ifelse(revision < 1000, "Others", author)) |>
  aggregate(revision ~ author2, FUN = sum)

dat2 <- within(dat1, {
  value <- 100 * revision / sum(revision)
  csum <- rev(cumsum(rev(value)))
  pos <- value / 1.5 + c(csum[-1], NA)
  pos <- ifelse(is.na(pos), value / 2, pos)
  label <- paste(author2, paste0(round(value, 2), "%"), sep = "\n")
})

ggplot(data = dat2, aes(x = 1, fill = author2, y = value)) +
  geom_col(show.legend = FALSE, color = "white") +
  coord_polar(theta = "y") +
  geom_label_repel(aes(y = pos, label = label),
    size = 4.5, nudge_x = 0.75, show.legend = FALSE
  ) +
  theme_void() +
  labs(x = NULL, y = NULL)
```

© 黄湘云

图 7.27: 维护者提交量占比

但是数量很多的情况下，也是无能为力的，当然，是否需要显示那么多，是否可以合并占比小的部分，也是值得考虑的问题。

表格 7.4: SVN 日志中的贡献者（部分）

SVN 花名	真实名字	主要贡献
rgentlem	Robert Gentleman	R 语言创始人
ihaka	Ross Ihaka	R 语言创始人
ripley	Brian Ripley	R Core Team 中的核心
murrell	Paul Murrell	grid 包及栅格绘图系统
maechler	Martin Maechler	cluster / Matrix 包维护者
hornik	Kurt Hornik	R FAQ 和 CRAN 维护者
jmc	John Chambers	S 语言的创始人之一
bates	Douglas Bates	nlme / lme4 包核心开发者
pd	Peter Dalgaard	《统计导论与 R 语言》作者

© 基础图形

表格 7.4: SVN 日志中的贡献者（部分）

SVN 花名	真实名字	主要贡献
ligges	Uwe Ligges	让 BUGS 与 R 同在
plummer	Martyn Plummer	让 JAGS 与 R 携手
luke	Luke Tierney	compiler 包核心开发者
iacus	Stefano M. Iacus	让 CRAN 拥抱 Fedora 系统
kalibera	Tomas Kalibera	编码问题终结者
deepayan	Deepayan Sarkar	lattice 包维护者
murdoch	Duncan Murdoch	R 软件的 Windows 版本维护者
duncan	Duncan Temple Lang	XML / RCurl 包开发者
urbaneks	Simon Urbanek	rJava / Rserve 包维护者

7.3.2 环形饼图

中间空了一块

```
aggregate(data = svn_trunk_log, revision ~ author, FUN = length) |>
  transform(author2 = ifelse(revision < 2000, "Others", author)) |>
  aggregate(revision ~ author2, FUN = sum) |>
  transform(label = paste0(round(revision / sum(revision), digits = 4) * 100, "%")) |>
  ggplot(aes(x = 1, fill = author2, y = revision)) +
  geom_col(position = "fill", show.legend = FALSE, color = "white") +
  scale_y_continuous(labels = scales::label_percent()) +
  coord_polar(theta = "y") +
  geom_text(aes(x = 1.2, label = author2),
    position = position_fill(vjust = 0.5), color = "black"
  ) +
  geom_text(aes(x = 1.7, label = label),
    position = position_fill(vjust = 0.5), color = "black"
  ) +
  theme_void() +
  labs(x = NULL, y = NULL) +
  xlim(c(0.2, 1.7))
```

© 黄湘云

图 7.28: 维护者提交量占比

7.3.3 扇形饼图

扇形饼图又叫风玫瑰图或南丁格尔图

```
aggregate(data = svn_trunk_log, revision ~ author, FUN = length) |>
  transform(author2 = ifelse(revision < 2000, "Others", author)) |>
  aggregate(revision ~ author2, FUN = sum) |>
  ggplot(aes(x = reorder(author2, revision), y = revision)) +
  geom_col(aes(fill = author2), show.legend = FALSE) +
  coord_polar() +
  theme_minimal() +
  theme(axis.text.y = element_blank()) +
  labs(x = NULL, y = NULL)
```

© 黄湘云

图 7.29: 维护者提交量分布

7.3.4 帕累托图

除了饼图，还常用堆积柱形图描述各个部分的数量，柱形图的优势在于简洁，准确，兼顾对比和趋势。下图 7.30 描述各年开发者们的贡献量及其变化趋势，饼图无法表达数量的变化趋势。

```
aggregate(data = svn_trunk_log, revision ~ year + author, FUN = length) |>
  ggplot(aes(x = year, y = revision, fill = author)) +
  geom_col() +
  theme_classic() +
  coord_cartesian(expand = FALSE) +
  theme(legend.position = "bottom") +
  labs(x = "年份", y = "提交量", fill = "开发者")
```

© 黄湘云

图 7.30: 代码提交量的比例趋势

百分比堆积柱形图在数量堆积柱形图的基础上，将纵坐标的数量转化为百分比，下图 7.31 展示各年开发者代码提交比例的变化趋势。

```
aggregate(data = svn_trunk_log, revision ~ year + author, FUN = length) |>
  ggplot(aes(x = year, y = revision, fill = author)) +
  geom_col(position = "fill") +
  scale_y_continuous(labels = scales::label_percent()) +
  theme_classic() +
  coord_cartesian(expand = FALSE) +
  theme(legend.position = "bottom") +
  labs(x = "年份", y = "提交量", fill = "开发者")
```

© 基础图形

图 7.31: 代码提交量的比例趋势

帕累托图描述各个部分的占比，特别是突出关键要素的占比。收入常服从帕累托分布，这是一个幂率分布，比如 80% 的财富集中在 20% 的人的手中。下图 7.32 展示过去 25 年各位开发者的代码累计提交量，提交量小于 1000 的已经合并为一类。不难看出，Ripley 的提交量远高于其他开发者。

```
dat <- aggregate(data = svn_trunk_log, revision ~ author, FUN = length) |>
  transform(author = ifelse(revision < 1000, "Others", author)) |>
  aggregate(revision ~ author, FUN = sum)
dat <- dat[order(-dat$revision), ]

ggplot(data = dat, aes(x = reorder(author, revision, decreasing = T), y = revision)) +
  geom_col(width = 0.75) +
  geom_line(aes(y = cumsum(revision), group = 1)) +
  geom_point(aes(y = cumsum(revision))) +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1)) +
  labs(x = "维护者", y = "累计提交量")
```

© 黄湘云

图 7.32: 代码提交量的比例分布

7.3.5 马赛克图

马赛克图常用于展示多个分类数据，如图 7.33 所示，展示加州伯克利分校院系录取情况。

```
library(ggmosaic)
ggplot(data = as.data.frame(UCBAdmissions)) +
  geom_mosaic(aes(x = product(Dept, Gender), weight = Freq, fill = Admit)) +
  theme_minimal()
```

© 黄湘云

图 7.33: 加州伯克利分校院系录取情况

提示

Base R 提供函数 `plot()` 和 `mosaicplot()` 对 `table` 表格类型的数据可视化，提供一套公式绘图语法，可以绘制类似的马赛克图。

```
mosaicplot(~ Gender + Dept + Admit,
  data = UCBAdmissions, color = TRUE,
  main = "", xlab = "性别", ylab = "院系"
)
```

对于多维列联表数据，Base R 提供函数 `loglin()` 拟合对数线性模型，以获取更加定量的结果。更进一步，**MASS** 包在函数 `loglin()` 的基础上，打包了另一个函数 `loglm()` ，它提供与函数 `lm()` 和 `glm()` 相一致的公式语法，使用起来更加方便。当然，函数 `glm()` 本身也是可以拟合对数线性模型的，毕竟它也是一种特殊的广义线性模型。

© 基础统计

7.3.6 矩阵树图

矩阵树图展示有层次的占比，比如 G20 国家的 GDP 按半球、地域分组。**treemapify** 包专门绘制矩阵树图，下图 7.34 展示南北半球，各地域内各个国家 GDP 的占比。

表格 7.5: G20 国家经济水平：GDP 总量、人类发展指数等

区域	国家	GDP	人类发展指数	经济水平	所属半球
Africa	South Africa	384315	0.629	Developing	Southern
North America	United States	15684750	0.937	Advanced	Northern
North America	Canada	1819081	0.911	Advanced	Northern
North America	Mexico	1177116	0.775	Developing	Northern
South America	Brazil	2395968	0.730	Developing	Southern
South America	Argentina	474954	0.811	Developing	Southern

每个瓦片的大小代表国家的 GDP 在所属半球里的比重。

```
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 = "G20 主要经济体", fill = "区域")
```

© 基础图形

图 7.34: G20 主要经济体的 GDP 占比

7.3.7 量表图

展示调查研究中的用户态度。量表在市场调查，问卷调查，App 用户体验反馈等方面应用十分广泛，已经成为调查研究中的金标准。量表由心理学家 Rensis Likert 于 1932 年提出 (Likert 1932)，Likert Scale 就是以他的名字命名的。

量表在互联网产品中应用非常广泛，比如美团 App 里消息页面中的反馈框，用以收集用户使用产品的体验情况，如表格 7.6 所示，从极其困难到极其方便，将用户反馈分成 7 个等级，目的是收集用户的反馈，以期改善产品的体验。

统计学 ©

表格 7.6: 您觉得在本页面，找想看的消息方便吗?

1	2	3	4	5	6	7
极其困难	非常困难	比较困难	一般	比较方便	非常方便	极其方便

量表中的问题、观点的描述极其简单明了，对回答、表明态度的任何人都不会造成歧义，以确保不受文化差异、学历差异等的影响，受调查的人只需在待选的几个选项中圈选即可。候选项一般为 5-7 个，下面是一组典型的选项：

1. Strongly disagree （强烈反对），
2. Disagree（反对），
3. Neither agree nor disagree（中立），
4. Agree（同意），
5. Strongly agree（强烈同意）。

Jason M. Bryer 开发了一个 R 包 **likert**，特别适合调查研究数据可视化，将研究对象的态度以直观有效的方式展示出来，内置多个数据集，其中表格 7.7 是一个数学焦虑量表调查的结果，调查数据来自统计课上的 20 个学生。

调查对象是 78 个来自不同学科的本科生，样本含有 36 个男性和 42 个女性，64% 的样本的年龄在 18 至 24 岁，36% 的样本年龄 25 岁及以上。更多数据背景信息 (Bai 等 2009)。

表格 7.7: 你对数学感到焦虑吗?

观点	强烈反对	反对	中立	同意	强烈同意
I find math interesting.	10	15	10	35	30
I get uptight during math tests.	10	20	20	25	25
I think that I will use math in the future.	0	0	20	25	55
Mind goes blank and I am unable to think clearly when doing my math test.	30	30	15	10	15
Math relates to my life.	5	20	10	40	25
I worry about my ability to solve math problems.	20	20	20	30	10
I get a sinking feeling when I try to do math problems.	35	10	15	35	5
I find math challenging.	5	10	15	45	25
Mathematics makes me feel nervous.	20	25	15	25	15

观点	强烈反对	反对	中立	同意	强烈同意
I would like to take more math classes.	20	25	30	20	5
Mathematics makes me feel uneasy.	25	15	20	25	15
Math is one of my favorite subjects.	35	15	25	20	5
I enjoy learning with mathematics.	15	25	30	20	10
Mathematics makes me feel confused.	15	20	15	35	15

相比于 **ggplot2** 绘制的普通条形图，图 7.35 有一些独特之处：对立型的渐变色表示两个不同方向的态度，左右两侧以中立态度为中间位置，非常形象，并且按照其中一个方向的态度数据排序，显得比较整齐有序，便于理解。

```
# 数据来自 likert 包
MathAnxiety <- readRDS(file = "data/MathAnxiety.rds")
# 宽转长格式
MathAnxiety_df <- reshape(data = MathAnxiety,
  varying = c("Strongly Disagree", "Disagree", "Neutral", "Agree", "Strongly Agree"),
  times = c("Strongly Disagree", "Disagree", "Neutral", "Agree", "Strongly Agree"),
  timevar = "Attitude", v.names = "Numbers",  idvar = "Item",
  new.row.names = 1:(5 * 14), direction = "long"
 )

MathAnxiety_df$Attitude <- factor(MathAnxiety_df$Attitude, levels = c(
  "Strongly Agree", "Agree", "Neutral", "Disagree", "Strongly Disagree"
), labels =  c(
  "强烈同意", "同意", "中立", "反对", "强烈反对"
), ordered = TRUE)

ggplot(data = MathAnxiety_df, aes(x = Numbers, y = Item)) +
  geom_col(aes(fill = Attitude), position = "fill") +
  scale_x_continuous(labels = scales::label_percent()) +
  scale_y_discrete(labels = scales::label_wrap(25)) +
  scale_fill_brewer(palette = "BrBG", direction = -1) +
  theme_classic() +
  guides(fill = guide_legend(reverse = TRUE)) +
  coord_cartesian(expand = FALSE) +
  labs(x = "占比", y = "问题", fill = "态度")
```

©黄湘云

图 7.35: 你喜欢数学吗

likert 包的函数 `likert()` 适合对聚合的调查数据绘图。

```
library(likert)
lmath <- likert(summary = MathAnxiety)
plot(lmath)
```

而 **ggstats** 包的函数 `gglikert()` 适合对明细的调查数据绘图。下面模拟一次调查收集到的数据，共计 150 人回答 6 个问题，每个问题都有 5 个候选项构成。

```
library(ggstats)
likert_levels <- c("强烈反对", "反对", "中立", "同意", "强烈同意")
set.seed(2023)
library(data.table)
df <- data.table(
  q1 = sample(likert_levels, 150, replace = TRUE),
  q2 = sample(likert_levels, 150, replace = TRUE, prob = 5:1),
  q3 = sample(likert_levels, 150, replace = TRUE, prob = 1:5),
  q4 = sample(likert_levels, 150, replace = TRUE, prob = 1:5),
  q5 = sample(c(likert_levels, NA), 150, replace = TRUE),
  q6 = sample(likert_levels, 150, replace = TRUE, prob = c(1, 0, 1, 1, 0))
)
fkt <- paste0("q", 1:6)
```

© 张俊

```
df[, (fkt) := lapply(.SD, factor, levels = likert_levels), .SDcols = fkt]
```

一个调查问卷共有 6 个题目，150 个人对 6 个问题的回答构成一个数据框 `df` 。

```
gglikert(df)
```

图 7.36: Likert 图

7.4 习题

1. 根据 Github 代码提交量数据制作日历图。

© 北京理工大学出版社

图 7.37: Github 打卡日历图

©黄湘云

第八章　统计图形

- 章节 8.1 探索、展示数据中隐含的分布信息，具体有箱线图、提琴图、直方图、密度图、岭线图和抖动图。
- 章节 8.2 探索、展示数据中隐含的相关信息，这种相关具有一般性，比如线性和非线性相关，位序关系、包含关系、依赖关系等，具体有散点图、气泡图、凹凸图、韦恩图、甘特图和网络图。
- 章节 8.3 探索、展示数据中隐含的不确定性，统计中描述不确定性有很多概念，比如置信区间、假设检验中的 P 值，统计模型中的预测值及其预测区间，模型残差隐含的分布，模型参数分量的边际分布及其效应，贝叶斯视角下的模型参数的后验分布。

8.1　描述分布

数据来自中国国家统计局发布的 2021 年统计年鉴，各省、直辖市和自治区分区域的性别比数据（部分）情况见表格 8.1 。

表格 8.1: 各省、直辖市和自治区分区域的性别比数据（部分）

地区	人口数/男	人口数/女	性别比（女 =100）	区域
北京	8937161	8814520	101.39	城市
天津	5610161	5322931	105.40	城市
河北	11010407	11119188	99.02	城市
山西	6588788	6608849	99.70	城市
内蒙古	4714495	4731924	99.63	城市
辽宁	12626419	12946058	97.53	城市

8.1.1　箱线图

```
library(ggplot2)
ggplot(data = province_sex_ratio, aes(x = `区域`, y = `性别比（女=100）`)) +
  geom_boxplot() +
  theme_classic()
```

统计学习基础 ©


```
library(lvplot)
ggplot(data = province_sex_ratio, aes(x = `区域`, y = `性别比（女=100）`)) +
  geom_lv() +
  theme_classic()

boxplot(`性别比（女=100）` ~ `区域` , data = province_sex_ratio)
```

箱线图的历史有 50 多年了，它的变体也有很多。lvplot 包绘制箱线图的变体 (McGill 和 Larsen 1978)

8.1.2 提琴图

```
ggplot(data = province_sex_ratio, aes(x = `区域`, y = `性别比（女=100）`)) +
  geom_violin(fill = "lightgray", draw_quantiles = c(0.25, 0.5, 0.75)) +
  theme_classic()

vioplot::vioplot(`性别比（女=100）` ~ `区域`,
  data = province_sex_ratio, col = "lightgray"
)

beanplot::beanplot(`性别比（女=100）` ~ `区域`,
  data = province_sex_ratio, col = "lightgray", log = "",
  xlab = "区域", ylab = "性别比（女=100）"
)
```

beanplot 包的名字是根据图形的外观取的，bean 即是豌豆，rug 用须线表示数据。

8.1.3 直方图

ggplot2 包绘制直方图的函数是 `geom_histogram()` ，而与之相关的函数 `geom_freqpoly()` 是绘制折线图，将直方图中每个柱子的顶点连接起来。

```
ggplot(data = province_sex_ratio, aes(x = `性别比（女=100）`, fill = `区域`)) +
  geom_histogram(binwidth = 5, color = "white", position = "stack") +
  theme_classic() +
  theme(legend.position = "inside", legend.position.inside = c(0.9, 0.8))

ggplot(data = province_sex_ratio, aes(x = `性别比（女=100）`, color = `区域`)) +
  geom_freqpoly(binwidth = 5, stat = "bin") +
  theme_classic() +
  theme(legend.position = "inside", legend.position.inside = c(0.9, 0.8))
```

© 黄湘云

(a) ggplot2 包

(b) lvplot 包

(c) Base R 包

图 8.1: 箱线图的几种绘制形式

© 黄湘云

(a) 函数 geom_violin()

(b) vioplot 包

(c) beanplot 包

图 8.2: 提琴图

© 黄湘云

(a) 函数 geom_histogram()　　(b) 函数 geom_freqpoly()

图 8.3: 直方图

8.1.4 密度图

ggplot2 包绘制密度图的函数是 geom_density()，图 8.4 展示分组密度曲线图

```
ggplot(data = province_sex_ratio, aes(x = `性别比（女=100）`)) +
  geom_density(aes(fill = `区域`), alpha = 0.5) +
  theme_classic()
```

图 8.4: 密度图

©黄湘云

8.1.4.1 堆积（条件）密度图

> **注意**
>
> Stacked density plots: if you want to create a stacked density plot, you probably want to 'count' (density * n) variable instead of the default density

堆积密度图正确的绘制方式是保护边际密度。

```
ggplot(data = province_sex_ratio, aes(x = `性别比（女=100）`, y = after_stat(density))) +
  geom_density(aes(fill = `区域`), position = "stack", alpha = 0.5) +
  theme_classic() +
  theme(legend.position = "inside", legend.position.inside = c(0.9, 0.8))
ggplot(data = province_sex_ratio, aes(x = `性别比（女=100）`, y = after_stat(density * n))) +
  geom_density(aes(fill = `区域`), position = "stack", alpha = 0.5) +
  theme_classic() +
  theme(legend.position = "inside", legend.position.inside = c(0.9, 0.8))
```

(a) 堆积密度图 `after_stat(density)`　　(b) 堆积密度图 `after_stat(density * n)`

图 8.5: 累积分布密度图

什么原因导致图 8.5 中两个子图看起来没什么差别呢？而换一组数据，就可以看出明显的差别。

```
ggplot(diamonds, aes(x = carat, y = after_stat(density), fill = cut)) +
  geom_density(position = "stack") +
  theme_classic() +
  theme(legend.position = "inside", legend.position.inside = c(0.8, 0.8))
ggplot(diamonds, aes(x = carat, y = after_stat(density * n), fill = cut)) +
  geom_density(position = "stack") +
  scale_y_continuous(
```

©李东风 统计学

```
    breaks = c(25000, 50000, 75000),
    labels = c("25K", "50K", "75K")) +
  theme_classic() +
  theme(legend.position = "inside", legend.position.inside = c(0.8, 0.8))
```

(a) 函数 `after_stat(density)` (b) 函数 `after_stat(density * n)`

图 8.6: 堆积密度图

8.1.4.2 联合密度图

```
state_x77 <- data.frame(state.x77,
  state_name = rownames(state.x77),
  state_region = state.region,
  check.names = FALSE
)
p1 <- ggplot(data = state_x77, aes(x = Income, y = `Life Exp`)) +
  geom_point() +
  geom_density_2d(aes(
    color = after_stat(level),
    alpha = after_stat(level)
  ), show.legend = F
  ) +
  scale_color_viridis_c(option = "inferno") +
  labs(
    x = "人均收入（美元）", y = "预期寿命（年）",
    title = "1977 年各州预期寿命与人均收入的关系",
    caption = "数据源：美国人口调查局"
  ) +
```

© 黄湘云


```
  theme_classic() +
  theme(
    panel.grid = element_line(colour = "gray92"),
    panel.grid.major = element_line(linewidth = rel(1.0)),
    panel.grid.minor = element_line(linewidth = rel(0.5))
  )
p1
```

图 8.7: 二维联合密度图

8.1.4.3 边际密度图

ggExtra 包 (Attali 和 Baker 2022) 添加边际密度曲线和边际直方图。

```
library(ggExtra)
ggMarginal(p1, type = "density")
ggMarginal(p1, type = "histogram")
```

© 黄湘云

图 8.8: 描述边际分布

8.1.4.4 填充密度图

ggplot2 包提供二维密度图层 `geom_density_2d_filled()` 绘制热力图，**ggdist** (Kay 2022) 进行了一些扩展。

```
ggplot(data = state_x77, aes(x = Income, y = `Life Exp`)) +
  geom_density_2d_filled(contour_var = "count") +
  theme_classic() +
  labs(
    x = "人均收入（美元）", y = "预期寿命（年）",
    title = "1977 年各州预期寿命与人均收入的关系",
    caption = "数据源：美国人口调查局"
  )
```

© 黄湘云

图 8.9: ggplot2 包绘制二维填充密度图

相比于 **ggplot2** 内置的二维核密度估计，**ggdensity** (Otto 和 Kahle 2023) 有一些优势，根据数据密度将目标区域划分，更加突出层次和边界。**gghdr** 包与 **ggdensity** 类似，展示 highest density regions (HDR)

```
library(ggdensity)
ggplot(data = state_x77, aes(x = Income, y = `Life Exp`)) +
  geom_hdr() +
  geom_point() +
  theme_classic() +
  labs(
    x = "人均收入（美元）", y = "预期寿命（年）",
    title = "1977 年各州预期寿命与人均收入的关系",
    caption = "数据源：美国人口调查局"
  )
```

©黄湘云

图 8.10: ggdensity 包绘制二维填充密度图

8.1.5 岭线图

叠嶂图，还有些其它名字，如峰峦图、岭线图等，详情参考统计之都主站《叠嶂图的前世今生》，主要用来描述数据的分布情况，在展示分布的对比上效果非常好。

图 8.11 设置窗宽为 1.5 个百分点

> 提示
>
> 除了中国国家统计年鉴，各省、自治区、直辖市及各级统计局每年都会发布一些统计年鉴、公告等数据，读者可以在此基础上继续收集更多数据，来分析诸多有意思的问题：
> 1. 城市、镇和乡村男女性别比呈现差异化分布的成因。
> 2. 城市、镇和乡村男女年龄构成。
> 3. 将上述问题从省级下钻到市、县级来分析。

8.1.6 抖动图

下面先用函数 `geom_point()` 绘制散点图展示原始数据，通过点的疏密程度暗示数据的分布。Base R 函数 `stripchart()` 可以实现类似的效果。当数据量比较大时，点相互覆盖比较严重，此时，抖动图比较适合用来展示原始数据。

©黄湘云

(a) 岭线图

(b) 岭线图和抖动图组合

(c) 岭线图和轴须图组合

图 8.11: 描述数据分布

©黄湘云

```
ggplot(data = province_sex_ratio, aes(x = `区域`, y = `性别比（女=100）`)) +
  geom_point() +
  theme_classic()

stripchart(
  `性别比（女=100）` ~ `区域`, vertical = TRUE, pch = 1,
  data = province_sex_ratio, xlab = "区域"
)

ggplot(data = province_sex_ratio, aes(x = `区域`, y = `性别比（女=100）`)) +
  geom_jitter(width = 0.25) +
  theme_classic()
```

Sidiropoulos 等 (2018) 提出一种新的方式描述数据的分布，集合抖动图和小提琴图的功能，在给定的分布界限内抖动。数据点受 violin 的曲线限制，蜂群图也是某种形式的抖动图，添加 violin 作为参考边界，与 sina 图是非常类似的。

```
library(ggforce)
ggplot(data = province_sex_ratio, aes(x = `区域`, y = `性别比（女=100）`)) +
  geom_sina() +
  theme_classic()

ggplot(data = province_sex_ratio, aes(x = `区域`, y = `性别比（女=100）`)) +
  geom_violin() +
  geom_sina() +
  theme_classic()

library(ggbeeswarm)
ggplot(data = province_sex_ratio, aes(x = `区域`, y = `性别比（女=100）`)) +
  geom_quasirandom() +
  theme_classic()

ggplot(data = province_sex_ratio, aes(x = `区域`, y = `性别比（女=100）`)) +
  geom_violin() +
  geom_quasirandom() +
  theme_classic()
```

函数 `geom_beeswarm()` 提供了另一种散点的组织方式，按一定的规则，而不是近似随机的方式组织

```
ggplot(data = province_sex_ratio, aes(x = `区域`, y = `性别比（女=100）`)) +
  geom_violin() +
  geom_beeswarm() +
```

©黄湘云

(a) 函数 geom_point()

(b) 函数 stripchart()

(c) 函数 geom_jitter()

图 8.12: 散点图

©黄湘云

(a) ggforce 包的函数 `geom_sina()`

(b) 函数 `geom_sina()` 叠加函数 `geom_violin()`

(c) ggbeeswarm 包的函数 `geom_quasirandom()`

(d) 函数 `geom_quasirandom()` 叠加函数 `geom_violin()`

图 8.13: 加强版的抖动图

© 基础统计


```
  theme_classic()
```

8.2 描述关系

8.2.1 散点图

散点图用以描述变量之间的关系，展示原始的数据，点的形态、大小、颜色等都可以随更多变量变化。

中国国家统计局 2021 年发布的统计年鉴，2020 年 31 个省、直辖市、自治区的抚养比、文盲率、人口数的关系。

表格 8.2: 2020 年各省、直辖市、自治区，总抚养比和文盲率相关数据（部分）

地区	人口数	15-64 岁	抚养比	15 岁及以上人口	文盲人口	文盲率
广东	126012510	91449628	37.79	102262628	1826344	1.79
山东	101527453	67100737	51.31	62464815	3308280	4.01
河南	99365519	62974661	57.79	76376565	2228594	2.92
江苏	84748016	58129537	45.79	71856068	2211291	3.08
四川	83674866	56036154	49.32	70203754	3330733	4.74
河北	74610235	49133330	51.85	59521267	1128423	1.90

其中，文盲人口是指 15 岁及以上不识字及识字很少人口，文盲率 = 文盲人口 / 15 岁及以上人口，抚养比 = (0-14 岁 + 65 岁及以上) / 15-64 岁人口数。

```
library(ggplot2)
ggplot(data = china_raise_illiteracy) +
  geom_point(aes(x = `总抚养比`, y = `文盲人口占15岁及以上人口的比重`)) +
  theme_classic() +
  labs(x = "抚养比 (%) ", y = "文盲率 (%) ")
```

©黄湘云

图 8.14: 文盲率与抚养比的关系

8.2.2 气泡图

气泡图在散点图的基础上，添加了散点大小的视觉维度，可以在图上多展示一列数据，下图 8.15 新增了人口数变量。此外，在气泡旁边添加地区名称，将气泡填充的颜色也映射给了人口数变量。

```
library(ggrepel)
library(scales)
ggplot(
  data = china_raise_illiteracy,
  aes(x = `总抚养比`, y = `文盲人口占15岁及以上人口的比重`)
) +
  geom_point(aes(size = `人口数`, color = `人口数`),
    alpha = 0.85, pch = 16,
    show.legend = c(color = FALSE, size = TRUE)
  ) +
  scale_size(labels = label_number(scale_cut = cut_short_scale())) +
  scale_color_viridis_c(option = "C") +
  geom_text_repel(
    aes(label = `地区`), size = 3, max.overlaps = 50,
    segment.colour = "gray", seed = 2022, show.legend = FALSE
```

©黄湘云


```
  ) +
  coord_cartesian(xlim = c(30, 60), ylim = c(0, 10.5), expand = FALSE) +
  theme_classic() +
  labs(x = "抚养比 (%) ", y = "文盲率 (%) ")
```

图 8.15: 文盲率和抚养比数据

8.2.3 凹凸图

凹凸图描述位置排序关系随时间的变化，比如前年、去年和今年各省的 GDP 排序变化，春节各旅游景点的人流量变化。**ggbump** 包专门用来绘制凹凸图，如图 8.16 所示，展示

```
library(ggbump)
# 位置排序变化
df <- data.frame(
  country = c(
    "印度", "印度", "印度", "瑞典",
    "瑞典", "瑞典", "德国", "德国",
    "德国", "芬兰", "芬兰", "芬兰"
  ),
  year = c(
```

R语言实战

```
    2018, 2019, 2020, 2018, 2019, 2020,
    2018, 2019, 2020, 2018, 2019, 2020
  ),
  value = c(
    492, 246, 246, 369, 123, 492,
    246, 369, 123, 123, 492, 369
  )
)

library(data.table)
df <- as.data.table(df)
df[, rank := rank(value, ties.method = "random"), by = "year"]

ggplot(df, aes(year, rank, color = country)) +
  geom_point(size = 7) +
  geom_text(data = df[df$year == min(df$year), ],
            aes(x = year - .1, label = country), size = 5, hjust = 1) +
  geom_text(data = df[df$year == max(df$year), ],
            aes(x = year + .1, label = country), size = 5, hjust = 0) +
  geom_bump(linewidth = 2, smooth = 8) +
  scale_x_continuous(limits = c(2017.6, 2020.4), breaks = seq(2018, 2020, 1)) +
  theme_minimal(base_size = 14) +
  theme(legend.position = "none",
        panel.grid.major = element_blank()) +
  labs(x = NULL, y = "排名") +
  scale_y_reverse() +
  coord_fixed(ratio = 0.5)
```

© 黄湘云

图 8.16: 凹凸图

8.2.4 韦恩图

韦恩图描述集合、群体的交叉关系，整体和部分的包含关系，**ggVennDiagram** 包展示 A、B、C 三个集合的交叉关系，如图 8.17 所示

```
x <- list(A = 1:5, B = 2:7, C = 5:10)
ggVennDiagram::ggVennDiagram(x) +
  scale_fill_gradient(low = "#F4FAFE", high = "#4981BF")
```

©黄湘云

图 8.17: A、B、C 三个集合的交叉关系

8.2.5 网络图

tidygraph 包基于 igraph 包操作图数据，计算网络图中节点重要性，**ggraph**包基于 ggplot2 包可视化网络关系。

```
library(ggraph)
data("highschool")
str(highschool)
```

```
#> 'data.frame':    506 obs. of  3 variables:
#>  $ from: num  1 1 1 1 1 2 2 3 3 4 ...
#>  $ to  : num  14 15 21 54 55 21 22 9 15 5 ...
#>  $ year: num  1957 1957 1957 1957 1957 ...
```

highschool 是一个数据框类型的数据，记录了 1957 年和 1958 年一些高中男生之间的关系，在数据集中，这些男生被编码成数字 1-71。

```
highschool[highschool$from == 1, ]
```

```
#>     from to year
#> 1      1 14 1957
```

©黄湘云


```
#> 2       1 15 1957
#> 3       1 21 1957
#> 4       1 54 1957
#> 5       1 55 1957
#> 244     1 15 1958
#> 245     1 21 1958
#> 246     1 22 1958
```

1 号男生在 1957 年与 14、15、21、54、55 男生关系密切，到了 1958 年，他与 15、21、22 关系比较密切。tidygraph 包在 igraph 的基础上，可以对图数据进行操作，下面先将数据框转化为图，然后计算中心度，作为高中生的受欢迎程度。

```
graph <- tidygraph::as_tbl_graph(highschool, directed = TRUE) |>
  dplyr::mutate(Popularity = tidygraph::centrality_degree(mode = 'in'))
graph
```

```
#> # A tbl_graph: 70 nodes and 506 edges
#> #
#> # A directed multigraph with 1 component
#> #
#> # Node Data: 70 x 1 (active)
#>    Popularity
#>         <dbl>
#>  1          2
#>  2          0
#>  3          0
#>  4          4
#>  5          5
#>  6          2
#>  7          2
#>  8          3
#>  9          4
#> 10          4
#> # i 60 more rows
#> #
#> # Edge Data: 506 x 3
#>    from    to  year
#>   <int> <int> <dbl>
#> 1     1    13  1957
#> 2     1    14  1957
#> 3     1    20  1957
```

```
#> # i 503 more rows
ggraph(graph, layout = "kk") +
  geom_edge_fan(aes(alpha = after_stat(index)), show.legend = FALSE) +
  geom_node_point(aes(size = Popularity)) +
  facet_edges(~year) +
  theme_graph(base_family = "sans", foreground = "steelblue", fg_text_colour = "white")
```

图 8.18: 高中男生间关系的变化

8.3 描述不确定性

统计是一门研究不确定性的学科，由不确定性引出许多的基本概念，比如用置信区间描述点估计的不确定性，用覆盖概率描述区间估计方法的优劣。下面以二项分布参数的点估计与区间估计为例，通过可视化图形介绍这一系列统计概念。就点估计来说，描述不确定性可以用标准差、置信区间。

ggdist 包可视化分布和不确定性 (Kay 2022)

- Michael Friendly 2021 年的课程 Psychology of Data Visualization
- Claus O. Wilke 2023 年的课程 SDS 375 Schedule Spring 2023

8.3.1 置信区间

©黄湘云

表格 8.3: 二项分布的分布列

0	1	2	$\cdots$	n
p_0	p_1	p_2	$\cdots$	p_n

二项分布 Binomial(n,p) 的参数 p 的精确区间估计如下：

$$\left(\mathrm{Beta}(\frac{\alpha}{2};x,n-x+1),\mathrm{Beta}(1-\frac{\alpha}{2};x+1,n-x)\right) \tag{8.1}$$

其中，x 表示成功次数，n 表示实验次数，Beta$(p;v,w)$ 表示形状参数为 v 和 w 的 Beta 贝塔分布的 p 分位数，参数 p 的置信区间的上下限 P_L,P_U 满足

$$\begin{aligned}\frac{\Gamma(n+1)}{\Gamma(x)\Gamma(n-x+1)}\int_0^{P_L} t^{x-1}(1-t)^{n-x}\mathrm{dt}=\frac{\alpha}{2}\\ \frac{\Gamma(n+1)}{\Gamma(x+1)\Gamma(n-x)}\int_0^{P_U} t^{x}(1-t)^{n-x-1}\mathrm{dt}=1-\frac{\alpha}{2}\end{aligned}$$

p_x 表示二项分布 Binomial(n,p) 第 x 项的概率，x 的取值为 $0,1,\cdots,n$

$$p_x=\binom{n}{x}p^x(1-p)^{n-x},\quad x=0,1,\cdots,n$$

二项分布的累积分布函数和 S_k 表示前 k 项概率之和

$$S_k=\sum_{x=0}^{k}p_x$$

S_k 服从形状参数为 $n-k,k+1$ 的贝塔分布 $I_x(a,b)$，对应于 R 函数 `pbeta(x,a,b)`。S_k 看作贝塔分布的随机变量 X

$$\begin{aligned}B_x(a,b)&=\int_0^x t^{a-1}(1-t)^{b-1}\mathrm{dt}\\ I_x(a,b)&=\frac{B_x(a,b)}{B(a,b)},\quad B(a,b)=B_1(a,b)\end{aligned}$$

考虑二项总体的参数 $p=0.7$，重复模拟 50 次，每次模拟获得的比例 $\hat{p}$ 及其置信区间，如图 8.19 所示，置信区间覆盖真值的情况用不同颜色表示，覆盖用 `TRUE` 表示，没有覆盖用 `FALSE` 表示

```
library(ggplot2)
# Clopper and Pearson (1934)
# 与 binom.test() 计算结果一致
clopper_pearson <- function(p = 0.1, n = 10, nsim = 100) {
```

© 王建平

```
  set.seed(2022)
  nd <- rbinom(nsim, prob = p, size = n)
  ll <- qbeta(p = 0.05 / 2, shape1 = nd, shape2 = n - nd + 1)
  ul <- qbeta(p = 1 - 0.05 / 2, shape1 = nd + 1, shape2 = n - nd)
  data.frame(nd = nd, ll = ll, ul = ul, cover = ul > p & ll < p)
}
# 二项分布的参数 p = 0.7
dat <- clopper_pearson(p = 0.7, n = 10, nsim = 50)
# 二项分布的参数的置信区间覆盖真值的情况
ggplot(data = dat, aes(x = 1:50, y = nd / 10, colour = cover)) +
  geom_hline(yintercept = 0.7, lty = 2, linewidth = 1.2, color = "gray") +
  geom_pointrange(aes(ymin = ll, ymax = ul)) +
  theme_classic() +
  labs(x = "第几次模拟", y = "置信区间上下限", color = "覆盖")
```

图 8.19: Clopper-Pearson 置信区间

8.3.2 假设检验

假设检验的目的是做决策，决策是有风险的，是可能发生错误的，为了控制犯第一类错误的可能性，我们用 P 值描述检验统计假设的不确定性，用功效描述检验方法的优劣。对同一个统计假设，同一组数据，不同的检验方法有不同的 P 值，本质是检验方法的功效不同。

© 基础统计

ggpval 在图上添加检验的 P 值结果，**ggsignif** (Constantin 和 Patil 2021) 在图上添加显著性注释。**ggstatsplot** (Patil 2021) 可视化统计检验、模型的结果，描述功效变化。**ggpubr** 制作出版级统计图形，两样本均值的比较。

```
with(
  aggregate(
    data = PlantGrowth, weight ~ group,
    FUN = function(x) c(dist_mean = mean(x), dist_sd = sd(x))
  ),
  cbind.data.frame(weight, group)
) |>
  ggplot(aes(x = group, y = dist_mean)) +
  geom_col(
    position = position_dodge(0.4), width = 0.4, fill = "gray"
  ) +
  geom_errorbar(aes(
    ymin = dist_mean - dist_sd,
    ymax = dist_mean + dist_sd
  ),
  position = position_dodge(0.4), width = 0.2
  ) +
  theme_classic() +
  labs(x = "组别", y = "植物干重")
```

图 8.20: 植物生长

> **注释**
>
> R 3.5.0 以后，函数 `aggregate()` 的参数 `drop` 默认值为 `TRUE`，表示扔掉未用来分组的变量，聚合返回的是一个矩阵类型的数据对象。

单因素方差分析 `oneway.test()` 检验各组的方差是否相等。

```
oneway.test(data = PlantGrowth, weight ~ group)
```

```
#>
#>  One-way analysis of means (not assuming equal variances)
#>
#> data:  weight and group
#> F = 5.181, num df = 2.000, denom df = 17.128, p-value = 0.01739
```

结果显示方差不全部相等，因此，采用函数 `t.test(var.equal = FALSE)` 来检验数据。图 8.21 展示假设检验的结果，分别标记了 ctrl 与 trt1、trt1 与 trt2 两组 t 检验的结果。

```
library(ggsignif)
ggplot(data = PlantGrowth, aes(x = group, y = weight)) +
  geom_boxplot(width = 0.25) +
  geom_jitter(width = 0.15) +
  geom_signif(comparisons = list(c("ctrl", "trt1"), c("trt1", "trt2")),
              map_signif_level = function(p) sprintf("p = %.2g", p),
              textsize = 6, test = "t.test") +
  theme_classic() +
  coord_cartesian(clip = "off")
```

©黄湘云

图 8.21: 展示假设检验的结果

为了了解其中的原理，下面分别使用函数 `t.test()` 检验数据，结果给出的 P 值与上图 8.21 完全一样。

```
t.test(data = PlantGrowth, weight ~ group, subset = group %in% c("ctrl", "trt1"))
```

```
#>
#>  Welch Two Sample t-test
#>
#> data:  weight by group
#> t = 1.1913, df = 16.524, p-value = 0.2504
#> alternative hypothesis: true difference in means between group ctrl and group trt1 is not equal to 0
#> 95 percent confidence interval:
#>  -0.2875162  1.0295162
#> sample estimates:
#> mean in group ctrl mean in group trt1
#>              5.032              4.661
```

```
t.test(data = PlantGrowth, weight ~ group, subset = group %in% c("trt1", "trt2"))
```

```
#>
#>  Welch Two Sample t-test
#>
#> data:  weight by group
```

```
#> t = -3.0101, df = 14.104, p-value = 0.009298
#> alternative hypothesis: true difference in means between group trt1 and group trt2 is not equal to 0
#> 95 percent confidence interval:
#>  -1.4809144 -0.2490856
#> sample estimates:
#> mean in group trt1 mean in group trt2
#>              4.661              5.526
```

8.3.3 模型预测

描述模型预测的不确定性，预测的方差、预测区间。线性回归来说，回归线及置信带。代码提交量的趋势拟合

```
svn_trunk_log <- readRDS(file = "data/svn-trunk-log-2022.rds")
svn_trunk_log$year <- as.integer(format(svn_trunk_log$stamp, "%Y"))
trunk_year <- aggregate(data = svn_trunk_log, revision ~ year, FUN = length)
ggplot(data = trunk_year, aes(x = year, y = revision)) +
  geom_point() +
  geom_smooth(aes(color = "LOESS", fill = "LOESS"),
    method = "loess", formula = "y~x",
    method.args = list(
      span = 0.75, degree = 2, family = "symmetric",
      control = loess.control(surface = "direct", iterations = 4)
    ), data = subset(trunk_year, year != c(1997, 2022))
  ) +
  geom_smooth(aes(color = "GAM", fill = "GAM"),
    formula = y ~ s(x, k = 12), method = "gam", se = TRUE,
    data = subset(trunk_year, year != c(1997, 2022))
  ) +
  geom_smooth(aes(color = "Cubic Spline", fill = "Cubic Spline"),
                method = "lm", formula = y ~ splines::bs(x, 3), se = T,
              data = subset(trunk_year, year != c(1997, 2022))) +
  scale_color_brewer(name = "模型", palette = "Set1") +
  scale_fill_brewer(name = "模型", palette = "Set1") +
  theme_classic() +
  theme(panel.grid.major.y = element_line(colour = "gray90")) +
  labs(x = "年份", y = "提交量")
```

©黄湘云

图 8.22: 趋势拟合、预测和推断

8.3.4 模型诊断

所有模型都是错误的，但有些是有用的。

— 乔治 · 博克斯

描述模型的敏感性，数据中存在的离群值，变量之间的多重共线性等。引入 Cook 距离、杠杆值、VIF 等来诊断模型。

```
# 准备数据
state_x77 <- data.frame(state.x77,
  state_name = rownames(state.x77),
  state_region = state.region,
  state_abb = state.abb,
  check.names = FALSE
)
# 线性模型拟合
fit <- lm(`Life Exp` ~ Income + Murder, data = state_x77)
# 模型诊断图
library(ggfortify)
```

```
autoplot(fit, which = 1:6, label.size = 3)
```

图 8.23: 线性模型的诊断图

对于复杂的统计模型，比如混合效应模型的诊断，**ggPMX** 包。

© 基础统计

8.3.5 边际效应

继续 state_x77 数据集，以预期寿命（1969-1971 年统计）为因变量，Income 人均收入（1974 年）和 Murder 谋杀和非过失杀人率（单位：十万分之一，1976 年统计）为自变量，建立线性模型如下：

$$\text{Life Exp} = \alpha + \beta_1 \text{Income} + \beta_2 \text{Murder} + \epsilon \tag{8.2}$$

在 R 语言中，可以用函数 `lm()` 拟合上述模型，

```
fit <- lm(`Life Exp` ~ Income + Murder, data = state_x77)
```

模型拟合结果输出如下：

```
summary(fit)
```

```
#>
#> Call:
#> lm(formula = `Life Exp` ~ Income + Murder, data = state_x77)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -1.48301 -0.62099 -0.01714  0.47768  2.20831
#>
#> Coefficients:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 71.2255815  0.9673952  73.626  < 2e-16 ***
#> Income       0.0003705  0.0001973   1.878   0.0666 .
#> Murder      -0.2697594  0.0328408  -8.214 1.22e-10 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.8259 on 47 degrees of freedom
#> Multiple R-squared:  0.637,  Adjusted R-squared:  0.6215
#> F-statistic: 41.23 on 2 and 47 DF,  p-value: 4.561e-11
```

ggeffects 描述单个自变量的作用，人均收入对预期寿命的边际效应

```
library(ggeffects)
income_pred <- ggpredict(fit, terms = "Income")
ggplot(income_pred, aes(x, predicted)) +
  geom_line() +
  geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = 0.1) +
  theme_classic() +
  labs(x = "人均收入", y = "预期寿命")
```

© 黄湘云

图 8.24: 边际效应

© 黄湘云

第九章　lattice 入门

> If you imagine that this pen is Trellis, then Lattice is not this pen.
>
> — Paul Murrell [1]

lattice 最初是受到商业统计软件 S/S-Plus 中的 Trellis 组件启发，打算在 R 软件中重新实现一套新的绘图系统，在使用接口上保持与 Trellis 兼容，Trellis 使用文档也同样适用于 **lattice**。

本章主要介绍 **lattice** 包 (Sarkar 2008) 及其相关的 **latticeExtra** 包。

```
library(lattice)
library(latticeExtra)
library(splines)
library(nlme)
library(mgcv)
library(maps)
library(sf)
library(RColorBrewer)
```

9.1　分组散点图

函数 `xyplot()` 在 lattice 包中非常具有代表性，掌握此函数的作图规律，其它函数学起来也就不难了。分组散点图是一个非常常见的、用来描述变量之间关系的图形，下面就以绘制一个分组散点图来介绍函数 `xyplot()` 的用法。

```
library(lattice)
xyplot(
  x = Sepal.Length ~ Petal.Length, groups = Species, scales = "free",
  data = iris, grid = TRUE, xlab = "萼片长度", ylab = "花瓣长度",
  auto.key = list(space = "top", columns = 3)
)
```

[1] Paul 在 DSC 2001 大会上的幻灯片。

©黄湘云

图 9.1: 分组散点图

- 参数 x 需要传递 R 语言中的表达式，这是一种被广泛使用的公式语法，示例中为 Sepal.Length ~ Petal.Length ，表示横坐标为 Petal.Length，纵坐标为 Sepal.Length 。
- 参数 groups 指定分组变量，此处为 Species 变量，表示鸢尾花种类。
- 参数 scales 设置坐标轴刻度，scales = "free" 表示去掉边框上面和右面的刻度线。
- 参数 data 指定绘图数据，此处为 iris 数据集。
- 参数 grid 控制是否添加背景网格线，此处为 TRUE 表示添加背景网格线。
- 参数 xlab 和参数 ylab 分别指定横、纵坐标轴标签。
- 参数 auto.key 设置图例，示例中设置 space = "top" 将图例置于图形上方，设置 columns = 3 使条目排成 3 列，此外，设置 reverse.rows = TRUE 还可以使图例中的条目顺序反向。

除了通过 space 参数设置图例的位置，还可以通过坐标设置图例的位置，比如下图 9.2b 中设置图例的位置坐标为 x = 1, y = 0 使得图例位于图的右下角。图例坐标的参考点是原点 x = 0, y = 0 就是左下角的位置，而右上角的位置为 x = 1, y = 1 。

```
xyplot(
  Sepal.Length ~ Petal.Length, groups = Species, data = iris,
  scales = "free", grid = TRUE, xlab = "萼片长度", ylab = "花瓣长度",
  auto.key = list(space = "right", columns = 1)
)
xyplot(
  Sepal.Length ~ Petal.Length, groups = Species, data = iris,
  scales = "free", grid = TRUE, xlab = "萼片长度", ylab = "花瓣长度",
```

©黄湘云


```
  auto.key = list(corner = c(1, 0))
)
```

除了上面介绍的几个参数，还有许多其它参数，其中一部分会在后续介绍其它种类的图形时顺带介绍，剩余的部分请感兴趣的读者查看函数 `xyplot()` 的帮助文档。

9.2 图形参数

类似 Base R 绘图系统中的图形参数设置函数 `par()` 和 **ggplot2** 包中的主题设置函数 `theme()`，**lattice** 包也有图形参数设置函数 `trellis.par.set()` ，而图形参数查询函数为 `trellis.par.get()` 。可设置的图形参数非常多，仅常用的也不少。首先来看看有哪些图形参数可以设置。

```
tp <- trellis.par.get()
names(tp)
```

```
#>  [1] "grid.pars"         "fontsize"          "background"
#>  [4] "panel.background"  "clip"              "add.line"
#>  [7] "add.text"          "plot.polygon"      "box.dot"
#> [10] "box.rectangle"     "box.umbrella"      "dot.line"
#> [13] "dot.symbol"        "plot.line"         "plot.symbol"
#> [16] "reference.line"    "strip.background"  "strip.shingle"
#> [19] "strip.border"      "superpose.line"    "superpose.symbol"
#> [22] "superpose.polygon" "regions"           "shade.colors"
#> [25] "axis.line"         "axis.text"         "axis.components"
#> [28] "layout.heights"    "layout.widths"     "box.3d"
#> [31] "par.title.text"    "par.xlab.text"     "par.ylab.text"
#> [34] "par.zlab.text"     "par.main.text"     "par.sub.text"
```

可以看到，图形参数着实非常多，知道了这么多图形参数，而每个参数又有哪些选项可取呢？不忙，再看看图形参数的结构，比如 `superpose.symbol` 。

```
tp$superpose.symbol
```

```
#> $alpha
#> [1] 1 1 1 1 1 1 1
#>
#> $cex
#> [1] 0.8 0.8 0.8 0.8 0.8 0.8 0.8
#>
#> $col
#>          blue        orange   bluishgreen    vermillion       skyblue
#>     "#0072B2"     "#E69F00"     "#009E73"     "#D55E00"     "#56B4E9"
#>        yellow reddishpurple
```

©黄湘云

(a) 图例位于图右侧

(b) 图例位于内内部

图 9.2: 调整图例位置

© 黄湘云


```
#>     "#F0E442"     "#CC79A7"
#>
#> $fill
#> [1] "#CCFFFF" "#FFCCFF" "#CCFFCC" "#FFE5CC" "#CCE6FF" "#FFFFCC" "#FFCCCC"
#>
#> $font
#> [1] 1 1 1 1 1 1 1
#>
#> $pch
#> [1] 1 1 1 1 1 1 1
```

这是一个列表，有 6 个元素，每个元素设置符号的不同属性，依次是透明度 `alpha`、大小 `cex`、颜色 `col`、填充色 `fill`、字型 `font` 和类型 `pch` ，这些属性的含义与函数 `par()` 是一致的。下图 9.3 展示所有的常用图形参数及其可设置的选项。

现在，知道了图形设置参数及其结构，还需要知道它们究竟在绘图时起什么作用，也就是说它们控制图形中的哪部分元素及效果。下图 9.4 展示 **lattice** 包图形参数效果。由图可知，图形参数 `superpose.symbol` 是控制散点图中的点，点可以是普通的点，也可以是任意的字母符号。

在之前的图 9.1 的基础上，设置 `type = c("p", "r")` 添加回归线。通过图形参数 `par.settings` 设置各类绘图元素的符号类型和大小，该参数接受一个列表类型的数据，列表的元素还是列表，列表的层层嵌套实现图中元素的精细控制。列表元素 `superpose.symbol` 控制点的符号，`pch = 16` 设置为 16，相比于默认的点要大一号。列表元素 `superpose.line` 控制线，`lwd = 2` 设置宽度为 2，比默认的宽度大一倍，`lty = 3` 设置线的类型为 3，表示虚线。通过参数 `auto.key` 设置图例位置，图例位于图形上方，图例中的条目排成 3 列。

```
xyplot(
  Sepal.Length ~ Petal.Length, groups = Species, data = iris,
  scales = "free", grid = TRUE, type = c("p", "r"),
  xlab = "萼片长度", ylab = "花瓣长度",
  auto.key = list(columns = 3, space = "top"),
  par.settings = list(
    superpose.symbol = list(pch = 16),
    superpose.line = list(lwd = 2, lty = 3)
  )
)
```

© 黄湘云

图 9.3: 常用图形参数

© 黄湘云

图 9.4: 图形参数效果预览

© 黄湘云

图 9.5: 调整点、线、图例元素

latticeExtra 包有两个函数专门用来设置图形风格，分别是 `theEconomist.theme()` 和 `ggplot2like()`，这两个主题函数提供一系列预设的图形参数，前者来自《经济学人》杂志的图形主题，后者来自 **ggplot2** 包的默认绘图主题。

```
library(latticeExtra)
xyplot(
  Sepal.Length ~ Petal.Length, groups = Species, data = iris,
  scales = "free", grid = TRUE, xlab = "萼片长度", ylab = "花瓣长度",
  auto.key = list(space = "top", columns = 3),
  par.settings = ggplot2like()
)
xyplot(
  Sepal.Length ~ Petal.Length, groups = Species, data = iris,
  scales = "free", grid = TRUE, xlab = "萼片长度", ylab = "花瓣长度",
  auto.key = list(space = "top", columns = 3),
  par.settings = theEconomist.theme(with.bg = TRUE, box = "transparent")
)
```

© 黄湘云

(a) ggplot2 包默认的绘图主题

(b) 《经济学人》杂志的绘图主题

图 9.6: latticeExtra 内置的两个主题

9.3 常见图形

9.3.1 分组柱形图

本节所用数据集 Insurance 来自 **MASS** 包，记录一家保险公司面临风险的投保人数量，以及在 1973 年第 3 季度他们提出汽车理赔的数量。数据类型、各个变量的类型及部分预览数据如下：

```
data(Insurance, package = "MASS")
str(Insurance)
```

```
#> 'data.frame':    64 obs. of  5 variables:
#>  $ District: Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
#>  $ Group   : Ord.factor w/ 4 levels "<1l"<"1-1.5l"<..: 1 1 1 1 2 2 2 2 3 3 ...
#>  $ Age     : Ord.factor w/ 4 levels "<25"<"25-29"<..: 1 2 3 4 1 2 3 4 1 2 ...
#>  $ Holders : int  197 264 246 1680 284 536 696 3582 133 286 ...
#>  $ Claims  : int  38 35 20 156 63 84 89 400 19 52 ...
```

其中，District 表示投保人居住的地区，因子型变量。Group 汽车按油箱大小分组的变量，有序的因子型变量。Age 投保人的年龄段标签，有序的因子型变量。Holders 投保人数量，整型变量。Claims 理赔数量，整型变量。下图 9.7 先按投保人的汽车类型分面，再按投保人所在地区分组，展示理赔频度与投保人年龄的关系。

```
barchart(
  Claims / Holders ~ Age | Group, groups = District,
  data = Insurance, xlab = "年龄段", ylab = "理赔频度",
  auto.key = list(space = "top", columns = 4, title = "地区", cex.title = 1)
)
```

© 黄湘云

图 9.7: 分组柱形图

函数 barchart() 中的公式 Claims / Holders ~ Age | Group ，斜杠 / 表示除法，波浪线 ~ 表示响应变量与自变量的分界，竖线 | 表示分面。

9.3.2 分组箱线图

```
bwplot(Petal.Length ~ Species, data = iris, scales = "free",
  xlab = "鸢尾花种类", ylab = "花瓣长度")
```

© 黄湘云

图 9.8: 分组箱线图

9.3.3 经验分布图

用阶梯图表示累积经验分布图，纵轴表示累积概率，不同种类的鸢尾花，花瓣长度的分布明显不同。根据 Glivenko–Cantelli 定理，经验分布函数以概率 1 收敛至累积分布函数。

```
library(latticeExtra)
ecdfplot(~ Petal.Length | Species, data = iris, scales = "free",
         xlab = "花瓣长度", ylab = "累积概率")
```

©黄湘云

图 9.9: 经验分布图

9.3.4 回归曲线图

- **splines** 自然立方样条 `ns()`
- **mgcv** 广义可加模型 `s()`

图 9.10: 调色板

图 9.11 中用不同的回归模型拟合数据中的趋势。1920s 汽车行驶距离和速度的关系图。函数 `panel.smoother()` 来自 **latticeExtra** 包

```
library(splines)
library(nlme)
library(mgcv)
xyplot(dist ~ speed,
  data = cars, scales = "free", xlab = "速度", ylab = "距离",
  panel = function(x, y, ...) {
    panel.xyplot(x, y, ...)
    panel.smoother(y ~ x,
      col.line = "#EA4335", method = "lm", ...
    )
  }
)
xyplot(dist ~ speed,
  data = cars, scales = "free", xlab = "速度", ylab = "距离",
  panel = function(x, y, ...) {
    panel.xyplot(x, y, ...)
    panel.smoother(y ~ x,
      col.line = "#4285f4", method = "loess", span = 0.9, ...
    )
  }
)
xyplot(dist ~ speed,
  data = cars, scales = "free", xlab = "速度", ylab = "距离",
  panel = function(x, y, ...) {
    panel.xyplot(x, y, ...)
    panel.smoother(y ~ ns(x, 5),
      col.line = "#34A853", method = "lm", ...
    )
  }
)
xyplot(dist ~ speed,
  data = cars, scales = "free", xlab = "速度", ylab = "距离",
  panel = function(x, y, ...) {
    panel.xyplot(x, y, ...)
    panel.smoother(y ~ s(x),
      col.line = "#FBBC05", method = "gam", ...
    )
  }
)
```

© 黄湘云

(a) 线性回归

(b) 局部多项式回归

(c) 自然样条回归

(d) 广义可加回归

图 9.11: 回归曲线图

©谢益辉

9.3.5 置信区间图

各个郡县每 10 万人当中因癌症死亡的人数。USCancerRates 数据集来自 **latticeExtra** 包，记录各个郡县的癌症死亡率及其置信区间，下图展示新泽西州各个郡县的癌症死亡率及其置信区间。

```
segplot(reorder(county, rate.male) ~ LCL95.male + UCL95.male,
  data = subset(USCancerRates, state == "New Jersey"),
  draw.bands = FALSE, centers = rate.male,
  scales = list(x = list(alternating = 1, tck = c(1, 0))),
  xlab = "癌症死亡率", ylab = "郡县"
)
```

图 9.12: 置信区间图

9.3.6 置信椭圆图

latticeExtra 包的函数 panel.ellipse() 可以绘制置信椭圆。二维数据，置信水平为 0.95 时，置信椭圆。

```
xyplot(Sepal.Length ~ Petal.Length,
  groups = Species, data = iris, scales = "free",
```

©黄湘云

```
  xlab = "萼片长度", ylab = "花瓣长度",
  par.settings = list(
    superpose.symbol = list(pch = 16),
    superpose.line = list(lwd = 2, lty = 3)
  ),
  panel = function(x, y, ...) {
    panel.xyplot(x, y, ...)
    panel.ellipse(x, y, level = 0.85, ...)
  },
  auto.key = list(space = "top", columns = 3)
)
```

图 9.13: 分组置信椭圆图

9.3.7 切片水平图

按照深度降序排列，根据震级 mag 划分 4 个区间，每个区间内数据点的数量比较平衡，相邻区间之间有重叠部分。对数据进行切片，观察连续的切片数据，增加一个维度。

对震深排序的目的是让数据点按照一定的顺序绘制在图上，数据点相距较近容易互相覆盖。使得在二维平面上，通过对数据点的染色，也能体现地震深度在空间中的层次变化。

不同的震级下，地震深度在空间中的变化是一致的。

```
# 震级区间
```

```
quakes$Magnitude <- equal.count(quakes$mag, number = 4)
# 震深
depth.ord <- rev(order(quakes$depth))
quakes.ordered <- quakes[depth.ord, ]
```

```
Intervals:
   min  max count
1 3.95 4.55   484
2 4.25 4.75   492
3 4.45 4.95   425
4 4.65 6.45   415

Overlap between adjacent intervals:
[1] 293 306 217
```

函数 `equal.count()` 内部调用函数 `co.intervals()` ，还有两个参数 `number` 和 `overlap`。如果要没有重叠的话，得设置 `overlap = 0` 。

```
quakes$Magnitude <- equal.count(quakes$mag, number = 4, overlap = 0)
```

```
levelplot(depth ~ long + lat | Magnitude,
  data = quakes.ordered, scales = "free",
  panel = panel.levelplot.points,
  prepanel = prepanel.default.xyplot,
  type = c("p", "g"), layout = c(2, 2)
)
```

©黄湘云

图 9.14: 分面水平图

9.3.8 三维散点图

lattice 包的函数 `cloud()` 三维散点图

```
cloud(Sepal.Length ~ Sepal.Width + Petal.Length,
  groups = Species, data = iris,
  # 去掉方向箭头
  scales = list(arrows = FALSE, col = "black"),
  xlab = list("萼片宽度", rot = 30),
  ylab = list("花瓣长度", rot = -35),
  zlab = list("萼片长度", rot = 90),
  # 减少三维图形的边空
  lattice.options = list(
```

© 黄湘云

```
    layout.widths = list(
      left.padding = list(x = -0.5, units = "inches"),
      right.padding = list(x = -1.0, units = "inches")
    ),
    layout.heights = list(
      bottom.padding = list(x = -1.5, units = "inches"),
      top.padding = list(x = -1.5, units = "inches")
    )
  ),
  par.settings = list(
    # 点的类型
    superpose.symbol = list(pch = 16),
    # 去掉外框线
    axis.line = list(col = "transparent")
  )
)
```

图 9.15: 三维散点图

下面是一个示例，自定义面板函数 `panel.3d.cloud` 。

©黄湘云

> **注释**
>
> 图底部的网格待改进，生成网格线的代码太死板。

```r
# 加载数据
rongelap <- readRDS(file = "data/rongelap.rds")
rongelap_coastline <- readRDS(file = "data/rongelap_coastline.rds")

library(lattice)
# 参考 lattice 书籍的图 6.5 的绘图代码
panel.3dcoastline <- function(..., rot.mat, distance, xlim, ylim, zlim,
                              xlim.scaled, ylim.scaled, zlim.scaled) {
  scale.vals <- function(x, original, scaled) {
    scaled[1] + (x - original[1]) * diff(scaled) / diff(original)
  }
  scaled.map <- rbind(
    scale.vals(rongelap_coastline$cX, xlim, xlim.scaled),
    scale.vals(rongelap_coastline$cY, ylim, ylim.scaled),
    zlim.scaled[1]
  )
  m <- ltransform3dto3d(scaled.map, rot.mat, distance)
  panel.lines(m[1, ], m[2, ], col = "black")
}

rongelap_grid_line <- rbind.data.frame(
  data.frame(x = 1000 * -6:0, y = -3500),
  data.frame(x = 1000 * 0:-6, y = -3000),
  data.frame(x = 1000 * -6:0, y = -2500),
  data.frame(x = 1000 * 0:-6, y = -2000),
  data.frame(x = 1000 * -6:0, y = -1500),
  data.frame(x = 1000 * 0:-6, y = -1000),
  data.frame(x = 1000 * -6:0, y = -500),
  data.frame(x = 1000 * 0:-6, y = 0),
  data.frame(x = -6000, y = -500 * 7:0),
  data.frame(x = -5000, y = -500 * 0:7),
  data.frame(x = -4000, y = -500 * 7:0),
  data.frame(x = -3000, y = -500 * 0:7),
  data.frame(x = -2000, y = -500 * 7:0),
  data.frame(x = -1000, y = -500 * 0:7),
  data.frame(x = 0, y = -500 * 7:0)
```

汪荣贵©著

```
)

panel.3dgridline <- function(..., rot.mat, distance, xlim, ylim, zlim,
                             xlim.scaled, ylim.scaled, zlim.scaled) {
  scale.vals <- function(x, original, scaled) {
    scaled[1] + (x - original[1]) * diff(scaled) / diff(original)
  }
  scaled.map <- rbind(
    scale.vals(rongelap_grid_line$x, xlim, xlim.scaled),
    scale.vals(rongelap_grid_line$y, ylim, ylim.scaled),
    zlim.scaled[1]
  )
  m <- ltransform3dto3d(scaled.map, rot.mat, distance)
  panel.lines(x = m[1,], y = m[2,], col = "gray", lty = 2)
}

cloud(counts / time ~ cX * cY,
  data = rongelap, col = "black",
  xlim = c(-6500, 100), ylim = c(-3800, 150),
  scales = list(arrows = FALSE, col = "black"),
  aspect = c(0.75, 0.5),
  xlab = list("横坐标（米）", rot = 20),
  ylab = list("纵坐标（米）", rot = -50),
  zlab = list("辐射强度", rot = 90),
  type = c("p", "h"), pch = 16, lwd = 0.5,
  panel.3d.cloud = function(...) {
    panel.3dgridline(...)
    panel.3dcoastline(...) # 海岸线
    panel.3dscatter(...)
  },
  # 减少三维图形的边空
  lattice.options = list(
    layout.widths = list(
      left.padding = list(x = -0.5, units = "inches"),
      right.padding = list(x = -1.0, units = "inches")
    ),
    layout.heights = list(
      bottom.padding = list(x = -1.5, units = "inches"),
      top.padding = list(x = -1.5, units = "inches")
    )
```

©黄湘云

```
  ),
  par.settings = list(
    # 移除几条内框线
    # box.3d = list(col = c(1, 1, NA, NA, 1, NA, 1, 1, 1)),
    # 刻度标签字体大小
    axis.text = list(cex = 0.8),
    # 去掉外框线
    axis.line = list(col = "transparent")
  ),
  # 设置三维图的观察方位
  screen = list(z = 30, x = -65, y = 0)
)
```

图 9.16: 添加三维网格参考线和透视曲线

9.3.9 三维透视图

有如下参数方程

$$\begin{cases} x(u,v) = \cos(u)\big(r + \cos(u/2)\big) \\ y(u,v) = \sin(u)\big(r + \cos(u/2)\sin(tv) - \sin(u/2)\sin(2tv)\big)\sin(tv) - \sin(u/2)\sin(2tv) \\ z(u,v) = \sin(u/2)\sin(tv) + \cos(u/2)\sin(tv) \end{cases}$$

其中，u 和 v 是参数，$\frac{u}{2\pi} \in [0.3, 1.25], \frac{v}{2\pi} \in [0,1]$，$r$ 和 t 是常量，不妨设 $r = 2$ 和 $t = 1$ 。

```r
# lattice 书 6.3.1 节 参数
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, drape = FALSE,
  xlab = expression(x[1]), ylab = expression(x[2]),
  zlab = list(expression(
    italic(f) ~ group("(", list(x[1], x[2]), ")")
  ), rot = 90), alpha = 0.75,
  scales = list(arrows = FALSE, col = "black"),
  # 减少三维图形的边空
  lattice.options = list(
    layout.widths = list(
      left.padding = list(x = -0.5, units = "inches"),
      right.padding = list(x = -1.0, units = "inches")
    ),
    layout.heights = list(
      bottom.padding = list(x = -1.5, units = "inches"),
      top.padding = list(x = -1.5, units = "inches")
    )
  ),
  par.settings = list(axis.line = list(col = "transparent")),
  screen = list(z = 30, x = -65, y = 0)
)
```

©黄湘云

图 9.17: 三维透视图

绘图函数 `wireframe()` 支持使用公式语法，也支持矩阵类型的数据绘制透视图。

```
wireframe(volcano,
  drape = TRUE, colorkey = FALSE, shade = TRUE,
  xlab = list("南北方向", rot = -40),
  ylab = list("东西方向", rot = 45),
  zlab = list("高度", rot = 90),
  # 减少三维图形的边空
  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")
    )
  ),
```

©黄湘云

```
  # 设置坐标轴字体大小
  par.settings = list(
    axis.line = list(col = "transparent"),
    fontsize = list(text = 12, points = 10)
  ),
  scales = list(arrows = FALSE, col = "black"),
  screen = list(z = -45, x = -50, y = 0)
)
```

图 9.18: 奥克兰火山地形图

9.3.10 地形轮廓图

绘图函数 `levelplot()` 支持使用公式语法，也支持矩阵类型的数据绘制轮廓图。基于奥克兰火山地形数据 volcano 绘制轮廓图，volcano 是矩阵类型的数据。

```
levelplot(volcano, useRaster = TRUE,
  # 去掉图形上、右边多余的刻度线
```

© 黄湘云


```
  scales = list(
    x = list(alternating = 1, tck = c(1, 0)),
    y = list(alternating = 1, tck = c(1, 0))
  ),
  par.settings = list(
    # x/y 轴标签字体，刻度标签字体
    par.xlab.text = list(fontfamily = "Noto Serif CJK SC"),
    par.ylab.text = list(fontfamily = "Noto Serif CJK SC"),
    axis.text = list(fontfamily = "sans")
  ),
  xlab = "南北方向", ylab = "东西方向"
)
```

图 9.19: 奥克兰火山的地形轮廓图

函数 levelplot() 的参数 col.regions 需要传递一个函数，示例中使用的默认设置。常见的函数有 hcl.colors() 、gray.colors() 、terrain.colors() 、cm.colors() 和 topo.colors() 等，函数 hcl.colors() 默认使用 viridis 调色板，还可以用函数 colorRampPalette() 构造调色板函数。

```
data(topo, package = "MASS")
levelplot(z ~ x * y, data = topo, scales = "free",
  panel = panel.2dsmoother, contour = TRUE,
  form = z ~ s(x, y, bs = "gp", k = 50), method = "gam",
  xlab = "水平方向", ylab = "垂直方向"
)
```

©黄湘云

图 9.20: 奥克兰火山的地形轮廓图

函数 `panel.2dsmoother()` 来自 **latticeExtra** 包，数据的二维分布，默认采用 `tp`

- `tp` thin plate regression spline 回归样条方法平滑。
- `cr` Cubic regression spline 立方回归样条。
- `gp` Gaussian process smooths 高斯过程平滑，默认为全秩 Full Rank，指定 k 低秩近似 Low Rank。

9.3.11 地区分布图

最后一个想要介绍的是地区分布图，也叫面量图、围栏图，描述空间栅格数据的分布，常见的一种情况是展示各个地区的人口、社会、经济指标。下面通过 **tigris** 包可以下载美国人口调查局发布的数据，本想下载与观测数据年份最近的地图数据，但是 2009 年及以前的地图数据缺失，因此，笔者下载了 2010 年的地图数据，它与得票率数据最近。

```
library(tigris)
us_state_map <- states(cb = TRUE, year = 2010, resolution = "20m", class = "sf")
us_state_map <- shift_geometry(us_state_map, geoid_column = "STATE", position = "below")
```

第一行代码用 **tigris** 包的函数 `states()` 下载 2010 年比例尺为 1:20000000 的多边形州边界矢量地图数据，返回一个 simple feature 类型的空间数据类型。第二行代码用该包的另一个函数 `shift_geometry()` 移动离岸的州和领地，将它们移动到主体部分的下方。

```
library(sf)
us_state_sf <- readRDS("data/us-state-map-2010.rds")
```

© 统计出版社

```
# sf 转 sp
us_state_sp <- as(us_state_sf, "Spatial")
library(maps)
# sp 转 map
us_state_map <- SpatialPolygons2map(us_state_sp, namefield = "NAME")
# 准备观测数据
data(votes.repub)
# 转为 data.frame 类型
votes_repub <- as.data.frame(votes.repub)
```

数据集 votes.repub 记录 1856-1976 年美国历届大选中共和党在各州的得票率。图中以由红到蓝的颜色变化表示由低到高的得票率，1964 年共和党在东南一隅得票率较高，在其它地方得票率普遍较低，形成一边倒的情况，最终由民主党的林登 · 约翰逊当选美国第 36 任总统。1968 年共和党在东南部得票率最低，与 1964 年相比，整个反过来了，最终由共和党的理查德 · 尼克松当选美国第 37 任总统。

```
library(RColorBrewer)
rdbu_pal <- colorRampPalette(colors = brewer.pal(n = 11, name = "RdBu"))
mapplot(rownames(votes_repub) ~ `1964` + `1968`, data = votes_repub,
  border = NA, map = us_state_map, colramp = rdbu_pal, layout = c(1, 2),
  scales = list(draw = FALSE), xlab = "", ylab = ""
)
```

©黄湘云

图 9.21: 共和党在各州的得票率

参数 `border` 设置州边界线的颜色，NA 表示去掉边界线。参数 `map` 设置州边界地图数据。参数 `colramp` 设置一个调色板，用于将得票率与调色板上的颜色映射起来。美国历届大选，共和党和民主党竞争总统职位，最终由得票率决定，用红蓝对抗型调色板表现竞争关系。基于 **RColorBrewer** 包的 `RdBu` 调色板，用函数 `colorRampPalette()` 构造一个新的红蓝调色板。参数 `layout` 将多个子图按照一定顺序排列，图中设置 2 行 1 列的多图布局。参数 `scales` 用来调整刻度，设置 `list(draw = FALSE)` 将图中的刻度去掉了。参数 `xlab` 设置一个空字符，即 `xlab = ""` 可去掉横坐标轴标签，参数 `ylab` 应用于设置纵坐标，用法与参数 `xlab` 一样。图中，主要表现得票率在各州的分布，因此，坐标轴刻度和标签都不太重要，可以去掉。

9.4 总结

现在回过头来看，无论是图形样式还是绘图语法，**lattice** 可以看作是介于 Base R 和 **ggplot2** 之间的一种绘图风格。举例来说，下面比较 Base R 和 **lattice** 的图形样式。

```
plot(Sepal.Length ~ Petal.Length, col = Species, data = iris,
  xlab = "萼片长度", ylab = "花瓣长度")
xyplot(Sepal.Length ~ Petal.Length, groups = Species, data = iris,
```

© 黄湘云


```
scales = "free", xlab = "萼片长度", ylab = "花瓣长度")
```

(a) Base R 图形

(b) lattice 图形

图 9.22: 对比 Base R 和 lattice 制作的分组散点图

与函数 `plot()` 对应的是函数 `xyplot()` ，它们共用一套公式语法，参数 `data` 的含义也是一样的。与参数 `col` 对应的是参数 `groups` ，作用是添加数据分组标识。在两个函数中，添加横纵坐标轴标签都是用参数 `xlab` 和 `ylab` 。函数 `xyplot()` 中参数 `scales` 的作用是对坐标轴刻度的调整，参数值 `"free"` 表示去掉图形上边和右边的刻度线，默认情况下是有刻度线的。

在高级的绘图函数方面，Base R 和 **lattice** 基本都有功能对应的函数，在低水平的绘图函数方面，二者截然不同，主要是因为后者基于另一套绘图系统 — **grid** 绘图系统。Base R 作图常常需要一个函数一个函数地不断叠加，在图中画上点、线、轴、标签等元素，而 **lattice** 主要通过面板函数，层层叠加的方式，每一个面板函数实现一个功能，整合一系列绘图操作。本章主要介绍 **lattice** 包和 **latticeExtra** 包，用到的高级绘图函数如下表。

表格 9.1: lattice 和 latticeExtra 包的部分函数

R 包	函数	图形	作用
lattice	`xyplot()`	（分组）散点图	描述关系
lattice	`bwplot()`	（分组）箱线图	描述分布
lattice	`barchart()`	（分组）柱形图	描述对比
lattice	`levelplot()`	切片水平图	描述趋势
lattice	`wireframe()`	三维透视图	描述趋势
lattice	`cloud()`	三维散点图	描述分布
latticeExtra	`panel.smoother()`	回归曲线图	描述趋势

R 包	函数	图形	作用
latticeExtra	`panel.2dsmoother()`	地形轮廓图	描述趋势
latticeExtra	`ecdfplot()`	经验分布图	描述分布
latticeExtra	`segplot()`	置信区间图	描述不确定性
latticeExtra	`panel.ellipse()`	置信椭圆图	描述不确定性
latticeExtra	`mapplot()`	地区分布图	描述分布

© 黄湘云

第十章　graphics 入门

不是把每个绘图函数都挨个讲一遍，也不是把它们统统归纳总结，而是比较深入地介绍一、两种图形，一、两个例子，重点阐述 Base R 的绘图特点，使用图形时，注意图形本身的作用，最终，希望读者能够达到举一反三的效果。

基础绘图系统。相比于 **ggplot2** 和 **lattice**，**graphics** 制作示意图是优势。

10.1　绘图基础

利用点、线等基础元素从零开始绘图。

10.1.1　`plot()`

本节将主要基于鸢尾花数据集介绍 R 语言基础绘图系统，该数据集最早来自埃德加 · 安德森，后来，被罗纳德 · 费希尔在介绍判别分析的论文中用到，从而，流行于机器学习社区。鸢尾花是非常漂亮的一种花，在统计和机器学习社区家喻户晓，更别提在植物界的名声。其实，远不止于此，在绘画艺术界也是如雷贯耳，印象派大师文森特 · 梵高画了一系列鸢尾花作品。万紫千红，但能入画的不多，故而，鸢尾花更显高雅。在生命最后的一段日子里，梵高受精神病折磨，在法国普罗旺斯的圣 · 雷米医院里，唯有盛开的鸢尾花陪着他，最著名的《星月夜》就是在这时候创作出来的。下面先让我们一睹鸢尾花芳容，图片来自维基百科鸢尾花词条。

(a) versicolor 杂色鸢尾

(b) setosa 山鸢尾

(c) virginica 弗吉尼亚鸢尾

图 10.1: 三种鸢尾花

© 草稿

鸢尾花数据集已经打包在 R 软件中，而且默认已经加载到命名空间，下面用函数 summary() 查看其概况。

```
summary(iris)
#>   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
#>  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
#>  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
#>  Median :5.800   Median :3.000   Median :4.350   Median :1.300
#>  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
#>  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
#>  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
#>        Species
#>  setosa    :50
#>  versicolor:50
#>  virginica :50
#>
#>
#>
```

函数 plot() 采用公式语法可以快速作图。

```
plot(Sepal.Length ~ Sepal.Width, data = iris)
plot(iris$Sepal.Width, iris$Sepal.Length, panel.first = grid())
```

(a) 公式语法绘制散点图　　(b) 带背景参考线的散点图

图 10.2: 快速作图函数 plot()

© 黄湘云

函数 plot() 是一个泛型函数，传递不同类型的参数值会调用不同的绘图方法，而不同的绘图方法的参数是不同的。当采用公式语法绘图时，会自动调用函数 plot.formula() ，此时，参数 panel.first 就不起作用。当不使用公式语法时，会调用函数 plot.default() ，此时，参数 panel.first 就起作用，利用该参数可以添加背景参考线。

10.1.2 标签

函数 plot() 的参数 xlab 、ylab 和 main 可以分别设置坐标轴横、纵标签和图主标题。

```
plot(
  Sepal.Length ~ Sepal.Width, data = iris,
  xlab = "Sepal Width", ylab = "Sepal Length",
  main = "Edgar Anderson's Iris Data"
)
```

图 10.3: 标签

© 基础统计

10.1.3 字体

作图函数 plot() 和 title() 都有参数 family ，设置该参数可以调整图形中的字体。下图 10.4 的横纵坐标轴标签和图标题设为宋体，坐标轴刻度标签设为无衬线字体。

```
plot(Sepal.Length ~ Sepal.Width, data = iris, ann = FALSE, family = "sans")
title(
  xlab = "萼片宽度", ylab = "萼片长度",
  main = "埃德加 · 安德森的鸢尾花数据", family = "Noto Serif CJK SC"
)
```

图 10.4: 字体

10.1.4 分组

分组有两种方式，其一按照数据中的分类变量分组，其二按照一定的规则分组。而图形表达的方式可以借助颜色或图形元素的样式。

函数 plot() 的参数 col 和 pch 都可以用来分组，前者通过颜色，后者通过点的类型。简单起见，将数据集 iris 中的 Species 列传递给参数 col ，实现不同种类的鸢尾花配以不同的颜色。

```
plot(Sepal.Length ~ Sepal.Width, data = iris, col = Species, pch = 16)
```

© 黄湘云

图 10.5: 分组

下面采用一个简单规则将数据分成两组，将鸢尾花中 setosa 山毛榉类型且 Sepal.Length 萼片长度大于 5 厘米的分成一组，以红色填充散点代表这部分数据，与余下的散点形成对比，达到区分的目的。

```
plot(Sepal.Length ~ Sepal.Width, data = iris)
points(Sepal.Length ~ Sepal.Width, data = iris,
  col = "#EA4335", pch = 16,
  subset = Species == "setosa" & Sepal.Length > 5
)
```

© 黄湘云

图 10.6: 分组

10.1.5 配色

经过探查，知道数据集 iris 中的 Species 列有三种取值。调用函数 palette() 设置一个超过 3 种颜色的调色板可以实现自定义配色。首先来看看当前调色板的颜色。

```
palette()
```

```
#> [1] "black"   "#DF536B" "#61D04F" "#2297E6" "#28E2E5" "#CD0BBC" "#F5C710"
#> [8] "gray62"
```

一共是 8 种颜色，效果预览见图 10.7 。

设置新的调色板也是用函数 palette() ，参数 value 设置新的颜色值向量，下面依次是红、蓝、绿、黄四种颜色。

```
palette(value = c("#EA4335", "#4285f4", "#34A853", "#FBBC05"))
```

函数 plot() 的调色板默认来自函数 palette() ，经过上面的调整，同一行绘图代码出来不同的效果，即图 10.5 变成图 10.8 。

```
plot(Sepal.Length ~ Sepal.Width, data = iris, col = Species, pch = 16)
```

©黄湘云

图 10.7: 默认调色板

图 10.8: 配色

10.1.6 注释

函数 `text()` 可以在图上任意位置添加文本或公式。下图在位置 (4,6.5) 处添加红色的文字 flower。

© 张贤


```
plot(Sepal.Length ~ Sepal.Width, data = iris)
text(x = 4, y = 6.5, labels = "flower", col = "#EA4335")
```

图 10.9: 注释

10.1.7 图例

函数 `plot()` 不会自动添加图例，需要使用函数 `legend()` 添加图例。

```
plot(Sepal.Length ~ Sepal.Width, data = iris, col = Species, pch = 16)
legend(x = "topright", title = "Species",
  legend = unique(iris$Species), box.col = NA, bg = NA,
  pch = 16, col = c("#EA4335", "#4285f4", "#34A853")
)
```

© 黄湘云

图 10.10: 图例

图例放置在绘图区域以外，比如右边空区域。此时，通过点和文本构造图例。

```
op <- par(mar = c(4, 4, 3, 6))
plot(
  Sepal.Length ~ Sepal.Width, data = iris,
  col = Species, pch = 16, main = "Edgar Anderson's Iris Data"
)
text(x = 4.7, y = 6.75, labels = "Species", pos = 4, offset = .5, xpd = T)
points(x = 4.7, y = 6.5, pch = 16, cex = 1, col = "#EA4335", xpd = T)
text(x = 4.7, y = 6.5, labels = "setosa", pos = 4, col = "#EA4335", xpd = T)
points(x = 4.7, y = 6.3, pch = 16, cex = 1, col = "#4285f4", xpd = T)
text(x = 4.7, y = 6.3, labels = "versicolor", pos = 4, col = "#4285f4", xpd = T)
points(x = 4.7, y = 6.1, pch = 16, cex = 1, col = "#34A853", xpd = T)
text(x = 4.7, y = 6.1, labels = "virginica", pos = 4, col = "#34A853", xpd = T)
on.exit(par(op), add = TRUE)
```

© 黄湘云

图 10.11: 图例

在函数 `plot()` 内设置较宽的坐标轴范围，获得一个较宽的绘图区域，再用函数 `points()` 添加数据点，最后，使用函数 `legend()` 添加图例。

```
plot(
  x = c(2, 6), y = range(iris$Sepal.Length), type = "n",
  xlab = "Sepal Width", ylab = "Sepal Length",
  main = "Edgar Anderson's Iris Data"
)
points(Sepal.Length ~ Sepal.Width,
  col = Species, pch = 16, data = iris
)
legend(x = "right", title = "Species",
  legend = unique(iris$Species), box.col = NA, bg = NA,
  pch = 16, col = c("#EA4335", "#4285f4", "#34A853")
)
```

©黄湘云

图 10.12: 图例

10.1.8 统计

添加分组线性回归线。按鸢尾花种类分组，线性回归模型拟合数据，抽取回归系数。首先，使用函数 `split()` 将数据集 iris 按变量 Species 分组拆分，得到一个列表，每个元素都是数据框。接着，调用函数 `lapply()` 将函数 `lm()` 作用到列表的每个元素上，得到一个列表，每个元素都是线性拟合对象。最后，再调函数 `lapply()` 将函数 `coef()` 应用到列表的每个元素上，得到回归模型的系数向量。

```
lapply(
  lapply(
    split(iris, ~Species), lm,
    formula = Sepal.Length ~ Sepal.Width
  ),
  coef
)
```

```
#> $setosa
#> (Intercept) Sepal.Width
#>   2.6390012   0.6904897
```

©清华大学出版社

```
#>
#> $versicolor
#> (Intercept) Sepal.Width
#>   3.5397347   0.8650777
#>
#> $virginica
#> (Intercept) Sepal.Width
#>   3.9068365   0.9015345
```

走到绘图这一步，往往是画什么内容比较清楚，分类数量、调色板都确定下来了。大致来说分 6 步：第一步，实现分组线性回归拟合；第二步，绘制分组散点图；第三步，添加分组回归线；第四步，添加图例并调整图例的位置；第五步，设置图形边界等绘图参数；第六步，添加背景网格线。输入线性拟合对象给函数 `abline()` 可以直接绘制回归线，不需要从拟合对象中提取回归系数。调用函数 `par()` 设置图形边界，特别是增加图形右侧边界以容纳图例，再调用函数 `legend()` 要设置 `xpd = TRUE` 以允许图例超出绘图区域。

```r
# 分组线性拟合
iris_lm <- lapply(
  split(iris, ~Species), lm, formula = Sepal.Length ~ Sepal.Width
)
# 将分组变量和颜色映射
cols <- c("setosa" = "#EA4335",  "versicolor" = "#4285f4", "virginica" = "#34A853")
# 设置图形边界以容纳标签和图例
op <- par(mar = c(4, 4, 3, 8))
# 绘制分组散点图
plot(
  Sepal.Length ~ Sepal.Width,
  data = iris, col = Species, pch = 16,
  xlab = "Sepal Width", ylab = "Sepal Length",
  main = "Edgar Anderson's Iris Data"
)
# 添加背景参考线
grid()
# 添加回归线
for (species in c("setosa", "versicolor", "virginica")) {
  abline(iris_lm[[species]], col = cols[species], lwd = 2)
}
# 添加图例
legend(
  x = "right", title = "Species", inset = -0.4, xpd = TRUE,
  legend = unique(iris$Species), box.col = NA, bg = NA, lty = 1, lwd = 2,
```

©黄湘云


```
  pch = 16, col = c("#EA4335", "#4285f4", "#34A853")
)
# 恢复图形参数设置
on.exit(par(op), add = TRUE)
```

图 10.13: 分组线性回归

10.2 绘图进阶

10.2.1 组合图形

点、线、多边形组合

```
x <- seq(-10, 10, length = 400)
y1 <- dnorm(x)
y2 <- dnorm(x, m = 3)
op <- par(mar = c(5, 4, 2, 1))
plot(x, y2,
```

```
  xlim = c(-3, 8), type = "n",
  xlab = quote(Z == frac(mu[1] - mu[2], sigma / sqrt(n))),
  ylab = "Density"
)
polygon(c(1.96, 1.96, x[240:400], 10),
  c(0, dnorm(1.96, m = 3), y2[240:400], 0),
  col = "grey80", lty = 0
)
lines(x, y2)
lines(x, y1)
polygon(c(-1.96, -1.96, x[161:1], -10),
  c(0, dnorm(-1.96, m = 0), y1[161:1], 0),
  col = "grey30", lty = 0
)
polygon(c(1.96, 1.96, x[240:400], 10),
  c(0, dnorm(1.96, m = 0), y1[240:400], 0),
  col = "grey30"
)
legend(x = 4.2, y = .4,
  fill = c("grey80", "grey30"),
  legend = expression(
    P(abs(Z) > 1.96, H[1]) == 0.85,
    P(abs(Z) > 1.96, H[0]) == 0.05
  ), bty = "n"
)
text(0, .2, quote(H[0]:~ ~ mu[1] == mu[2]))
text(3, .2, quote(H[1]:~ ~ mu[1] == mu[2] + delta))
on.exit(par(op), add = TRUE)
```

©黄湘云

图 10.14: 正态总体下两样本均值之差的检验

10.2.2 多图布局

布局函数 `layout()` 和图形参数设置函数 `par()`

```
data(anscombe)
form <- sprintf("y%d ~ x%d", 1:4, 1:4)
fit <- lapply(form, lm, data = anscombe)
op <- par(mfrow = c(2, 2), mgp = c(2, 0.7, 0),
          mar = c(3, 3, 1, 1) + 0.1, oma = c(0, 0, 2, 0))
for (i in 1:4) {
  plot(as.formula(form[i]),
    data = anscombe, col = "black",
    pch = 20, xlim = c(3, 19), ylim = c(3, 13),
    xlab = as.expression(substitute(x[i], list(i = i))),
    ylab = as.expression(substitute(y[i], list(i = i))),
    family = "sans"
  )
  abline(fit[[i]], col = "black")
  text(
```

© 张雄

```
    x = 7, y = 12, family = "sans",
    labels = bquote(R^2 == .(round(summary(fit[[i]])$r.squared, 3)))
  )
}
mtext("数据集的四重奏", outer = TRUE)
on.exit(par(op), add = TRUE)
```

图 10.15: 数据可视化很重要

10.3 图形选择

以不同的二维或三维图形可视化同一份多元数据。颜色图、透视图、等值线图和填充等值线图存在某种相似性，又有区别。

©黄湘云

10.3.1 颜色图

$$f(x,y)=\begin{cases}\frac{\sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}, & (x,y)\neq(0,0)\\ 1, & (x,y)=(0,0)\end{cases}$$

```
y <- x <- seq(from = -8, to = 8, length.out = 51)
z <- outer(x, y, FUN = function(x, y) sin(sqrt(x^2 + y^2)) / sqrt(x^2 + y^2))
z[26, 26] <- 1
```

将绘图区域划分成网格，每个小网格对应一个颜色值。函数 `image()` 绘制颜色图

```
image(x = x, y = y, z = z, xlab = "$x$", ylab = "$y$")
```

图 10.16: 颜色图

10.3.2 透视图

函数 `persp()` 绘制透视图

```
op <- par(mar = c(0, 1, 2, 1))
persp(
```

© 谢湘

```
  x = x, y = y, z = z, main = "二维函数的透视图",
  theta = 30, phi = 30, expand = 0.5, col = "lightblue",
  xlab = "$x$", ylab = "$y$", zlab = "$f(x,y)$"
)
on.exit(par(op), add = TRUE)
```

图 10.17: 透视图

10.3.3 等值线图

地理上，常用等高线图描述地形，等高线图和等值线图其实是一个意思。函数 `contour()` 绘制等值线图。

```
contour(x = x, y = y, z = z, xlab = "$x$", ylab = "$y$")
```

©黄湘云

图 10.18: 等值线图

10.3.4 填充等值线图

函数 `filled.contour()` 绘制填充等值线图。

```
filled.contour(
  x = x, y = y, z = z, asp = 1,
  color.palette = hcl.colors,
  plot.title = {
    title(
      main = "二维函数的填充等值线图",
      xlab = "$x$", ylab = "$y$"
    )
  },
  plot.axes = {
    grid(col = "gray")
    axis(1, at = 2 * -4:4, labels = 2 * -4:4)
    axis(2, at = 2 * -4:4, labels = 2 * -4:4)
    points(0, 0, col = "blue", pch = 16)
```

© 非洲出版社

```
  },
  key.axes = {
    axis(4, seq(-0.2, 1, length.out = 9))
  }
)
```

图 10.19: 填充等值线图

10.4 总结

10.4.1 plot2 包

plot2 包扩展 Base R 函数 `plot()` 的功能，在公式语法方面和 lattice 包很接近。另一个值得一提的 R 包是 **basetheme** ，用来设置 Base R 绘图主题。

```
library(plot2)
plot2(Sepal.Length ~ Sepal.Width | Species, data = iris,
      palette = "Tableau 10", pch = 16)
```

© 黄湘云

图 10.20: plot2 包绘制分组散点图

或者使用参数 by 指定分组变量，效果和上图一样。

```
with(iris, {
  plot2(y = Sepal.Length, x = Sepal.Width, by = Species,
      palette = "Tableau 10", pch = 16)
})
```

还可以使用参数 legend 调整图例的位置，比如放置在绘图区域下方。

```
op <- par(mar = c(5, 4, .5, .5))
plot2(Sepal.Length ~ Sepal.Width | Species,
  data = iris, palette = "Tableau 10", pch = 16,
  legend = legend("bottom!", title = "Species of iris", bty = "o")
)
on.exit(par(op), add = TRUE)
```

© 黄湘云

图 10.21: plot2 包调整图例位置

还可以绘制其它类型的图形，如分组密度曲线图等。

```
with(iris, plot2(
  density(Sepal.Length), by = Species,
  bg = "by",   # 分组填充
  grid = TRUE, # 背景网格
  palette = "Tableau 10",
  legend = list("topright", bty = "o") # 右上角图例
))
```

©黄湘云

图 10.22: plot2 包绘制密度曲线图

10.4.2 plot3D 包

虽然不提倡大量使用三维图形，但如何绘制三维图形却是生生不息的命题，以下仅是 R 语言社区的冰山一角。

- **plotrix** (Lemon 2006) 一个坐落于 R 的红灯区的 R 包。基于 Base R 各类绘图函数。
- **scatterplot3d** (Ligges 和 Mächler 2003) 基于 Base R 绘制三维散点图。
- **misc3d** (Feng 和 Tierney 2008) 绘制三维图形的杂项，支持通过 Base R、**tcltk** 包和 **rgl** 包渲染图形。
- **plot3D** (Soetaert 2021) 依赖 **misc3d** 包，加强 Base R 在制作三维图形方面的能力。

举个比较新颖的一个例子，**plot3D 包**的函数 `image2D()` 绘制二维颜色图，细看又和 `image()` 函数不同，渲染出来的图形有三维的立体感。归根结底，很多时候束缚住自己的不是工具，而是视野和思维。以奥克兰 Maunga Whau 火山地形数据 `volcano` 为例。

```
library(plot3D)
image2D(volcano,
  shade = 0.2, rasterImage = TRUE, asp = 0.7,
  xlab = "南北方向", ylab = "东西方向",
```

```
  main = "奥克兰 Maunga Whau 地形图", clab = "高度",
  contour = FALSE, col = hcl.colors(100),
  colkey = list(
    at = 90 + 20 * 0:5, labels = 90 + 20 * 0:5,
    length = 1, width = 1
  )
)
op <- par(mar = c(1, 1.5, 0, 0))
persp3D(
  x = 1:87, y = 1:61, z = volcano, col = hcl.colors(100),
  ticktype = "detailed", colkey = FALSE, expand = 1,
  phi = 35, theta = 125, bty = "b2", shade = TRUE,
  ltheta = 100, lphi = 45,
  xlab = "\n南北方向", ylab = "\n东西方向", zlab = "\n高度"
)
on.exit(par(op), add = TRUE)
```

值得一提，Python 社区的绘图模块 matplotlib 同样具有强大的绘图能力，三维图形也不在话下。不过，不同的绘图系统所采用的透视法不同，如下图所示。

© 黄湘云

(a) 函数 `image2D()` 二维颜色图

(b) 函数 `persp3D()` 三维透视图

图 10.23: 奥克兰火山地形图

© 黄湘云

图 10.24: matplotlib 绘制三维透视图

© 吉湘石

第十一章　TikZ 入门

有一些示意图是用来表达数据探索的思路的，而不是直接探索数据的工具。比如象限图、甘特图、思维导图、网络图等，可以用 TikZ 绘制这类图形来阐述分析维度、思路、结构等。当然，绘制这类示意图不仅限于 TikZ，还有很多其它工具，如 LaTeX 社区的 PSTricks，JavaScripts 社区的 Mermaid，软件 Graphviz 等。

11.1　standalone 宏包

最常见的 LaTeX 文档类有 article、report、beamer、book，分别对应文章、报告、演示和书籍。有的宏包在此基础上扩展功能，比如 ctex 宏包提供中文支持，有四个文档类 ctexart、ctexrep 、ctexbeamer 和 ctexbook 与之对应起来。standalone 宏包提供 standalone 文类主要用于绘制独立的图片，默认情况下，文档四周多余的空白部分会被裁剪掉。在 LaTeX 环境中，推荐使用 TikZ 来绘图。standalone 文类可与 tikz 宏包一起使用，生成一张张由 TikZ 代码绘制的独立图片。下面举个简单的例子，用 TikZ 绘制两个坐标轴。

```
\documentclass[tikz,border=1mm]{standalone}
\begin{document}
\begin{tikzpicture}
\draw[<->] (6,0) -- (0,0) node[left]{O} -- (0,6);
\end{tikzpicture}
\end{document}
```

standalone 文类启用 `tikz` 选项来绘图，选项 `border=1mm` 表示图片四周的边空保留 1 毫米，文档内容放在 document 环境里，TikZ 绘图代码放在 tikzpicture 环境中，命令 `\draw` 负责绘制具体的图形，用 XeLaTeX 编译，效果如图 11.1 所示。

standalone 文类有很多选项，下面介绍 4 个选项的常用内容。

- `class` 选项指定文类环境，默认值为 `article`，表示在 article 文类中绘图。其它选项还有 `beamer` ，表示在演示环境中绘图。在不同的文类中，图片渲染出来的效果不同。
- `tikz=true|false` 选项是否启用 TikZ 绘图，默认值是 `false` 。当显式地在 standalone 文类中启用 `tikz` 选项，就表示用 TikZ 绘图，将自动加载 tikz 宏包。与之类似的选项 `pstricks=true|false` ，表示是否启用 PSTricks 绘图，PSTricks 是 LaTeX 社区中一套语法不同于 TikZ 的绘图工具。

© 黄湘云

图 11.1: TikZ 绘图

- `crop=true|false` 选项是否裁剪变空，默认值是 `true` ，表示绘图区域以外的部分都裁剪掉。与之相关的另一个选项是 `border` ，可以更加精细地控制图片四周的各个边空。
- `border` 选项指定边空大小，默认值是 0，表示无边空。当 `crop=true` 时，再指定 `border` 选项，比如 `border=1mm` 表示图片四周的边空保留 1 毫米。图片四周的边空大小可以按照左、下、右、上的顺序指定，比如 `border={5mm 6mm 0mm -2mm}` 表示图片左边空 5 毫米、下边空 6 毫米、右边空 0 毫米、上边空负 2 毫米。

standalone 文类是支持 PSTricks 绘图的，下面在直角坐标系中绘制一个带阴影效果的圆，示例代码如下：

```
\documentclass[pstricks,border={5mm 6mm 0mm -2mm}]{standalone}
\usepackage{pst-plot}
\begin{document}
\psset{xunit=0.15in, yunit=0.15in}
\begin{pspicture}(0,0)(11,11)
\psaxes[Dx=4,Dy=4, subticks=4]{->}(0,0)(0,0)(10,10)[$x$,0][$y$,0]
\pscircle[runit=0.15in, fillcolor=orange!50, fillstyle=solid,shadow=true](5,5){3}
\end{pspicture}
\end{document}
```

standalone 文类的选项 `pstricks` 表示启用 PSTricks 绘图环境，加载 pst-plot 宏包提供额外的命令，PSTricks 是基于 PostScript 语言的，每一个绘图命令都是 `\ps` 开头的，比如 `\psset` 、`\psaxes`、`\pscircle` 等。`\begin{pspicture}` 和 `\end{pspicture}` 之间是 PSTricks 绘图代码，`\begin{pspicture}` 之后的 `(0,0)(11,11)` 是左下和右上角两个坐标，定义了一个绘图区域。和 TikZ 绘图代码一样，也用 XeLaTeX 编译，效果如图 11.2 所示。

可以在 Quarto 和 R Markdown 文档中插入 PSTricks 绘图代码，使用 **knitr** 包的 `tikz` 引擎绘图。只要修改模版文件 `tikz2pdf.tex` ，移除一行 `\usetikzlibrary{matrix}` ，不再加载 tikz 宏

©黄渊石

图 11.2: PSTricks 绘图

包及其 matrix 库。TIKZ_CLASSOPTION 不再仅限于 TikZ ，而是 standalone 文类的选项，相应地，EXTRA_TIKZ_PREAMBLE_CODE 变成一般的 LaTeX 文档的导言区，TIKZ_CODE 可以是 PSTricks 代码。新的模版文件 tikz2pdf.tex 如下：

```
\documentclass[
%% TIKZ_CLASSOPTION %%
]{standalone}
%% EXTRA_TIKZ_PREAMBLE_CODE %%
\begin{document}
%% TIKZ_CODE %%
\end{document}
```

上图 11.2 即是由 **knitr** 包的 tikz 引擎渲染出来的。在代码块选项 engine-opts 中，传递一个列表，分别包含 classoption（standalone 文类选项）、extra.preamble（导言区）、template （TikZ 模版文件）三块内容。生成图 11.2 的 engine-opts 设置如下：

```
list(
  classoption = "pstricks,border={5mm 6mm 0mm -2mm}",
  extra.preamble = "\\usepackage{pst-plot}",
  template = "code/tikz2pdf.tex"
)
```

其它选项和更多详细介绍见 standalone 宏包帮助文档。

11.2 PGF 宏包

PGF 宏包提供一套易于学习和使用的绘图语法 TikZ，TikZ 是 TikZ ist kein Zeichenprogramm 的简写，命名有 Linux 哲学意味。下面比较详细的介绍 LaTeX 宏包 PGF 绘制曲线图的过程。

```
\begin{tikzpicture}
\draw[<->] (6,0) -- (0,0) node[left]{O} -- (0,6);
```

©黄海洋

```
\end{tikzpicture}
```

图 11.3: PGF 绘制曲线图

首先，`\draw` 命令绘制带箭头的坐标轴，坐标轴的范围 $[0,6]\times[0,6]$ 。坐标轴是由线构成的，线有虚线、实线，也有宽度和颜色，虚线还有不同类型，这些都是可以设置的参数，比如将 `\draw[<->]` 改为 `\draw[color=red,<->]` ，坐标轴颜色设置为红色。

```
\begin{tikzpicture}
\draw[<->] (6,0) node[below]{$q$} -- (0,0) node[left]{O} -- (0,6) node[left]{$V(q)$};
\end{tikzpicture}
```

图 11.4: PGF 绘制曲线图

然后，在位置 (6,0) 和 (0,6) 分别添加节点 `node[below]{$q$}` 和 `node[left]{$V(q)$}` 。node 表示节点，节点的标签内容在大括号内，标签的位置在中括号内，这里，below 表示在位置 (6,0) 的下方。

```
\begin{tikzpicture}
```

©黄海宏


```
\draw[<->] (6,0) node[below]{$q$} -- (0,0) node[left]{O} -- (0,6) node[left]{$V(q)$};
\draw[very thick] (0,0) to [out=90,in=145] (5,4.5);
\end{tikzpicture}
```

图 11.5: PGF 绘制曲线图

最后，从点 (0,0) 至点 (5,4.5) 绘制一条非常粗的曲线。曲线从点 (0,0) 出去的时候，是以 90 度垂直水平轴的方向出去的，到点 (5,4.5) 是以 145 度方向进入的。角度是按照逆时针方向计算的。线的粗细、方向都是由参数决定的。

在这里，TikZ 是用来绘制示意图的，不需要知道每个命令的每个参数的取值有哪些。关键是知道自己想要画什么，其实，可以用铅笔在纸上以最快的方式绘制草图，了解每个绘图元素，然后查找 PGF 帮助手册，找到对应的命令和参数，将草图工整地誊抄一遍。

11.3 三维图

顾名思义，pgfplots 宏包基于 PGF 的，用它来绘制三维图形是非常方便的。

```
\documentclass[tikz]{standalone}
\usepackage{pgfplots}
\pgfplotsset{width=7cm,compat=1.17}
\begin{document}
%% TikZ 代码%%
\end{document}
```

首先加载 pgfplots 宏包，设置全局的绘图参数，`width=7cm` 表示绘图页面宽度，`compat=1.17` 表示使用 pgfplots 的版本。

```
\begin{tikzpicture}
\begin{axis}[
```

```
    hide axis,
    colormap/viridis
]
\addplot3[
    mesh,
    samples=50,
    domain=-8:8
]
{ sin(deg(sqrt(x^2+y^2)))/sqrt(x^2+y^2) };
\addlegendentry{$\frac{\sin(r)}{r}$}
\end{axis}
\end{tikzpicture}
```

图 11.6: TikZ 绘制三维图 viridis 调色板

```
\begin{tikzpicture}
\begin{axis}[
    hide axis,
    colormap/jet
]
\addplot3[
    mesh,
    samples=50,
    domain=-8:8
]
{ sin(deg(sqrt(x^2+y^2)))/sqrt(x^2+y^2) };
\addlegendentry{$\frac{\sin(r)}{r}$}
\end{axis}
\end{tikzpicture}
```

© 黄湘云

图 11.7: TikZ 绘制三维图 jet 调色板

```
\begin{tikzpicture}
\begin{axis}[
    hide axis,
    colormap/cool,
    colorbar sampled,
    domain=-8:8
]
\addplot3[
    contour filled={
      number=20,
    },
    ]{sin(deg(sqrt(x^2+y^2)))/sqrt(x^2+y^2)};
\addlegendentry{$\frac{\sin(r)}{r}$}
\end{axis}
\end{tikzpicture}
```

图 11.8: TikZ 绘制三维图 cool 调色板

- `\begin{axis}` 和 `\end{axis}` 环境有很多配置选项，参数值 `[hide axis, colormap/viridis]` 中 `hide axis` 表示隐藏坐标轴，`colormap/viridis` 表示三维图形的调色板采用 viridis 。`colormap` 支持很多不同的调色板，上面列举了两个。其实还可以增加不同的选项，比如添加选项 `colorbar sampled` 会生成一个颜色条，还可以添加选项 `colorbar horizontal` 来水平放置颜色条。

- 可以在导言区加载 `\usetikzlibrary{pgfplots.colorbrewer}` 导入 ColorBrewer 系列调色板，方便后续绘图时调用。作用与 R 语言中的 **RColorBrewer** 包类似，调色板名称略有不同，前者 `PuBu-9` 对应后者 `PuBu` 。
- `\addplot3` 命令绘制函数 `sin(deg(sqrt(x^2+y^2)))/sqrt(x^2+y^2)` 的三维图像，即函数 $f(x,y)=\frac{\sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}$ 的三维图像。参数值 `[mesh, samples=50, domain=-8:8]` 中 `mesh` 表示三维图形是网格状，其它可选值还有曲面图 `surf` 、填充等值线图 `contour filled` 等，`samples=50` 表示网格密度是 50，`domain=-8:8` 表示横纵坐标的范围都是 $[-8,8]$ 。
- `\addlegendentry` 添加图例，图例标签是 $\frac{\sin(r)}{r}$ ，颜色会随着调色板变化。

11.4 网络图

绘制网络图用 tikz-network 宏包，也是 PGF 的一个扩展包。图结构是根据顶点和边来定义的，图的复杂程度也可以用顶点和边的规模来衡量。图描述一种非线性的关系，有自己的一套语言，定义顶点 `\Vertex` 和边 `\Edge` 的两个命令是最基础的。下面绘制柯尼斯堡七桥问题对应的图。

```
\documentclass[tikz]{standalone}
\usepackage{tikz-network}
\begin{document}
%% TikZ 代码%%
\end{document}

\begin{tikzpicture}
\Vertex[IdAsLabel, x=5, color=gray, size=1, fontsize=\large]{A}
\Vertex[IdAsLabel, x=10, color=gray, size=1, fontsize=\large]{B}
\Vertex[IdAsLabel, x=15, color=gray, size=1, fontsize=\large]{C}
\Vertex[IdAsLabel, x=10, y=6, color=gray, size=1, fontsize=\large]{D}

\Edge[label=2, bend=45, fontscale=2](A)(B)
\Edge[label=6, bend=30, fontscale=2](A)(D)
\Edge[label=3, bend=45, fontscale=2](B)(A)
\Edge[label=5, bend=45, fontscale=2](B)(C)
\Edge[label=4, bend=45, fontscale=2](C)(B)
\Edge[label=7, bend=30, fontscale=2](D)(C)
\Edge[label=1, fontscale=2](D)(B)
\end{tikzpicture}
```

©黄渊石

图 11.9: 柯尼斯堡七桥

- \Vertex 命令定义顶点（含标签），参数 IdAsLabel 表示顶点 ID 作为标签，参数 x 和 y 表示坐标位置，参数 color 表示顶点的填充色，参数 size 表示顶点的大小，参数 fontsize 表示标签文本的大小。
- \Edge 命令在已有顶点的基础上定义边，(A)(B) 表示从顶点 A 到顶点 B 有一条边，参数 label 表示边上的标签文本，参数 bend 表示边的弧度，参数 fontscale 表示标签文本的大小。

不难看出，无论是顶点还是边，都有颜色、大小、标签等参数，尽管参数名称有所不同。

11.5 思维导图

思维导图是非常常见的一种树状图，用于梳理层次关系。TikZ 绘制思维导图是通过 mindmap 库实现的，它是 PGF 的一个库。如图 11.10 所示，看着和脑神经网络有某种相似性，所以，有时候，思维导图也叫脑图。

```
\documentclass[tikz,svgnames]{standalone}
\usepackage[fontset=fandol]{ctex}
\usetikzlibrary{mindmap}
\begin{document}
%% TikZ 代码%%
\end{document}
```

```
\begin{tikzpicture}[
    mindmap, every node/.style=concept, concept color=orange, text=white,
    level 1/.append style={level distance=5cm, sibling angle=60, font=\LARGE},
```

©黄湘云

```
  level 2/.append style={level distance=3.5cm, sibling angle=45, font=\large}
]

\node{\huge{\textsf{数据分析}}} [clockwise from=60]
child [concept color=DarkMagenta] {
    node {\textit{数据准备}} [clockwise from=120]
    child { node {数据对象}}
    child { node {数据获取}}
    child { node {数据清洗}}
    child { node {数据操作}}
  }
child [concept color=DarkBlue] {
    node {\textit{数据探索}} [clockwise from=30]
    child { node {ggplot2 入门}}
    child { node {基础图形}}
    child { node {统计图形}}
  }
child [concept color=Brown] {
    node {\textit{数据交流}} [clockwise from=-30]
    child { node {交互图形}}
    child { node {交互表格}}
    child { node {交互应用}}
  }
child [concept color=teal] {
    node {\textit{统计分析}} [clockwise from=-75]
    child { node {统计检验}}
    child { node {回归分析}}
    child { node {功效分析}}
  }
child [concept color=purple] {
    node {\textit{数据建模}} [clockwise from=-120]
    child { node {网络分析}}
    child { node {文本分析}}
    child { node {时序分析}}
  }
child [concept color=DarkGreen] {
    node {\textit{优化建模}} [clockwise from=180]
    child { node {统计计算}}
    child { node {数值优化}}
    child { node {优化问题}}
```

©黄湘云


```
    };
\end{tikzpicture}
```

图 11.10: TikZ 绘制思维导图

根节点视为一层，则该思维导图有三层。不同的颜色和字体来区分不同的层次或分类，数据分析划分为不同的部分，每个部分有若干章。根节点字体为黑体、第二、三级节点分别为楷体、宋体。

```
\node{\huge{\textsf{数据科学}}} [clockwise from=60]
```

定义根节点，节点的文本设置为黑体，大小设置为 \huge 。由根节点向外辐射生成 6 个子节点，每隔 60 度设置一个子节点。

```
  child [concept color=DarkMagenta] {
```

© 笨鸟出版

```
    node {\textit{数据准备}} [clockwise from=120]
    child { node {数据对象}}
    child { node {数据获取}}
    child { node {数据清洗}}
    child { node {数据操作}}
  }
```

第一个子节点，颜色为饱和的紫色 DarkMagenta，二级子节点为「数据准备」，三级子节点有 4 个，逆时针 120 度的位置设置第一个三级子节点「数据对象」，然后顺时针往下，依次是「数据获取」、「数据清洗」和「数据操作」。

11.6 SmartArt 图

Office 办公软件中有一个 SmartArt 绘图模块，专门用来绘制各类示意图。LaTeX 宏包 smartdiagram 基于 TikZ 定制了一套风格类似的绘图库。smartdiagram 宏包的主要绘图命令是 `\smartdiagram[参数值]`，设置不同的参数值可以绘制不同的图形，如气泡图 `bubble diagram` 和描述图 `descriptive diagram` 等。

```
\smartdiagram[bubble diagram]{
  Pandoc,
  编程语言~\\ (Python\\R/Julia\\JavaScript),
  编译引擎~\\ (Jupyter\\Knitr\\Observable),
  扩展Pandoc~\\ (交叉引用\\悬浮引用\\布局面板),
  文档项目~\\ (批量渲染\\共享配置),
  扩展接口~\\ (RStudio\\VS Code\\JupyterLab)
}
```

© 黄湘云

图 11.11: 气泡图

```
\smartdiagram[descriptive diagram]{
  {编程语言, {Python、R、Julia、JavaScript}},
  {编译引擎, {Jupyter、Knitr、Observable}},
  {扩展Pandoc, {交叉引用、悬浮引用、布局面板}},
  {文档项目, {批量渲染、共享配置}},
  {扩展接口, {RStudio、VS Code、JupyterLab}},
}
```

©黄湘云

图 11.12: 描述图

11.7 TikZ 与 R

TikZ 绘图的优势有很多，语法简单、易于上手、功能强大、资源丰富、成熟稳定等，可以说几乎是集所有优点于一身。正因如此，**knitr** 包和 **tikzDevice** 包将其引入 R 语言社区中。**knitr** 包的 `tikz` 引擎是用来编译 TikZ 代码的，默认使用的是 standalone 文类。

R 语言绘图遇到公式时，略显不足，而排版公式是 LaTeX 的优势。正因为有所不足，所以我也不会纠结于工具层面的东西，什么好用用什么！三维图 11.6 是用 LaTeX 里的优秀绘图工具 TikZ 制作的，细心的读者会发现本书多次用到这个工具。

众所周知，Donald Knuth 十年磨一剑，开发了 TeX 排版系统，就是解决排版数学公式的痛点。如图 11.13 所示，因图形中包含数学公式和符号，为了获得原汁原味的渲染效果，在使用 Base R 绘图的过程中通过 **tikzDevice** 包引入了 LaTeX 中的 TikZ 绘图引擎。

```
opar <- par(mgp = c(2, 0.7, 0), mar = c(4, 3, 4, 1) + 0.1, no.readonly = TRUE)
set.seed(2021)
x <- rnorm(10)
y <- x + rnorm(5, sd = 0.25)
lab <- sample(
  x = paste0("$\\mathcal{", LETTERS, "}$"),
  size = 10, replace = FALSE
)
```

© 张韩


```
model <- lm(y ~ x)
rsq <- summary(model)$r.squared
rsq <- signif(rsq, 4)
plot(x, y,
  main = "你好 \\LaTeX!", # 引入 7 号文本字体
  sub = "$\\mathcal{N}(x;\\mu,\\Sigma)$",
  xlab = "$x$", ylab = "$y$", type = "n"
)
text(x = x, y = y, labels = lab)
abline(model, col = "black")
# 引入 7 号数学字体
mtext(paste("线性模型: $\\mathsf{R}^{2}=", rsq, "$"), line = 0.5)
legend("bottomright",
  legend = paste0(
    "$y = ", round(coef(model)[2], 3), "x +",
    round(coef(model)[1], 3), "$"
  ),
  bty = "n"
)
on.exit(par(opar), add = TRUE)
```

图 11.13: 简单线性模型

© 谢益辉

图 11.14 是贝塞尔函数 $\mathcal{K}_\kappa(u)$ 在区间 $(10^{-8}, 10^2)$ 和 $(0, 4)$ 上的图像。其中，图 11.14a 是区间 $(10^{-8}, 10^2)$ 上的贝塞尔函数 $\mathcal{K}_\kappa(u)$，图 11.14b 是区间 $(0, 4)$ 上的贝塞尔函数 $\mathcal{K}_\kappa(u)$ 。

```
x0 <- 2^(-20:10)
nus <- c(2:5, 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
)

axis(1,
  at = 10^(-8 + 2 * 0:5),
  labels = paste0("$\\mathsf{10^{", -8 + 2 * 0:5, "}}$")
)
axis(2,
  at = 10^(-8 + 16 * 0:4),
  labels = paste0("$\\mathsf{10^{", -8 + 16 * 0:4, "}}$"), 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 = TRUE), lwd = 2, cex = 1
)

x <- seq(0, 4, length.out = 501)
x <- x[x > 0]
plot(x, x,
  frame.plot = TRUE, ylim = c(1e+0, 1e+20), log = "y",
  xlab = "$u$", ylab = "$\\mathcal{K}_{\\kappa}(u)$",
  type = "n", yaxt = "n", ann = TRUE, panel.first = grid()
```

© 王建平


```
)
axis(2,
  at = c(1e+0, 1e+05, 1e+10, 1e+15, 1e+20),
  labels = paste0("$\\mathsf{10^{", 5 * 0:4, "}}$"), las = 1
)

for (i in seq(length(nus))) {
  lines(x, besselK(x, nu = nus[i]), col = hcl.colors(9)[i], lwd = 2)
}
legend("topright",
  legend = paste0("$\\kappa=", rev(nus), "$"),
  col = hcl.colors(9, rev = TRUE), lwd = 2, cex = 1
)
```

(a) 区间 $(10^{-8}, 10^{2})$ 上的贝塞尔函数

(b) 区间 $(0, 4)$ 上的贝塞尔函数

图 11.14: 贝塞尔函数图像

© 黄湘云

第十二章　探索实践

12.1　分析老忠实间歇泉喷发规律

图 12.1 展示美国怀俄明州黄石国家公园老忠实间歇泉喷发规律，横轴表示喷发持续时间（以分钟计），纵轴表示等待时间（以分钟计），点的亮暗程度（白到黑）代表附近点密度的高低，亮度值通过二维核密度估计方法得到，具体实现借助了 **KernSmooth** (Wand 和 Jones 1995) 包提供的 `bkde2D()` 函数，设置了喷发时间的窗宽为 0.7 分钟，等待时间的窗宽为 7 分钟。不难看出，每等待 55 分钟左右间歇泉喷发约 2 分钟，或者每等待 80 分钟左右间歇泉喷发 4.5 约分钟，非常守时，表现得很老实，故而得名。说实话，二维核密度估计在这里有点大材小用了，因为数据点比较少，肉眼也能分辨出来哪里聚集的点多，哪里聚集的点少。

> 提示
>
> 函数 `bkde2D()` 实现二维带窗宽的核密度估计（2D Binned Kernel Density Estimate），R 语言存在多个版本，**grDevices** 包的函数 `densCols()` 直接调用 **KernSmooth** 包的函数 `bkde2D()`，**graphics** 包的函数 `smoothScatter()` 与函数 `densCols()` 一样，内部也是调用 `bkde2D()` 函数，**ggplot2** 包的图层 `geom_density_2d()` 采用 **MASS** 包的函数 `kde2d()`，在算法实现上，`MASS::kde2d()` 与 `KernSmooth::bkde2D()` 不同，前者是二维核密度估计（Two-Dimensional Kernel Density Estimation）。Tarn Duong 的著作《Multivariate Kernel Smoothing and Its Applications》(José. Chacón 2018) 对多元核平滑方法及其应用作了专门的论述，相关实现见书籍配套的 **ks** 包。

12.2　中国地区级男女性别比分布

图 12.2a 展示 2020 年中国各省、自治区和直辖市分城市、镇和乡村的性别比数据。数据来自中国国家统计局发布的 2021 年统计年鉴，在数据量不太大的情况下，尽可能展示原始数据，可以捕捉到更加细致的差异。社会经济相关的数据常常显示有马太效应，对原始数据适当做一些变换有利于比较和展示数据，图 12.2b 展示对数变换后的性别比数据的分布。大部分地区的性别比数据都在 100:100 左右，流动人口的性别比波动大很多。

- 「住本乡、镇、街道，户口在本乡、镇、街道」土著和已获得当地户口的。性别比分布有明显的层

© 黄湘云

(a) faithful 数据集的散点图

(b) 点的亮暗表示核密度估计值的大小

(c) 等高线表示核密度估计值

(d) 等高线表示核密度估计值

图 12.1: 二维核密度估计

© 黄湘云

(a) 原始性别比数据

(b) 对数变换后的性别比数据

图 12.2: 2020 年中国地区级男女性别比分布

© 黄湘云

次差异，性别比均值从大到小依次是城市、乡村、镇。城市里，男性略多于女性，镇里，男性明显少于女性，乡村里，男性略低于女性。

- 「住本乡、镇、街道，户口待定」黑户或其它。性别比分布有明显的层次差异。同上。
- 「住本乡、镇、街道，户口在外乡、镇、街道，离开户口登记地半年以上」流出人口，流出乡、镇、街道。城市、镇、乡村的性别比数据充分混合了，性别比分布没有明显的层次差异。
- 「居住在港澳台或国外，户口在本乡、镇、街道」流出人口，流出国。同上。

12.3 美国历年各年龄死亡率变化

图 12.3 展示美国 1933-2020 年男性分年龄的死亡率数据[1]。图分上下两部分，上半部分展示死亡率原值随年龄的变化情况，以 ggplot2 默认的调色板给各个年份配色，下半部分展示死亡率对数变换后随年龄的变化情况，并以红、橙、黄、绿、青、蓝、紫构造彩虹式的调色板给各个年份配色。作图过程中，使用对数变换和调用彩虹式的调色板，帮助我们观察到更多的细节、层次。对数变换后，更加清晰地展示死亡率的变化，尤其是 0-20 岁之间的死亡率起伏变化。调用彩虹式的调色板后，约 20 年为一个阶段，每个阶段内呈现梯度变化，多个阶段体现层次性，更加清晰地展示死亡率曲线的变动趋势。透过层次看到 80 多年来，美国在医疗和公共卫生方面取得的显著改善。

图 12.3 也展示了很多基础信息，出生时有很高的死亡率，出生后死亡率迅速下降，一直到 10 岁，死亡率才又开始回升，直到 20 岁，死亡率才回到出生时的水平。之后，在青年阶段死亡率缓慢增加，直至老年阶段达到很高的死亡率水平。相比于老年阶段，医疗水平的改善作用主要体现在婴儿、儿童、青少年阶段。

图 12.3 还展示了一个潜在的数据质量问题，在 100 岁之后，死亡率波动程度明显在变大，这是因为高龄人数变得很少，即死亡率的分母变得很小，分子的细小波动会被放大，也因为同样的原因，100 岁以上的死亡率主要依赖模型估计，甚至出现死亡率大于 1 的罕见情况。因此，就对比医疗和公共卫生水平的变化而言，从数据的实际情况出发，100 岁以上的情况可以不参与比较。

图 12.4 以年份为横轴，以年龄为纵轴绘制网格，网格内部根据男性死亡率数据填充颜色制作热力图，死亡率数据是对数尺度，颜色的变化和死亡率的变化关系同前，采用了相同的调色板。更加深入的分析和建模，详见 Marron 和 Dryden (2022) 的第一章。

12.4 美国弗吉尼亚州城乡死亡率

VADeaths 数据来自 Base R 内置的 datasets 包，记录 1940 年美国弗吉尼亚州死亡率，如下表。

[1] 数据来自德国马克斯普朗克人口研究所、美国加州大学伯克利分校、法国人口研究所共同建立的人类死亡率数据库 (https://www.mortality.org/)。

© 黄湘云

图 12.3: 1933-2020 年美国男性死亡率曲线

© 黄湘云

图 12.4: 1933-2020 年美国男性死亡率热力图

© 黄湘云

表格 12.1: 1940 年美国弗吉尼亚州死亡率

	农村男	农村女	城市男	城市女
50-54	11.7	8.7	15.4	8.4
55-59	18.1	11.7	24.3	13.6
60-64	26.9	20.3	37.0	19.3
65-69	41.0	30.9	54.6	35.1
70-74	66.0	54.3	71.1	50.0

死亡率数据是按年龄段、城乡、性别分组统计的，这是一个三因素交叉统计表，表格中第 1 行第 1 列的数据表示 50-54 岁乡村男性的死亡率为 11.7 ‰ ，即在 50-54 岁乡村男性群体中，1000 个人中死亡 11.7 个，这是抽样调查出来的数字。下图分年龄段、城乡、性别展示弗吉尼亚州死亡率数据，从图例来看，突出的是各年龄段的对比，图主要传递的信息是各年龄段的死亡率差异。无论城市还是乡村，也无论男性还是女性，年龄越大死亡率越高，这完全是一个符合生物规律的客观事实，这是众人皆知的，算不上洞见。

图 12.5: 弗吉尼亚州各年龄段死亡率的对比

© 黄湘云

将对比对象从年龄段转变为城乡，描述城乡差距在死亡率上的体现，是不是一下子更深刻了呢？城市降低各个年龄段的死亡率，暗示着城市的居住条件、医疗水平比乡村好，提高城市化率增加全民的寿命。严格来说，就这个粗糙的数据集不能如此快地下这个结论，但是，它暗示这个信息，同样也符合人们的常识。

图 12.6: 弗吉尼亚州城乡死亡率的对比

12.5 如何用图表示累积概率分布

不失一般性，考虑连续随机变量的条件分布和累积分布。不妨设随机变量 x 的概率分布函数和概率密度函数分别是 $F(x)$ 和 $f(x)$ 。已知概率分布函数和概率密度函数之间有如下关系。

$$F(x) = \int_{-\infty}^{x} f(t)\mathrm{dt}$$

diamonds 数据来自 ggplot2 包，记录了约 54000 颗钻石的价格、重量、切工、颜色、纯净度、尺寸等属性信息。图 12.7 展示这批不同切工的钻石随价格的分布，在这个示例中，如何表达累积分布？概率分布的密度曲线是根据直方图估计得来的，根据不同价格区间内钻石的数量占总钻石的比例估计概率。固

© 黄湘石

定窗宽，即在同一价格区间内累积不同切工的钻石数量，得到累积概率，最后获得累积概率密度曲线，更多理论细节见数据可视化陷阱 (Pu 和 Kay 2020) 。

图 12.7: 不同切工的钻石随价格的分布

12.6 解释二项总体参数的置信带

0 和 1 是世界的原点，蕴含大道真意，从 0-1 分布也叫伯努利分布，独立同 0-1 分布之和可以衍生出二项分布。在一定条件下，可以用泊松分布近似二项分布。根据中心极限定理，独立同二项分布的极限和与正态分布可以发生关系。在二项分布、正态分布的基础上，可以衍生出超几何分布、贝塔分布等等。本书多个地方以二项分布为例介绍基本统计概念和模型。

在给定置信水平为 0.95，即 $\alpha = 0.05$，固定样本量 $n = 10$，观测到的成功次数 x 可能为 $0, 1, \cdots, 10$。对于给定的 p，不同 x 值出现的机率 $P(X = x)$ 由 $(p+q)^{10}$ 二项展开式的项给出，这里 $q = 1 - p$，即：

$$P(X = x) = \binom{n}{x} p^x (1-p)^{n-x} \tag{12.1}$$

在给定 $p = 0.1$ 的情况下，求二项分布的上 $\alpha/2 = 0.025$ 分位点，尾项之和不应超过 $\alpha/2$，最大的 x 值

©黄湘云

可有如下方程给出：

$$\sum_{r=x}^{n}\binom{n}{x}p^x(1-p)^{n-x}=\frac{\alpha}{2} \tag{12.2}$$

在 R 语言中，函数 `qbinom()` 可以计算上述二项分布的上分位点 $x=3$，即

```
qbinom(0.025, size = 10, prob = 0.1, lower.tail = F)
```

```
#> [1] 3
```

反过来，若已知上分位点为 $x=3$，则概率为

$$P(X>3)=\sum_{x>3}^{10}\binom{10}{x}0.1^x*(1-0.1)^{10-x} \tag{12.3}$$

在 R 语言中，函数 `pbinom()` 可以计算上述二项分布的上分位点对应的概率为 0.0127952。

```
pbinom(q = 3, size = 10, prob = 0.1, lower.tail = F)
```

```
#> [1] 0.0127952
```

首先简单回顾一下置信区间，在学校和教科书里，有两种说法如下：

1. $1-\alpha$ 的把握确定区间包含真值。
2. 区间包含真值的概率是 $1-\alpha$。

为什么要采纳第一种说法而不是第二种呢？这其实涉及到置信区间的定义问题，历史上 E. S. Pearson 和 R. A. Fisher 曾有过争论。和大多数以正态分布为例介绍参数的置信估计不同，下面以二项分布为例展开介绍。我们知道二项分布是 N 个伯努利分布的卷积，而伯努利分布又称为 0-1 分布，最形象的例子要数抛硬币了，反复投掷硬币，将正面朝上记为 1，反面朝上记为 0，记录正反面出现的次数，正面朝上的总次数又叫成功次数。

1934 年 C. J. Clopper 和 E. S. Pearson 在给定置信水平 $1-\alpha=0.95$ 和样本量 $n=10$ 的情况下，给出二项分布 $B(n,p)$ 参数 p 的区间估计（即所谓的 Clopper-Pearson 精确区间估计）和置信带 (Clopper 和 Pearson 1934)，如图 12.8 所示，横坐标为观测到的成功次数，纵坐标为参数 p 的置信限。具体来说，固定样本量为 10，假定观测到的成功次数为 2，在置信水平为 0.95 的情况下，Base R 内置的二项精确检验函数 `binom.test()`，可以获得参数 p 的精确区间估计为 $(p_1,p_2)=(0.025,0.556)$，即：

```
# 精确二项检验 p = 0.2
binom.test(x = 2, n = 10, p = 0.2)
```

```
#>
#>  Exact binomial test
#>
#> data:  2 and 10
#> number of successes = 2, number of trials = 10, p-value = 1
#> alternative hypothesis: true probability of success is not equal to 0.2
```

```
#> 95 percent confidence interval:
#>  0.02521073 0.55609546
#> sample estimates:
#> probability of success
#>                    0.2
```

值得注意，这个估计的区间与函数 `binom.test()` 中参数 `p` 的取值无关，也就是说，当 $p = 0.4$，区间估计结果是一样的，如下：

```
# 精确二项检验 p = 0.4
binom.test(x = 2, n = 10, p = 0.4)
```

```
#>
#>  Exact binomial test
#>
#> data:  2 and 10
#> number of successes = 2, number of trials = 10, p-value = 0.3335
#> alternative hypothesis: true probability of success is not equal to 0.4
#> 95 percent confidence interval:
#>  0.02521073 0.55609546
#> sample estimates:
#> probability of success
#>                    0.2
```

由此，也可以看出区间估计与假设检验的一些关系。

12.7 解释置信区间及其覆盖概率

统计图形很重要的一个作用是解释统计概念，这就要求不拘泥于抽象的严格数学表达，借助数值模拟，可视化等手段帮助读者发散思维，加深理解复杂的逻辑概念，建立统计直觉，正如顾恺之所言「以形写神，形神兼备」。下面仅以二项分布为例讲讲区间估计及其覆盖概率。众所周知，在置信水平为 $1-\alpha$ 的情况下，二项分布 $\mathrm{Bin}(n,p)$ 的参数 p （也叫成功概率）的 Wald 区间估计为

$$(\hat{p} - Z_{1-\alpha/2}\sqrt{\hat{p}*(1-\hat{p})/n}, \hat{p} + Z_{1-\alpha/2}\sqrt{\hat{p}*(1-\hat{p})/n}) \tag{12.4}$$

其中，n 为样本量，$Z_{1-\alpha/2}$ 为标准正态分布 $\mathcal{N}(0,1)$ 在 $1-\alpha/2$ 处的分位点。α 一般取 0.05，进而 $Z_{1-\alpha/2} \approx 1.96$。用通俗的话说，有 $1-\alpha$ 的把握确定参数真值 p 在该估计区间内。可见区间估计的意义是解决点估计可靠性问题，但是可靠性和精度往往不能兼得。统计上，通常的做法是先给定可靠性，去尽可能提升精度，即给定置信水平，使估计区间的长度尽可能短，这就涉及到区间估计的方法问题。

下面通过数值模拟的方式辅助说明 Wald 和 Agresti-Coull 两种区间估计方法，现固定样本量 $n = 10$ 或 $n = 100$，重复抽样 1000 次，将参数 p 以 0.01 的间隔离散化，从 0.01 取值到 0.99。已知给定参数 p，

© 黄湘云

图 12.8: 二项分布参数的置信带

每次抽样都可以得到参数 p 的估计值 $\hat{p}$ 及其置信区间，1000 次的重复抽样可以计算出来 1000 个置信区间，每个区间要么覆盖真值，要么没有覆盖真值，覆盖的比例可以近似为覆盖概率。

如图 12.9 所示，从上往下分别代表 Wald、Agresti-Coull、Wilson 和 Clopper-Pearson 区间估计，纵坐标是覆盖概率，横坐标是参数 p 的真值，图中黑虚线表示置信水平 $1-\alpha=0.95$，红、蓝点线分别表示样本量 $n=10$ 和 $n=100$ 的模拟情况。不难看出，Wald 区间估计方法在小样本情况下表现很差，覆盖概率很少能达到置信水平的，而 Agresti-Coull 区间估计在 Wald 基础上添加了修正后，情况得到显著改善。更多区间估计方法的详细比较见文献 Blyth 和 Hutchinson (1960);Brown, Cai, 和 DasGupta (2001);Geyer 和 Meeden (2005) 。

通过图 12.9 一看就明白了几种区间估计方法的优劣，以及为什么软件普遍默认采用 Wilson 估计方法?因为它又稳定又准确。Wilson 区间估计用的更加广泛的，Base R 内置的比例检验函数 `prop.test()` 在不启用 Yates 修正时，就是用该方法获得比例 p 的区间估计 (Wilson 1927)。Clopper-Pearson 区间估计特别适合小样本情形，它是精确区间估计方法，Base R 内置的二项比例检验函数 `binom.test()` 就是用该方法获得比例 p 的区间估计 (Clopper 和 Pearson 1934)。

> **提示**
>
> 请读者再思考两个问题：图 12.9 为什么呈现对称的形式，泊松分布会和二项分布有类似的现象吗?如果有的话，连续分布，如正态分布和指数分布也会有吗?

12.8 习题

1. 1888 年，瑞士开始进入一个人口转变的阶段，从发展中国家的高出生率开始下滑。分析生育率和经济指标的关系。数据集 swiss 记录了 1888 年瑞士 47 个说法语的省份的生育率和社会经济指标数据。Fertility（生育率，采用常见的标准生育率统计口径）、Agriculture（男性从事农业生产的比例）、Examination（应征者在军队考试中获得最高等级的比例）、Education（应征者有小学以上教育水平的比例）、Catholic（信仰天主教的比例）、Infant.Mortality（婴儿死亡率，仅考虑出生一年内死亡），各个指标都统一标准化为百分比的形式。其中，Examination 和 Education 是 1887 年、1888 年和 1889 年的平均值。瑞士 182 个地区 1888 年及其它年份的数据可从网站获得。

© 黄湘云

图 12.9: 二项分布参数的几种区间估计：覆盖概率随成功概率的变化

© 黄湘云

第三部分

数据交流

©黄湘云

第十三章　交互图形

在之前的数据探索章节介绍了 **ggplot2** 包，本章将介绍 **plotly** 包，绘制交互图形，包含基础元素、常用图形和技巧，沿用日志提交数据和 Base R 内置的斐济及周边地震数据。写作上，仍然以一个数据串联尽可能多的小节，从 **ggplot2** 包到 **plotly** 包，将介绍其间的诸多联系，以便读者轻松掌握。

13.1　基础元素

13.1.1　图层

plotly 包封装了许多图层函数，可以绘制各种各样的统计图形，见下表格 13.1 。

表格 13.1: **plotly** 包可以绘制丰富的统计图形

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

下面以散点图为例，使用方式非常类似 **ggplot2** 包，函数 `plot_ly()` 类似 `ggplot()`，而函数 `add_markers()` 类似 `geom_point()`，效果如图 13.1 所示。

```
# https://plotly.com/r/reference/scatter/
plotly::plot_ly(data = quakes, x = ~long, y = ~lat) |>
  plotly::add_markers()
```

或者使用函数 `add_trace()`，层层添加图形元素，效果和上图 13.1 是一样的。

© 王建硕

图 13.1: 默认风格的简单散点图

```
plotly::plot_ly(data = quakes, x = ~long, y = ~lat) |>
  plotly::add_trace(type = "scatter", mode = "markers")
```

提示

plotly 包的函数 `plot_ly()` 又与 **ggplot2** 包中函数 `qplot()` 类似，可以将大部分设置塞进去。

```
plotly::plot_ly(
  data = quakes, x = ~long, y = ~lat,
  type = "scatter", mode = "markers"
)
```

所以，总的来说，`add_markers()` 、`add_trace(type = "scatter", mode = "markers")` 和 `plot_ly(type = "scatter", mode = "markers")` 是等价的。

13.1.2 配色

在图 13.1 的基础上，将颜色映射到震级变量上。

```
plotly::plot_ly(data = quakes, x = ~long, y = ~lat) |>
  plotly::add_markers(color = ~mag)
```

©黄湘云

13.1.3 刻度

东经和南纬

```
plotly::plot_ly(data = quakes, x = ~long, y = ~lat) |>
  plotly::add_markers(color = ~mag) |>
  plotly::layout(
    xaxis = list(title = "经度", ticksuffix = 'E'),
    yaxis = list(title = "纬度", ticksuffix = 'S')
  )
```

13.1.4 标签

添加横轴、纵轴以及主副标题

```
plotly::plot_ly(
  data = quakes, x = ~long, y = ~lat,
  marker = list(
    color = ~mag,
    colorscale = "Viridis",
    colorbar = list(title = list(text = "震级"))
  )
) |>
  plotly::add_markers() |>
  plotly::layout(
    xaxis = list(title = "经度"),
    yaxis = list(title = "纬度"),
    title = "斐济及其周边地区的地震活动"
  )
```

13.1.5 主题

plotly 内置了一些主题风格

```
plotly::plot_ly(
  data = quakes, x = ~long, y = ~lat,
  marker = list(
    color = ~mag,
    colorscale = "Viridis",
    colorbar = list(title = list(text = "震级"))
  )
```

©黄湘云

```
) |>
  plotly::add_markers() |>
  plotly::layout(
    xaxis = list(title = "经度"),
    yaxis = list(title = "纬度"),
    title = "斐济及其周边地区的地震活动"
  )
```

13.1.6 字体

13.1.7 图例

13.2 常用图形

13.2.1 散点图

plotly 包支持绘制许多常见的散点图，从直角坐标系 `scatter` 到极坐标系 `scatterpolar` 和地理坐标系 `scattergeo`，从二维平面 `scatter` 到三维空间 `scatter3d`，借助 WebGL 可以渲染大规模的数据点 `scattergl`。

表格 13.2: plotly 包支持绘制的散点图类型

类型	名称
`scatter`	二维平面散点图
`scatter3d`	三维立体散点图
`scattergl`	散点图（WebGL 版）
`scatterpolar`	极坐标下散点图
`scatterpolargl`	极坐标下散点图（WebGL 版）
`scattergeo`	地理坐标下散点图
`scattermapbox`	地理坐标下散点图（MapBox 版）
`scattercarpet`	地毯图
`scatterternary`	三元图

图 13.2 展示斐济及其周边的地震分布

```
plotly::plot_ly(
  data = quakes, x = ~long, y = ~lat,
  type = "scatter", mode = "markers"
) |>
  plotly::layout(
```

©黄湘云

```
    xaxis = list(title = "经度"),
    yaxis = list(title = "纬度")
  )
```

图 13.2: 普通散点图

13.2.2 柱形图

```
# https://plotly.com/r/reference/bar/
plotly::plot_ly(
  data = trunk_year, x = ~year, y = ~revision, type = "bar"
) |>
  plotly::layout(
    xaxis = list(title = "年份"),
    yaxis = list(title = "代码提交量")
  )
```

13.2.3 曲线图

```
plotly::plot_ly(
  data = trunk_year, x = ~year, y = ~revision, type = "scatter",
  mode = "markers+lines", line = list(shape = "spline")
```

©黄湘云

```
) |>
  plotly::layout(
    xaxis = list(title = "年份"),
    yaxis = list(title = "代码提交量")
  )
```

13.2.4 直方图

地震次数随震级的分布变化，下图 13.3 为频数分布图

```
# https://plotly.com/r/reference/histogram/
plotly::plot_ly(quakes, x = ~mag, type = "histogram") |>
  plotly::layout(
    xaxis = list(title = "震级"),
    yaxis = list(title = "次数")
  )
```

图 13.3: 地震震级的频数分布图

地震震级的概率分布，下图 13.4 为频率分布图

```
plotly::plot_ly(
  data = quakes, x = ~mag, type = "histogram",
  histnorm = "probability",
```

©黄湘云

```
  marker = list(
    color = "lightblue",
    line = list(color = "white", width = 2)
  )
) |>
  plotly::layout(
    xaxis = list(title = "震级"),
    yaxis = list(title = "频率")
  )
```

图 13.4: 地震震级的频率分布图

`histnorm = "probability"` 意味着纵轴表示频率，即每个窗宽下地震次数占总地震次数的比例。地震常常发生在地下，不同的深度对应着不同的地质构造、不同的地震成因，下图 13.5 展示海平面下不同深度的地震震级分布。

```
quakes$depth_bin <- cut(quakes$depth, breaks = 150 * 0:5)
```

```
plotly::plot_ly(quakes,
  x = ~mag, colors = "viridis",
  color = ~depth_bin, type = "histogram"
) |>
  plotly::layout(
    xaxis = list(title = "震级"),
    yaxis = list(title = "次数")
```

©黄湘云

```
)
```

图 13.5: 地震震级的频率分布图

13.2.5 箱线图

```
plotly::plot_ly(quakes,
  x = ~depth_bin, y = ~mag, colors = "viridis",
  color = ~depth_bin, type = "box"
) |>
  plotly::layout(
    xaxis = list(title = "深度"),
    yaxis = list(title = "震级")
  )
```

```
plotly::plot_ly(quakes,
  x = ~depth_bin, y = ~mag, split = ~depth_bin,
  type = "violin", color = ~depth_bin, colors = "viridis",
  box = list(visible = TRUE),
  meanline = list(visible = TRUE)
) |>
  plotly::layout(
    xaxis = list(title = "深度"),
```

```
    yaxis = list(title = "震级")
  )
```

13.2.6 热力图

plotly 整合了开源的 Mapbox GL JS，可以使用 Mapbox 提供的瓦片地图服务（Mapbox Tile Maps），对空间点数据做核密度估计，展示热力分布，如图 13.6 所示。图左上角为所罗门群岛（Solomon Islands）、瓦努阿图（Vanuatu）和新喀里多尼亚（New Caledonia），图下方为新西兰北部的威灵顿（Wellington）和奥克兰（Auckland），图中部为斐济（Fiji）。

```
plotly::plot_ly(
  data = quakes, lat = ~lat, lon = ~long, radius = 10,
  type = "densitymapbox", coloraxis = "coloraxis"
) |>
  plotly::layout(
    mapbox = list(
      style = "stamen-terrain", zoom = 3,
      center = list(lon = 180, lat = -25)
    ),
    coloraxis = list(colorscale = "Viridis")
  )
```

图 13.6: 空间点数据的核密度估计

统计学习与深度学习

图中设置瓦片地图的风格 `style` 为 `"stamen-terrain"`，还可以使用其他开放的栅格瓦片地图服务，比如 `"open-street-map"` 和 `"carto-positron"`。如果使用 MapBox 提供的矢量瓦片地图服务，则需要访问令牌 Mapbox Access Token。图中设置中心坐标 `center` 以及缩放倍数 `zoom`，目的是突出图片中的数据区域。设置调色板 Viridis 展示热力分布，黄色团块的地方表示地震频次高。

13.2.7 面量图

在之前我们介绍过用 ggplot2 绘制地区分布图，实际上，地区分布图还有别名，如围栏图、面量图等。本节使用 plotly 绘制交互式的地区分布图，如图 13.7 所示。

```
# https://plotly.com/r/reference/choropleth/
dat <- data.frame(state.x77,
  stats = rownames(state.x77),
  stats_abbr = state.abb
)
# 绘制图形
plotly::plot_ly(
  data = dat,
  type = "choropleth",
  locations = ~stats_abbr,
  locationmode = "USA-states",
  colorscale = "Viridis",
  colorbar = list(title = list(text = "人均收入")),
  z = ~Income
) |>
  plotly::layout(
    geo = list(scope = "usa"),
    title = "1974年美国各州的人均收入"
  )
```

13.2.8 动态图

本节参考 plotly 包的官方示例渐变动画，数据来自 SVN 代码提交日志，统计 Martin Maechler 和 Brian Ripley 的年度代码提交量，他们是 R Core Team 非常重要的两位成员，长期参与维护 R 软件及社区。下图展示 1999-2022 年 Martin Maechler 和 Brian Ripley 的代码提交量变化。

```
# https://plotly.com/r/animations/
trunk_year_author <- aggregate(data = svn_trunk_log, revision ~ year + author, FUN = length)
# https://plotly.com/r/cumulative-animations/
accumulate_by <- function(dat, var) {
  var <- lazyeval::f_eval(f = var, data = dat)
```

©黄湘云

图 13.7: 1974 年美国各州的人均收入

```
  lvls <- plotly:::getLevels(var)
  dats <- lapply(seq_along(lvls), function(x) {
    cbind(dat[var %in% lvls[seq(1, x)], ], frame = lvls[[x]])
  })
  dplyr::bind_rows(dats)
}

subset(trunk_year_author, year >= 1999 & author %in% c("ripley", "maechler")) |>
  accumulate_by(~year) |>
  plotly::plot_ly(
    x = ~year, y = ~revision, split = ~author,
    frame = ~frame, type = "scatter", mode = "lines",
    line = list(simplyfy = F)
  ) |>
  plotly::layout(
    xaxis = list(title = "年份"),
    yaxis = list(title = "代码提交量")
  ) |>
  plotly::animation_opts(
    frame = 100, transition = 0, redraw = FALSE
```

©谢益辉

```
  ) |>
  plotly::animation_button(
    visible = TRUE, # 显示播放按钮
    label = "播放", # 按钮文本
    font = list(color = "gray")# 文本颜色
  ) |>
  plotly::animation_slider(
    currentvalue = list(
      prefix = "年份 ",
      xanchor = "right",
      font = list(color = "gray", size = 30)
    )
  )
```

lazyeval 的非标准计算采用 Base R 实现，目前，已经可以被 **rlang** 替代。

13.3 常用技巧

13.3.1 数学公式

正态分布的概率密度函数形式如下：

$$f(x;\mu,\sigma^2)=\frac{1}{\sqrt{2\pi}\sigma}\exp\{-\frac{(x-\mu)^2}{2\sigma^2}\}$$

下图展示两个正态分布，分别是 $\mathcal{N}(3,1^2)$ 和 $\mathcal{N}(2,1.5^2)$ 。函数 `plotly::TeX()` 包裹 LaTeX 书写的数学公式，**plotly** 包调用 MathJax 库渲染图中的公式符号。

```
x <- seq(from = -4, to = 8, length.out = 193)
y1 <- dnorm(x, mean = 3, sd = 1)
y2 <- dnorm(x, mean = 2, sd = 1.5)

plotly::plot_ly(
  x = x, y = y1, type = "scatter", mode = "lines",
  fill = "tozeroy", fillcolor = "rgba(0, 204, 102, 0.2)",
  text = ~ paste0(
    "x: ", x, "<br>",
    "y: ", round(y1, 3), "<br>"
  ),
  hoverinfo = "text",
  name = plotly::TeX("\\mathcal{N}(3,1^2)"),
```

© 张贤达


```
  line = list(shape = "spline", color = "#009B95")
) |>
  plotly::add_trace(
    x = x, y = y2, type = "scatter", mode = "lines",
    fill = "tozeroy", fillcolor = "rgba(51, 102, 204, 0.2)",
    text = ~ paste0(
      "x: ", x, "<br>",
      "y: ", round(y2, 3), "<br>"
    ),
    hoverinfo = "text",
    name = plotly::TeX("\\mathcal{N}(2, 1.5^2)"),
    line = list(shape = "spline", color = "#403173")
  ) |>
  plotly::layout(
    xaxis = list(showgrid = F, title = plotly::TeX("x")),
    yaxis = list(showgrid = F, title = plotly::TeX("f(x)")),
    legend = list(x = 0.8, y = 1, orientation = "v")
  ) |>
  plotly::config(mathjax = "cdn", displayModeBar = FALSE)
```

图 13.8: 设置数学公式

13.3.2 动静转化

在出版书籍，发表期刊文章，打印纸质文稿等场景中，需要将交互图形导出为静态图形，再插入到正文之中。

```
library(ggplot2)
p <- ggplot(data = quakes, aes(x = long, y = lat)) +
  geom_point()
p
```

图 13.9: ggplot2 绘制的静态图形

将 **ggplot2** 包绘制的散点图转化为交互式的散点图，只需调用 **plotly** 包的函数 `ggplotly()`。

```
plotly::ggplotly(p)
```

当使用配置函数 `config()` 设置参数选项 `staticPlot = TRUE`，可将原本交互式的动态图形转为非交互式的静态图形。

```
plotly::ggplotly(p) |>
  plotly::config(staticPlot = TRUE)
```

> 提示
>
> 函数 `style()` 设置动态点的注释，比如点横纵坐标、坐标文本，以及整个注释标签的样式，如背景色。
>
> ```
> plotly::ggplotly(p, dynamicTicks = "y") |>
> ```

©清华大学出版社


```
  plotly::style(hoveron = "points", hoverinfo = "x+y+text",
        hoverlabel = list(bgcolor = "white"))
```

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。

```
# orca
plotly::orca(p, "plotly-quakes.svg")
# kaleido
plotly::save_image(p, "plotly-quakes.svg")
```

13.3.3 坐标系统

quakes 是一个包含空间位置的数据集，plotly 的 `scattergeo` 图层针对空间数据提供多边形矢量边界地图数据，支持设定坐标参考系。下图 13.10 增加了地震震级维度，在空间坐标参考系下绘制散点。

```
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, colorscale = "Viridis",
    size = 10, opacity = 0.8,
    line = list(color = "white", width = 1)
  )
) |>
  plotly::layout(geo = list(
    showland = TRUE,
    landcolor = plotly::toRGB("gray95"),
    countrycolor = plotly::toRGB("gray85"),
    subunitcolor = plotly::toRGB("gray85"),
    countrywidth = 0.5,
    subunitwidth = 0.5,
    lonaxis = list(
      showgrid = TRUE,
      gridwidth = 0.5,
```

©南京大学出版社

```
      range = c(160, 190),
      dtick = 5
    ),
    lataxis = list(
      showgrid = TRUE,
      gridwidth = 0.5,
      range = c(-40, -10),
      dtick = 5
    )
  ))
```

图 13.10: 空间点数据图

13.3.4 添加水印

在图片右下角添加水印图片

```
plotly::plot_ly(quakes,
  x = ~long, y = ~lat, color = ~mag,
  type = "scatter", mode = "markers"
) |>
  plotly::config(staticPlot = TRUE) |>
  plotly::layout(
```

©黄湘云

```
    images = list( # 水印图片
      source = "https://images.plot.ly/language-icons/api-home/r-logo.png",
      xref = "paper", # 页面参考
      yref = "paper",
      x = 0.90, # 横坐标
      y = 0.20, # 纵坐标
      sizex = 0.2, # 长度
      sizey = 0.2, # 宽度
      opacity = 0.5 # 透明度
    )
  )
```

13.3.5 多图布局

将两个图形做上下排列

```
p1 <- plotly::plot_ly(
  data = trunk_year, x = ~year, y = ~revision, type = "bar"
) |>
  plotly::layout(
    xaxis = list(title = "年份"),
    yaxis = list(title = "代码提交量")
  )

p2 <- plotly::plot_ly(
  data = trunk_year, x = ~year, y = ~revision, type = "scatter",
  mode = "markers+lines", line = list(shape = "spline")
) |>
  plotly::layout(
    xaxis = list(title = "年份"),
    yaxis = list(title = "代码提交量")
  )

htmltools::tagList(p1, p2)
```

plotly 包提供的函数 `subplot()` 专门用于布局排列，下图的上下子图共享 x 轴。

```
plotly::subplot(plotly::style(p1, showlegend = FALSE),
                plotly::style(p2, showlegend = FALSE),
                nrows = 2, margin = 0.05, shareX = TRUE, titleY = TRUE)
```

下图展示更加灵活的布局形式，嵌套使用布局函数 `subplot()` 实现。

```
p11 <- plotly::subplot(plotly::style(p1, showlegend = FALSE),
  plotly::style(p2, showlegend = FALSE),
  nrows = 1, margin = 0.05, shareY = TRUE, titleX = TRUE
)

plotly::subplot(p11,
  plotly::style(p2, showlegend = FALSE),
  nrows = 2, margin = 0.05, shareY = FALSE, titleX = FALSE
)
```

13.3.6 图表联动

crosstalk 包可将 **plotly** 包绘制的图形和 **DT** 包制作的表格联动起来。**plotly** 绘制交互图形，在图形上用套索工具筛选出来的数据显示在表格中。

```
library(crosstalk)
# quakes 数据变成可共享的
quakes_sd <- SharedData$new(quakes)
# 绘制交互图形
p <- plotly::plot_ly(quakes_sd, x = ~long, y = ~lat) |>
  plotly::add_markers() |>
  plotly::highlight(on = "plotly_selected", off = "plotly_deselect")
# 制作表格
d <- DT::datatable(quakes_sd, options = list(dom = "tp"))
# 将图表组合一起展示
bscols(list(p, d))
```

©黄湘云

第十四章　交互表格

表格常用来汇总展示数据，交互表格附带更多的功能，可以按列分组、排序、搜索，也可以按行分组、折叠，还可以上下滚动、左右滚动、前后翻页、页面挑转、导出数据等。交互表格是需要放在网页中的，制作这样的表格需要谢益辉开发的 **DT** 包，它的覆盖测试达到 31%，它基于 jQuery 框架的衍生品 DataTables 库，提供了一个 R 的封装，封装工具和许多其他基于 JS 库的 R 包一样，都依赖于 **htmlwidgets**。

14.1　基础功能

14.1.1　创建表格

14.1.2　添加标题

14.1.3　添加注释

14.1.4　水平滚动

14.1.5　垂直滚动

14.1.6　数据分页

14.1.7　适应宽度

14.1.8　行列分组

14.1.9　列格式化

14.1.10　数据配色

```
library(tibble)
```

©谢乐

```
dat <- tribble(
  ~name1, ~name2,
  as.character(htmltools::tags$b("加粗")), as.character(htmltools::a(href = "https://rstudio.com", "超链
  as.character(htmltools::em("强调")), '<a href="#" onclick="alert(\'Hello World\');">Hello</a>',
  as.character(htmltools::span(style = "color:red", "正常")), "正常"
)
```

根据数据的大小配上颜色

```
colorize_num <- function(x) {
  ifelse(x > 0,
    sprintf("<span style='color:%s'>%s</span>", "green", x),
    sprintf("<span style='color:%s'>%s</span>", "red", x)
  )
}
colorize_pct <- function(x) {
  ifelse(x > 0,
    sprintf("<span style='color:%s'>%s</span>", "green", scales::percent(x, accuracy = 0.01)),
    sprintf("<span style='color:%s'>%s</span>", "red", scales::percent(x, accuracy = 0.01))
  )
}

colorize_pp <- function(x) {
  ifelse(x > 0,
    sprintf("<span style='color:%s'>%s</span>", "green", paste0(round(100*x, digits = 2), "PP")),
    sprintf("<span style='color:%s'>%s</span>", "red", paste0(round(100*x, digits = 2), "PP"))
  )
}

colorize_text <- function(x, color = "red") {
    sprintf("<span style='color:%s'>%s</span>", color, x )
}

library(DT)
datatable(
  data = dat, escape = FALSE,
  colnames = c(colorize_text("第1列", "red"),
               as.character(htmltools::em("第2列"))),
  options = list(
    pageLength = 5, # 每页显示5行
    dom = "t"
```

©黄湘云

表格 14.1: 数据配色

	第1列	第2列
1	加粗	超链
2	强调	Hello
3	正常	正常

```
  )
)
```

Base R 内置的 R 包含有丰富的数据集，非常适合演示图形和阐述统计理论，后面技术和理论部分的介绍大多围绕内置的数据集展开，数据集及其描述如下表所示：

```
# 抽取 R 包信息
Pkgs <- sapply(list.files(R.home("library")), function(x) {
  packageDescription(pkg = x, fields = "Priority")
})
# 抽取内置 R 包列表
CorePkgs <- names(Pkgs[Pkgs %in% c("base", "recommended") & !is.na(Pkgs)])
# 抽取 R 包的数据集
BaseDataSets <- data(package = CorePkgs)$results[, c("Package", "Item", "Title")]

library(DT)
datatable(BaseDataSets,
  rownames = FALSE, # 不显示行名
  extensions = c("Buttons", "RowGroup"),
  options = list(
    pageLength = 10, # 每页显示的行数
    language = list(url = "//cdn.datatables.net/plug-ins/1.10.11/i18n/Chinese.json"), # 汉化
    dom = "Bfrtp", # 去掉显示行数 i、过滤 f 的能力，翻页用 p 表示
    ordering = F, # 去掉列排序
    buttons = c("copy", "csv", "excel", "print"), # 提供打印按钮
    rowGroup = list(dataSrc = 0), # 按 Package 列分组
    columnDefs = list(
      list(className = "dt-center", targets = 0), # 不显示行名，则 targets 从 0 开始，否则从 1 开始
      list(visible = FALSE, targets = 0) # 不显示 Package 列
    )
  )
```

表格 14.2: Base R 包内置的数据集

Copy CSV Excel Print Search:

Item	Title
datasets	
AirPassengers	Monthly Airline Passenger Numbers 1949-1960
BJsales	Sales Data with Leading Indicator
BJsales.lead (BJsales)	Sales Data with Leading Indicator
BOD	Biochemical Oxygen Demand
CO2	Carbon Dioxide Uptake in Grass Plants
ChickWeight	Weight versus age of chicks on different diets
DNase	Elisa assay of DNase
EuStockMarkets	Daily Closing Prices of Major European Stock Indices, 1991-1998
Formaldehyde	Determination of Formaldehyde
HairEyeColor	Hair and Eye Color of Statistics Students

Previous 1 2 3 4 5 … 37 Next

)

14.2 扩展功能

14.2.1 汉化表格

14.2.2 下载数据

14.3 其它工具

© 黄湘云

第十五章　交互应用

```
library(shiny)
```

一个简单示例，介绍一个 Shiny 应用的各个常见组成部分。一个快速改变风格的主题包。介绍交互表格、交互图形与 Shiny 集成，如 **DT**、**plotly**、**leaflet** 等。介绍 Shiny 工业化应用的开发过程。

15.1　简单示例

```
library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "n", label = "观测记录的数目",
              min = 1, max = nrow(faithful), value = 100),
  plotOutput("plot")
)

server <- function(input, output) {
  output$plot <- renderPlot({
    hist(faithful$eruptions[seq_len(input$n)],
      breaks = 40,
      main = "美国黄石公园喷泉",
      xlab = "喷发持续时间"
    )
  })
}

shinyApp(ui, server)
```

©黄湘云

15.1.1 UI 前端

15.1.2 Server 后端

15.2 Shiny 组件

组件又很多，下面想重点介绍 4 个，它们使用频次很高，很有代表性。

15.2.1 筛选器

单个筛选器、独立筛选器、筛选器联动

15.2.2 输入框

数值型、文本型

15.2.3 动作按钮

提交按钮、响应按钮

15.2.4 书签

书签记录输入状态，链接可以指向页面状态

```
library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "n", label = "观测记录的数目",
              min = 1, max = nrow(faithful), value = 100),
  plotOutput("plot"),
  bookmarkButton(id = "bookmark1", label = "书签", title = "记录、分享此时应用的状态")
)

server <- function(input, output) {
  output$plot <- renderPlot({
    hist(faithful$eruptions[seq_len(input$n)],
      breaks = 40,
      main = "美国黄石公园喷泉",
      xlab = "喷发持续时间"
```

©黄湘云


```
    )
  })
}

enableBookmarking(store = "url")
shinyApp(ui, server)
```

15.3 Shiny 扩展

页面布局

- shinydashboard / shinydashboardPlus Shiny 应用
- flexdashboard R Markdown 文档中制作 Shiny 应用
- bs4Dash

交互表格

- DT
- reactable

交互图形

- plotly
- ggiraph

15.3.1 页面布局

15.3.2 交互表格

下面在 Shiny 应用中插入 DT 包制作的交互表格

```
# 前端
library(shiny)
ui <- fluidPage(
  # 应用的标题名称
  titlePanel("鸢尾花数据集"),
  # 边栏
  fluidRow(
    column(12, DT::dataTableOutput("table"))
  )
)

# 服务端
```

©黄湘云

```
server <- function(input, output, session) {
  output$table <- DT::renderDataTable(iris,
    options = list(
      pageLength = 5, # 每页显示5行
      initComplete = I("function(settings, json) {alert('Done.');}")
    ), server = F
  )
}

shinyApp(ui, server)
```

> **❗ 重要**
>
> 加载 shiny 包后再加载 DT 包，函数 `dataTableOutput()` 和 `renderDataTable()` 显示冲突，因为两个 R 包都有这两个函数。在创建 shiny 应用的过程中，如果我们需要呈现动态表格，就需要使用 DT 包的 `DT::dataTableOutput()` 和 `DT::renderDataTable()` ，否则会报错，详见 https://github.com/rstudio/shiny/issues/2653。

reactable 基于 JS 库 React Table 提供交互式表格渲染，和 **shiny** 无缝集成，是替代 **DT** 的不二选择，在 app.R 用 reactable 包的 `reactableOutput()` 和 `renderReactable()` 函数替代 **shiny** 里面的 `dataTableOutput()` 和 `renderDataTable()`。再也不用忍受 **DT** 和 **shiny** 的函数冲突了，且其覆盖测试达到 99%。

```
library(shiny)
```

下面在 Shiny 应用中插入 **reactable** 包制作的交互表格

```
library(shiny)
library(reactable)

ui <- fluidPage(
  reactableOutput("table")
)

server <- function(input, output) {
  output$table <- renderReactable({
    reactable(iris,
      filterable = TRUE, # 过滤
      searchable = TRUE, # 搜索
      showPageSizeOptions = TRUE, # 页面大小
      pageSizeOptions = c(5, 10, 15), # 页面大小可选项
      defaultPageSize = 10, # 默认显示10行
```

© 黄湘云


```
      highlight = TRUE, # 高亮选择
      striped = TRUE, # 隔行高亮
      fullWidth = FALSE, # 默认不要全宽填充，适应数据框的宽度
      defaultSorted = list(
        Sepal.Length = "asc", # 由小到大排序
        Petal.Length = "desc" # 由大到小
      ),
      columns = list(
        Sepal.Width = colDef(style = function(value) {
          # Sepal.Width 添加颜色标记
          if (value > 3.5) {
            color <- "#008000"
          } else if (value > 2) {
            color <- "#e00000"
          } else {
            color <- "#777"
          }
          list(color = color, fontWeight = "bold") # 字体加粗
        })

      )
    )
  })
}

shinyApp(ui, server)
```

除了 **DT** 和 **reactable** 包，其它支持 Shiny 集成的 R 包还有 gt 、formattable 和 **kableExtra** 等。

15.3.3 交互图形

ggiraph 包

15.4 Shiny 仪表盘

dashboard 翻译过来叫仪表盘，就是驾驶仓的那个玩意，形象地表达作为掌舵者应该关注的对象。R 包 shiny 出现后，仪表盘的制作显得非常容易，也很快形成了一个生态，比如 shinydashboard、flexdashboard 等，此外 bs4Dash 基于 Bootstrap 4 的仪表盘，目前 shiny 和 rmarkdown 都在向 Bootstrap 4 升级，这是未来的方向。shinydashboardPlus 主要目的在于扩展 **shinydashboard** 包

©李东风

15.4.1 shinydashboard 包

将如下内容保存为 app.R 文件。

```
library(shiny)
library(shinydashboard)
ui <- dashboardPage(
  dashboardHeader(title = "Basic dashboard"),
  ## 边栏
  dashboardSidebar(
    sidebarMenu(
      menuItem("Dashboard", tabName = "dashboard", icon = icon("dashboard")),
      menuItem("Widgets", tabName = "widgets", icon = icon("th"))
    )
  ),
  ## 主体内容
  dashboardBody(
    tabItems(
      # 第一个 Tab 页内容
      tabItem(
        tabName = "dashboard",
        fluidRow(
          box(plotOutput("plot1", height = 250)),
          box(
            title = "Controls",
            sliderInput("slider", "Number of observations:", 1, 100, 50)
          )
        )
      ),

      # 第二个 Tab 页内容
      tabItem(
        tabName = "widgets",
        h2("Widgets tab content")
      )
    )
  )
)

server <- function(input, output) {
```

©清华大学出版社

```
  set.seed(122)
  histdata <- rnorm(500)

  output$plot1 <- renderPlot({
    data <- histdata[seq_len(input$slider)]
    hist(data)
  })
}

shinyApp(ui, server)
```

15.4.2 shinydashboardPlus 包

shinydashboardPlus 包的函数 descriptionBlock()

```
library(shiny)
library(shinydashboard)
library(shinydashboardPlus)

shinyApp(
  ui = dashboardPage(
    dashboardHeader(),
    dashboardSidebar(),
    dashboardBody(
      box(
        solidHeader = FALSE,
        title = "状态概览",
        background = NULL,
        width = 4,
        status = "danger",
        footer = fluidRow(
          column(
            width = 6,
            descriptionBlock(
              number = "17%",
              numberColor = "green",
              numberIcon = "fa fa-caret-up",
              header = "$35,210.43",
              text = "总收入",
              rightBorder = TRUE,
```

©清华大学出版社

```
            marginBottom = FALSE
          )
        ),
        column(
          width = 6,
          descriptionBlock(
            number = "18%",
            numberColor = "red",
            numberIcon = "fa fa-caret-down",
            header = "1200",
            text = "目标完成",
            rightBorder = FALSE,
            marginBottom = FALSE
          )
        )
      )
    )
  ),
  title = "Description Blocks"
),
server = function(input, output) { }
)
```

15.4.3 bs4Dash 包

```
library(bs4Dash)
ui <- dashboardPage(
  dashboardHeader(title = "Basic dashboard"),
  dashboardSidebar(),
  dashboardBody(
    # Boxes need to be put in a row (or column)
    fluidRow(
      box(plotOutput("plot1", height = 250)),

      box(
        title = "Controls",
        sliderInput("slider", "Number of observations:", 1, 100, 50)
      )
    )
```

©清华大学出版社

```
  )
)

server <- function(input, output) {
  set.seed(122)
  histdata <- rnorm(500)

  output$plot1 <- renderPlot({
    data <- histdata[seq_len(input$slider)]
    hist(data)
  })
}

shinyApp(ui, server)
```

15.4.4 miniUI 包

miniUI 包制作迷你版 Shiny 应用，适用于小屏幕显示。

```
library(shiny)
library(miniUI)
library(leaflet)
library(ggplot2)

ui <- miniPage(
  gadgetTitleBar("Shiny gadget example"),
  miniTabstripPanel(
    miniTabPanel(title = "参数",
      icon = icon("sliders"),
      miniContentPanel(
        sliderInput("year", "年份", 1978, 2010, c(2000, 2010), sep = "")
      )
    ),
    miniTabPanel(title = "可视化",
      icon = icon("area-chart"),
      miniContentPanel(
        plotOutput("quakes", height = "100%")
      )
    ),
    miniTabPanel(title = "地图",
```

©清华大学出版社

```
      icon = icon("map-o"),
      miniContentPanel(
        padding = 0,
        leafletOutput("map", height = "100%")
      ),
      miniButtonBlock(
        actionButton("resetMap", "Reset")
      )
    ),
    miniTabPanel(title = "数据",
      icon = icon("table"),
      miniContentPanel(
        DT::dataTableOutput("table")
      )
    ),
    selected = "Map"
  )
)

server <- function(input, output, session) {
  output$quakes <- renderPlot({
    ggplot(quakes, aes(long, lat)) +
      geom_point()
  })

  output$map <- renderLeaflet({
    force(input$resetMap)

    leaflet(quakes, height = "100%") |>
      addTiles() |>
      addMarkers(lng = ~long, lat = ~lat)
  })

  output$table <- DT::renderDataTable({
    quakes
  })

  observeEvent(input$done, {
    stopApp(TRUE)
  })
```

©袁清

```
}

shinyApp(ui, server)
```

15.5 Shiny 主题

15.5.1 bslib 包

- bslib

15.5.2 shinymaterial 包

shinymaterial 包实现 Material Design

```
library(shiny)
library(shinymaterial)

ui <- material_page(
  title = "用户画像",
  nav_bar_fixed = TRUE,
  # 每个 sidebar 内容
  material_side_nav(
    fixed = TRUE,
    # Place side-nav tabs within side-nav
    material_side_nav_tabs(
      side_nav_tabs = c(
        "数据汇总" = "tab_1",
        "趋势信息" = "tab_2"
      ),
      icons = c("cast", "insert_chart")
    )
  ),
  # 每个 tab 页面的内容
  material_side_nav_tab_content(
    side_nav_tab_id = "tab_1",
    tags$h2("第一个tab页")
  ),
  material_side_nav_tab_content(
    side_nav_tab_id = "tab_2",
```

©黄湘云

```
    tags$h2("第二个tab页")
  )
)

server <- function(input, output) {

}
shinyApp(ui = ui, server = server)
```

15.6 Shiny 优化

提升 shiny 仪表盘访问性能的 4 个建议

15.7 Shiny 部署

15.7.1 promises 并发

shiny 异步编程实现并发访问，多人同时访问 Shiny 应用的情况下，解决必须等另一个人完成访问的情况下才能继续访问的问题。

```
library(shiny)
library(future)
library(promises)

plan(multiprocess)

ui <- fluidPage(
  h2("测试异步下载"),
  tags$ol(
    tags$li("Verify that plot appears below"),
    tags$li("Verify that pressing Download results in 5 second delay, then rock.csv being downloaded"),
    tags$li("Check 'Throw on download?' checkbox and verify that pressing Download results in 5 second d
  ),
  hr(),
  checkboxInput("throw", "Throw on download?"),
  downloadButton("download", "下载 (等待5秒)"),
  plotOutput("plot")
)
```

© 华章出版社


```
server <- function(input, output, session) {
  output$download <- downloadHandler("rock.csv", function(file) {
    future({Sys.sleep(5)}) %...>%
      {
        if (input$throw) {
          stop("boom")
        } else {
          write.csv(rock, file)
        }
      }
  })

  output$plot <- renderPlot({
    plot(cars)
  })
}

shinyApp(ui, server)
```

15.8 Shiny 替代品

R Markdown + Shiny 文档

- crosstalk 交互
- flexdashboard 布局
- DT 交互表格
- leaflet 交互地图
- ggiraph 交互图形

Quarto Dashboard 文档

15.9 Shiny 案例

- radiant 探索性数据分析解决方案

15.10 总结

事实上，作为 BI 工程师，相当一部分工作是与数据开发结合的。从 Kafka 接入埋点上报的原始日志（ODS 层）、清洗抽取特定业务/领域内的数据（Fact 事实层）、面向某一类任务的主题数据（topic 主题

© 基础设施

层)、面向特定数据产品的应用数据（app 应用层)。

- 数据仓库 Hive 数据开发：事实、主题和应用层
- 数据计算 Spark 数据开发工具 Spark SQL / Hive SQL 任务调度
- 数据报表 MySQL / Doris 数据同步工具 Hive2MySQL 同步应用层数据
- 数据展示 Dashboard 应用开发工具 Shiny RStudio Shiny Server

报表开发从数据仓库的 DWD 层开始，可能一些业务原因，我们需要从 ODS 层甚至从点击流的日志数据开始，经过数据清洗、提取、聚合成为支撑 BI 报表最底层的基础表，存储在 Hive 中，然后对这一系列的基础表根据 BI 展示的需要进行第二层聚合形成中间表，这两层数据根据业务情况做增量更新或者全量更新，并将中间表同步到 MySQL 仓库中，全量更新的情况，往往更新数据比较大，建议用 sqoop 做数据的同步。创建第二层的中间表稍有些灵活性，原则是在中间表之上对应的数据操作和可视化是容易实现且效率较高的，否则应该构造第三层的中间表，绝不能将大规模的数据集直接导入 R 中进行分析和可视化，拖慢前端展示的速度，占用过多的服务器资源。

图 15.1: Shiny 生态系统

- 连接数据库。根据数据库的情况选择相应的 R 接口包，比如连接 MySQL 数据库可以用 RMySQL 包，值得一提，odbc 包支持连接相当多的数据库。
- 数据操作。根据需要处理的数据规模，可以选择 Base R、data.table 或者 dplyr 做数据操作，推荐和管道操作一起使用，增加代码可读性。
- 交互表格。推荐 reactable 和 DT 包做数据呈现。
- 交互图形。推荐功能强大的 plotly 包，可以先用 ggplot2 绘制，然后调用 plotly 包的 `ggplotly()` 函数将静态图转化为交互图。

© 黄湘云

- 针对特定应用场景的其它交互可视化工具包，比如 leaflet 可以将地图嵌入 Shiny 应用，dygraphs 可以将时间序列塞进去。
- Shiny 组件。shinyFeedback 提供用户输入的反馈。shinyWidgets 提供自定义 widget 的功能。miniUI 专为小屏设计，shinyMobile 在 IOS 和安卓手机上访问 shiny 应用。
- Shiny 主题。比如 shinythemes 包可以统一配色，dashboardthemes 提供更加深度的主题，shinytableau 提供仿 Tableau 的 dashboard 框架。sass 在 CSS 样式层面重定义风格。bslib 通过 Bootstrap 3/4/5 定制 Shiny 和 R Markdown 主题。
- Shiny 权限。shinymanager / shinyauthr 支持单个 shiny 应用的权限管理，firebase 提供访问权限设置 https://firebase.john-coene.com/。
- Shiny 框架。ShinyStudio 打造基于容器架构的协作开发环境的开源解决方案，golem 构建企业级 shiny 应用的框架，RinteRface 开发的系列 R 包也试图打造一套完整的解决方案，并配有速查小抄 cheatsheets。
- Shiny 部署。shiny-server 以网络服务的方式支持 shiny 应用，shinyproxy 提供企业级部署 shiny 应用的开源解决方案。

自 RStudio 推出 Shiny 系列产品以来，一些公司进一步根据所需扩展和定制，比如 Appsilon、RinteRface、ThinkR-open、dreamRs 和datastorm-open 等。经过商业公司和个人开发者的努力，Shiny 生态非常庞大，资源非常丰富。

- Shiny 入门 https://shiny.posit.co/r/getstarted/。
- Shiny 扩展包 https://github.com/nanxstats/awesome-shiny-extensions。
- Shiny 常用技巧和提示 https://github.com/daattali/advanced-shiny。
- Shiny 各类资源列表 https://github.com/grabear/awesome-rshiny。

特别值得一提，Shiny 方面的三本专著。

- Hadley Wickham 的书 Mastering Shiny。
- Colin Fay, Sébastien Rochette, Vincent Guyader, Cervan Girard 的书 Engineering Production-Grade Shiny Apps。
- David Granjon 的书 Outstanding User Interfaces with Shiny。

©黄湘云

第十六章　HTML 文档

从本章开始，接下来的三个章节都围绕数据交流的工具及其案例展开。日常工作中，有的需要产出具有丰富交互内容的网页文档（HTML 文档），有的需要产出排版精美、符合格式要求的、可打印的便携式文档（PDF 文档），有的需要可协作共享、可编辑修改的办公文档（Office 文档）。在 R 语言社区，陆续出现两套解决方案，一个是以 rmarkdown 包为核心的 R Markdown 生态，另一个是以 Quarto 为核心的文档写作和发布系统。继 R Markdown 出现 10 余年后，2022 年 RStudio 公司发布 Quarto 系统，整合 R Markdown 生态，提供统一的语法。截止写作时间，相比于成熟的 R Markdown 生态，Quarto 系统还在路上。因此，不拘泥于 R Markdown 还是 Quarto，根据使用场景、实践经验、工具现状，选择最合适的工具介绍，整体上，以 Quarto 为主，R Markdown 补位的方式介绍。

16.1　文档元素

无论是 R Markdown 还是 Quarto，都是站在巨人 Pandoc 的肩膀上，Pandoc 在普通 Markdown 的基础上提供了许多扩展支持，通过一些简单的标记，大大丰富了文档内容，下面介绍的内容适用于 R Markdown 和 Quarto，无论文档最终的输出格式如何。

16.1.1　样式

文字样式，如加粗、倾斜、上下标等。

Markdown 语法	输出
`*斜体*，**加粗**，***粗斜体***`	斜体，**加粗**，***粗斜体***
`上角标^2^ / 下角标~2~`	上角标^{2} / 下角标$_{2}$
`~~删除线~~`	~~删除线~~
`` `代码` ``	`代码`

16.1.2　图片

其一插入现成的图片，其二插入代码生成的图片

© 黄湘云

(a) versicolor 杂色鸢尾

(b) setosa 山鸢尾

(c) virginica 弗吉尼亚鸢尾

图 16.1: 三种鸢尾花

```
flowchart LR
  A[Hard edge] --> B(Round edge)
  B --> C{Decision}
  C --> D[Result one]
  C --> E[Result two]
```

图 16.2: 流程图

ggplot2 绘制的图形

```
library(ggplot2)
ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point(aes(color = Species)) +
  theme_classic()
```

© 谢益辉

图 16.3: 一幅简单的 ggplot2 图形

16.1.3 表格

Markdown 原生支持的表格和 **knitr** 包制作的表格。

表格 16.2: 鸢尾花数据集

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

```
knitr::kable(head(iris, 3))
```

表格 16.3: 鸢尾花数据集

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

© 吉祥出版

16.1.4 列表

常见的列表有无序列表、有序列表及其嵌套。

表格 16.4: 几种列表

Markdown 语法	输出

`* 无序列表`
`  + 子条目 1`
`  + 子条目 2`
`  - 子子条目 1`
`*  条目 2`
 	
`  继续（缩进 4 格）`
	• 无序列表
	  – 子条目 1
	  – 子条目 2
	  * 子子条目 1
	• 条目 2
	  继续 (缩进 4 格)

`1. 有序列表`
`2. 条目 2`
`  i) 子条目 1`
`  A. 子子条目 1`
	1. 有序列表
	2. 条目 2
	  i) 子条目 1
	  A. 子子条目 1

`(@) 第一个人是好的`
 	
`第二个人是坏的`
 	
`(@) 第三个人是丑陋的`
	(1) 第一个人是好的
	第二个人是坏的
	(2) 第三个人是丑陋的

`::: {}`
`1. 一个列表`
`:::`
 	
`::: {}`
`1. 又一个列表`
`:::`
	1. 一个列表
	1. 又一个列表

`术语` `: 定义`	**术语** 定义

在 (@) 中添加标识符，如 (@good) 就可以引用列表中的条目 (1)。

16.1.5 引用

除了引用外部书籍、文章、刊物等的内容，还有长文档内部的交叉引用，这项功能是非常需要的，涉及图、表、公式、定理，参考文献，列表条目等。

16.1.6 脚注

> If you imagine that this pen is Trellis, then Lattice is not this pen.[1]
>
> — Paul Murrell

16.1.7 公式

公式分两种情况，其一是行内公式，其二是行间公式。前者一对美元符号夹住数学公式，美元符号与字母之间不能有空格，比如 `$\beta$` 渲染出来的效果是 β 。后者是两对美元符号夹住公式，比如 `$$ \beta$$` 渲染出来的效果如下：

$$\beta$$

行内公式一般用来写数学符号，行间公式一般用来排版数学公式，特别是多行公式。行间公式可以编号，也可以不编号，编号通常是了交叉引用。

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

排版行间公式有很多不同的 LaTeX 环境，最常见的有两种，一种是多个公式逐行排，一种是长公式折行，常常都要求对齐。举例来说，线性模型的两种表示方式，一种是矩阵向量式，一种是数据结构式，见方程式 16.1 。

$$\begin{aligned} \mathbf{y} &= X\boldsymbol{\beta} + \boldsymbol{\epsilon} \\ y_i &= \mathbf{x}_i\boldsymbol{\beta} + \epsilon_i \end{aligned} \tag{16.1}$$

在行间公式中，使用 `split` 公式环境排版一个长公式，这个公式是折成多行的，表达一个计算过程。举例来说，线性模型回归系数的最小二乘估计 $\hat{\boldsymbol{\beta}}$ 的方差的计算过程，见方程式 16.2 。

$$\begin{aligned} \mathsf{Var}\{\hat{\boldsymbol{\beta}}\} &= \mathsf{Var}\{(X^\top X)^{-1}X^\top\mathbf{y}\} \\ &= (X^\top X)^{-1}X^\top\mathsf{Var}\{\mathbf{y}\}\big((X^\top X)^{-1}X^\top\big)^\top \\ &= (X^\top X)^{-1}X^\top\mathsf{Var}\{\mathbf{y}\}X(X^\top X)^{-1} \\ &= (X^\top X)^{-1}X^\top\sigma^2 IX(X^\top X)^{-1} \\ &= (X^\top X)^{-1}\sigma^2 \end{aligned} \tag{16.2}$$

值得注意，

1. LaTeX 命令 `\mathbf` 只对英文字母 a, b, c, A, B, C 加粗，对希腊字母 $\theta, \alpha, \beta, \ldots, \gamma$ 加粗应该使用命令 `\boldsymbol`。

[1] (on the difference of Lattice (which eventually was called grid) and Trellis) DSC 2001, Wien (March 2001)

©黄湘云

2. Quarto 文档中将行间公式中成对 $$ 转化为 LaTeX 中的 equation 环境。Quarto 不支持在多行公式逐行编号，也不支持在多行公式中对某一（些）行编号。而在 LaTeX 文档中，这些全都支持，可以说公式排版是 LaTeX 最突出的优势。

3. MathJax 支持公式宏定义，如定义命令 \bm 对希腊字母加粗。在 Quarto 文档中插入如下代码，用命令 \boldsymbol 定义一个新的命令 \bm，这种做法很常见，用来简少公式排版的工作量。

```
$$
\def\bm#1{{\boldsymbol #1}}
$$
```

16.2 制作报告

Quarto Report 文档

16.2.1 SQL 查询

```
library(DBI)
conn <- DBI::dbConnect(RSQLite::SQLite(),
  dbname = system.file("db", "datasets.sqlite", package = "RSQLite")
)
```

Base R 内置的数据集都整合进 RSQLite 的样例数据库里了，

```
dbListTables(conn)
```

```
 [1] "BOD"              "CO2"              "ChickWeight"      "DNase"
 [5] "Formaldehyde"     "Indometh"         "InsectSprays"     "LifeCycleSavings"
 [9] "Loblolly"         "Orange"           "OrchardSprays"    "PlantGrowth"
[13] "Puromycin"        "Theoph"           "ToothGrowth"      "USArrests"
[17] "USJudgeRatings"   "airquality"       "anscombe"         "attenu"
[21] "attitude"         "cars"             "chickwts"         "esoph"
[25] "faithful"         "freeny"           "infert"           "iris"
[29] "longley"          "morley"           "mtcars"           "npk"
[33] "pressure"         "quakes"           "randu"            "rock"
[37] "sleep"            "stackloss"        "swiss"            "trees"
[41] "warpbreaks"       "women"
```

随意选择 5 行数据记录，将结果保存到变量 iris_preview

```
SELECT * FROM iris LIMIT 5;
```

查看变量 iris_preview 的内容

```
iris_preview
  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         0.2  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
```

结束后关闭连接

```
dbDisconnect(conn = conn)
```

16.3 制作演示

Quarto Presentation

16.4 编写书籍

Quarto Book 网页格式

© 高等教育出版社

第十七章　PDF 文档

在国内外，有不少使用场景需要用到 LaTeX 排版，在期刊论文、毕业论文、毕业答辩、学术报告、课程作业、课程笔记、学术专著等科技写作方面，LaTeX 编辑的 PDF 文档。近 10 年来，可重复性研究成为一个热门的话题，不少权威期刊的投稿论文要求公开数据处理的过程，图表的生成代码等。走在前列的《R Journal》杂志，采用 R Markdown 投稿，整篇论文可以一键生成。经过 10 年的发展，R Markdown 生态已经比较成熟，为了进一步降低学习和使用的成本，基于 Pandoc ，Quarto 统一了论文排版，学术论文、演示报告等应用场景，提供一整套科技写作的解决方案。LaTeX 、R Markdown 和 Quarto 建立了紧密的联系，Pandoc 在其间起了非常重要的桥梁作用。Pandoc 将 Markdown 语法转为 LaTeX 语法，Pandoc 在 R Markdown 和 Quarto 中的作用类似。

17.1　LaTeX 基础

LaTeX 是一个非常方便用户使用的排版工具，提供一套精确的编程语言，下面是一个简单示例。短短 14 行代码展示了大量的常用功能，生成文章标题、作者、目录，设置文档布局、排版公式、交叉引用等。

```
\documentclass[b5paper]{article}
\usepackage[heading=true, UTF8]{ctex} % 设置中文环境
\usepackage{amsmath,bm} % 处理数学公式
\title{LaTeX 入门}
\author{张三}
\begin{document}
\maketitle
\tableofcontents
\section{线性模型} \label{sec:lm}
第 \ref{sec:lm} 节介绍线性模型，线性模型的矩阵表示见公式 \ref{eq:lm} 。
\begin{align} \label{eq:lm}
\bm{\mathsf{y}} = \bm{\mathsf{X}}\bm{\beta} + \bm{\epsilon}
\end{align}
\end{document}
```

接下来，逐行解释上面的 LaTeX 代码。

- \documentclass 命令用来加载文类，常用的有 article、report、book 等，文类的选项 b5paper 表示布局为 B5 纸。
- \usepackage 命令用来加载 LaTeX 宏包，上面的第 2 行设置中文环境，加载了 ctex 宏包，并设置了两个选项 heading=true 和 UTF8 。
- \title 和 \author 命令分别用来设置文档标题和作者。\documentclass 和 \begin{document} 之间的部分叫导言区，常常用来加载宏包和自定义 LaTeX 命令。\begin{document} 和 \end{document} 之间的部分叫正文。
- \maketitle 和 \tableofcontents 命令分别用来生成标题和文档目录。\section 命令设置小节的标题。\label 命令设置小节标签，用于交叉引用。
- \begin{align} 和 \end{align} 是一个公式环境，其间的命令 \bm 来自 bm 宏包，用于加粗数学符号，命令 \mathsf 、\beta 和 \epsilon 都来自 amsmath 宏包。
- \begin{align} 之后的命令 \label{eq:lm} 设置公式标签，eq:lm 是用户指定的唯一标识符，不同公式不能重复使用同一标签，\ref{eq:lm} 在正文中交叉引用公式。

所有的 LaTeX 命令都是以反斜杠 \ 开头的。文类和红包的选项说明可查看其帮助文档。

17.1.1 中英字体

大部分情况下，加载 **ctex** 宏包就够了，但也有的场景需要使用特定的中文字体，比如学位论文排版、项目申请书等，这些对文档格式有极其严格的要求。此时，可以在导言区使用 **xecjk** 宏包配置字体，或者加载 **ctex** 宏包时添加选项 fontset=none ，加载 **ctex** 宏包会自动加载 **xecjk** 宏包。

```
\usepackage[heading=true, fontset=none, UTF8]{ctex} % 设置中文环境
```

下面的代码表示在 LaTeX 文档里使用黑体、宋体、仿宋、楷体四款中文字体。正文字体是宋体，中文没有斜体，倾斜中文使用楷体，加粗中文使用黑体，等宽字体使用仿宋。

```
\setCJKmainfont[ItalicFont={KaiTi_GB2312}, BoldFont={SimHei}]{SimSun}
\setCJKsansfont{SimHei}
\setCJKmonofont{FangSong_GB2312}
% 黑体
\setCJKfamilyfont{heiti}{SimHei}
\newcommand{\heiti}{\CJKfamily{heiti}}
% 楷体 GB2312
\setCJKfamilyfont{kaishu}{KaiTi_GB2312}
\newcommand{\kaishu}{\CJKfamily{kaishu}}
% 宋体
\setCJKfamilyfont{songti}{SimSun}
\newcommand{\songti}{\CJKfamily{songti}}
% 仿宋 GB2312
\setCJKfamilyfont{fangsong}{FangSong_GB2312}
\newcommand{\fangsong}{\CJKfamily{fangsong}}
```

© 吉祥图书

LaTeX 提供很多字体宏包，支持英文字体、数学字体单独设定。在加载 **amsmath** 宏包后，加载 **mathpazo** 设置数学字体，加载 **palatino** 设置正文中的英文字体，加载 **courier** 设置代码中的等宽字体，加载 **fontenc** 设置字体编码方式。确保已安装 **dvips** 宏包，它用来处理字体文件。

```
\usepackage{mathpazo} % 数学符号
\usepackage{palatino} % 英文衬线字体
\usepackage{courier}  % 英文无衬线字体
\usepackage[T1]{fontenc} % 字体编码 T1
```

17.1.2 数学公式

排版数学公式分三部分，其一是排版的环境，其二是使用的符号、其三是使用的字体。公式环境都是由成对的命令组成，前面已经提及 align 环境，这是一个可对公式编号的适用于对齐多行公式的排版环境。

表格 17.1: LaTeX 公式排版环境

可编号	无编号	作用
align	align*	多行公式对齐
equation	equation*	可与 split / cases 等环境嵌套使用
multline	multline*	长公式折行
gather	gather*	多行公式居中

不可编号的排版环境，行内公式排版，用一对美元符号 `$ $` 或一对小括号 `\( \)`。行间公式排版，用一对双美元符号 `$$ $$` 或一对中括号 `\[ \]` 。

LaTeX 支持丰富的数学符号大、小写英文字母，大、小写希腊字母，字母可以加粗、倾斜，字母也可以设置为等宽字体或衬线字体，还可以设置花体、空心体等。一些常用的数学符号样式见下表。

大写	小写	加粗	无衬线
X	x	$\boldsymbol{X}$	X
衬线	花体	空心体	花体
X	$\mathcal{X}$	$\mathbb{X}$	$\mathscr{X}$
大写	小写	加粗	无衬线
Γ	γ	$\boldsymbol{\gamma}$	$\mathsf{\Gamma}$

```
\[
\Bigg(\sqrt{\frac{M}{1 - \big(\frac{r}{\widetilde{x_1 + \cdots + u_N}} \big)^2}
\big(\sum_{\beta =1}^{N} \sum_{i=1}^{n}\frac{\partial u_{\beta}}{\partial x_i} + 1 \big) }
+ \sqrt{XY} \Bigg)^3
\]
```

amsmath 宏包渲染效果如下：

$$\Bigg(\sqrt{\frac{M}{1-\big(\frac{r}{\widetilde{x_1+\cdots+u_N}}\big)^2}\big(\sum_{\beta=1}^{N}\sum_{i=1}^{n}\frac{\partial u_\beta}{\partial x_i}+1\big)}+\sqrt{XY}\Bigg)^3$$

newtxtext 和 **newtxmath** 宏包常组合在一起，提供一套 New Times 字体风格的文本和数学公式，一种介于。

```
\documentclass[b5paper]{article}
\usepackage{amsmath}
\usepackage{newtxtext,newtxmath}
\begin{document}
\[
\Bigg(\sqrt{\frac{M}{1 - \big(\frac{r}{\widetilde{x_1 + \cdots + u_N}} \big)^2}
\big(\sum_{\beta =1}^{N} \sum_{i=1}^{n}\frac{\partial u_{\beta}}{\partial x_i} + 1 \big) }
+ \sqrt{XY} \Bigg)^3
\]
\end{document}
```

newtxmath 宏包渲染效果如下：

$$\Bigg(\sqrt{\frac{M}{1-\big(\frac{r}{\widetilde{x_1+\cdots+u_N}}\big)^2}\big(\sum_{\beta=1}^{N}\sum_{i=1}^{n}\frac{\partial u_\beta}{\partial x_i}+1\big)}+\sqrt{XY}\Bigg)^3$$

图 17.1: newtxmath 包渲染的公式效果

17.1.3 代码抄录

`verbatim` 环境是用来抄录代码的。

```
\begin{verbatim}
library(stats) % 提供 lowess, rpois, rnorm 等函数
library(graphics) % 提供 plot 方法
plot(cars)
lines(lowess(cars))
\end{verbatim}
```

渲染出来的效果如下：

© 黄湘云


```
library(stats) % 提供 lowess, rpois, rnorm 等函数
library(graphics) % 提供 plot 方法
plot(cars)
lines(lowess(cars))
```

图 17.2: verbatim 抄录环境

listings 宏包提供丰富的配置，下面在导言区设置代码字体样式和大小，代码块的背景、代码块的行号。

```
\usepackage{xcolor}
\definecolor{shadecolor}{rgb}{.97, .97, .97}
\usepackage{listings}
\lstset{
  basicstyle=\ttfamily, % 代码是等宽字体
  backgroundcolor=\color{shadecolor},  % 代码块的背景颜色
  breaklines=true, % 可以段行
  numbers=left,    % 行序号
  numberstyle=\footnotesize, % 行序号字体大小
  commentstyle=\ttfamily     % 注释是等宽字体
}
```

启用 `lstlisting` 环境抄录代码，设置参数 `language=R` 指定抄录环境中的编程语言类型，以便提供语法高亮。

```
\begin{lstlisting}[language=R]
library(stats)    % 提供 lowess, rpois, rnorm 等函数
library(graphics) % 提供 plot 方法
plot(cars)
lines(lowess(cars))
\end{lstlisting}
```

渲染出来的效果如下：

```
library(stats)    % 提供 lowess, rpois, rnorm 等函数
library(graphics) % 提供 plot 方法
plot(cars)
lines(lowess(cars))
```

图 17.3: lstlisting 抄录环境

©梦溪笔谈

17.1.4 插入图表

首先在导言区加载 graphicx 宏包，然后可以使用 `\includegraphics` 命令插入图片，该命令有一些选项，`[width=.65\textwidth]` 表示插入的图片占页面宽度的 65%。

```
\usepackage{graphicx}
```

在正文中 figure 环境是专门用来处理的图片，选项 `[h]` 表示将图片就插入此处，不要浮动。center 环境将图片居中，`\caption` 和 `\label` 命令分别用来指定图片的标题和标签。

```
\begin{figure}[h]
  \begin{center}
    \includegraphics[width=.65\textwidth]{images/peaks.png}
    \caption{图片的标题}
    \label{fig:figure}
  \end{center}
\end{figure}
```

渲染效果如下

图 17.4: 图片的标题

table 环境用于制作控制表格位置，tabular 用于制作表格，控制表头、每个列和每个格子。

```
\begin{table}[h!]
  \begin{center}
    \begin{tabular}{|c c c|}
     \hline
      列1 & 列2 & 列3 \\
      \hline
      1 & 6 & 77 \\
      2 & 7 & 15 \\
      3 & 8 & 44 \\
```

©黄沛文

```
      \hline
    \end{tabular}
  \caption{表格的标题}
  \label{tbl:table}
  \end{center}
\end{table}
```

`\begin{table}[h!]` 表格环境中 `[h!]` 让表格不要浮动，center 环境使表格居中，`\begin{tabular}{|c c c|}` 表格各个列的元素居中，表格整体封闭，`\hline` 制作水平线，`\\` 用于换行，`&` 用于表格格子对齐，`\caption` 添加表格的标题，`\label` 添加表格标识符，以便后续引用。

渲染出来的效果如下：

表格 17.3: 表格的标题

列 1	列 2	列 3
1	6	77
2	7	15
3	8	44

17.1.5 交叉引用

`\ref` 命令用于图、表、公式、章节的交叉引用，引用图片 `\ref{fig:figure}` 、引用表格 `\ref{tbl:table}` 、引用公式 `\ref{eq:lm}` 等。

`\cite` 命令用于参考文献的引用。

17.1.6 PDF-A/X

启用 PDF/X 或 PDF/A 标准，导言区加载 pdfx 宏包

```
% PDF/A-1b 标准
\usepackage[a-1b]{pdfx}
```

示例文档内容如下：

```
\documentclass[b5paper]{article}
\usepackage{amsmath} % boldsymbol
\usepackage[a-1b]{pdfx}
\title{Math in LaTeX}
\author{Zhang San}
\begin{document}
\maketitle
```

张三 ©

```
\tableofcontents
\section{Math}
\begin{align}
\boldsymbol{x},\mathbf{x},\mathsf{x},x
\end{align}
\begin{verbatim}
require(stats) # for lowess, rpois, rnorm
require(graphics) # for plot methods
plot(cars)
lines(lowess(cars))
\end{verbatim}
\end{document}
```

编译 LaTeX 文档的命令如下：

```
xelatex --shell-escape -output-driver="xdvipdfmx -z 0" <filename>.tex
```

17.2 R Markdown 基础

```
---
title: "R Markdown 入门"
author: "张三"
documentclass: article
output:
  bookdown::pdf_book:
    extra_dependencies:
      ctex:
        - UTF8
        - heading=true
      bm: null
    toc: yes
    template: null
    base_format: rmarkdown::pdf_document
    latex_engine: xelatex
    number_sections: yes
mathspec: true
colorlinks: yes
classoptions: "b5paper"
---
```

©黄文强

```
# 线性模型 {#sec:lm}

第 \@ref(sec:lm) 节介绍线性模型，线性模型的矩阵表示见公式 \@ref(eq:lm) 。

```{=tex}
\begin{align}
\bm{\mathsf{y}} = \bm{\mathsf{X}}\bm{\beta} + \bm{\epsilon}
(\#eq:lm)
\end{align}
```
```

17.2.1 中英字体

17.2.2 数学公式

17.2.3 代码抄录

17.2.4 插入图表

17.2.5 交叉引用

17.2.6 布局排版

```
---
title: "R Markdown 双栏排版"
subtitle: "副标题"
author: "张三"
date: "`r Sys.Date()`"
mathspec: yes
fontsize: 10pt
graphics: yes
lof: yes
geometry: margin=1.18in
output:
  bookdown::pdf_book:
    number_sections: yes
    toc: yes
    fig_crop: no
    latex_engine: xelatex
    base_format: rmarkdown::pdf_document
```

谢益辉©著

```
    citation_package: natbib
    template: null
    extra_dependencies:
      sourcecodepro:
       - scale=0.85
      ctex:
       - heading=true
       - fontset=fandol
      caption:
       - labelfont=bf
       - singlelinecheck=off
       - textfont=it
       - justification=centering
      Alegreya: null
keywords:
  - 动态文档
  - 双栏排版
subject: "可重复研究与动态文档"
abstract: |
  这里是摘要内容
bibliography:
 - packages.bib
biblio-style: plainnat
natbiboptions: "authoryear,round"
link-citations: true
colorlinks: true
classoption: "UTF8,a4paper,twocolumn"
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

R Markdown 文档混合了代码、图形和文字内容[@rmarkdown]。

代码 {#sec:code}

```{r cars}
```
```

©谢湘宜

```
summary(cars)
```

插图 {#sec:plot}

```{r}
#| fig-iris,
#| fig.cap="鸢尾花数据集",
#| fig.width=5,
#| fig.height=4,
#| fig.showtext=TRUE,
#| out.width="95%",
#| echo=FALSE

library(ggplot2)
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) "
  )
```

参考文献 {#chap:refer}

17.3 Quarto 基础

```
---
title: "Quarto 入门"
author: "张三"
lang: zh
format:
  pdf:
    include-in-header:
      - text: |
          \usepackage[heading=true,UTF8]{ctex}
          \usepackage{amsmath,bm}
```

```
    toc: true
    mathspec: true
    number-sections: true
    colorlinks: true
    documentclass: article
    papersize: b5paper
---

# 线性模型 {#sec-lm}

@sec-lm 介绍线性模型，线性模型的矩阵表示见 @eq-lm 。

$$
\bm{\mathsf{y}} = \bm{\mathsf{X}}\bm{\beta} + \bm{\epsilon}
$$ {#eq-lm}
```

17.3.1 中英字体

17.3.2 数学公式

17.3.3 代码抄录

17.3.4 插入图表

插入图片的语法是 `![]({}`，中括号内是插图标题，小括号内是插图存放路径，大括号内是插图的标识符和属性，比如 `width="65%"` 设置图片的宽度为页面宽度的 65%。

```
![An Elephant](images/peaks.png){#fig-quarto-figure width="65%"}
```

渲染效果如下：

表格默认左对齐，冒号加虚线、虚线加冒号、虚线两侧加冒号分别对应左对齐、右对齐和居中对齐。

```
| Default | Left | Right | Center |
|---------|:-----|------:|:------:|
| 12      | 12   |    12 |   12   |
| 123     | 123  |   123 |  123   |
| 1       | 1    |     1 |   1    |

: 制作表格的管道语法 {#tbl-quarto-table}
```

渲染效果如下：

©黄湘云

图 17.5: Peaks 函数图像

表格 17.4: 制作表格的管道语法

Default	Left	Right	Center
12	12	12	12
123	123	123	123
1	1	1	1

17.3.5 交叉引用

`@tbl-quarto-table` 插入表格 17.4 ，`@fig-quarto-figure` 插入图 17.5 。引用表格的标识符前缀必须是 tbl，引用插图的标识符前缀必须是 fig，后以连字符连接其它内容。对比 LaTeX 文档中的图、表引用 `\ref{fig:figure}` ，Quarto 中的 `@` 符号对应于命令 `\ref` 。值得注意，在 LaTeX 文档中，对标识符的命名没有这般要求，但为了区分引用内容，通常会加上不同的前缀。

17.4 Quarto beamer

Quarto 制作 beamer 幻灯片

```
---
title: "Quarto 幻灯片模版"
author:
  - 张三
  - 李四
institute:
  - XX 大学
  - XX 学院
```

非梁©梅

```yaml
date: today
date-format: long
documentclass: beamer
classoption:
  - 11pt
  - compress
  - xcolor=x11names
  - UTF8
lang: zh
format:
  beamer:
    theme: Singapore
    fonttheme: structurebold
    pdf-engine: lualatex
    include-in-header:
      text: |
        \usecolortheme[named=SpringGreen4]{structure}
        \usepackage[fontset=fandol]{ctex}
    keep-tex: false
    mathspec: true
    toc: true
    navigation: horizontal
    latex-min-runs: 2
    latex-auto-install: false
link-citations: true
---

# In the morning

## Getting up

-   Turn off alarm
-   Get out of bed

## Breakfast

-   Eat eggs
-   Drink coffee

# In the evening
```

©清华大学出版社

```
## Dinner

-   Eat spaghetti
-   Drink wine

## Going to sleep

-   Get in bed
-   Count sheep
```

Pandoc's Markdown 制作 beamer 幻灯片

```
---
title: "Quarto 幻灯片模版"
author:
  - 张三
  - 李四
institute:
  - XX 大学
  - XX 学院
mathspec: true
toc: true
toc-title: "目录"
---

# In the morning

## Getting up

-   Turn off alarm
-   Get out of bed

## Breakfast

-   Eat eggs
-   Drink coffee

# In the evening

## Dinner
```

```
-   Eat spaghetti
-   Drink wine

## Going to sleep

-   Get in bed
-   Count sheep
```

将 Markdown 文档转化为 beamer 幻灯片的命令

```
pandoc --pdf-engine lualatex -t beamer \
  --variable theme="Singapore" --variable fonttheme="structurebold" \
  --variable classoption="xcolor=x11names" \
  --variable header-includes="\usecolortheme[named=SpringGreen4]{structure}" \
  --variable header-includes="\usepackage[fontset=fandol]{ctex}" \
  -f markdown pandoc-beamer.md -o pandoc-beamer.pdf
```

© 黄湘云

第十八章　Office 文档

本章主要介绍办公文档 Word、演示报告 PowerPoint 和电子邮件 Email 三类应用。在 R 语言社区中，Quarto 文档支持输出 Word 和 PowerPoint 格式，**blastula** 包可以将 R Markdown 文档转化为电子邮件内容，从而实现代码化、可重复和批量化，提高工作效率。

18.1　Word 文档

18.1.1　Markdown 制作 Word 文档

本节探索 (R) Markdown + Pandoc 以 Word 格式作为最终交付的可能性。

18.1.2　R Markdown 制作 Word 文档

docxtools、officer 和 officedown 大大扩展了 rmarkdown 包在制作 Word/PPT 方面的功能。

18.1.3　自定义 Word 模版

R Markdown 借助 Pandoc 将 Markdown 转化为 Word 文档，继承自 Pandoc 的扩展性，R Markdown 也支持自定义 Word 模版，那如何自定义呢？首先，我们需要知道 Pandoc 内建的 Word 模版长什么样子，然后我们依样画葫芦，制作适合实际需要的模版。获取 Pandoc 自带的 Word 和 PPT 模版，只需在命令行中执行

```
# DOCX 模版
pandoc -o custom-reference.docx --print-default-data-file reference.docx
# PPTX 模版
pandoc -o custom-reference.pptx --print-default-data-file reference.pptx
```

这里其实是将 Pandoc 自带的 docx 文档 reference.docx 拷贝一份到 custom-reference.docx，而后将 custom-reference.docx 文档自定义一番，但仅限于借助 MS Word 去自定义样式。

- Word 文档的 YAML 元数据定义
- 如何深度自定义文档模版

bookdown 提供的函数 `word_document2()` 相比于 **rmarkdown** 提供的 `word_document()` 支持图表的交叉引用，更多细节详见帮助 `?bookdown::word_document2`。

18.2 PowerPoint 演示

18.3 电子邮件

Rahul Premraj 基于 **rJava** 包开发的 mailR 虽然还未在 CRAN 上正式发布，但是已得到很多人的关注，也被广泛的使用，目前作者已经不维护了，继续使用有一定风险。RStudio 公司 Richard Iannone 新开发的 **blastula** 扔掉了 Java 的重依赖，更加轻量化、现代化，支持发送群组邮件。

18.3.1 curl 包

curl 包提供的函数 `send_mail()` 本质上是在利用 curl 软件发送邮件，举个例子，邮件内容如下：

```
From: "张三" <邮箱地址>
To: "李四" <邮箱地址>
Subject: 测试邮件

你好：

这是一封测试邮件！
```

将邮件内容保存为 mail.txt 文件，然后使用 curl 命令行工具将邮件内容发出去。

```
curl --url 'smtp://公司邮件服务器地址:开放的端口号' \
  --ssl-reqd --mail-from '发件人邮箱地址' \
  --mail-rcpt '收件人邮箱地址' \
  --upload-file data/mail.txt \
  --user '发件人邮箱地址:邮箱登陆密码'
```

> **注释**
>
> Gmail 出于安全性考虑，不支持这种发送邮件的方式，会将邮件内容阻挡，进而接收不到邮件。

18.3.2 blastula 包

下面以 **blastula** 包为例怎么支持 Gmail、Outlook、QQ 等邮件发送，先安装系统软件依赖，CentOS 8 上安装依赖

```
sudo dnf install -y libsecret-devel libsodium-devel
```

©谢湘

然后安装 **keyring** 和 **blastula**

```
install.packages(c("keyring", "blastula"))
```

接着配置邮件帐户，这一步需要邮件账户名和登陆密码，配置一次就够了，不需要每次发送邮件的时候都配置一次

```
library(blastula)
create_smtp_creds_key(
  id = "outlook",
  user = "zhangsan@outlook.com",
  provider = "outlook"
)
```

第二步，准备邮件内容，包括邮件主题、发件人、收件人、抄送人、密送人、邮件主体和附件等。

```
attachment <- "data/mail.txt" # 如果没有附件，引号内留空即可。
# 这个Rmd文件渲染后就是邮件的正文，交互图形和交互表格不适用
body <- "examples/html-document.Rmd"
# 渲染邮件内容，生成预览
email <- render_email(body) |>
  add_attachment(file = attachment)
email
```

最后，发送邮件

```
smtp_send(
  from = c("张三" = "xxx@outlook.com"), # 发件人
  to = c("李四" = "xxx@foxmail.com",
         "王五" = "xxx@gmail.com"), # 收件人
  cc = c("赵六" = "xxx@outlook.com"), # 抄送人
  subject = "这是一封测试邮件",
  email = email,
  credentials = creds_key(id = "outlook")
)
```

密送人实现群发单显，即一封邮件同时发送给多个人，每个收件人只能看到发件人地址而看不到其它收件人地址。

```
email <- compose_email(
  body = md("
Markdown 格式的邮件内容
")
)
```

```
smtp_send(
  from = c("发件人" = "xx@outlook.com"),
  to = c("收件人" = "xx@outlook.com"),
  bcc = c(
    "抄送人" = "xx@outlook.com"
    ),
  subject = "邮件主题",
  email = email,
  credentials = creds_key(id = "outlook")
)
```

© 黄湘云

第四部分

统计分析

© 黄湘云

第十九章　常见的统计检验

本章亮点

1. 比较全面地展示各类统计检验问题的 R 语言实现，其覆盖面之广，远超市面上同类 R 语言书籍。从连续数据到离散数据，从单样本到两样本，再到一般的多样本，触及最前沿的热门话题。
2. 对每类统计检验问题都给出示例及 R 语言实现，涉及近 40 个统计检验方法。在组织结构上，本章按照数据情况对检验方法分类，方便读者根据手头的数据情况，快速从众多的方法中定位最合适的检验方法，符合从数据出发进行分析实战的要求。

> The Earth is Round ($p < 0.05$)
>
> — Jacob Cohen (Cohen 1994)

Jacob Cohen 实际谈的是更加深刻的问题。开篇介绍为什么需要假设检验，做检验和不做检验有什么区别？R. A. Fisher 将抽样分布、参数估计和假设检验列为统计推断的三个中心内容，可见假设检验的重要地位。经过近百余年的发展，假设检验具有丰富的内容，可以从不同的角度对检验方法进行归类。

- 检验方法归类：参数与非参数检验方法。
- 检验计算方式：近似 Approximate、精确 Extract、模拟 Simulation 和重抽样 Bootstrap 等方式。
- 检验对象归类：位置参数（均值）和尺度参数（方差）的检验。
- 检验总体数量归类：单总体、两个总体和多个总体。
- 检验总体分布归类：正态、二项、泊松、多项分布等。
- 检验总体维度归类：分一维、二维和多维的情形。
- 检验样本的数量：小样本 $n < 30$ 和大样本 $n \geq 30$。

χ^2 分布、t 分布和 F 分布作为最基础的三大抽样分布，分别是由 K. Pearson 卡尔 · 皮尔逊、W. S. Gosset 哥塞特、R. A. Fisher 费希尔提出，并以他们的名字命名的。在假设检验中，也有许多检验方法是以提出者的名字命名的。本来名字具有突出效果，由检验方法联系人物名称，可以帮助记忆，但如此之多，以至于很难一一记住。因此，本文也不按检验方法罗列，但是，推荐读者了解这些统计大师的工作和故事，相信会加深对这些检验方法的理解。关于检验方法，如有不明白的地方，可以查看维基百科词条。对每个检验问题，本章给出原假设和备择假设，检验统计量及其服从的分布，R 语言实现（自编

©黄湘云

或调用函数，如果调用函数，说明参数及其含义)，不讲公式推导过程。

在 R 语言中，有大量的函数可以对样本数据做检验，每一个函数对应一个或多个检验问题。为了让读者根据手头数据可以快速地找到最合适的检验方法。单样本检验、两样本检验和多样本检验都只针对连续数据。计数数据检验针对离散数据，不区分总体数量。配对样本检验是两样本检验中的特殊情况，不分连续还是离散，不分两个样本还是多个样本，多个样本就是两两配对检验。前面都是关于某个特征统计量的检验，对分布的检验涉及样本点是否来自正态分布，样本点是否独立和平稳，样本点是否来自某一分布，两个样本是否来自相同分布等。

```
library(nlme)
library(ggplot2)
library(pwr)          # 计算检验的功效和实验样本量
library(dunn.test)    # dunn.test
library(car)        # leveneTest 可替代 bartlett.test
library(survival)
library(coin)       # 补充更多的检验方法
# library(multcomp)  # 多重比较
# library(MKpower)   # power.welch.t.test
# library(rstatix)   # 管道操作整合检验方法
# library(pwrss)     # 常见检验的功效和样本量计算
```

19.1 单样本检验

19.1.1 正态总体均值检验

19.1.1.1 方差已知

$$
\begin{aligned}
\text{I} \quad & H_0: \mu-\mu_0 \le 0 \quad vs. \quad H_1: \mu-\mu_0 > 0 \\
\text{II} \quad & H_0: \mu-\mu_0 \ge 0 \quad vs. \quad H_1: \mu-\mu_0 < 0 \\
\text{III} \quad & H_0: \mu-\mu_0 = 0 \quad vs. \quad H_1: \mu-\mu_0 \ne 0
\end{aligned}
$$

设 $x_1, \cdots, x_n$ 是来自总体 $\mathcal{N}(\mu, \sigma^2)$ 的样本，样本均值和方差分别

$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$ ， $s^2 = \frac{1}{n-1}\sum_{i=1}^n (x_i - \bar{x})^2$

考虑到 $\bar{x} \sim \mathcal{N}(\mu, \sigma^2/n)$ ，则检验统计量服从正态分布

$$u = \frac{\bar{x}-\mu_0}{\sigma/\sqrt{n}}$$

假定 $\mu_0 = 1$ 对于检验问题 I 拒绝域 $\{u \ge u_{1-\alpha}\}$

© 基础统计

图 19.1: 单样本检验

```
set.seed(20232023)
n <- 20
# 样本
x <- rnorm(n, mean = 1.8, sd = 2)
# 检验统计量
u <- (mean(x) - 1) / (2 / sqrt(n))
# 临界值
qnorm(p = 1 - 0.05, mean = 0, sd = 1)
```

```
#> [1] 1.644854
```

```
# P 值
1 - pnorm(q = u)
```

```
#> [1] 0.005082465
```

❗ 重要

随机变量 X 服从标准正态分布，它的概率分布函数如下：

$$P(X \leq u) = \phi(u) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{u} \mathrm{e}^{-t^2/2} \mathrm{d}t$$

统计学基础 ©

若已知概率 $p = 0.95$ ，则对应的下分位点可用函数 `qnorm()` 计算。

```
qnorm(p = 0.95, mean = 0, sd = 1)
#> [1] 1.644854
```

19.1.1.2 方差未知

$$
\begin{aligned}
\text{I} \quad & H_0 : \mu - \mu_0 \leq 0 \quad vs. \quad H_1 : \mu - \mu_0 > 0 \\
\text{II} \quad & H_0 : \mu - \mu_0 \geq 0 \quad vs. \quad H_1 : \mu - \mu_0 < 0 \\
\text{III} \quad & H_0 : \mu - \mu_0 = 0 \quad vs. \quad H_1 : \mu - \mu_0 \neq 0
\end{aligned}
$$

考虑到

$$
\begin{aligned}
& \frac{\bar{x} - \mu}{\sigma / \sqrt{n}} \sim \mathcal{N}(0, 1) \\
& \frac{(n-1)s^2}{\sigma^2} \sim \chi^2(n-1) \\
& \mathsf{E}\{s^2\} = \sigma^2 \quad \mathsf{Var}\{s^2\} = \frac{2\sigma^4}{n-1}
\end{aligned}
$$

根据 t 分布的定义，检验统计量服从 t 分布，即 $t \sim t(n-1)$

$$
t = \frac{\bar{x} - \mu_0}{s / \sqrt{n}}
$$

假定 $\mu_0 = 1$ 对于检验问题 I ，拒绝域 $\{t \geq t_{1-\alpha}(n-1)\}$

```
# 检验统计量
t0 <- (mean(x) - 1) / sqrt(var(x) / n)
# 临界值
qt(p = 1 - 0.05, df = n - 1)
```

```
#> [1] 1.729133
```

```
# P 值
1 - pt(q = t0, df = n - 1)
```

```
#> [1] 0.01569596
```

注释

英国统计学家 William Sealy Gosset (1876-1937) 于 1908 年在杂志《Biometrics》上以笔名 Student 发表论文《The Probable Error of a Mean》(”Student” 1908)，论文中展示了独立同正态分布的样本 $x_1, \ldots, x_n \overset{i.i.d}{\sim} \mathcal{N}(\mu, \sigma^2)$ 的样本方差 s^2 和样本标准差 s 的抽样分布，根据均值和标准差不相关的性质导出 t 分布，宣告 t 分布的诞生，因其在小样本领域的突出贡献，W. S. Gosset 进入世纪

名人录 (Heyde 等 2001)。

表格 19.1: t 分布的分位数表

	0.75	0.8	0.9	0.95	0.975	0.99	0.995	0.999
1	1.0000	1.3764	3.0777	6.3138	12.7062	31.8205	63.6567	318.3088
2	0.8165	1.0607	1.8856	2.9200	4.3027	6.9646	9.9248	22.3271
3	0.7649	0.9785	1.6377	2.3534	3.1824	4.5407	5.8409	10.2145
4	0.7407	0.9410	1.5332	2.1318	2.7764	3.7469	4.6041	7.1732
5	0.7267	0.9195	1.4759	2.0150	2.5706	3.3649	4.0321	5.8934
6	0.7176	0.9057	1.4398	1.9432	2.4469	3.1427	3.7074	5.2076
7	0.7111	0.8960	1.4149	1.8946	2.3646	2.9980	3.4995	4.7853
8	0.7064	0.8889	1.3968	1.8595	2.3060	2.8965	3.3554	4.5008
9	0.7027	0.8834	1.3830	1.8331	2.2622	2.8214	3.2498	4.2968
10	0.6998	0.8791	1.3722	1.8125	2.2281	2.7638	3.1693	4.1437

19.1.2 正态总体方差检验

卡方检验 χ^2 检验统计量服从卡方分布。

$$
\begin{aligned}
\text{I} \quad & H_0 : \sigma^2 - \sigma_0^2 \leq 0 \quad vs. \quad H_1 : \sigma^2 - \sigma_0^2 > 0 \\
\text{II} \quad & H_0 : \sigma^2 - \sigma_0^2 \geq 0 \quad vs. \quad H_1 : \sigma^2 - \sigma_0^2 < 0 \\
\text{III} \quad & H_0 : \sigma^2 - \sigma_0^2 = 0 \quad vs. \quad H_1 : \sigma^2 - \sigma_0^2 \neq 0
\end{aligned}
$$

一般假定均值 μ 是未知的。检验统计量服从卡方分布 $\chi^2(n-1)$

$$\chi^2 = \frac{(n-1)s^2}{\sigma_0^2}$$

设 $\sigma_0^2 = 1.5^2$ ，考虑检验问题 I

```
# 检验统计量
chi <- (n - 1) * var(x) / 1.5^2
# 临界值
qchisq(p = 1 - 0.05, df = n -1)
```

```
#> [1] 30.14353
```

```
# P 值
1 - pchisq(q = chi, df = n -1)
```

```
#> [1] 0.002183653
```

试读

R 软件提供很多统计分布的计算，因此，不再需要查分位数表，现算即可。计算自由度为 n 概率为 p 的 χ^2 分布的分位数 $\chi_p^2(n)$ ，即

$$P(\chi^2(n) \leq \chi_p^2(n)) = p$$

若已知自由度为 1 ，概率为 0.05，则可借助分位数函数 `qchisq()` 计算对应的（下）分位点。

```
qchisq(p = 0.05, df = 1)
```

```
#> [1] 0.00393214
```

同理，也可以获得 χ^2 分布的分位数表格 19.2 ，计算出来的分位数保留 4 位小数。

表格 19.2: χ^2 分布的分位数表

	0.005	0.01	0.025	0.05	0.1	0.9	0.95	0.975	0.99	0.995
1	0.0000	0.0002	0.0010	0.0039	0.0158	2.7055	3.8415	5.0239	6.6349	7.8794
2	0.0100	0.0201	0.0506	0.1026	0.2107	4.6052	5.9915	7.3778	9.2103	10.5966
3	0.0717	0.1148	0.2158	0.3518	0.5844	6.2514	7.8147	9.3484	11.3449	12.8382
4	0.2070	0.2971	0.4844	0.7107	1.0636	7.7794	9.4877	11.1433	13.2767	14.8603
5	0.4117	0.5543	0.8312	1.1455	1.6103	9.2364	11.0705	12.8325	15.0863	16.7496
6	0.6757	0.8721	1.2373	1.6354	2.2041	10.6446	12.5916	14.4494	16.8119	18.5476
7	0.9893	1.2390	1.6899	2.1673	2.8331	12.0170	14.0671	16.0128	18.4753	20.2777
8	1.3444	1.6465	2.1797	2.7326	3.4895	13.3616	15.5073	17.5345	20.0902	21.9550
9	1.7349	2.0879	2.7004	3.3251	4.1682	14.6837	16.9190	19.0228	21.6660	23.5894
10	2.1559	2.5582	3.2470	3.9403	4.8652	15.9872	18.3070	20.4832	23.2093	25.1882

19.1.3 总体未知均值检验

有了均值和方差，为什么还要位置参数和尺度参数？为了更一般地描述问题，扩展范围。特别是在总体分布未知或知之甚少的情况下做检验，不再仅限于均值和方差这样的特征量。

考虑前面正态总体均值检验中的假设 I 的形式，若总体的分布形式未知，则需要 Wilcoxon （威尔科克森）秩和检验 `wilcox.test()` 来做均值的比较。

```
wilcox.test(x = x, mu = 1, alternative = "greater")
```

```
#>
#>  Wilcoxon signed rank exact test
#>
#> data:  x
#> V = 163, p-value = 0.01479
#> alternative hypothesis: true location is greater than 1
```

© 草稿

相比于 t 检验，P 值更小。

19.1.4 总体未知方差检验

19.2 两样本检验

图 19.2: 两样本检验

设 $x_1, \cdots, x_{n_1}$ 是来自总体 $\mathcal{N}(\mu_1, \sigma_1^2)$ 的样本，设 $y_1, \cdots, y_{n_2}$ 是来自总体 $\mathcal{N}(\mu_2, \sigma_2^2)$ 的样本。

19.2.1 正态总体均值检验

两样本均值之差的检验

©黄湘云

图 19.3: 两样本均值之差的检验

常见检验问题

$$
\begin{array}{rlcl}
\text{I} & H_0: \mu_1 - \mu_2 \le 0 & vs. & H_1: \mu_1 - \mu_2 > 0 \\
\text{II} & H_0: \mu_1 - \mu_2 \ge 0 & vs. & H_1: \mu_1 - \mu_2 < 0 \\
\text{III} & H_0: \mu_1 - \mu_2 = 0 & vs. & H_1: \mu_1 - \mu_2 \ne 0
\end{array}
$$

19.2.1.1 方差已知

$$
u = \frac{(\bar{x} - \bar{y}) - (\mu_1 - \mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}
$$

检验统计量服从标准正态分布 $u \sim \mathcal{N}(0,1)$，检验统计量 u 对应的样本值 u_0，检验的拒绝域和 P 值如下

$$
W_1 = \{u \ge u_{1-\alpha}\}, \quad p_1 = 1 - \Phi(u_0)
$$

```
n_1 <- 100
n_2 <- 80
mu_1 <- 10
sigma_1 <- 2.5
mu_2 <- 6
sigma_2 <- 4.5
```

小数据 © 拓展

```
set.seed(20232023)
x1 <- rnorm(n_1, mean = mu_1, sd = sigma_1)
y1 <- rnorm(n_2, mean = mu_2, sd = sigma_2)
u0 <- (mean(x1) - mean(y1)) / sqrt(sigma_1^2 / n_1 + sigma_2^2 / n_2)
u0
```

```
#> [1] 6.779039
```

对检验问题 I，给定显著性水平 $\alpha = 0.05$，得出拒绝域 $\{u \geq 1.645\}$，计算样本观察值得到的检验统计量的值 $u_0 = 6.779$，而该值落在拒绝域，所以拒绝原假设，即拒绝 $\mu_1 - \mu_2 \leq 0$，则接受 $\mu_1 - \mu_2 > 0$。

```
# 计算拒绝域
qnorm(1 - 0.05)
```

```
#> [1] 1.644854
```

```
# 计算 P 值
1 - pnorm(u0)
```

```
#> [1] 6.048939e-12
```

19.2.1.2 方差未知但相等

设 $x_1, \cdots, x_{n_1}$ 是来自总体 $\mathcal{N}(\mu_1, \sigma^2)$ 的样本，设 $y_1, \cdots, y_{n_2}$ 是来自总体 $\mathcal{N}(\mu_2, \sigma^2)$ 的样本。

t 检验，检验统计量服从自由度为 $n_1 + n_2 - 2$ 的 t 分布

$$t = \frac{(\bar{x} - \bar{y}) - (\mu_1 - \mu_2)}{s_0 \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

其中，

$$\bar{x} = \sum_{i=1}^{n_1} x_i \quad \bar{y} = \sum_{i=1}^{n_2} y_i$$
$$s_0^2 = \frac{1}{n_1 + n_2 - 2} \left(\sum_{i=1}^{n_1} (x_i - \bar{x})^2 + \sum_{i=1}^{n_2} (y_i - \bar{y})^2 \right)$$

```
s_w <- sqrt(1 / (n_1 + n_2 - 2) * ((n_1 - 1) * var(x1) + (n_2 - 1) * var(y1)))
t0 <- (mean(x1) - mean(y1)) / (s_w * sqrt(1 / n_1 + 1 / n_2))
t0
```

```
#> [1] 8.155781
```

样本观察值 $t_0 = 8.155 > t_{0.95}(n_1 + n_2 - 2) = 1.653$ 落在拒绝域内，对于检验问题 I 我们要拒绝原假设

```r
# 临界值: 0.95 分位点对应的分位数
qt(1 - 0.05, df = n_1 + n_2 - 2)

#> [1] 1.653459

# p 值
1 - pt(t0, df = n_1 + n_2 - 2, lower.tail = TRUE)

#> [1] 3.019807e-14
```

利用 R 内置的 `t.test()` 函数计算

```r
t.test(x = x1, y = y1, alternative = "greater", var.equal = TRUE)

#>
#>  Two Sample t-test
#>
#> data:  x1 and y1
#> t = 8.1558, df = 178, p-value = 3.016e-14
#> alternative hypothesis: true difference in means is greater than 0
#> 95 percent confidence interval:
#>  3.036384      Inf
#> sample estimates:
#> mean of x mean of y
#> 10.338905  6.530406
```

检验统计量的值及对应的 P 值都是一样的。睡眠数据 sleep 记录了两种药物对病人睡眠时间的影响，此数据集由 “Student”（哥塞特的笔名）收集。

```r
# 方差未知但相等
t.test(extra ~ group, data = sleep, var.equal = TRUE)

#>
#>  Two Sample t-test
#>
#> data:  extra by group
#> t = -1.8608, df = 18, p-value = 0.07919
#> alternative hypothesis: true difference in means between group 1 and group 2 is not equal to 0
#> 95 percent confidence interval:
#>  -3.363874  0.203874
#> sample estimates:
#> mean in group 1 mean in group 2
#>            0.75            2.33
```

19.2.1.3 方差未知且不等

两个样本的样本量不是很大，总体方差也未知，两样本均值之差的显著性检验，即著名的 Behrens-Fisher 问题，Welch 在 1938 年提出近似服从自由度为 l 的 t 分布。

两样本的样本量很大，尽管总体方差未知，两样本均值之差的显著性检验，极限分布是正态分布，可以用 Z 检验。在样本量很大的情况下，Welch t 检验也可以用。

设 $x_1, \cdots, x_{n_1}$ 是来自总体 $\mathcal{N}(\mu_1, \sigma_1^2)$ 的 IID 样本，设 $y_1, \cdots, y_{n_2}$ 是来自总体 $\mathcal{N}(\mu_2, \sigma_2^2)$ 的 IID 样本。

Welch（韦尔奇）t 检验

$$T = \frac{(\bar{x} - \bar{y}) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_x^2}{n_1} + \frac{s_y^2}{n_2}}}$$

其中，s_x^2 表示样本 x 的方差 $s_x^2 = \frac{1}{n_1 - 1} \sum_{i=1}^{n_1} (x_i - \bar{x})^2$ ，s_y^2 表示样本 y 的方差 $s_y^2 = \frac{1}{n_2 - 1} \sum_{i=1}^{n_2} (y_i - \bar{y})^2$ 。检验统计量 T 服从自由度为 l 的 t 分布。

$$l = \frac{s_0^4}{\frac{s_x^4}{n_1^2(n_1 - 1)} + \frac{s_y^4}{n_2^2(n_2 - 1)}}$$

其中，$s_0^2 = s_x^2/n_1 + s_y^2/n_2$，$l$ 通常不是整数，实际使用时，l 可取最近的整数。

```
s0 <- var(x1) / n_1 + var(y1) / n_2
l <- s0^2 / (var(x1)^2 / (n_1^2 * (n_1 - 1)) + var(y1)^2 / (n_2^2 * (n_2 - 1)))
l
```

```
#> [1] 126.7708
```

所以，l 可取 127。检验统计量的值如下

```
t0 <- (mean(x1) - mean(y1)) / sqrt(s0)
t0
```

```
#> [1] 7.77002
```

```
# 临界值：0.95 分位点对应的分位数
qt(1 - 0.05, df = 127)
```

```
#> [1] 1.65694
```

```
# p 值
1 - pt(t0, df = 126.7708, lower.tail = TRUE)
```

```
#> [1] 1.162404e-12
```

```
# 就近取整
1 - pt(t0, df = 127, lower.tail = TRUE)
```

```
#> [1] 1.153078e-12
```

与函数 t.test() 比较，值得注意，t 分布的自由度可以为非整数。

```
t.test(x = x1, y = y1, alternative = "greater", var.equal = FALSE)
```

```
#>
#>  Welch Two Sample t-test
#>
#> data:  x1 and y1
#> t = 7.77, df = 126.77, p-value = 1.162e-12
#> alternative hypothesis: true difference in means is greater than 0
#> 95 percent confidence interval:
#>  2.996334      Inf
#> sample estimates:
#> mean of x mean of y
#> 10.338905  6.530406
```

举例：sleep 数据集

图 19.4: 学生睡眠数据的分布

```
# 方差未知且不等
t.test(extra ~ group, data = sleep, var.equal = FALSE)
```

```
#>
```

©谢湘 统计课

```
#>  Welch Two Sample t-test
#>
#> data:  extra by group
#> t = -1.8608, df = 17.776, p-value = 0.07939
#> alternative hypothesis: true difference in means between group 1 and group 2 is not equal to 0
#> 95 percent confidence interval:
#>  -3.3654832  0.2054832
#> sample estimates:
#> mean in group 1 mean in group 2
#>            0.75            2.33
```

> **注释**
>
> Egon Pearson 接过他父亲 Karl Pearson 的职位，担任伦敦大学学院的高尔顿统计教授。许宝𝔽（Pao-Lu Hsu）在 Jerzy Neyman 和 Egon Pearson 主编的杂志《Statistical Research Memoirs》发表第一篇关于 Behrens-Fisher 问题的论文 (HSU 1938)，1998 年关于 Behrens-Fisher 问题的综述 (S.-H. Kim 和 Cohen 1998)。陈家鼎和郑忠国一起整理了许宝𝔽的生平事迹和学术成就，见《许宝𝔽先生的生平和学术成就》。钟开莱（Kai-Lai Chung）将许宝𝔽的论文集整理出版 (HSU 1983)。

t 检验的影响是如此巨大，以至于广泛存在于具有统计功能的软件中，比如办公软件里的 t 检验。以 MacOS 上的 Numbers 表格软件为例，如图 19.5 所示，首先打开 Numbers 软件，新建工作表，输入两组数值，然后点击空白处，再从顶部导航栏找到「插入」菜单，「公式」选项，点击扩展选项「新建公式」，在弹出的会话条里输入 TTEST，依次选择第一组，第二组值，检验类型和样本类型，最后点击确认，即可得到两样本 t 检验的 P 值结果。

图 19.5: 办公软件 Numbers 的两样本 t 检验

微软 Excel 办公软件也提供 t 检验计算器，和 MacOS 系统上的 Numbers 办公软件类似，它提供 `T.TEST` 函数，计算结果也一样，此处从略。R 软件自带 `t.test()` 函数，也是用于做 t 检验，如下：

```
t.test(x = c(3, 4, 5, 8, 9, 1, 2, 4, 5), y = c(6, 19, 3, 2, 14, 4, 5, 17, 1))
```

```
#>
#>  Welch Two Sample t-test
#>
#> data:  c(3, 4, 5, 8, 9, 1, 2, 4, 5) and c(6, 19, 3, 2, 14, 4, 5, 17, 1)
#> t = -1.3622, df = 10.255, p-value = 0.2023
#> alternative hypothesis: true difference in means is not equal to 0
#> 95 percent confidence interval:
#>  -8.767183  2.100516
#> sample estimates:
#> mean of x mean of y
#>  4.555556  7.888889
```

19.2.2 正态总体方差检验

比较两个正态总体的方差是否相等，F 检验。

```r
# 两样本
var.test(extra ~ group, data = sleep)
```

```
#>
#>  F test to compare two variances
#>
#> data:  extra by group
#> F = 0.79834, num df = 9, denom df = 9, p-value = 0.7427
#> alternative hypothesis: true ratio of variances is not equal to 1
#> 95 percent confidence interval:
#>  0.198297 3.214123
#> sample estimates:
#> ratio of variances
#>          0.7983426
```

```r
# 或者
bartlett.test(extra ~ group, data = sleep)
```

```
#>
#>  Bartlett test of homogeneity of variances
#>
#> data:  extra by group
#> Bartlett's K-squared = 0.10789, df = 1, p-value = 0.7426
```

注意：函数 `bartlett.test()` 支持多样本情况。

©李东风

19.2.3 总体未知均值检验

在总体分布未知的情况下，比较均值是否相等的检验。

- `wilcox.test()` 适用于单样本和两样本的均值检验，单样本 Wilcoxon 秩和检验，两样本 Wilcoxon 符号秩和检验，后者也叫 Mann-Whitney 检验。
- `kruskal.test()` 适用于两样本和多样本，比较多个均值是否相等的检验，Kruskal-Wallis 秩和检验。

单样本和两样本 `wilcox.test()`。

```
wilcox.test(extra ~ group, data = sleep)
```

```
#> Warning in wilcox.test.default(x = DATA[[1L]], y = DATA[[2L]], ...): cannot
#> compute exact p-value with ties
```

```
#>
#>  Wilcoxon rank sum test with continuity correction
#>
#> data:  extra by group
#> W = 25.5, p-value = 0.06933
#> alternative hypothesis: true location shift is not equal to 0
```

coin 包提供渐进 Wilcoxon-Mann-Whitney 检验

```
# Asymptotic Wilcoxon-Mann-Whitney Test
wilcox_test(extra ~ group, data = sleep, conf.int = TRUE)
```

```
#>
#>  Asymptotic Wilcoxon-Mann-Whitney Test
#>
#> data:  extra by group (1, 2)
#> Z = -1.8541, p-value = 0.06372
#> alternative hypothesis: true mu is not equal to 0
#> 95 percent confidence interval:
#>  -3.500000e+00  1.270214e-10
#> sample estimates:
#> difference in location
#>              -1.347344
```

```
# Exact Wilcoxon-Mann-Whitney Test
wilcox_test(
  extra ~ group, data = sleep,
  distribution = "exact", conf.int = TRUE
)
```

©黄湘云


```
#>
#>  Exact Wilcoxon-Mann-Whitney Test
#>
#> data:  extra by group (1, 2)
#> Z = -1.8541, p-value = 0.06582
#> alternative hypothesis: true mu is not equal to 0
#> 95 percent confidence interval:
#>  -3.5  0.0
#> sample estimates:
#> difference in location
#>                  -1.35
```

两样本和多样本 `kruskal.test()` 。

```
kruskal.test(extra ~ group, data = sleep)
```

```
#>
#>  Kruskal-Wallis rank sum test
#>
#> data:  extra by group
#> Kruskal-Wallis chi-squared = 3.4378, df = 1, p-value = 0.06372
```

能用参数检验的一定也可以用非参数检验，一般来说，非参数检验的功效不小于参数检验，非参数检验不要求分布是正态，比如此时 P 值从 0.07939 降至 0.06372。

19.2.4 总体未知方差检验

对总体没有分布要求的方差齐性检验方法有三个，按适用范围分类，见下表格 19.3 。

表格 19.3: 检验方法分类

两个样本	多个样本
• Ansari-Bradley 检验 `ansari.test()` • Mood 检验 `mood.test()`	• Fligner-Killeen 检验 `fligner.test()`

以 A. R. Ansari 和 R. A. Bradley 命名的 Ansari-Bradley 检验 (Ansari 和 Bradley 1960)，对应的 R 函数是 `ansari.test()` ，以 A. M. Mood 命名的 Mood 检验 (Mood 1954)，对应的 R 函数是 `mood.test()` ，这两者都属于两样本的非参数检验，检验尺度参数是否相同（齐性）。以 M. A. Fligner 和 T. J. Killeen 命名的 Fligner-Killeen 检验 (Fligner 和 Killeen 1976)，对应的 R 函数是 `fligner.test()` ，也属于非参数检验，适用于两样本和多样本的情况。非参数检验常涉及位置参数和尺度参数这一对概念，就正态分布而言，位置参数可以理解为均值 μ ，尺度参数可以理解为方差 σ^2 。

```
ansari.test(extra ~ group, data = sleep)
```

©黄湘云

```
#> Warning in ansari.test.default(x = DATA[[1L]], y = DATA[[2L]], ...): cannot
#> compute exact p-value with ties
#>
#>  Ansari-Bradley test
#>
#> data:  extra by group
#> AB = 50.5, p-value = 0.4927
#> alternative hypothesis: true ratio of scales is not equal to 1
```

```
mood.test(extra ~ group, data = sleep)
```

```
#>
#>  Mood two-sample test of scale
#>
#> data:  extra by group
#> Z = 0.44761, p-value = 0.6544
#> alternative hypothesis: two.sided
```

```
fligner.test(extra ~ group, data = sleep)
```

```
#>
#>  Fligner-Killeen test of homogeneity of variances
#>
#> data:  extra by group
#> Fligner-Killeen:med chi-squared = 0.21252, df = 1, p-value = 0.6448
```

19.3 多样本检验

本节考虑 Base R 内置的 PlantGrowth 数据集，它收集自 Annette J. Dobson 所著书籍《An Introduction to Statistical Modelling》(Dobson 1983) 第 2 章第 2 节的案例 — 研究植物在两种不同试验条件下的生长情况，植物通过光合作用吸收土壤的养分和空气中的二氧化碳，完成积累，故以植物的干重来刻画植物的生长情况，首先将几乎相同的种子随机地分配到实验组和对照组，基于完全随机实验设计（completely randomized experimental design），经过预定的时间后，将植物收割，干燥并称重。

```
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 ...
```

设立对照组（控制组）ctrl 和实验组 trt1 和 trt2，比较不同的处理方式对植物干重的影响

```
summary(PlantGrowth)
```

©黄湘云

图 19.6: 多样本检验

```
#>      weight       group
#>  Min.   :3.590   ctrl:10
#>  1st Qu.:4.550   trt1:10
#>  Median :5.155   trt2:10
#>  Mean   :5.073
#>  3rd Qu.:5.530
#>  Max.   :6.310
```

每个组都有 10 颗植物，生长情况如图 19.7 所示

```
## Annette J. Dobson 扩展的 Plant Weight Data 数据，见 59 页
library(ggplot2)
ggplot(data = PlantGrowth, aes(x = group, y = weight, color = group)) +
  geom_boxplot() +
  geom_jitter() +
  theme_minimal()
```

图 19.7: 植物干重

©黄湘云

19.3.1 正态总体均值检验

19.3.1.1 假定同方差

讲清楚原假设和备择假设。讲清楚假设检验、方差分析、一般线性模型（包含广义线性模型和线性混合效应模型）的关系。

$\sigma_i^2 = \mathsf{Var}\{\epsilon_{ij}\}, i = 1, 2, 3$ 表示第 i 组的方差，

$$y_{ij} = \mu + \epsilon_{ij}, i = 1, 2, 3$$

其中 μ 是固定的未知参数。单因素方差分析 `oneway.test()`

```
# 假设各组方差相同
oneway.test(weight ~ group, data = PlantGrowth, var.equal = TRUE)
```

```
#>
#>  One-way analysis of means
#>
#> data:  weight and group
#> F = 4.8461, num df = 2, denom df = 27, p-value = 0.01591
```

线性模型也假定各个组的方差是相同的，模型显著性检验的结果和上面是一致的。

```
fit_lm <- lm(weight ~ group, data = PlantGrowth)
anova(fit_lm) # 或者 summary(fit)
```

```
#> Analysis of Variance Table
#>
#> Response: weight
#>           Df  Sum Sq Mean Sq F value  Pr(>F)
#> group      2  3.7663  1.8832  4.8461 0.01591 *
#> Residuals 27 10.4921  0.3886
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

模型输出整理成表格 19.4

表格 19.4: 线性回归的输出

	估计值	标准差	t 统计量	P 值
α	5.032	0.1971	25.5265	0.0000
β_1	-0.371	0.2788	-1.3308	0.1944
β_2	0.494	0.2788	1.7720	0.0877

©梁文佳

19.3.1.2 假定异方差

```
# 计算各个组的方差
aggregate(data = PlantGrowth, weight ~ group, FUN = var)
```

```
#>   group    weight
#> 1  ctrl 0.3399956
#> 2  trt1 0.6299211
#> 3  trt2 0.1958711
```

```
# 或者
with(PlantGrowth, tapply(weight, group, var))
```

```
#>      ctrl      trt1      trt2
#> 0.3399956 0.6299211 0.1958711
```

各个组的方差确实不太相同。

```
# 假设各组方差不同
oneway.test(weight ~ group, data = PlantGrowth, var.equal = FALSE)
```

```
#>
#>  One-way analysis of means (not assuming equal variances)
#>
#> data:  weight and group
#> F = 5.181, num df = 2.000, denom df = 17.128, p-value = 0.01739
```

线性混合效应模型，假定每一组（层）有不同的方差。

```
fit_gls <- nlme::gls(weight ~ 1,
  data = PlantGrowth, method = "ML",
  weights = nlme::varIdent(form = ~ 1 | group)
)
summary(fit_gls)
```

```
#> Generalized least squares fit by maximum likelihood
#>   Model: weight ~ 1
#>   Data: PlantGrowth
#>        AIC      BIC    logLik
#>   67.99884 73.60363 -29.99942
#>
#> Variance function:
#>  Structure: Different standard deviations per stratum
#>  Formula: ~1 | group
#>  Parameter estimates:
#>      ctrl      trt1      trt2
```

©黄湘云


```
#> 1.0000000 1.6028758 0.9103568
#>
#> Coefficients:
#>                Value Std.Error  t-value p-value
#> (Intercept) 5.205999  0.115762 44.97158       0
#>
#> Standardized residuals:
#>         Min          Q1         Med          Q3         Max
#> -1.78654574 -0.92900218 -0.08794552  0.61374803  2.09128348
#>
#> Residual standard error: 0.5798892
#> Degrees of freedom: 30 total; 29 residual
```

考虑每个组有不同的方差，放开同方差的假设，发现，从对数似然的角度来看，有一定提升。

```
logLik(fit_lm)
```

```
#> 'log Lik.' -26.80952 (df=4)
```

```
logLik(fit_gls)
```

```
#> 'log Lik.' -29.99942 (df=4)
```

19.3.2 正态总体方差检验

总体服从正态分布，有四种常见的参数检验方法：

1. Hartley 检验：各组样本量必须相等。
2. Bartlett 检验：各组样本量可以相等或不等，但每个组的样本量必须不低于 5。
3. 修正的 Bartlett 检验：在样本量较大或较小、相等或不等场合都可使用。
4. Levene 检验：相当于单因素组间方差分析，相比于 Bartlett 检验，Levene 检验更加稳健。

> **提示**
>
> 在总体分布未知的情况下，检验方差齐性的非参数方法也都可以用在这里。

设 $x_1, \cdots, x_{n_1}$ 是来自总体 $\mathcal{N}(\mu_1, \sigma_1^2)$ 的样本，设 $y_1, \cdots, y_{n_2}$ 是来自总体 $\mathcal{N}(\mu_2, \sigma_2^2)$ 的样本，设 $z_1, \cdots, z_{n_3}$ 是来自总体 $\mathcal{N}(\mu_3, \sigma_3^2)$ 的样本。

$$\sigma_1^2 = \sigma_2^2 = \sigma_3^2 \quad vs. \quad \sigma_1^2, \sigma_2^2, \sigma_3^2 \quad \text{不全相等}$$

Bartlett （巴特利特）检验 `bartlett.test()` 要求总体的分布为正态分布，检验各个组的方差是否有显著性差异，即方差齐性检验，属于参数检验，适用于多个样本的情况。

```r
# 三样本
bartlett.test(weight ~ group, data = PlantGrowth)
```

```
#>
#>  Bartlett test of homogeneity of variances
#>
#> data:  weight by group
#> Bartlett's K-squared = 2.8786, df = 2, p-value = 0.2371
```

```r
# 或者
car::leveneTest(weight ~ group, data = PlantGrowth)
```

```
#> Levene's Test for Homogeneity of Variance (center = median)
#>       Df F value Pr(>F)
#> group  2  1.1192 0.3412
#>       27
```

19.3.3 总体未知均值检验

Kruskal-Wallis 秩和检验 `kruskal.test()` 检验均值是否齐性。

```r
kruskal.test(weight ~ group, data = PlantGrowth)
```

```
#>
#>  Kruskal-Wallis rank sum test
#>
#> data:  weight by group
#> Kruskal-Wallis chi-squared = 7.9882, df = 2, p-value = 0.01842
```

等价的线性模型表示

```r
fit_lm <- lm(rank(weight) ~ group, data = PlantGrowth)
anova(fit_lm) # summary(fit_lm)
```

```
#> Analysis of Variance Table
#>
#> Response: rank(weight)
#>           Df  Sum Sq Mean Sq F value  Pr(>F)
#> group      2  618.95 309.475  5.1324 0.01291 *
#> Residuals 27 1628.05  60.298
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Friedman 秩和检验是非参数检验。适用于单因素重复测量数据的方差分析，检验是否存在一组值显著高于或低于其他组。针对 unreplicated blocked data

典型场景：n 个品酒师对 k 瓶葡萄酒打分，是否存在一组打分显著高于其他组。检验睡眠质量一组人显著好于另一组人。

```
friedman.test(extra ~ group | ID, data = sleep)
```

```
#>
#>  Friedman rank sum test
#>
#> data:  extra and group and ID
#> Friedman chi-squared = 9, df = 1, p-value = 0.0027
```

`formula` 参数取值为 `a ~ b | c` ，`a` 表示数据值，`b` 分组变量 groups，`c` 表示 blocks。

Quade 检验 `quade.test()` 与 Friedman 检验类似，Quade 检验应用于 unreplicated complete block designs。

```
# 睡眠实验
quade.test(extra ~ group | ID, data = sleep)
```

```
#>
#>  Quade test
#>
#> data:  extra and group and ID
#> Quade F = 28.557, num df = 1, denom df = 9, p-value = 0.0004661
```

术语涉及实验设计，比如完全区组设计 complete block designs 。1879 年迈克尔逊光速测量数据记录了五次实验，每次试验测量 20 次光速。数据集 morley 中光速 Speed 已经编码过了，为了展示方便，原始观测速度减去了 299000 (km/sec)。

```
# 光速实验
quade.test(Speed ~ Expt | Run, data = morley)
```

```
#>
#>  Quade test
#>
#> data:  Speed and Expt and Run
#> Quade F = 3.6494, num df = 4, denom df = 76, p-value = 0.008976
```

```
ggplot(data = morley, aes(x = Expt, y = Speed, group = Expt)) +
  geom_boxplot() +
  geom_jitter() +
  theme_minimal() +
  labs(x = "Expt", y = "Speed (km/sec)")
```

© 黄湘云

图 19.8: 1879 年迈克尔逊光速实验数据

19.3.4 总体未知方差检验

三个及以上样本的方差齐性检验。进一步地，我们在线性模型的基础上考虑每个实验组有不同的方差，先做方差齐性检验。

```
# 非参数检验
fligner.test(weight ~ group, data = PlantGrowth)
```

```
#>
#>  Fligner-Killeen test of homogeneity of variances
#>
#> data:  weight by group
#> Fligner-Killeen:med chi-squared = 2.3499, df = 2, p-value = 0.3088
```

检验的结果显示，可以认为三个组的方差没有显著差异。

19.4 配对样本检验

配对样本检验算是两样本检验的一种特殊情况。若待检验的样本不止两个，则两两配对检验。

© 王建平

表格 19.5: 配对样本检验

样本	R 函数
两样本	• t.test(paired = TRUE) 正态总体均值检验 • wilcox.test(paired = TRUE) 总体未知均值检验

19.4.1 配对 t 检验

做两个组的配对 t 检验，函数 t.test() 的参数 paired 设置为 TRUE ，两个组的样本当作配对样本处理。

```
t.test(extra ~ group, data = sleep, paired = TRUE)
```

```
#>
#>  Paired t-test
#>
#> data:  extra by group
#> t = -4.0621, df = 9, p-value = 0.002833
#> alternative hypothesis: true mean difference is not equal to 0
#> 95 percent confidence interval:
#>  -2.4598858 -0.7001142
#> sample estimates:
#> mean difference
#>           -1.58
```

做多个组的两两配对 t 检验，函数 pairwise.t.test() 的参数 paired 设置为 TRUE ，当仅做两个组的配对 t 检验时，检验结果与前面的等价。

```
with(sleep, pairwise.t.test(x = extra, g = group, paired = TRUE))
```

```
#>
#>  Pairwise comparisons using paired t tests
#>
#> data:  extra and group
#>
#>   1
#> 2 0.0028
#>
#> P value adjustment method: holm
```

输出结果中，组 1 和组 2 配对 t 检验的 P 值为 0.0028。

提示

两个组的配对 t 检验还与变截距的线性混合效应模型等价。

```
library(nlme)
m <- lme(fixed = extra ~ group, random = ~ 1 | ID, data = sleep)
summary(m)
#> Linear mixed-effects model fit by REML
#>   Data: sleep
#>        AIC      BIC    logLik
#>   77.95588 81.51737 -34.97794
#>
#> Random effects:
#>  Formula: ~1 | ID
#>         (Intercept)  Residual
#> StdDev:      1.6877 0.8697384
#>
#> Fixed effects:  extra ~ group
#>             Value Std.Error DF  t-value p-value
#> (Intercept)  0.75 0.6003979  9 1.249172  0.2431
#> group2       1.58 0.3889588  9 4.062127  0.0028
#>  Correlation:
#>        (Intr)
#> group2 -0.324
#>
#> Standardized Within-Group Residuals:
#>         Min          Q1         Med          Q3         Max
#> -1.63372282 -0.34157076  0.03346151  0.31510644  1.83858572
#>
#> Number of Observations: 20
#> Number of Groups: 10
```

输出结果中，固定效应部分 `group2` 意味着相对于第 1 组，第 2 组的增加值，其为 1.58，对应的 t 统计量的值为 4.062127，P 值为 0.0028。调用 **nlme** 包的函数 `intervals()` 计算固定效应部分 95% 的置信区间。

```
intervals(m, which = "fixed")
#> Approximate 95% confidence intervals
#>
#>  Fixed effects:
#>                  lower est.    upper
#> (Intercept) -0.6081944 0.75 2.108194
```

©黄湘云


```
#> group2       0.7001140 1.58 2.459886
```

group2 对应的 95% 的置信区间是 $(0.7001140, 2.459886)$ 。

19.4.2 配对 Wilcoxon 检验

Wilcoxon 检验函数 `wilcox.test()` 设置 `paired = TRUE` 可以做配对检验，但是仅限于两个组。

```
# 不支持
# wilcox.test(weight ~ group, data = PlantGrowth, paired = TRUE)
# 支持
wilcox.test(extra ~ group, data = sleep, paired = TRUE)
```

```
#> Warning in wilcox.test.default(x = DATA[[1L]], y = DATA[[2L]], ...): cannot
#> compute exact p-value with ties
```

```
#> Warning in wilcox.test.default(x = DATA[[1L]], y = DATA[[2L]], ...): cannot
#> compute exact p-value with zeroes
```

```
#>
#>  Wilcoxon signed rank test with continuity correction
#>
#> data:  extra by group
#> V = 0, p-value = 0.009091
#> alternative hypothesis: true location shift is not equal to 0
```

19.5 多重假设检验

同时检验多个统计假设。

表格 19.6: 多重假设检验

样本	R 函数
多样本	• `pairwise.t.test()` 正态总体均值检验 • `pairwise.prop.test()` 二项总体比例检验 • `pairwise.wilcox.test()` 总体未知均值检验

19.5.1 多重 t 检验

数据集 sleep 仅有两个组，数据集 PlantGrowth 包含三个组，下面以数据集 PlantGrowth 为例，介绍做多个组同时进行两两比较的 t 检验。

©黄湘云

```r
# 样本成对的情况
with(PlantGrowth, pairwise.t.test(x = weight, g = group, paired = TRUE))
```

```
#>
#>  Pairwise comparisons using paired t tests
#>
#> data:  weight and group
#>
#>      ctrl  trt1
#> trt1 0.346 -
#> trt2 0.220 0.058
#>
#> P value adjustment method: holm
```

函数 pairwise.t.test() 以 P 值给出两两配对比较的结果，trt1 和 ctrl 配对比较，P 值为 0.346，trt2 和 ctrl 配对比较，P 值为 0.220，以此类推。

```r
# 样本非成对的情况
with(PlantGrowth, pairwise.t.test(x = weight, g = group))
```

```
#>
#>  Pairwise comparisons using t tests with pooled SD
#>
#> data:  weight and group
#>
#>      ctrl  trt1
#> trt1 0.194 -
#> trt2 0.175 0.013
#>
#> P value adjustment method: holm
```

19.5.2 多重比例检验

对于离散数据，做两两比例检验，采用函数 pairwise.prop.test() ，如下示例含有 4 个组。

```r
smokers <- c(83, 90, 129, 70)
patients <- c(86, 93, 136, 82)
pairwise.prop.test(smokers, patients)
```

```
#> Warning in prop.test(x[c(i, j)], n[c(i, j)], ...): Chi-squared approximation
#> may be incorrect

#> Warning in prop.test(x[c(i, j)], n[c(i, j)], ...): Chi-squared approximation
```

©清华大学出版社


```
#> may be incorrect

#> Warning in prop.test(x[c(i, j)], n[c(i, j)], ...): Chi-squared approximation
#> may be incorrect

#>
#>  Pairwise comparisons using Pairwise comparison of proportions
#>
#> data:  smokers out of patients
#>
#>   1     2     3
#> 2 1.000 -     -
#> 3 1.000 1.000 -
#> 4 0.119 0.093 0.124
#>
#> P value adjustment method: holm
```

19.5.3 Wilcoxon 检验

Wilcoxon 检验的是两个总体的均值是否相等。

函数 `pairwise.wilcox.test()` 做两个及以上组的两两比较检验。

```
with(PlantGrowth, pairwise.wilcox.test(x = weight, g = group))
```

```
#> Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot compute
#> exact p-value with ties

#>
#>  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
#>
#> data:  weight and group
#>
#>      ctrl  trt1
#> trt1 0.199 -
#> trt2 0.126 0.027
#>
#> P value adjustment method: holm
```

19.5.4 Dunn 检验

dunn.test 包提供函数 `dunn.test()` 实现 Dunn 检验，将 Kruskal-Wallis 秩和检验用于两两比较。

```r
library(dunn.test)
with(PlantGrowth, dunn.test(x = weight, g = group, method = "holm", altp = TRUE))
#>   Kruskal-Wallis rank sum test
#>
#> data: weight and group
#> Kruskal-Wallis chi-squared = 7.9882, df = 2, p-value = 0.02
#>
#>
#>                          Comparison of weight by group
#>                                      (Holm)
#> Col Mean-|
#> Row Mean |       ctrl       trt1
#> ---------+----------------------
#>     trt1 |   1.117725
#>          |     0.2637
#>          |
#>     trt2 |  -1.689289  -2.807015
#>          |     0.1823    0.0150*
#>
#> alpha = 0.05
#> Reject Ho if p <= alpha
```

19.6 总体分布的检验

前面介绍的检验方法都是对总体的某个特征数（均值、方差）进行检验，下面介绍的检验方法是针对分布的性质。比如样本是否来自正态分布，两个样本是否来自同一分布，样本点之间是否相互独立，样本点列是否平稳等。通过检验方法探索样本的分布性质。

19.6.1 正态性检验

什么样的数据是正态的，理论上是清楚的，对统计建模来说，更实际的问题是什么样的数据是够正态的！探索性数据分析是不断提出假设和验证假设的过程。

> Usually (but not always) doing tests of normality reflect a lack of understanding of the power of rank tests, and an assumption of high power for the tests (qq plots don't always help with that because of their subjectivity). When possible it's good to choose a robust method. Also, doing pre-testing for normality can affect the type I error of the overall analysis.
>
> — Frank Harrell [1]

[1] https://stat.ethz.ch/pipermail/r-help/2005-April/070508.html

内部使用 ©

检验：拒绝原假设和接受原假设的风险，数据本身和理论的正态分布的距离，抛开 P 值

Shapiro 和 Wilk 提出的 W 检验 (Shapiro 和 Wilk 1965)，对应的 R 函数为 `shapiro.test()`

```
set.seed(20232023)
x <- rnorm(100, mean = 5, sd = 3)
shapiro.test(x)
```

```
#>
#>  Shapiro-Wilk normality test
#>
#> data:  x
#> W = 0.98635, p-value = 0.3954
```

> The issue really comes down to the fact that the questions: "exactly normal?", and "normal enough?" are 2 very different questions (with the difference becoming greater with increased sample size) and while the first is the easier to answer, the second is generally the more useful one.
>
> — Greg Snow [2]

EP 检验对多种备择假设有较高的效率，利用样本的特征函数和正态分布的特征函数的差的模的平方产生的一个加权积分得到 EP 检验统计量 (Epps 和 Pulley 1983)

提示

样本量 $n \geq 200$ EP 检验统计量 T_{EP} 非常接近 $n = \infty$ 时 T_{EP} 的分位数。

设 $x_1, \ldots, x_n$ 是来自正态总体 $\mathcal{N}(\mu, \sigma^2)$ 的样本，EP 检验统计量定义为

$$T_{EP} = 1 + \frac{n}{\sqrt{3}} + \frac{2}{n} \sum_{i=2}^{n} \sum_{j=1}^{i-1} \exp\left\{ -\frac{(x_j - x_i)^2}{2s_\star^2} \right\} - \sqrt{2} \sum_{i=1}^{n} \exp\left\{ -\frac{(x_i - \bar{x})^2}{4s_\star^2} \right\}$$

其中 $\bar{x}, s_\star^2$ 分别是样本均值和（除以 n 的）样本方差。

19.6.2 同分布检验

Lilliefors 检验 [3] 和单样本的 ks 检验的关系

> As to whether you can do a **Lilliefors test** for several groups, that depends entirely on your ability to understand what the underlying question would be (see Adams D 1979).
>
> — Knut M. Wittkowski [4]

[2] https://stat.ethz.ch/pipermail/r-help/2009-May/390164.html
[3] https://personal.utdallas.edu/~herve/Abdi-Lillie2007-pretty.pdf
[4] https://stat.ethz.ch/pipermail/r-help/2004-February/045597.html

Kolmogorov-Smirnov 检验：单样本或两样本的同分布检验 `ks.test()`

```
# 数据 x 与正态分布比较
ks.test(x, y = "pnorm")

#>
#>  Asymptotic one-sample Kolmogorov-Smirnov test
#>
#> data:  x
#> D = 0.85897, p-value < 2.2e-16
#> alternative hypothesis: two-sided
```

19.6.3 相关性检验

样本的相关性检验 `cor.test()`：Pearson's 相关系数检验，Kendall's τ 检验或者 Spearman's ρ 检验。基于美国高等法院律师对州法官的评级数据集 USJudgeRatings 介绍各项评分之间的相关性。

```
# cor.test(method = "pearson")  # lm(y ~ 1 + x)
cor.test(~ CONT + INTG, data = USJudgeRatings)

#>
#>  Pearson's product-moment correlation
#>
#> data:  CONT and INTG
#> t = -0.8605, df = 41, p-value = 0.3945
#> alternative hypothesis: true correlation is not equal to 0
#> 95 percent confidence interval:
#>  -0.4168591  0.1741182
#> sample estimates:
#>        cor
#> -0.1331909
```

其中，变量 CONT 表示律师与法官的联系次数，INTG 表示司法公正。

```
# cor.test(method = "kendall")
# cor.test(method = "spearman") # lm(rank(y) ~ 1 + rank(x))
```

19.6.4 独立性检验

时间序列独立性检验 `Box.test()` 计算 Box-Pierce 或 Ljung-Box 检验统计量来检查给定时间序列的独立性假设。

©黄湘云

19.6.5 平稳性检验

时间序列单位根检验，检验时间序列平稳性 Phillips-Perron 的单位根检验 `PP.test()`

```
PP.test(x, lshort = TRUE)
```

19.7 多元分布情形

- Hotelling T^2 检验：总体服从多元正态分布，两样本均值之差的检验。
- Mauchly 球形检验：总体服从多元正态分布，单样本协方差矩阵的检验。

19.7.1 Hotelling T^2 检验

Hotelling T^2 检验是一维情形下两样本 t 检验的多维推广。

19.7.2 Mauchly 球形检验

Mauchly 球形检验 `mauchly.test()` 检验：Wishart 分布的协方差矩阵是否正比于给定的矩阵。一组样本来自多元正态分布，样本的协方差矩阵是关于样本的随机矩阵，随机矩阵的分布服从 Wishart 分布。

如果 $\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_m}$，$\boldsymbol{x_i} \in \mathbb{R}^p$，$\boldsymbol{x_i} \overset{i.i.d}{\sim} \mathrm{MVN}(0, \Sigma)$，即 m 个样本点都服从均值为 0，协方差矩阵为 Σ 的 p 维多元正态分布 $\mathrm{MVN}(0, \Sigma)$，且样本点之间相互独立。则 $X = \boldsymbol{x}^\top \boldsymbol{x}$ 服从参数为 Σ ，自由度为 m 的 Wishart 分布 $W_p(\Sigma, m)$。概率密度函数如下

$$f(X) = \frac{1}{2^{\frac{mp}{2}} |\Sigma|^{\frac{m}{2}} \Gamma_p(\frac{m}{2})} |X|^{(m-p-1)/2} \exp\{-\frac{1}{2} \mathrm{tr}(\Sigma^{-1} X)\}$$

其中，Γ_p 是多元伽马函数，定义如下

$$\Gamma_p(\frac{m}{2}) = \pi^{p(p-1)/4} \prod_{j=1}^{p} \Gamma(\frac{m}{2} - \frac{j-1}{2})$$

R 语言内置了一个模拟数生成器，可以直接模拟出服从 Wishart 分布 $W_p(\Sigma, m)$ 的样本，$m = \mathrm{df}, \Sigma = \mathrm{Sigma}$。R 语言命令如下：

```
rWishart(n, df, Sigma)
```

其中，整型参数 `n` 指定样本量，数值参数 `df` 指定自由度，正定的 $p \times p$ 矩阵 `Sigma` 指定 Wishart 分布的矩阵参数。`rWishart()` 返回一个 $p \times p \times n$ 数组 R，其中 $R[, , i]$ 是正定矩阵，是服从 Wishart 分布 $W_p(\Sigma, m)$ 的一个样本点，其中 $m = \mathrm{df}, \Sigma = \mathrm{Sigma}$。

汪荣鑫©著

```r
set.seed(2022)
# 构造 n 个随机矩阵
S <- matrix(c(1.2, 0.9, 0.9, 1.2), nrow = 2, ncol = 2)
rWishart(n = 3, df = 2, Sigma = S)
```

```
#> , , 1
#>
#>          [,1]      [,2]
#> [1,] 3.213745 1.2445391
#> [2,] 1.244539 0.5032642
#>
#> , , 2
#>
#>          [,1]     [,2]
#> [1,] 4.443057 3.387850
#> [2,] 3.387850 2.605341
#>
#> , , 3
#>
#>          [,1]     [,2]
#> [1,] 3.614911 4.797919
#> [2,] 4.797919 6.846811
```

随机矩阵 M 的期望 $\mathsf{E}(M) = m \times \Sigma$，随机矩阵 M 中每个元素的方差

$$\mathsf{Var}(M_{ij}) = m(\Sigma_{ij}^2 + \Sigma_{ii}\Sigma_{jj}), \quad S = \Sigma$$

若 $p = 1$，即 Σ 是一个标量 σ^2，Wishart 分布退化为自由度为 df 的卡方分布 χ^2，即 $W_1(\sigma^2, m) = \sigma^2\chi_m^2$。下面计算随机矩阵 M 的期望。

```r
set.seed(2022)
Wish <- rWishart(n = 3000, df = 2, Sigma = S)
# 计算随机矩阵 M 的期望
apply(Wish, MARGIN = 1:2, FUN = mean)
```

```
#>          [,1]     [,2]
#> [1,] 2.375915 1.792558
#> [2,] 1.792558 2.430074
```

```r
# 随机矩阵 M 的期望理论值
2 * S
```

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

© 王湘君


```
#> [1,]  2.4  1.8
#> [2,]  1.8  2.4
```

接着计算随机矩阵 M 的方差。

```
# 样本方差
apply(Wish, MARGIN = 1:2, var)
```

```
#>          [,1]     [,2]
#> [1,] 5.668746 4.472606
#> [2,] 4.472606 5.729270
```

```
# 理论方差
2*(S^2 + tcrossprod(diag(S)))
```

```
#>      [,1] [,2]
#> [1,] 5.76 4.50
#> [2,] 4.50 5.76
```

19.8 假设检验的一些注记

真实数据的情况是复杂多样的，本章按照数据情况对检验方法分类，方便读者根据手头的数据情况，快速从众多的方法中定位最合适的检验方法。依次是单样本检验、两样本检验、多样本检验、计数数据检验、配对样本检验。如果已知符合参数检验的条件，优先考虑参数检验。如果不确定是否符合参数检验的条件，对参数检验和非参数检验方法都适用，非参数检验方法的功效更大，方法更优。在总体分布未知的情况下，无论是对均值检验还是对方差检验，大部分情况下都需要非参数检验方法。

在假设检验理论方面作出贡献的人非常多，自 Karl Pearson 提出卡方统计量、卡方分布和卡方检验以来，陆续涌现出来一批人及载入史册的工作，见下表。不难看出，19 世纪后半叶至 20 世纪前半叶，假设检验理论经过一个世纪的发展趋于成熟。从假设检验这个细分领域也印证了世界的统计中心从英国逐渐转移到美国，相比而言，中国在这方面的贡献微乎其微。笔者同时也注意到很多检验方法都是以人名命名的，且已经被编写到各类统计软件中。R 语言中有十分丰富的统计检验函数，根据这些函数及其帮助文档可以追溯到检验方法的发明者，再从维基百科中找到学者及其提出的检验方法的详情页，最后，根据学者的出生日期排序整理成表格。

表格 19.7: 对假设检验理论有重要贡献的学者

姓名	国籍	出生	死亡	寿命	贡献
K. Pearson	英国	1857-03-27	1936-04-27	79	卡方分布、卡方检验
C. Spearman	英国	1863-09-10	1945-09-17	82	Spearman's ρ
W. S. Gosset	英国	1876-06-13	1937-10-16	61	t 分布、t 检验
R. A. Fisher	英国	1890-02-17	1962-07-29	72	F 检验、Fisher 精确检验
F. Wilcoxon	美国	1892-09-02	1965-11-18	73	Wilcoxon 秩检验

姓名	国籍	出生	死亡	寿命	贡献
H. Cramér	瑞士	1893-09-25	1985-10-05	92	Cramér's V
J. Neyman	波兰、美国	1894-04-16	1981-08-05	87	Neyman-Pearson 引理
E. S. Pearson	英国	1895-08-11	1980-06-12	84	Neyman-Pearson 引理
H. Hotelling	美国	1895-09-29	1973-12-26	78	Hotelling T^2 检验
E. J. G. Pitman	澳大利亚	1897-10-29	1993-07-21	95	Pitman 估计
J. Wishart	英国	1898-11-28	1956-07-14	57	Wishart 分布
Q. M. McNemar	美国	1900-02-20	1986-07-03	86	McNemar 检验
F. Yates	英国	1902-05-12	1994-06-17	92	Yates 矫正
A. Wald	匈牙利	1902-10-31	1950-12-13	48	Wald 检验
A. Kolmogorov	苏联	1903-04-25	1987-10-20	84	Kolmogorov-Smirnov 检验
S. S. Wilks	美国	1906-06-17	1964-03-07	57	Wilks 检验/似然比检验
J. W. Mauchly	美国	1907-08-30	1980-01-08	72	Mauchly 球形检验
M. Kendall	英国	1907-09-06	1983-03-29	76	Kendall's τ
W. G. Cochran	英国、美国	1909-07-15	1980-03-29	70	Cochran–Mantel–Haenszel 检验
M. S. Bartlett	英国	1910-06-18	2002-01-08	91	Bartlett 检验
W. M. Haenszel	美国	1910-06-19	1998-03-13	87	Cochran–Mantel–Haenszel 检验
B. L. Welch	英国	1911	1989-12-29	78	Welch t 检验
H. O. Hartley	德国	1912-04-13	1980-12-30	68	Hartley 检验
M. Friedman	美国	1912-07-31	2006-11-16	94	Friedman 秩和检验
W. A. Wallis	美国	1912-11-05	1998-10-12	85	Kruskal-Wallis 检验
H. Levene	美国	1914-01-17	2003-07-02	89	Levene 检验
J. W. Tukey	美国	1915-06-16	2000-07-26	85	Tukey's HSD 检验
O. J. Dunn	美国	1915-09-01	2008-01-12	92	Dunn 检验
E. L. Lehmann	法国、美国	1917-11-20	2009-09-12	91	Lehmann-Scheffé 定理
T. W. Anderson	美国	1918-06-05	2016-09-17	98	Anderson–Darling 检验
N. Mantel	美国	1919-02-16	2002-05-25	83	Cochran–Mantel–Haenszel 检验
W. Kruskal	美国	1919-10-10	2005-04-21	85	Kruskal-Wallis 检验
George E. P. Box	英国、美国	1919-10-18	2013-03-28	93	Box-Pierce 检验
C. R. Rao	印度、美国	1920-09-10	2023-08-22	102	Score 检验
M. Wilk	加拿大	1922-12-18	2013-02-19	90	Shapiro-Wilk 检验
J. Durbin	英国	1923-06-30	2012-06-23	88	Durbin 检验
L. Le Cam	法国	1924-11-18	2000-04-25	75	渐近理论
H. Lilliefors	美国	1928-06-14	2008-02-23	80	Lilliefors 检验
S. S. Shapiro	美国	1930-07-13	-	93	Shapiro-Wilk 检验

©黄湘云

> **注释**
>
> 笔者仅根据自己搜集了解的材料制作此表，受一定的局限，或有缺漏和主观。20 世纪 60 年代后，假设检验理论开始成熟起来了，所以仅考虑 1930 年以前出生的。此外，学者需具有一定的名气，至少收录在维基百科词条里。

其中，最重要的统计学家及其学术传承关系见下图 19.9 。

图 19.9: 最重要的统计学家及其学术传承关系

F. Galton 是 K. Pearson 的老师，E. S. Pearson 是 K. Pearson 的儿子。E. L. Lehmann 是 J. Neyman 的学生，J. Neyman 和 E. S. Pearson 一起提出 N-P 引理，是置信区间和假设检验理论的奠基人。假设检验和区间估计、决策理论是紧密相关的，A. Wald 是继 J. Neyman 和 E. S. Pearson 之后，继续开疆拓土的一位统计学家，不幸的是，在一场飞机事故中英年早逝。

19.8.1 假设检验和多重比较的关系

FDR 是 False Discovery Rate 的简称

19.8.2 假设检验和方差分析的关系

19.8.2.1 单因素一元方差分析

函数 `aov()` 可以做单、双因素一元方差分析

```
fit_aov <- aov(weight ~ group, data = PlantGrowth)
```

两两比较，多重比较

```
TukeyHSD(fit_aov)
```

```
#>   Tukey multiple comparisons of means
#>     95% family-wise confidence level
#>
#> Fit: aov(formula = weight ~ group, data = PlantGrowth)
#>
```

统计学习 ©

```
#> $group
#>             diff        lwr       upr     p adj
#> trt1-ctrl -0.371 -1.0622161 0.3202161 0.3908711
#> trt2-ctrl  0.494 -0.1972161 1.1852161 0.1979960
#> trt2-trt1  0.865  0.1737839 1.5562161 0.0120064
```

自己实现方差分析

```
# 自由度
df1 <- 2
df2 <- 27
# 每组样本量
group.size <- 10
# 组间方差
sq.between <- sum(tapply(
  PlantGrowth$weight, PlantGrowth$group,
  function(x) (mean(x) - mean(PlantGrowth$weight))^2
)) * group.size

mean.sq.between <- sq.between / df1

# 组内方差
sq.within <- sum(tapply(
  PlantGrowth$weight, PlantGrowth$group,
  function(x) sum((x - mean(x))^2)
))

mean.sq.within <- sq.within / df2
# F 统计量
f.value <- mean.sq.between / mean.sq.within
f.value
```

```
#> [1] 4.846088
```

```
# P 值
p.value <- 1 - pf(f.value, df1, df2)
p.value
```

```
#> [1] 0.01590996
```

从假设检验角度看单因素方差分析，方差分析其实是在比较多个组的均值是否有显著差异。

```
oneway.test(weight ~ group, data = PlantGrowth, var.equal = TRUE)
```

```
#>
```

©黄湘云


```
#>  One-way analysis of means
#>
#> data:  weight and group
#> F = 4.8461, num df = 2, denom df = 27, p-value = 0.01591
```

方差分析还可以纳入线性模型的框架内

```
fit <- lm(weight ~ group, data = PlantGrowth)
summary(fit)
```

```
#>
#> Call:
#> lm(formula = weight ~ group, data = PlantGrowth)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -1.0710 -0.4180 -0.0060  0.2627  1.3690
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   5.0320     0.1971  25.527   <2e-16 ***
#> grouptrt1    -0.3710     0.2788  -1.331   0.1944
#> grouptrt2     0.4940     0.2788   1.772   0.0877 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.6234 on 27 degrees of freedom
#> Multiple R-squared:  0.2641, Adjusted R-squared:  0.2096
#> F-statistic: 4.846 on 2 and 27 DF,  p-value: 0.01591
```

```
anova(fit)
```

```
#> Analysis of Variance Table
#>
#> Response: weight
#>           Df  Sum Sq Mean Sq F value  Pr(>F)
#> group      2  3.7663  1.8832  4.8461 0.01591 *
#> Residuals 27 10.4921  0.3886
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

假定各个组来自正态总体，且它们的方差相同，从 F 统计量的值和检验的 P 值看，方差分析 `aov()` 、假设检验 `oneway.test()` 和线性模型 `lm()` 在这里等价了。

© 科学出版社

19.8.2.2 双因素一元方差分析

```
with(ToothGrowth, interaction.plot(supp, dose, len))
```

图 19.10: OJ 和 VC 的交互作用

如果 dose = 2，则 len 与提供的方式 supp 没有关系。

```
fit_aov <- aov(len ~ supp * dose, data = ToothGrowth)
fit_aov

#> Call:
#>    aov(formula = len ~ supp * dose, data = ToothGrowth)
#>
#> Terms:
#>                      supp      dose supp:dose Residuals
#> Sum of Squares   205.3500 2224.3043   88.9201  933.6349
#> Deg. of Freedom         1         1         1        56
#>
#> Residual standard error: 4.083142
#> Estimated effects may be unbalanced
```

© 王建平

19.8.2.3 单因素多元方差分析

PlantGrowth 属于一元方差分析，观测变量只有植物干重一个变量。如果推广到多个变量，就是多元方差分析 multivariate analysis of variance 。不同种类的鸢尾花的萼片长度的分布有所不同。

```
library(ggplot2)
library(ggridges)
ggplot(data = iris, aes(x = Sepal.Length, y = Species, fill = Species)) +
  scale_fill_brewer(palette = "Greys") +
  geom_density_ridges(bandwidth = 0.2) +
  theme_ridges(font_size = 12, font_family = "sans")
```

图 19.11: 鸢尾花萼片长度的分布

```
fit <- manova(cbind(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width) ~ Species, data = iris)
summary(fit, test = "Wilks")
```

```
#>            Df    Wilks approx F num Df den Df    Pr(>F)
#> Species     2 0.023439   199.15      8    288 < 2.2e-16 ***
#> Residuals 147
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

P 值小于 0.0.5，说明 iris 数据集三个组的均值向量有显著差异。关于均值向量的检验方法，请看 ? summary.manova 。

©黄湘云

按 Species 分组统计各个变量的样本均值、样本方差

```
aggregate(data = iris, cbind(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width) ~ Species, 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
```

```
aggregate(data = iris, cbind(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width) ~ Species, var)
```

```
#>      Species Sepal.Length Sepal.Width Petal.Length Petal.Width
#> 1     setosa    0.1242490  0.14368980   0.03015918  0.01110612
#> 2 versicolor    0.2664327  0.09846939   0.22081633  0.03910612
#> 3  virginica    0.4043429  0.10400408   0.30458776  0.07543265
```

19.8.3 假设检验与区间估计的关系

区间估计的意义是解决点估计可靠性问题，它用置信系数解决了对估计结果的信心问题，弥补了点估计的不足。置信系数是最大的置信水平。

Base R 提供的 `binom.test()` 函数可以精确计算置信区间，即所谓的 Clopper-Pearson 区间，而 `prop.test()` 函数可近似计算置信区间，即所谓的 Wilson 区间。以单样本的比例检验为例。

```
# 近似区间估计
prop.test(x = 2, n = 10, p = 0.95, conf.level = 0.95, correct = TRUE)
```

```
#> Warning in prop.test(x = 2, n = 10, p = 0.95, conf.level = 0.95, correct =
#> TRUE): Chi-squared approximation may be incorrect
```

```
#>
#>  1-sample proportions test with continuity correction
#>
#> data:  2 out of 10, null probability 0.95
#> X-squared = 103.16, df = 1, p-value < 2.2e-16
#> alternative hypothesis: true p is not equal to 0.95
#> 95 percent confidence interval:
#>  0.03542694 0.55781858
#> sample estimates:
#>   p
#> 0.2
```

```
# 精确区间估计
binom.test(x = 2, n = 10, p = 0.95, conf.level = 0.95)
```

```
#>
```

©黄湘云

```
#>  Exact binomial test
#>
#> data:  2 and 10
#> number of successes = 2, number of trials = 10, p-value = 1.605e-09
#> alternative hypothesis: true probability of success is not equal to 0.95
#> 95 percent confidence interval:
#>  0.02521073 0.55609546
#> sample estimates:
#> probability of success
#>                    0.2
```

实际达到的置信度水平随真实的未知参数值和样本量的变化而**剧烈**波动，这意味着这种参数估计方法在实际应用中不可靠、真实场景中参数真值是永远未知的，样本量是可控的，并且是可以变化的。根本原因在于这类分布是离散的，比如这里的二项分布。当样本数据服从离散的分布，置信区间的端点也是离散的。这种缺陷是无法避免的，清晰的置信区间和离散的数据之间存在无法调和的冲突。

19.8.4 常见的统计检验是线性模型

两样本的均值检验：非参数检验方法

19.8.4.1 Wilcoxon 符号秩检验

与 `wilcox.test()` 等价的线性模型

```
signed_rank <- function(x) sign(x) * rank(abs(x))
fit <- lm(signed_rank(extra) ~ group, data = sleep)
summary(fit)
```

```
#>
#> Call:
#> lm(formula = signed_rank(extra) ~ group, data = sleep)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -14.55  -6.55   0.90   6.90  13.95
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)    3.050      2.872   1.062   0.3022
#> group2         8.300      4.061   2.044   0.0559 .
#> ---
```

```
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 9.081 on 18 degrees of freedom
#> Multiple R-squared:  0.1884, Adjusted R-squared:  0.1433
#> F-statistic: 4.177 on 1 and 18 DF,  p-value: 0.05589
```

19.8.4.2 Kruskal-Wallis 秩和检验

与 kruskal.test() 等价的线性模型表示。

```
fit <- lm(rank(extra) ~ group, data = sleep)
summary(fit)
```

```
#>
#> Call:
#> lm(formula = rank(extra) ~ group, data = sleep)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -8.450 -3.925 -0.500  5.275  8.950
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)    8.050      1.738   4.633 0.000207 ***
#> group2         4.900      2.457   1.994 0.061520 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 5.495 on 18 degrees of freedom
#> Multiple R-squared:  0.1809, Adjusted R-squared:  0.1354
#> F-statistic: 3.976 on 1 and 18 DF,  p-value: 0.06152
```

19.8.5 假设检验的工业应用

传统的试验设计为什么不适用于互联网？因为 Fisher 的实验设计和方差分析，主要针对的是受控对象，比如测试武器、肥料配比、飞机制造等实体的东西。互联网是虚拟经济，实验的对象是人，对平台来说，人的行为是半知半解，更不受控，所以需要成千上万、乃至几十万的样本才能抵消样本内部的随机性。互联网数据的噪声太多、太大了，微小的变化就好像一粒小石子扔进大海里，要获得样本间显著的差异性，需要累积相当的样本量。另一方面，大型的互联网公司，搜索、推荐、广告等业务相对成熟，提升关键指标，拿到好的结果，往往比较困难。成熟的业务几乎不太可能一次实验拿到很好的结果，所以，

© 黄湘云

方向比努力重要，更快地迭代，跑在同行前面，更快地试错（想法），试更多的错（想法），更好地试错（想法），累积更多的经验，做更多地创新，这是 A/B 实验平台的核心价值。

曾经，在学校里，我总想获得一个全局最优解，并且还有这样的情结，到了厂里，发现没人研究全局最优解，大家都在做 A/B 实验优化自己的子业务和方向。有时候这个细分业务方向甚至也就小几万的用户了。全局最优解和局部最优解，我们不太可能获得全局最优解，一则全局最优解受影响的因素很多，而这些因素变化很快，所以，即使可以获得全局最优解，代价会非常大，那么，怎么办呢？还不如获取局部最优解，研究一个个局部显然比研究全局要简单的多，此外，研究局部的好处是可以快速地随业务迭代。

一个完整的实验周期包含提出问题、设计实验、收集数据、组织数据、统计检验、分析结论、数据解读、数据交流、决策行动、业务价值。这是一个闭环，根据业务中发现的问题，提出解决方法，并设计实验验证。问题有时候就是机会，奋斗的方向，解决问题自然就会带来业务价值。实验又可以按业务问题、数据问题和统计问题划分三个阶段。

- 业务问题：根据目标确定方向，找到有价值的、可以解决的业务问题，再提出合理的统计假设。
- 数据问题：数据收集、数据组织、数据管理、数据治理，验证数据流的完整性、一致性等。
- 统计问题：设计实验方案，包括分流、实验周期等，利用假设检验、区间估计和功效分析等统计工具完成显著性分析、可靠性分析，撰写数据分析和评估报告。

19.9 习题

1. 分析《红楼梦》的情景描写。参考 2009 年东南大学韦博成教授将两个独立二项总体的等价性检验应用于《红楼梦》前 80 回与后 40 回某些文风差异的统计分析 (韦博成 2009)。

2. 根据数据集 chickwts 分析不同喂食方式对小鸡体重的影响。(单因素方差分析)

```
ggplot(data = chickwts, aes(x = feed, y = weight)) +
  geom_boxplot() +
  geom_jitter() +
  theme_minimal()
```

© 基础统计

图 19.12: 不同喂食方式对小鸡的影响

3. 根据数据集 ChickWeight 分析 4 种喂食方式对小鸡体重有影响，每个小鸡本身对喂食方式的接受、吸收程度不一样、它们本身的素质不一样（个体差异），要考察喂食的方式的影响，应该剔除掉个体差异，才是喂食方式的真正影响。

```
ggplot(data = ChickWeight, aes(x = Time, y = weight, group = Chick, color = Diet)) +
  geom_point() +
  geom_line() +
  facet_wrap(~Diet) +
  theme_minimal()
```

© 黄湘云

图 19.13: 不同喂食方式对小鸡的影响（续）

© 王湘君

第二十章　回归与相关分析

20.1　子代身高与亲代身高的关系

弗朗西斯 · 高尔顿（Francis Galton, 1822-1911）是历史上著名的优生学家、心理学家、遗传学家和统计学家，是统计学中相关和回归等一批概念的提出者，是遗传学中回归现象的发现者。1885 年，高尔顿以保密和给予金钱报酬的方式，向社会征集了 205 对夫妇及其 928 个成年子女的身高数据 (Galton 1886)。

目前，Michael Friendly 从原始文献中整理后，将该数据集命名为 `GaltonFamilies`，放在 R 包 **HistData** (Friendly 2021) 内，方便大家使用。篇幅所限，下表格 20.1 展示该数据集的部分内容。

表格 20.1: 高尔顿收集的 205 对夫妇及其子女的身高数据（部分）

家庭编号	父亲身高	母亲身高	中亲身高	子女数量	子女编号	子女性别	子女身高
001	78.5	67.0	75.43	4	1	male	73.2
001	78.5	67.0	75.43	4	2	female	69.2
001	78.5	67.0	75.43	4	3	female	69.0
001	78.5	67.0	75.43	4	4	female	69.0
002	75.5	66.5	73.66	4	1	male	73.5
002	75.5	66.5	73.66	4	2	male	72.5

表中子女性别一栏，Male 表示男性，Female 表示女性。表中 1 号家庭父亲身高 78.5 英寸，母亲身高 67.0 英寸，育有 4 个成年子女，1 男 3 女，子女身高依次是 73.2 英寸、69.2 英寸、69.0 英寸和 69.0 英寸。1 英寸相当于 2.54 厘米，78.5 英寸相当于 199.39 厘米，约等于 2 米的身高。

高尔顿提出「中亲」概念，即父母的平均身高，认为子代身高只与父母平均身高相关，而与父母身高差无关，为了消除性别给身高带来的差异，女性身高均乘以 1.08。

根据数据统计的均值和协方差，椭圆 level = 0.95

女儿的身高乘以 1.08 后，两条回归线将几乎重合。(Hanley 2004)

$$\text{height}_{children} = \alpha + \beta * \text{height}_{midparent} + \epsilon$$

© 黄湘云

图 20.1: 子代身高与亲代身高的关系

© 黄湘云

图 20.2: 子代身高与亲代身高的关系

© 黄湘云

表格 20.2: 子女身高向中亲平均身高回归

性别	截距	中亲身高
male	19.91346	0.7132745
female	19.80016	0.7136104

图 20.3: 二维核密度估计与二元正态分布

向均值回归现象最早是高尔顿在甜豌豆实验中发现的，实际上，均值回归现象在社会经济和自然界中广泛存在，比如一个人的智力水平受家族平均水平的影响。

20.2 预期寿命与人均收入的关系

生物遗传的回归现象，更确切地说是因果而不是相关，是一种近似的函数关系。与回归紧密相连的是另一个统计概念是相关，主要刻画数量指标之间的关系深浅程度，相关系数是其中一个度量。在经济、社会领域中，很多数据指标存在相关性，接下来的这个例子基于 1977 年美国人口调查局发布的统计数据，篇幅所限，下表格 20.3 展示美国各州的部分统计数据。

© 基础统计

表格 20.3: 1977 年美国人口调查局发布的各州统计数据（部分）

州名	区域划分	人口数量	人均收入	预期寿命
Alabama	South	3615	3624	69.05
Alaska	West	365	6315	69.31
Arizona	West	2212	4530	70.55
Arkansas	South	2110	3378	70.66
California	West	21198	5114	71.71
Colorado	West	2541	4884	72.06

该数据集在 R 环境中的结构如下：

```
str(state_x77)
```

```
#> 'data.frame':    50 obs. of  10 variables:
#>  $ Population  : num  3615 365 2212 2110 21198 ...
#>  $ Income      : num  3624 6315 4530 3378 5114 ...
#>  $ Illiteracy  : num  2.1 1.5 1.8 1.9 1.1 0.7 1.1 0.9 1.3 2 ...
#>  $ Life Exp    : num  69 69.3 70.5 70.7 71.7 ...
#>  $ Murder      : num  15.1 11.3 7.8 10.1 10.3 6.8 3.1 6.2 10.7 13.9 ...
#>  $ HS Grad     : num  41.3 66.7 58.1 39.9 62.6 63.9 56 54.6 52.6 40.6 ...
#>  $ Frost       : num  20 152 15 65 20 166 139 103 11 60 ...
#>  $ Area        : num  50708 566432 113417 51945 156361 ...
#>  $ state_name  : chr  "Alabama" "Alaska" "Arizona" "Arkansas" ...
#>  $ state_region: Factor w/ 4 levels "Northeast","South",..: 2 4 4 2 4 4 1 2 2 2 ...
```

它是一个 50 行 10 列的数据框，其中，state_name（州名）是字符型变量，state_region（区域划分）是因子型变量。除了这两个变量外，Population（人口数量，单位：1000），Income（人均收入，单位：美元），Life Exp（预期寿命，单位：岁）等都是数值型的变量。下图 20.4 展示了 1977 年美国各州的预期寿命和人均收入的关系，通过此图，可以初步观察出两个指标存在一些明显的正向相关性，也符合常识。

为了更加清楚地观察到哪些州预期寿命长，哪些州人均收入高，在图 20.4 基础上，在散点旁边添加州名。此外，为了观察各州的地域差异，根据各州所属区域，给散点分类，最后，将各州人口数量映射给散点的大小，形成如下图 20.5 所示的分类气泡图。

整体来说，预期寿命与人均收入息息相关。

> **提示**
>
> 从图 20.5 到图 20.6 ，尝试初步量化两个变量之间的相关性之前，有没有想过，回归线应该更加陡峭一些，即回归线的斜率应该更大一些，是什么原因导致平缓了这么多？是阿拉斯加州和内华

© 黄湘云

图 20.4: 预期寿命与人均收入的关系图

© 统计出版社

图 20.5: 分地域预期寿命与人均收入的气泡图

©黄湘云

图 20.6: 1977 年美国各州预期寿命与人均收入的关系：回归分析

©黄湘云

> 达州的数据偏离集体太远。那又是什么原因导致阿拉斯加州人均收入全美第一，而预期寿命倒数呢？同样的，内华达州的人均收入也不低，但预期寿命为什么上不去呢？

```
m <- lm(data = state_x77, `Life Exp` ~ Income)
summary(m)
```

```
#>
#> Call:
#> lm(formula = `Life Exp` ~ Income, data = state_x77)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -2.96547 -0.76381 -0.03428  0.92876  2.32951
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 6.758e+01  1.328e+00  50.906   <2e-16 ***
#> Income      7.433e-04  2.965e-04   2.507   0.0156 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.275 on 48 degrees of freedom
#> Multiple R-squared:  0.1158, Adjusted R-squared:  0.09735
#> F-statistic: 6.285 on 1 and 48 DF,  p-value: 0.01562
```

输出结果中各个量的计算公式及 R 语言实现，比如方差 Variance、偏差 Deviance/Bias、残差 Residual Error

20.3 分析影响入院等待时间的因素

医院的床位是非常重要的资源。

```
hospital_waiting_time <- readRDS(file = "data/hospital_waiting_time.rds")
```

```
str(hospital_waiting_time)
```

```
#> 'data.frame':    2625 obs. of  11 variables:
#>  $ 等待时间    : num  1 1.2 20 6 8.9 2.9 7.9 2.8 2.7 5 ...
#>  $ 门诊次      : int  2 7 43 1 3 1 10 3 6 2 ...
#>  $ 住院次      : int  1 1 1 1 1 1 1 1 1 1 ...
#>  $ 开住院条日期: int  3 3 3 3 3 3 3 3 3 3 ...
#>  $ 性别        : int  0 0 1 1 1 1 0 1 1 1 ...
```

© 黄湘云


```
#>  $ 年龄        : int  42 32 59 9 45 73 50 25 14 20 ...
#>  $ 入院疾病分类: int  3 1 1 3 3 3 4 1 2 3 ...
#>  $ 入院目的    : int  1 1 1 1 1 1 1 1 1 1 ...
#>  $ 住院类别    : int  2 2 2 2 2 2 2 2 2 2 ...
#>  $ 入院病情    : int  1 1 1 1 1 1 1 1 1 1 ...
#>  $ 医生        : int  2 2 2 2 2 4 2 2 4 4 ...
```

20.4 习题

1. R 软件内置的数据集 `esoph` 是一份关于法国伊勒-维莱讷地区食道癌的数据，请读者根据这份数据研究年龄组、烟草消费量、酒精消费量（每日喝酒量）和患食道癌的关系。

©黄湘云

第二十一章　分类数据的分析

1. 最常见的两个广义线性模型：泊松和逻辑回归
2. 理论公式、R 输出及其解释，应用案例
3. 与计数/离散数据的假设检验的关系
4. 辛普森悖论，分类数据处理，高维列联表的压缩和分层，边际和条件
5. 泰坦尼克号 4x2x2x2 高维复杂列联表分析

```
library(MASS)
```

计数数据，通俗来说，对象是一个一个或一份一份的，可数的、离散的数据，比如人数。列联表来组织数据，分二维和多维的情况。

21.1　比例检验

21.1.1　单样本检验

比例检验函数 `prop.test()` 检验比例是否等于给定的值。单样本的比例检验结果中比例的区间估计与 Wilson 区间估计 (Wilson 1927) 是相关的。区间估计与假设检验是有紧密关系的，对于二项分布比例的 11 种区间估计方法的比较 (Newcombe 1998)。

21.1.1.1　近似检验

21.1.1.2　精确检验

函数 `binom.test()` 来做二项检验，函数 `binom.test()` 用来检验伯努利试验中成功概率 p 和给定概率 p_0 的关系，属于精确检验 (Clopper 和 Pearson 1934)。

比例 p 的检验，做 n 次独立试验，样本 $X_1, \ldots, X_n \sim b(1, p)$，事件发生的总次数 $\sum_{i=1}^{n} X_i$。

```r
# 模拟一组样本
set.seed(20232023)
x <- sample(x = c(0, 1), size = 100, replace = TRUE, prob = c(0.8, 0.2))
```

二项分布中成功概率的检验

```r
binom.test(sum(x), n = 100, p = 0.5)
```

```
#>
#>  Exact binomial test
#>
#> data:  sum(x) and 100
#> number of successes = 23, number of trials = 100, p-value = 5.514e-08
#> alternative hypothesis: true probability of success is not equal to 0.5
#> 95 percent confidence interval:
#>  0.1517316 0.3248587
#> sample estimates:
#> probability of success
#>                   0.23
```

检验成功概率 p 是否等于 0.5，P 值 5.514×10^{-8} 结论是拒绝原假设

```r
binom.test(sum(x), n = 100, p = 0.2)
```

```
#>
#>  Exact binomial test
#>
#> data:  sum(x) and 100
#> number of successes = 23, number of trials = 100, p-value = 0.4534
#> alternative hypothesis: true probability of success is not equal to 0.2
#> 95 percent confidence interval:
#>  0.1517316 0.3248587
#> sample estimates:
#> probability of success
#>                   0.23
```

检验成功概率 p 是否等于 0.2，P 值 0.4534 结论是不能拒绝原假设

切比雪夫不等式（Chebyshev, 1821-1894）。设随机变量 X 的数学期望和方差都存在，则对任意常数 $\epsilon > 0$，有

$$P(|X - EX| \geq \epsilon) \leq \frac{Var(X)}{\epsilon^2}$$
$$P(|X - EX| \leq \epsilon) \geq 1 - \frac{Var(X)}{\epsilon^2}$$

© 基础统计

21.1.2 两样本检验

关于两样本的比例检验问题

$$H_0: P_A = P_B \quad vs. \quad H_1: P_A > P_B$$
$$H_0: P_A = P_B \quad vs. \quad H_1: P_A < P_B$$

H_0 成立的情况下，暗示着两个样本来自同一总体。

比例检验函数 `prop.test()` 用来检验两组或多组二项分布的成功概率（比例）是否相等。

设随机变量 X 服从参数为 p 的二项分布 $b(n,p)$，Y 服从参数为 θ 的二项分布 $b(m,\theta)$，m,n 都假定为较大的正整数，检验如下问题

$$H_0: P_A \geq P_B \quad vs. \quad H_1: P_A < P_B$$

根据中心极限定理

$$\frac{\bar{X}-\bar{Y}}{\sqrt{\frac{p(1-p)}{n}+\frac{\theta(1-\theta)}{m}}}$$

近似服从标准正态分布 $N(0,1)$。如果用矩估计 $\bar{X}$ 和 $\bar{Y}$ 分别替代总体参数 p 和 θ，构造检验统计量

$$T = \frac{\bar{X}-\bar{Y}}{\sqrt{\frac{\bar{X}(1-\bar{X})}{n}+\frac{\bar{Y}(1-\bar{Y})}{m}}}$$

根据 Slutsky 定理，检验统计量 T 近似服从标准正态分布，当 T 偏大时，拒绝 H_0。该方法的优势在于当 n,m 比较大时，二项分布比较复杂，无法建立统计表，利用标准正态分布表来给出检验所需要的临界值，简便易行！

当 p 和 θ 都比较小，上述方法检验效果不好，原因在于由中心极限定理对 $\bar{X}$ 和 $\bar{Y}$ 的正态分布近似效果不好，或者间接地导致 $\bar{X}-\bar{Y}$ 的方差偏小，进而 T 的分辨都不好，而且当 p,θ 很接近 1 时，上述现象也会产生！

下面介绍新的解决办法，办法来自两个二项总体成功概率的比较 (宋泽熙 2011)。

上面的检验问题等价于

$$H_0: \frac{P_A}{P_B} \geq 1 \quad vs. \quad H_1: \frac{P_A}{P_B} < 1$$

引入检验统计量

$$T^\star = \frac{\bar{X}}{\bar{Y}}$$

©黄湘云

同样由 Slutsky 定理和中心极限定理可知，$\bar{X}/\bar{Y}$ 近似服从正态分布 $\mathcal{N}(1, \frac{1-\theta}{m\theta})$

当 $(T^\star - 1)/\hat{\sigma}$ 偏大时接受 H_0，临界值可通过 $\mathcal{N}(0, \hat{\sigma}^2)$ 分布表计算得到，$\hat{\sigma}^2$ 是对 $\frac{1-\theta}{m\theta}$ 的估计，比如取 $\hat{\sigma}^2 = \frac{1-\bar{Y}}{m} \cdot \frac{1}{\bar{Y}}$ 或取 $\hat{\sigma}^2 = \frac{1-\bar{Y}}{m} \cdot \frac{1}{\bar{X}}$

由于渐近方差形如 $\frac{1-\theta}{m\theta}$，因而在 θ 较小，渐近方差较大，克服了之前 $\bar{X} - \bar{Y}$ 的方差较小的问题

p, θ 很接近 1 时，我们取检验统计量

$$T^{\star\star} = \frac{1-\bar{Y}}{1-\bar{X}}$$

结论和 $T^\star$ 类似，当 $T^{\star\star}$ 偏大时，拒绝 H_0。

21.1.3 多样本检验

21.1.3.1 比例齐性检验

对多组数据的比例检验，可以理解为比例齐性检验。

21.1.3.2 比例趋势检验

比例趋势检验函数 `prop.trend.test()` 的原假设：四个组里面病人中吸烟的比例是相同的。备择假设：四个组的吸烟比例是有趋势的。

$$H_0 : P_1 = P_2 = P_3 = P_4$$
$$H_1 : P_1 < P_2 < P_3 < P_4 \text{ 或者 } P_1 > P_2 > P_3 > P_4$$

```
smokers <- c(83, 90, 129, 70)
patients <- c(86, 93, 136, 82)
prop.test(smokers, patients)
```

```
#>
#>  4-sample test for equality of proportions without continuity correction
#>
#> data:  smokers out of patients
#> X-squared = 12.6, df = 3, p-value = 0.005585
#> alternative hypothesis: two.sided
#> sample estimates:
#>    prop 1    prop 2    prop 3    prop 4
#> 0.9651163 0.9677419 0.9485294 0.8536585
```

```
prop.trend.test(smokers, patients)
```

©李东风

```
#>
#>  Chi-squared Test for Trend in Proportions
#>
#> data:  smokers out of patients ,
#>  using scores: 1 2 3 4
#> X-squared = 8.2249, df = 1, p-value = 0.004132
```

21.2 泊松检验

泊松分布是 1837 年由法国数学家泊松（Poisson, 1781-1840）首次提出。

$$p(x) = \frac{\lambda^x \exp(-\lambda)}{x!}, x = 0, 1, \cdots .$$

泊松分布的期望和方差都是 λ ，一般要求 $\lambda > 0$。

21.2.1 单样本

`poisson.test()` 泊松分布的参数 λ 的精确检验，适用于单样本和两样本。

```
poisson.test(x,
  T = 1, r = 1,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

参数 `T` 数据的时间单位

21.2.2 两样本

21.3 列联表描述

泰坦尼克号乘客生存死亡统计数据，Titanic 数据集

```
Titanic
```

```
#> , , Age = Child, Survived = No
#>
#>       Sex
#> Class  Male Female
#>   1st     0      0
#>   2nd     0      0
```

©黄湘云

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

21.3.1 行列分组表格

```
# 长格式转宽格式
titanic_data <- reshape(
  data = as.data.frame(Titanic), direction = "wide",
  idvar = c("Class", "Sex", "Age"),
  timevar = "Survived", v.names = "Freq", sep = "_"
)
```

©李东风 统计课

```r
# 制作表格
gt::gt(titanic_data) |>
  gt::cols_label(
    Freq_Yes = "存活",
    Freq_No = "死亡",
    Class = "船舱",
    Sex = "性别",
    Age = "年龄"
  )
```

表格 21.1: 泰坦尼克号乘客生存死亡统计数据

船舱	性别	年龄	死亡	存活
1st	Male	Child	0	5
2nd	Male	Child	0	11
3rd	Male	Child	35	13
Crew	Male	Child	0	0
1st	Female	Child	0	1
2nd	Female	Child	0	13
3rd	Female	Child	17	14
Crew	Female	Child	0	0
1st	Male	Adult	118	57
2nd	Male	Adult	154	14
3rd	Male	Adult	387	75
Crew	Male	Adult	670	192
1st	Female	Adult	4	140
2nd	Female	Adult	13	80
3rd	Female	Adult	89	76
Crew	Female	Adult	3	20

21.3.2 百分比堆积图

泰坦尼克号处女航乘客数量按船舱、性别、年龄和存活情况分层，**ggstats** 包绘制百分比堆积柱形图展示多维分类数据。

```r
library(ggplot2)
library(ggstats)
ggplot(as.data.frame(Titanic)) +
  aes(x = Class, fill = Survived, weight = Freq, by = Class) +
  geom_bar(position = "fill") +
```

©清华大学出版社

```
  scale_y_continuous(labels = scales::label_percent()) +
  geom_text(stat = "prop", position = position_fill(.5)) +
  facet_grid(~Sex) +
  labs(x = "船舱", y = "比例", fill = "存活")
```

图 21.1: 百分比堆积柱形图展示多维分类数据

ggstats 包提供的图层 `stat_prop()` 是 `stat_count()` 的变种，`as.data.frame(Titanic)` 中 Age 一列会自动聚合吗？by = Class 按 Class 分组聚合，统计 Survived 的比例，提供 prop 计算的变量，传递给 `geom_text()` 以添加注释，position 设置将注释放在柱子的中间

21.3.3 桑基图

用 **ggalluvial** 包 (Brunson 2020) 绘制桑基图展示多维分类数据。

```
library(ggplot2)
library(ggalluvial)
ggplot(
  data = as.data.frame(Titanic),
```

©李漫工作室

```
  aes(axis1 = Class, axis2 = Sex, axis3 = Age, y = Freq)
) +
  scale_x_discrete(limits = c("Class", "Sex", "Age")) +
  geom_alluvium(aes(fill = Survived)) +
  geom_stratum() +
  geom_text(stat = "stratum", aes(label = after_stat(stratum))) +
  theme_classic() +
  labs(
    x = "分层维度", y = "人数", fill = "存活",
    title = "泰坦尼克号处女航乘客分层情况"
  )
```

图 21.2: 桑基图展示多维分类数据

21.3.4 马赛克图

```
op <- par(mar = c(2.5, 2.5, 1.5, 0.5))
mosaicplot(~ Class + Sex + Age + Survived,
```

```
  data = Titanic, # shade = TRUE,
  color = TRUE, border = "white",
  xlab = "船舱", ylab = "性别", main = "泰坦尼克号")
par(op)
```

图 21.3: 马赛克图展示多维分类数据

vcd 包针对分类数据做了很多专门的可视化工作，内置了很多数据集和绘图函数，在 Base R 绘图基础上，整合了许多统计分析功能，提供了一个统一的可视化框架 (Meyer, Zeileis, 和 Hornik 2006; Zeileis, Meyer, 和 Hornik 2007)，更多细节见著作《Discrete Data Analysis with R: Visualization and Modeling Techniques for Categorical and Count Data》及其附带的 R 包 **vcdExtra**(Friendly 和 Meyer 2016)。

```
library(grid)
library(vcd)
mosaic(~ Class + Sex + Age + Survived,
  data = Titanic, shade = TRUE, legend = TRUE
)
```

© 黄湘云

图 21.4: 马赛克图展示多维分类数据

21.4 列联表分析

是否应该按照列联表的维度分类？还是应该从分析的目的和作用出发？比如我的目的是检验独立性。二者似乎也并不冲突。

列联表中的数据服从多项分布，关于独立性检验，有如下几种常见类型：

1. 相互独立 Mutual independence 所有变量之间相互独立，$X \perp Y \perp Z$ 。
2. 联合独立 Joint independence 两个变量的联合与第三个变量独立，$XY \perp Z$ 。
3. 边际独立 Marginal independence 当忽略第三个变量时，两个变量是独立的。列联表压缩
4. 条件独立 Conditional independence 当固定第三个变量时，两个变量是独立的，$X \perp Y|Z$。

本节数据来自著作《An Introduction to Categorical Data Analysis》(Agresti 2007) 的第 2 章习题 2.33，探索 1976-1977 年美国佛罗里达州的凶杀案件中被告肤色和死刑判决的关系。

©黄湘云

表格 21.2: 佛罗里达州的凶杀案件统计数据

		死刑	
被告	被害人	是	否
白人	白人	19	132
黑人	白人	11	52
白人	黑人	0	9
黑人	黑人	6	97

21.4.1 相互独立性

皮尔逊卡方检验（Pearson's χ^2 检验）`chisq.test()` 常用于列联表独立性检验和方差分析模型的拟合优度检验。下面是一个 2×2 的列联表。

表格 21.3: 卡方独立性检验

	第一列	第二列	合计
第一行	a	b	$a+b$
第二行	c	d	$c+d$
合计	$a+c$	$b+d$	$a+b+c+d$

```
# Death 死刑与 Defend （被告）独立性检验
m <- xtabs(Freq ~ Death + Defend, data = ethnicity)
m
```

```
#>      Defend
#> Death 白人 黑人
#>   Yes   19   17
#>   No   141  149
```

```
chisq.test(m, correct = TRUE)
```

```
#>
#>  Pearson's Chi-squared test with Yates' continuity correction
#>
#> data:  m
#> X-squared = 0.086343, df = 1, p-value = 0.7689
```

```
chisq.test(m, correct = FALSE)
```

```
#>
#>  Pearson's Chi-squared test
```

统计学习 ©黄湘

```
#>
#> data:  m
#> X-squared = 0.22145, df = 1, p-value = 0.6379
```

当被告是白人时，死刑判决 19 个，占总的死刑判决数量的 $19/36 = 52.78\%$，当被告是黑人时，死刑判决 17 个，占总的死刑判决数量的 $17/36 = 47.22\%$。判决结果与被告种族没有显著关系，但与原告（受害人）种族是有关系的，请继续往下看。

```
# Death 死刑与 Victim （原告）独立性检验
m <- xtabs(Freq ~ Death + Victim, data = ethnicity)
chisq.test(m, correct = TRUE)
```

```
#>
#>  Pearson's Chi-squared test with Yates' continuity correction
#>
#> data:  m
#> X-squared = 4.7678, df = 1, p-value = 0.029
```

```
chisq.test(m, correct = FALSE)
```

```
#>
#>  Pearson's Chi-squared test
#>
#> data:  m
#> X-squared = 5.6149, df = 1, p-value = 0.01781
```

当受害人是白人时，死刑判决 30 个，占总的死刑判决数量的 $30/36 = 83.33\%$，当受害人是黑人时，死刑判决 6 个，占总的死刑判决数量的 $6/36 = 16.67\%$。受害人是白人时，死刑判决明显多于黑人。

多维列联表

```
m <- xtabs(Freq ~ Death + Defend + Victim, data = ethnicity)
m
```

```
#> , , Victim = 白人
#>
#>      Defend
#> Death 白人 黑人
#>   Yes   19   11
#>   No   132   52
#>
#> , , Victim = 黑人
#>
#>      Defend
#> Death 白人 黑人
```

©黄湘云


```
#>   Yes    0    6
#>   No     9   97
```

判决结果、被告种族、原告种族三者是否存在联合独立性，即考虑 (Victim, Death) 是否与 Defend 独立，(Victim, Defend) 是否与 Death 独立，(Death, Defend) 与 Victim 是否相互独立。

```
fm <- loglin(table = m, margin = list(c(1, 2), c(1, 3), c(2, 3)), print = FALSE)
fm
```

```
#> $lrt
#> [1] 0.7007504
#>
#> $pearson
#> [1] 0.3751739
#>
#> $df
#> [1] 1
#>
#> $margin
#> $margin[[1]]
#> [1] "Death"  "Defend"
#>
#> $margin[[2]]
#> [1] "Death"  "Victim"
#>
#> $margin[[3]]
#> [1] "Defend" "Victim"
```

```
# 拟合对数线性模型
# fm <- loglin(m, list(c(1), c(2), c(3)))
# fm
```

似然比检验统计量 (Likelihood Ratio Test statistic), 皮尔逊 χ^2 统计量 (Pearson X-square Test statistic)

```
1 - pchisq(fm$lrt, fm$df)
```

```
#> [1] 0.4025317
```

拟合对数线性模型

```
fit_dvp <- glm(Freq ~ ., data = ethnicity, family = poisson(link = "log"))
```

模型输出

```
summary(fit_dvp)
```

```
#>
```

```
#> Call:
#> glm(formula = Freq ~ ., family = poisson(link = "log"), data = ethnicity)
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  2.45087    0.18046  13.582  < 2e-16 ***
#> DeathNo      2.08636    0.17671  11.807  < 2e-16 ***
#> Defend黑人   0.03681    0.11079   0.332     0.74
#> Victim黑人  -0.64748    0.11662  -5.552 2.83e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for poisson family taken to be 1)
#>
#>     Null deviance: 395.92  on 7  degrees of freedom
#> Residual deviance: 137.93  on 4  degrees of freedom
#> AIC: 181.61
#>
#> Number of Fisher Scoring iterations: 5
```

Pearson χ^2 统计量

```
sum(residuals(fit_dvp, type = "pearson")^2)
```

```
#> [1] 122.3975
```

MASS 包计算模型参数的置信区间

```
confint(fit_dvp, trace = FALSE)
```

```
#>                  2.5 %     97.5 %
#> (Intercept)  2.0802598  2.7893934
#> DeathNo      1.7546021  2.4493677
#> Defend黑人  -0.1803969  0.2543149
#> Victim黑人  -0.8790491 -0.4213701
```

对于单元格总样本量小于 40 或 T 小于 1 时，需采用费希尔精确检验（Fisher 's Exact 检验）。

21.4.2 边际独立性

费希尔精确检验：固定边际的情况下，检验列联表行和列之间的独立性 `fisher.test()` 。

`fisher.test()` 函数用法，统计原理和公式，适用范围和条件，概念背景和历史。

©黄湘云

费舍尔 (Sir Ronald Fisher, 1890.2 – 1962.7)[1] 和一位女士打赌，女士说能品出奶茶中奶和茶的添加顺序。

`fisher.test()` 针对计数数据，检验列联表中行和列的独立性。

```
TeaTasting <- matrix(c(3, 1, 1, 3),
  nrow = 2,
  dimnames = list(
    Guess = c("Milk", "Tea"),
    Truth = c("Milk", "Tea")
  )
)
TeaTasting
```

```
#>       Truth
#> Guess  Milk Tea
#>   Milk    3   1
#>   Tea     1   3
```

```
# 单边 P 值
fisher.test(TeaTasting, alternative = "greater")
```

```
#>
#>  Fisher's Exact Test for Count Data
#>
#> data:  TeaTasting
#> p-value = 0.2429
#> alternative hypothesis: true odds ratio is greater than 1
#> 95 percent confidence interval:
#>  0.3135693       Inf
#> sample estimates:
#> odds ratio
#>   6.408309
```

```
# 双边 P 值
fisher.test(TeaTasting, alternative = "two.sided")
```

```
#>
#>  Fisher's Exact Test for Count Data
#>
#> data:  TeaTasting
#> p-value = 0.4857
#> alternative hypothesis: true odds ratio is not equal to 1
#> 95 percent confidence interval:
```

[1] https://en.wikipedia.org/wiki/Ronald_Fisher

统计学习 ©

```
#>    0.2117329 621.9337505
#> sample estimates:
#> odds ratio
#>   6.408309
```

```
# 单边 P 值
sum(dhyper(x = c(3, 4), m = 4, n = 4, k = 4))
```

```
#> [1] 0.2428571
```

21.4.3 对称性

用于计数数据的 McNemar 卡方检验（McNemar χ^2 检验）：检验二维列联表行和列的对称性 `mcnemar.test()`。怎么理解对称性？其实是配对检验。看帮助实例。

```
Performance <- matrix(c(794, 86, 150, 570),
  nrow = 2,
  dimnames = list(
    "1st Survey" = c("Approve", "Disapprove"),
    "2nd Survey" = c("Approve", "Disapprove")
  )
)
Performance
```

```
#>             2nd Survey
#> 1st Survey   Approve Disapprove
#>   Approve        794        150
#>   Disapprove      86        570
```

```
mcnemar.test(Performance)
```

```
#>
#>  McNemar's Chi-squared test with continuity correction
#>
#> data:  Performance
#> McNemar's chi-squared = 16.818, df = 1, p-value = 4.115e-05
```

21.4.4 条件独立性

用于分层分类数据的 Cochran-Mantel-Haenszel 卡方检验：两个枚举（分类）变量的条件独立性，假定不存在三个因素的交互作用。Cochran-Mantel-Haenszel 检验 `mantelhaen.test()`

```
str(UCBAdmissions)
```

```
#>  'table' num [1:2, 1:2, 1:6] 512 313 89 19 353 207 17 8 120 205 ...
#>  - attr(*, "dimnames")=List of 3
#>   ..$ Admit : chr [1:2] "Admitted" "Rejected"
#>   ..$ Gender: chr [1:2] "Male" "Female"
#>   ..$ Dept  : chr [1:6] "A" "B" "C" "D" ...
```

UCBAdmissions 数据集是一个 $2 \times 2 \times 6$ 的三维列联表，R 语言中常用 table 类型表示。实际上，table 类型衍生自 array 数组类型，当把 UCBAdmissions 当作一个数组操作时，1、2、3 分别表示 Admit、Gender、Dept 三个维度。

```
mantelhaen.test(UCBAdmissions)
```

```
#>
#>  Mantel-Haenszel chi-squared test with continuity correction
#>
#> data:  UCBAdmissions
#> Mantel-Haenszel X-squared = 1.4269, df = 1, p-value = 0.2323
#> alternative hypothesis: true common odds ratio is not equal to 1
#> 95 percent confidence interval:
#>  0.7719074 1.0603298
#> sample estimates:
#> common odds ratio
#>         0.9046968
```

没有证据表明院系与性别之间存在关联。在给定院系的情况下，是否录取和性别没有显著关系。

```
# 按系统计
apply(UCBAdmissions, 3, function(x) (x[1, 1] * x[2, 2]) / (x[1, 2] * x[2, 1]))
```

```
#>         A         B         C         D         E         F
#> 0.3492120 0.8025007 1.1330596 0.9212838 1.2216312 0.8278727
```

```
woolf <- function(x) {
  x <- x + 1 / 2
  k <- dim(x)[3]
  or <- apply(x, 3, function(x) (x[1, 1] * x[2, 2]) / (x[1, 2] * x[2, 1]))
  w <- apply(x, 3, function(x) 1 / sum(1 / x))
  1 - pchisq(sum(w * (log(or) - weighted.mean(log(or), w))^2), k - 1)
}
woolf(UCBAdmissions)
```

```
#> [1] 0.0034272
```

©黄湘云

21.5 加州伯克利分校的录取情况

1973 年加州伯克利分校 6 个最大的院系的录取情况见下表格 21.4，研究目标是加州伯克利分校在招生录取工作中是否有性别歧视?

表格 21.4: 加州伯克利分校的录取情况

	录取		拒绝	
院系	男性	女性	男性	女性
A	512	89	313	19
B	353	17	207	8
C	120	202	205	391
D	138	131	279	244
E	53	94	138	299
F	22	24	351	317

借助马赛克图图 21.5 可以更加直观的看出数据中的比例关系。

接下来进行定量的分析，首先，按性别和录取情况统计人数，如下:

```
m <- xtabs(Freq ~ Gender + Admit, data = as.data.frame(UCBAdmissions))
m
```

```
#>         Admit
#> Gender   Admitted Rejected
#>   Male       1198     1493
#>   Female      557     1278
```

可以看到，申请加州伯克利分校的女生当中，只有 $557/(557+1278) = 30.35\%$ 录取了，而男生则有 $1198/(1198+1493) = 44.52\%$ 的录取率。根据皮尔逊 χ^2 检验:

```
# 不带耶茨矫正
chisq.test(m, correct = FALSE)
```

```
#>
#>  Pearson's Chi-squared test
#>
#> data:  m
#> X-squared = 92.205, df = 1, p-value < 2.2e-16
```

可知 χ^2 统计量的值为 92.205 且 P 值远远小于 0.05，差异达到统计显著性，不是随机因素导致的。因此，加州伯克利分校被指控在招生录取工作中存在性别歧视。然而，当我们细分到各个院系去看录取率(录取人数 / 申请人数)，结果显示院系 A 的录取率为 64.41%，院系 B 的录取率为 63.24%，依次类推，各院系情况如下:

©黄湘云

图 21.5: 加州伯克利分校院系录取情况

©黄湘云 课稿

```
proportions(xtabs(Freq ~ Dept + Admit,
  data = as.data.frame(UCBAdmissions)
), margin = 1)
#>     Admit
#> Dept   Admitted   Rejected
#>    A 0.64415863 0.35584137
#>    B 0.63247863 0.36752137
#>    C 0.35076253 0.64923747
#>    D 0.33964646 0.66035354
#>    E 0.25171233 0.74828767
#>    F 0.06442577 0.93557423
```

图 21.6: 加州伯克利分校各院系录取情况

对每个院系，单独使用皮尔逊 χ^2 检验，发现只有 A 系的男、女生录取率的差异达到统计显著性，其它系的差异都不显著。辛普森悖论在这里出现了，在分类数据的分析中，常常遇到。

```
# 以 A 系为例
ma <- xtabs(Freq ~ Gender + Admit,
  subset = Dept == "A",
```

©黄湘云

```
  data = as.data.frame(UCBAdmissions)
)
chisq.test(ma, correct = FALSE)
```

```
#>
#>  Pearson's Chi-squared test
#>
#> data:  ma
#> X-squared = 17.248, df = 1, p-value = 3.28e-05
```

为了经一步说明此现象的原因，建立对数线性模型来拟合数据，值得一提的是皮尔逊卡方检验可以从对数线性模型的角度来看，而对数线性模型是一种特殊的广义线性模型，针对计数数据建模。

```
fit_ucb0 <- glm(Freq ~ Dept + Admit + Gender,
  family = poisson(link = "log"),
  data = as.data.frame(UCBAdmissions)
)
summary(fit_ucb0)
```

```
#>
#> Call:
#> glm(formula = Freq ~ Dept + Admit + Gender, family = poisson(link = "log"),
#>     data = as.data.frame(UCBAdmissions))
#>
#> Coefficients:
#>               Estimate Std. Error z value Pr(>|z|)
#> (Intercept)    5.37111    0.03964 135.498  < 2e-16 ***
#> DeptB         -0.46679    0.05274  -8.852  < 2e-16 ***
#> DeptC         -0.01621    0.04649  -0.349 0.727355
#> DeptD         -0.16384    0.04832  -3.391 0.000696 ***
#> DeptE         -0.46850    0.05276  -8.879  < 2e-16 ***
#> DeptF         -0.26752    0.04972  -5.380 7.44e-08 ***
#> AdmitRejected  0.45674    0.03051  14.972  < 2e-16 ***
#> GenderFemale  -0.38287    0.03027 -12.647  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for poisson family taken to be 1)
#>
#>     Null deviance: 2650.1  on 23  degrees of freedom
#> Residual deviance: 2097.7  on 16  degrees of freedom
#> AIC: 2272.7
```

©李漫 仅供试读

```
#>
#> Number of Fisher Scoring iterations: 5
```

添加性别和院系的交互效应后，对数线性模型的 AIC 下降一半多，说明模型的交互效应是显著的，也就是说性别和院系之间存在非常强的关联。

```
fit_ucb1 <- glm(Freq ~ Dept + Admit + Gender + Dept * Gender,
  family = poisson(link = "log"),
  data = as.data.frame(UCBAdmissions)
)
summary(fit_ucb1)
```

```
#>
#> Call:
#> glm(formula = Freq ~ Dept + Admit + Gender + Dept * Gender, family = poisson(link = "log"),
#>     data = as.data.frame(UCBAdmissions))
#>
#> Coefficients:
#>                    Estimate Std. Error z value Pr(>|z|)
#> (Intercept)         5.76801    0.03951 145.992  < 2e-16 ***
#> DeptB              -0.38745    0.05475  -7.076 1.48e-12 ***
#> DeptC              -0.93156    0.06549 -14.224  < 2e-16 ***
#> DeptD              -0.68230    0.06008 -11.356  < 2e-16 ***
#> DeptE              -1.46311    0.08030 -18.221  < 2e-16 ***
#> DeptF              -0.79380    0.06239 -12.722  < 2e-16 ***
#> AdmitRejected       0.45674    0.03051  14.972  < 2e-16 ***
#> GenderFemale       -2.03325    0.10233 -19.870  < 2e-16 ***
#> DeptB:GenderFemale -1.07581    0.22860  -4.706 2.52e-06 ***
#> DeptC:GenderFemale  2.63462    0.12343  21.345  < 2e-16 ***
#> DeptD:GenderFemale  1.92709    0.12464  15.461  < 2e-16 ***
#> DeptE:GenderFemale  2.75479    0.13510  20.391  < 2e-16 ***
#> DeptF:GenderFemale  1.94356    0.12683  15.325  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for poisson family taken to be 1)
#>
#>     Null deviance: 2650.10  on 23  degrees of freedom
#> Residual deviance:  877.06  on 11  degrees of freedom
#> AIC: 1062.1
#>
```

©黄湘云


```
#> Number of Fisher Scoring iterations: 5
```

此辛普森悖论现象的解释是女生倾向于申请录取率低的院系，而男生倾向于申请录取率高的院系，最终导致整体上，男生的录取率显著高于女生。至于为什么女生会倾向于申请录取率低的院系？这可能要看具体的院系是哪些，招生政策如何？这已经不是仅仅依靠招生办的统计数字就可以完全解释得了的，更多详情见文献 Bickel, Hammel, 和 O'Connell (1975) 。

> 提示
>
> 对数线性模型的皮尔逊 χ^2 检验的统计量
>
> ```
> sum(residuals(fit_ucb1, type = "pearson")^2)
> #> [1] 797.7045
> ```
>
> 比较多个广义线性模型的拟合效果，除了看 AIC，还可以看对数似然，它越大越好。可以看到添加性别和院系的交互效应后，对数似然增加了一倍多。
>
> ```
> # 基础模型
> logLik(fit_ucb0)
> #> 'log Lik.' -1128.365 (df=8)
> # 添加交互效应
> logLik(fit_ucb1)
> #> 'log Lik.' -518.0581 (df=13)
> ```

21.6 分析泰坦尼克号乘客生存率

分析存活率的影响因素。

除了从条件独立性检验的角度，下面从逻辑回归模型的角度分析这个高维列联表数据，由此，我们可以知道假设检验和广义线性模型之间的联系，针对复杂高维列联表数据进行关联分析和解释。

响应变量是乘客的状态，存活还是死亡，titanic_data 是按船舱 Class、性别 Sex 和年龄 Age 分类汇总统计的数据，因此，下面的逻辑回归模型是对乘客群体的建模。

```
# 建立模型
fit_titanic <- glm(cbind(Freq_Yes, Freq_No) ~ Class + Sex + Age,
  data = titanic_data, family = binomial(link = "logit")
)
```

接着，我们查看模型输出的情况

```
# 模型输出
summary(fit_titanic)
```

```
#>
#> Call:
#> glm(formula = cbind(Freq_Yes, Freq_No) ~ Class + Sex + Age, family = binomial(link = "logit"),
```

```
#>     data = titanic_data)
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)   0.6853     0.2730   2.510   0.0121 *
#> Class2nd     -1.0181     0.1960  -5.194 2.05e-07 ***
#> Class3rd     -1.7778     0.1716 -10.362  < 2e-16 ***
#> ClassCrew    -0.8577     0.1573  -5.451 5.00e-08 ***
#> SexFemale     2.4201     0.1404  17.236  < 2e-16 ***
#> AgeAdult     -1.0615     0.2440  -4.350 1.36e-05 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 671.96  on 13  degrees of freedom
#> Residual deviance: 112.57  on  8  degrees of freedom
#> AIC: 171.19
#>
#> Number of Fisher Scoring iterations: 5
```

©黄湘云

第二十二章　统计检验的功效

22.1　三大检验方法

统计检验的一般方法。

22.1.1　Wald 检验

22.1.2　Wilks 检验

也叫似然比检验

22.1.3　Rao 检验

也叫得分检验

22.2　t 检验的功效

检验的功效常用于样本量的计算

`power.t.test()` 计算单样本或两样本的 t 检验的功效，或者根据功效计算参数，如样本量

```
power.t.test(
  n = 100, delta = 2.2,
  sd = 1, sig.level = 0.05,
  type = "two.sample",
  alternative = "two.sided"
)
```

© 黄湘云

图 22.1: t 检验的功效

```
#>
#>      Two-sample t test power calculation
#>
#>               n = 100
#>           delta = 2.2
#>              sd = 1
#>       sig.level = 0.05
#>           power = 1
#>     alternative = two.sided
#>
#> NOTE: n is number in *each* group
```

表格 22.1: 函数 `power.t.test()` 的参数及其含义

参数	含义
`n`	每个组的样本量
`delta`	两个组的均值之差
`sd`	标准差，默认值 1
`sig.level`	显著性水平，默认是 0.05 （犯第 I 类错误的概率）
`power`	检验的功效（1 - 犯第 II 类错误的概率）
`type`	t 检验的类型 `"two.sample"` 两样本、`"one.sample"` 单样本或 `"paired"` 配对样本

©黄湘云

参数	含义
alternative	单边或双边检验，取值为 "two.sided" 或 "one.sided"

参数 n，delta，power，sd 和 sig.level 必须有一个值为 NULL，为 NULL 的参数是由其它参数决定的。

```
# 前面 t 检验的等价功效计算
library(pwr)
pwr.t.test(
  d = 2.2 / 6.4,
  n = 100,
  sig.level = 0.05,
  type = "two.sample",
  alternative = "two.sided"
)
```

```
#>
#>      Two-sample t test power calculation
#>
#>               n = 100
#>               d = 0.34375
#>       sig.level = 0.05
#>           power = 0.6768572
#>     alternative = two.sided
#>
#> NOTE: n is number in *each* group
```

sleep 数据集为例，计算功效

```
# 分组计算均值
aggregate(data = sleep, extra ~ group, FUN = mean)
```

```
#>   group extra
#> 1     1  0.75
#> 2     2  2.33
```

```
# 分组计算标准差
aggregate(data = sleep, extra ~ group, FUN = sd)
```

```
#>   group    extra
#> 1     1 1.789010
#> 2     2 2.002249
```

```
# 代入计算功效
power.t.test(
```

©曹滨 江湖

```
  delta = 2.33 - 0.75,            # 两组均值之差
  sd = (2.002249 + 1.789010) / 2, # 标准差
  sig.level = 0.05,          # 显著性水平
  type = "two.sample",       # 两样本
  power = 0.95,              # 功效水平
  alternative = "two.sided" # 双边检验
)
```

```
#>
#>      Two-sample t test power calculation
#>
#>               n = 38.39795
#>           delta = 1.58
#>              sd = 1.89563
#>       sig.level = 0.05
#>           power = 0.95
#>     alternative = two.sided
#>
#> NOTE: n is number in *each* group
```

经检验，上面取两组的平均方差代替共同方差和下面精确计算的结果差不多。各组至少需要 39 个样本。**MKpower** 包精确计算 Welch t 检验的功效

```
library(MKpower)
power.welch.t.test(
  delta = 2.33 - 0.75,
  sd1 = 2.002249,
  sd2 = 1.789010,
  sig.level = 0.05,
  power = 0.95,
  alternative = "two.sided"
)
```

我国著名统计学家许宝[F]先生对此功效计算方法做出过巨大贡献。

22.3 比例检验的功效

```
# power.prop.test()
```

`power.prop.test()` 计算两样本比例检验的功效

功效可以用来计算实验所需要的样本量，检验统计量的功效越大/高，检验方法越好，实验所需要的样本量越少

©李漫漫


```r
# p1 >= p2 的检验 单边和双边检验
power.prop.test(
  p1 = .65, p2 = 0.6, sig.level = .05,
  power = 0.90, alternative = "one.sided"
)
```

```
#>
#>      Two-sample comparison of proportions power calculation
#>
#>               n = 1603.846
#>              p1 = 0.65
#>              p2 = 0.6
#>       sig.level = 0.05
#>           power = 0.9
#>     alternative = one.sided
#>
#> NOTE: n is number in *each* group
```

```r
power.prop.test(
  p1 = .65, p2 = 0.6, sig.level = .05,
  power = 0.90, alternative = "two.sided"
)
```

```
#>
#>      Two-sample comparison of proportions power calculation
#>
#>               n = 1968.064
#>              p1 = 0.65
#>              p2 = 0.6
#>       sig.level = 0.05
#>           power = 0.9
#>     alternative = two.sided
#>
#> NOTE: n is number in *each* group
```

pwr 包 `pwr.2p.test()` 函数提供了类似 `power.prop.test()` 函数的功能

```r
library(pwr)
# 明确 p1 > p2 的检验
# 单边检验拆分更加明细，分为大于和小于
pwr.2p.test(
  h = ES.h(p1 = 0.65, p2 = 0.6),
  sig.level = 0.05, power = 0.9, alternative = "greater"
```

©黄湘云

```
)
#>
#>     Difference of proportion power calculation for binomial distribution (arcsine transformation)
#>
#>                h = 0.1033347
#>                n = 1604.007
#>        sig.level = 0.05
#>            power = 0.9
#>      alternative = greater
#>
#> NOTE: same sample sizes
```

已知两样本的样本量不等，检验 H_0: $p_1 = p_2$ H_1: $p_1 \neq p_2$ 的功效

```
pwr.2p2n.test(
  h = 0.30, n1 = 80, n2 = 245,
  sig.level = 0.05, alternative = "greater"
)
```

```
#>
#>     difference of proportion power calculation for binomial distribution (arcsine transformation)
#>
#>                h = 0.3
#>               n1 = 80
#>               n2 = 245
#>        sig.level = 0.05
#>            power = 0.7532924
#>      alternative = greater
#>
#> NOTE: different sample sizes
```

h 表示两个样本的差异，计算得到的功效是 0.75

22.4 方差分析的功效

`power.anova.test()` 计算平衡的单因素方差分析检验的功效

```
power.anova.test(
  groups = 4,        #  4 个组
  between.var = 1,   # 组间方差为 1
  within.var = 3,    # 组内方差为 3
```

仅供试读


```
  power = 0.95      # 1 - 犯第二类错误的概率
)

#>
#>      Balanced one-way analysis of variance power calculation
#>
#>          groups = 4
#>               n = 18.18245
#>     between.var = 1
#>      within.var = 3
#>       sig.level = 0.05
#>           power = 0.95
#>
#> NOTE: n is number in each group
```

```
library(pwr)
# f 是如何和上面的组间/组内方差等价指定的
pwr.anova.test(
  k = 4,            # 组数
  f = 0.5,          # 效应大小
  sig.level = 0.05, # 显著性水平
  power = 0.95      # 检验的效
)

#>
#>      Balanced one-way analysis of variance power calculation
#>
#>               k = 4
#>               n = 18.18244
#>               f = 0.5
#>       sig.level = 0.05
#>           power = 0.95
#>
#> NOTE: n is number in each group
```

© 黄湘云

第五部分

数据建模

©黄湘云

第二十三章　网络分析

网络数据分析又是另一个大话题，微信、微博、论坛、知乎、豆瓣、美团等等应用都或多或少自带了社交属性。人不能脱离社会存在，社交是人的一种本能，身处洪流之中，即是复杂社会网络中的一个节点。网络分析的内容有很多，比如社区探测、节点影响力分析等。网络图是表示节点之间关系的图，核心在于关系的刻画。用来表达网络关系的是稀疏矩阵，以及为处理这种矩阵而专门优化的矩阵计算库，如 **Matrix** 包、**rsparse** 包和 **RcppEigen** 包 (D. Bates 和 Eddelbuettel 2013) 等。图关系挖掘和计算的应用场景非常广泛，如社交推荐（社交 App）、风险控制（银行征信、企业查）、深度学习（图神经网络）、知识图谱（商户、商家、客户的实体关系网络）、区块链、物联网（IoT）、反洗钱（金融监管）、数据治理（数据血缘图谱）等。

本文将分析 R 语言社区开发者之间的协作关系网络。首先基于 CRAN（The Comprehensive R Archive Network）上发布的 R 包元数据信息，了解 R 语言社区 R 包及其维护者的规模，以及根据元数据中的信息发掘社区中的组织，最后，分析开发者在协作网络中的影响力，并将结果可视化。本文主要用到的工具有 igraph 包，操作图数据和图计算的 tidygraph 包，以及可视化图数据的 ggraph 包。

23.1　R 语言社区的规模

从 CRAN 上的 R 包及其开发者数量来看目前 R 语言社区规模。

```
# 设置就近的 CRAN 镜像站点
Sys.setenv(R_CRAN_WEB = "https://mirrors.tuna.tsinghua.edu.cn/CRAN")
# 获取 R 包元数据
pdb <- tools::CRAN_package_db()
```

截止 2022 年 12 月 31 日，CRAN 上发布的 R 包有 18976 个，CRAN 进入年末维护期 2022-12-22 至 2023-01-05。

```
pdb <- subset(
  x = pdb, subset = !duplicated(Package),
  select = c("Package", "Maintainer", "Title", "Authors@R", "Date", "Published")
)
```

距离上次更新的时间分布，有的包是一周内更新的，也有的是 10 多年未更新的。

©黄湘云

```
pdb$date_diff <- as.integer(as.Date("2022-12-31") - as.Date(pdb$Published))
```

根据发布日期 Published 构造新的一列 — 发布年份。

```
pdb$published_year <- as.integer(format(as.Date(pdb$Published), "%Y"))
```

然后按年统计更新的 R 包数量，如图 23.1 所示，以 2020 年为例，总数 18976 个 R 包当中有 2470 个 R 包的更新日期停留在 2020 年，占比 2470 / 18976 = 13.02%。过去 1 年内更新的 R 包有 8112 个（包含新出现的 R 包），占总数 8112 / 18976 = 42.75%，过去 2 年内更新的 R 包有 11553 个，占总数 11553 / 18976 = 60.88%，这个占比越高说明社区开发者越活跃。

```
library(ggplot2)
aggregate(data = pdb, Package ~ published_year, FUN = length) |>
  ggplot(aes(x = published_year, y = Package)) +
  geom_col(fill = NA, color = "gray20") +
  theme_classic() +
  coord_cartesian(expand = F) +
  labs(x = "年份", y = "R 包数量")
```

图 23.1: CRAN 上 R 包的更新情况

截止 2022-12-31，CRAN 上 R 包的维护者有 10049 人，其中有多少人在 2022 年更新了自己的 R 包呢？有 4820 个维护者，占比 47.96%，也就是说 2022 年，有 4820 个开发者更新了 8112 个 R 包，人均更新 1.68 个 R 包，下图 23.2 按 R 包发布年份统计开发者数量。

```
# 清理维护者字段，同一个开发者可能有多个邮箱
extract_maintainer <- function(x) {
  x <- gsub(pattern = "<.*?>", replacement = "", x = x)
```

©黄湘云


```
  trimws(x, which = "both", whitespace = "[ \t\r\n]")
}
# 极少量的维护者名字大小写不一样
pdb$Maintainer2 <- tolower(extract_maintainer(pdb$Maintainer))
# 维护者总数
length(unique(pdb$Maintainer2))
```

```
#> [1] 10049
```

图 23.2: CRAN 上的维护者活跃情况

23.2 R 语言社区的组织

有的组织基本停止了开发，如 Omegahat，有的被商业公司收购后，不再活跃了，如 Revolution Analytics。除了众所周知的 tidyverse (H. Wickham 等 2019) 和 tidymodels (Kuhn 和 Wickham 2020)，还有很多数据分析、建模的工具箱，如 mlr3verse (Lang 和 Schratz 2023)、easystats (Lüdecke 等 2022)、strengejacke (Lüdecke 2019) 和 DrWhy (Biecek 2023)。它们作为解决方案大都属于一些组织，还有深藏功与名，有待笔者挖掘的。因不存在明显的规律，下面从开发者的邮箱出发，隶属企业、组织往往有统一的邮箱后缀。

```
str_extract <- function(text, pattern, ...) regmatches(text, regexpr(pattern, text, ...))
# 移除 ORPHANED
pdb <- subset(pdb, subset = Maintainer != "ORPHANED")
# 抽取邮件后缀
extract_email_suffix <- function(x) {
```

```
  x <- str_extract(text = x, pattern = "<.*?>")
  sub(x = x, pattern = ".*?@(.*?)>", replacement = "\\1")
}
pdb$Email_suffix <- extract_email_suffix(pdb$Maintainer)
```

按组织统计扩展包的数量（总的 R 包数量约 2 万），即各个组织开发的 R 包。

```
pdb_pkg <- aggregate(
  data = pdb, Package ~ Email_suffix, FUN = function(x) { length(unique(x)) }
)
head(pdb_pkg[order(pdb_pkg$Package, decreasing = TRUE), ], 20)
```

```
#>        Email_suffix Package
#> 876       gmail.com    6968
#> 2044    rstudio.com     208
#> 979     hotmail.com     185
#> 1825    outlook.com     152
#> 1971  R-project.org     106
#> 2           163.com      94
#> 210    berkeley.edu      91
#> 2559      umich.edu      91
#> 2819         uw.edu      74
#> 1927 protonmail.com      73
#> 2564        umn.edu      69
#> 581      debian.org      68
#> 2951      yahoo.com      68
#> 1828     outlook.fr      63
#> 2212   stanford.edu      58
#> 155  auckland.ac.nz      57
#> 887          gmx.de      55
#> 2911       wisc.edu      55
#> 895  googlemail.com      50
#> 1970  r-project.org      50
```

不难看出，至少有如下几类：

1. 邮件服务提供商。6968 个 R 包使用 gmail 邮箱作为联系维护者的方式，googlemail.com 也是谷歌提供的服务。hotmail.com 和 outlook.com 都是微软提供的邮箱服务，outlook.fr（法国）也是，除此之外，比较大的邮件服务提供商就是 163.com（网易）、protonmail.com 和 yahoo.com （雅虎）等。
2. 商业组织。208 个 R 包来自 RStudio 公司的员工，这些维护者使用 RStudio 公司提供的邮箱。
3. 开源组织。R-project.org 和 r-project.org 都是 R 语言组织的联系方式，自不必多说，R 语言核心团队成员不仅维护 R 软件源码，还维护了很多 R 包。debian.org 是 Debian 组织的联系方式，都

©黄湘云

是开源组织（Open Source Org）。

4. 大学学院机构。berkeley.edu 、umich.edu 等以 edu 结尾的北美（国）的大学，gmx.de、posteo.de 等以 de 结尾的德国大学，ucl.ac.uk 等以 uk 结尾的英国的大学，auckland.ac.nz 等以 nz 结尾的新西兰的大学，uwaterloo.ca 等以 ca 结尾的加拿大的大学。

按组织统计开发者的数量（总的开发者数量约 1 万），即各个组织的 R 包开发者。

```
pdb_org <- aggregate(
  data = pdb, Maintainer2 ~ Email_suffix, FUN = function(x) { length(unique(x)) }
)
head(pdb_org[order(pdb_org$Maintainer2, decreasing = TRUE), ], 20)
```

```
#>        Email_suffix Maintainer2
#> 876       gmail.com        3795
#> 979     hotmail.com         109
#> 1825    outlook.com          87
#> 2           163.com          57
#> 2559      umich.edu          54
#> 2951      yahoo.com          51
#> 2564        umn.edu          47
#> 1927 protonmail.com          46
#> 2819         uw.edu          46
#> 887          gmx.de          34
#> 210    berkeley.edu          33
#> 2044    rstudio.com          30
#> 895  googlemail.com          28
#> 2212   stanford.edu          27
#> 468    columbia.edu          26
#> 1114       inrae.fr          26
#> 2451      ucl.ac.uk          25
#> 2964       yale.edu          25
#> 635        duke.edu          23
#> 1906      posteo.de          23
```

可见，大部分开发者采用邮件服务提供商的邮件地址。3795 个开发者使用来自谷歌的 gmail.com、196 个开发者使用来自微软的 hotmail.com 或 outlook.com，57 个开发者使用来自网易的 163.com，51 个开发者使用来自雅虎的 yahoo.com，46 个开发者使用来自 Proton 的 protonmail.com。

无论从开发者数量还是 R 包数量的角度看，都有两个显著特点。其一马太效应，往头部集中，其二，长尾分布，尾部占比接近甚至超过 50%。1666 个开发者来自以 edu 为后缀的邮箱。各个大学及其 R 包开发者数据如下：

```
sum(pdb_org[grepl(pattern = "edu$", x = pdb_org$Email_suffix), "Maintainer2"])
```

```
#> [1] 1666
pdb_org_edu <- pdb_org[grepl(pattern = "edu$", x = pdb_org$Email_suffix), ]
pdb_org_edu[order(pdb_org_edu$Maintainer2, decreasing = TRUE), ] |> head(20)
#>        Email_suffix Maintainer2
#> 2559      umich.edu          54
#> 2564        umn.edu          47
#> 2819         uw.edu          46
#> 210    berkeley.edu          33
#> 2212   stanford.edu          27
#> 468    columbia.edu          26
#> 2964       yale.edu          25
#> 635        duke.edu          23
#> 2911       wisc.edu          23
#> 482     cornell.edu          22
#> 2444    ucdavis.edu          21
#> 1929        psu.edu          19
#> 2449   uchicago.edu          19
#> 2830 vanderbilt.edu          19
#> 1660       ncsu.edu          18
#> 1663         nd.edu          18
#> 1008    iastate.edu          17
#> 1919  princeton.edu          17
#> 1815        osu.edu          16
#> 2523      uiowa.edu          16
```

好吧，几乎全是美国各个 NB 大学的，比如华盛顿大学（uw.edu）、密歇根大学（umich.edu）、加州伯克利大学（berkeley.edu）等等。顺便一说，欧美各个大学的网站，特别是统计院系很厉害的，已经帮大家收集得差不多了，有留学打算的读者自取，邮箱后缀就是学校/院官网。

有些邮箱后缀带有院系，但是并没有向上合并到学校这一级，比如 `stanford.edu` 、`stat.stanford.edu` 和 `alumni.stanford.edu` 等没有合并统计，所以学校排名仅供参考。有的邮箱来自教育机构，但是不以 `edu` 结尾，实际上，使用 `edu` 邮箱的教育机构大部份位于美国。美国以外的教育机构，比如新西兰奥克兰大学 `auckland.ac.nz` 、瑞士苏黎世联邦理工学院 `stat.math.ethz.ch` 等。如果读者还知道其他一般规律的，或者提供大学邮箱列表或者有其它更好的办法，就可以把这个排序数字做得更加精准一些。

下面根据邮箱后缀匹配抽取 CRAN 团队及开发的 R 包，规则也许不能覆盖所有的情况，读者若有补充，欢迎 PR 给我。举个例子，Brian Ripley 的邮箱 ripley@stats.ox.ac.uk 就不是一路，需要单独添加。

Kurt Hornik、Simon Urbanek、Achim Zeileis 、Martin Maechler、Torsten Hothorn 等真是高产呐！除了维护 R 语言核心代码，还开发维护了 **20** 多个 R 包。以 Brian Ripley 为例，看看他都具体维护了哪些 R 包。

© 高等教育出版社

表格 23.1: CRAN 团队开发维护 R 包数量情况

(a) 表

团队成员	R 包数量
Kurt Hornik	28
Simon Urbanek	26
Achim Zeileis	25
Martin Maechler	25
Torsten Hothorn	25
Paul Murrell	19
Toby Dylan Hocking	17
Brian Ripley	12
Thomas Lumley	12
Uwe Ligges	9
Duncan Murdoch	7
David Meyer	6
CRAN Team	5

(b) 续表

团队成员	R 包数量
Friedrich Leisch	5
Luke Tierney	5
Michael Lawrence	5
Stefan Theussl	5
Bettina Grün	3
John Chambers	3
Simon Wood	3
Bettina Gruen	2
Deepayan Sarkar	2
Douglas Bates	2
Martyn Plummer	2
Peter Dalgaard	1

表格 23.2: Brian Ripley 维护的 R 包

Package	Title
boot	Bootstrap Functions (Originally by Angelo Canty for S)
class	Functions for Classification
fastICA	FastICA Algorithms to Perform ICA and Projection Pursuit
gee	Generalized Estimation Equation Solver
KernSmooth	Functions for Kernel Smoothing Supporting Wand & Jones (1995)
MASS	Support Functions and Datasets for Venables and Ripley's MASS
mix	Estimation/Multiple Imputation for Mixed Categorical and Continuous Data
nnet	Feed-Forward Neural Networks and Multinomial Log-Linear Models
pspline	Penalized Smoothing Splines
RODBC	ODBC Database Access
spatial	Functions for Kriging and Point Pattern Analysis
tree	Classification and Regression Trees

震惊！有一半收录在 R 软件中，所以已经持续维护 **20** 多年了。下面继续根据邮箱后缀将 RStudio 团队的情况统计出来，结果见下表。

CRAN 和 RStudio 团队是 R 语言社区最为熟悉的，其它团队需借助一些网络分析算法挖掘了。

© 统计出版社

表格 23.3: RStudio 团队开发维护 R 包数量情况（部分）

(a) 表

团队成员	R 包数量
Hadley Wickham	48
Yihui Xie	22
Max Kuhn	18
Lionel Henry	15
Winston Chang	15
Daniel Falbel	13
Jennifer Bryan	13
Davis Vaughan	11
Carson Sievert	10
Tomasz Kalinowski	8
Barret Schloerke	6
Thomas Lin Pedersen	6
Hannah Frick	5
Christophe Dervieux	4
Joe Cheng	4
Julia Silge	4

(b) 续表

团队成员	R 包数量
Cole Arendt	3
Edgar Ruiz	3
JJ Allaire	3
Kevin Kuo	3
Kevin Ushey	3
Richard Iannone	3
Aron Atkins	2
Romain François	2
Yitao Li	2
Brian Smith	1
Emil Hvitfeldt	1
Garrick Aden-Buie	1
James Blair	1
Nathan Stephens	1
Nick Strayer	1

©黄湘云

23.3 R 语言社区的开发者

23.3.1 最高产的开发者

继续基于数据集 pdb ，将维护 R 包数量比较多的开发者统计出来。

图 23.3: 高产的 R 包开发者

发现，开发 1 个 R 包的开发者有 7656 人，开发 2 个 R 包的开发者有 1678 人，第二名是第一名的五分之一，递减规律非常符合指数分布。

```
table(pdb_ctb$Package)
```

```
#>
#>    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
```

```
#> 6716 1683  720  328  180   81   82   52   37   36   31   15   18    8   11    7
#>   17   18   19   20   21   22   23   24   25   26   27   28   31   32   33   52
#>    1    3    4    4    2    3    3    1    5    5    2    1    1    1    1    3
#>   58   63   69
#>    1    1    1
```

过滤掉非常高产的开发者，可以发现变化规律服从幂律分布。

```
ggplot(data = pdb_ctb, aes(x = Package)) +
  geom_histogram(binwidth = 1) +
  theme_classic() +
  labs(x = "R 包数量", y = "开发者")
ggplot(data = pdb_ctb[pdb_ctb$Package <= 20, ], aes(x = Package)) +
  geom_histogram(binwidth = 1, fill = NA, color = "gray20") +
  scale_y_log10() +
  theme_classic() +
  labs(x = "R 包数量", y = "开发者")
```

(a) 直方图 (b) 直方图（对数尺度）

图 23.4: 开发者数量的分布

最高产 Top 1% 的开发者 117 人（开发 R 包超过 10 个的开发者）贡献了 2060 / 18976 = 10.8% 扩展包，高产的是商业公司、开源组织、大学机构。

```
dim(pdb_ctb[pdb_ctb$Package > 10, ])
```

```
#> [1] 133   2
```

```
sum(pdb_ctb[pdb_ctb$Package > 10, "Package"])
```

```
#> [1] 2351
```

最低产 Bottom 的开发者 7656 人（仅开发一个 R 包的开发者）贡献了 7656 / 18976 扩展包 40.3 %，

©黄湘云

低产的人是主体。

23.3.2 开发者协作关系

如果一个开发者维护了一个 R 包，就成为维护者。一个 R 包有唯一的一个维护者，可能有一个至多个贡献者，这样，维护者和贡献者之间就形成了有向关系，贡献者可能又是另一个 R 包的维护者，也可能不是。不仅有向而且可能存在环。在一个 R 包中，A 是 B 的贡献者，而在另一个 R 包中，B 是 A 的贡献者，A 和 B 之间可能通过多个 R 包存在多次互相协作关系，这也表明 A 和 B 之间的关系密切。有向环的节点可能有 2 个以上，一个人可能同时属于多个环。

维护者 A 接受来自多个开发者的贡献，接受次数（所有贡献者人数的累和，A 的每个 R 包的贡献者人数相加）视为 A 的入度。维护者 A 作为开发者给多个维护者贡献，贡献次数（作为开发者给其它 R 包做贡献的次数，向外参与贡献的 R 包数目）视为 A 的出度。注意，A 作为维护者，必然包含 A 作为开发者，忽略 A 到 A 的贡献，只考虑贡献/协作关系。

```
# 过滤重复和缺失的记录
pdb <- subset(
  x = pdb, subset = !duplicated(Package) & !is.na(`Authors@R`),
  select = c("Package", "Maintainer", "Authors@R")
)
# 提取维护者的名字
pdb$Maintainer <- extract_maintainer(pdb$Maintainer)
```

有些包的元数据中没有 Authors@R 字段，有可能是没有贡献者，比如 mgcv 包、gam 包等，但也有可能是有贡献者，只是维护者没有填写这个字段，比如 Rcpp 包、RcppEigen 包等，因此将这些先过滤出来。总之，本文是以 `Authors@R` 字段作为贡献者的来源，共计 12503 个 R 包含有 `Authors@R` ，有 6000+ 个 R 包没有该字段，缺失约占 R 包总数的 1/3，在不那么考虑准确性的情况下，也可以使用。Author 字段是一段没有结构的文本，相比于 Author 字段，`Authors@R` 字段是以 R 语言中的 person 类型为存储结构的，比较规范，因此，提取贡献者的操作比较方便。作为示例，下面提取 Matrix 包的贡献者。

```
tmp <- eval(parse(text = pdb[pdb$Package == "Matrix", "Authors@R"]))
tmp <- unlist(lapply(tmp, function(x) format(x, include = c("given", "family"))))
# 返回一个整洁的数据框
tmp <- data.frame(Package = "Matrix", Maintainer = pdb[pdb$Package == "Matrix", "Maintainer"], Auth
# 去掉 Authors 是 Maintainer 的记录
subset(tmp, subset = Maintainer != Authors)
```

```
#>   Package      Maintainer           Authors
#> 1  Matrix Martin Maechler     Douglas Bates
#> 3  Matrix Martin Maechler      Mikael Jagan
#> 4  Matrix Martin Maechler  Timothy A. Davis
#> 5  Matrix Martin Maechler Jens Oehlschlägel
#> 6  Matrix Martin Maechler       Jason Riedy
```

©黄湘云

```
#> 7  Matrix Martin Maechler       R Core Team
```

数据框包含 R 包（Package 字段）、及其维护者（Maintainer 字段）和贡献者（Authors 字段）。将上述过程写成一个函数，接着，将所有 R 包的贡献者提取出来，形成一个大的数据框。

```
extract_authors <- function(pkg) {
  sub_pdb <- pdb[pdb$Package == pkg, ]
  tmp <- eval(parse(text = sub_pdb[, "Authors@R"]))
  tmp <- unlist(lapply(tmp, function(x) format(x, include = c("given", "family"))))
  tmp <- data.frame(Package = pkg, Maintainer = sub_pdb[, "Maintainer"], Authors = tmp)
  subset(tmp, subset = Maintainer != Authors)
}
extract_authors("Matrix")
```

```
#>   Package      Maintainer           Authors
#> 1  Matrix Martin Maechler     Douglas Bates
#> 3  Matrix Martin Maechler      Mikael Jagan
#> 4  Matrix Martin Maechler  Timothy A. Davis
#> 5  Matrix Martin Maechler Jens Oehlschlägel
#> 6  Matrix Martin Maechler       Jason Riedy
#> 7  Matrix Martin Maechler       R Core Team
```

```
# lapply(c("Matrix", "gt"), extract_authors)
# 抽取所有 R 包的贡献者，运行需要1-2分钟时间
pdb_authors_list <- lapply(pdb[, "Package"], extract_authors)
# 合并列表
pdb_authors_dt <- data.table::rbindlist(pdb_authors_list)
```

最后整理出来的大数据框 **pdb_authors_dt** 含有近 26000 条记录，即边的规模大小。考虑到有些维护者和贡献者之间可能存在多次合作的情况，下面统计一下合作次数。

```
pdb_authors_dt[ ,.(cnt = length(Package)) , by = c("Maintainer", "Authors")][cnt >= 10, ][order(cnt, dec
```

```
#>                 Maintainer               Authors cnt
#>  1:         Hadley Wickham               RStudio  36
#>  2:          Pablo Sanchez            Windsor.ai  25
#>  3:           Jan Wijffels                BNOSAC  24
#>  4:           Gábor Csárdi               RStudio  19
#>  5:               Hong Ooi             Microsoft  16
#>  6:               Max Kuhn               RStudio  14
#>  7:           Lionel Henry               RStudio  14
#>  8:      Robrecht Cannoodt        Wouter Saelens  13
#>  9:      Scott Chamberlain              rOpenSci  13
#> 10:            Joe Thorley    Poisson Consulting  13
```

© 高等教育出版社


```
#> 11:      Frederic Bertrand Myriam Maumy-Bertrand  12
#> 12:          Winston Chang               RStudio  12
#> 13:          Daniel Falbel               RStudio  12
#> 14:           David Kretch           Adam Banker  12
#> 15:           David Kretch      Amazon.com, Inc.  12
#> 16:         Victor Perrier           Fanny Meyer  11
#> 17:         Jennifer Bryan               RStudio  11
#> 18: William Michael Landau Eli Lilly and Company  11
#> 19:        Adrian Baddeley             Ege Rubak  11
#> 20:           Gábor Csárdi            Jim Hester  10
#> 21:          Kirill Müller               RStudio  10
#> 22:         Carson Sievert               RStudio  10
#> 23:    Thomas Lin Pedersen               RStudio  10
#> 24:           Lionel Henry        Hadley Wickham  10
#> 25:        Adrian Baddeley           Rolf Turner  10
#>                 Maintainer               Authors cnt
```

Authors 字段出现了不少组织的名字，这是因为有许多 R 包的维护者受雇于该组织，版权归属于该组织，组织不仅提供持续的资金，而且还提供其它帮助。以 **dplyr** 包为例，Hadley Wickham 受雇于 RStudio 公司，在 dplyr 包的元数据中，字段 `Authors@R` 中 RStudio 的角色是 cph 和 fnd ，即版权所有和资金支持。角色 cre 就是维护者，负责与 CRAN 团队的沟通。角色 aut 就是对 R 包有实质贡献的人。

```
format(eval(parse(text = pdb[pdb$Package == "dplyr", "Authors@R"])),
       include = c("given", "family", "role"))
```

```
#> [1] "Hadley Wickham [aut, cre]" "Romain François [aut]"
#> [3] "Lionel Henry [aut]"        "Kirill Müller [aut]"
#> [5] "RStudio [cph, fnd]"
```

此外，同属于一个组织的维护者之间常常合作紧密，从上面的结果可以看到，Gábor Csárdi 和 Jim Hester ，Lionel Henry 和 Hadley Wickham，Carson Sievert 和 Joe Cheng ，Jennifer Bryan 和 Hadley Wickham 等同属于 RStudio 公司，常常协作开发项目。对 RStudio、CRAN Team 和 rOpenSci 不再赘述，下面对排名靠前的其它组织略作说明。

1. Windsor.ai 提供一系列可以连接各大营销平台，获取营销效果数据 R 包。
2. BNOSAC 提供一系列计算机视觉、图像识别、自然语言处理方面的 R 包，比如 word2vec 包、doc2vec 包等。
3. Microsoft 提供一系列连接和操作 Azure 云套件的 R 包，比如 AzureR 包。
4. Wouter Saelens 提供一系列单细胞轨迹推理（**single-cell trajectory inference**）相关的 R 包，形成一个 **dynverse 家族**。
5. Poisson Consulting 提供一系列用于计算生物学和统计生态学的 R 包和相关研究论文。
6. Amazon.com, Inc. 提供一系列用于存储、管理、操作等 Amazon 云服务的 R 包，形成一个 paws 套件。

©李东风

7. Eli Lilly and Company 可能是 rOpenSci 的一员，赞助了旗下的 targets 和 jagstargets 等 R 包。

最后，统计协作次数的分布，网络中边的权重的分布。

```
pdb_authors_net <- pdb_authors_dt[, .(cnt = .N), by = c("Maintainer", "Authors")]
```

可以发现，绝大多数人之间协作只有一次。

```
table(pdb_authors_net$cnt)
```

```
#>
#>     1     2     3     4     5     6     7     8     9    10    11    12    13
#> 20432  1511   365   121    44    28    14     8     3     6     4     5     3
#>    14    16    19    24    25    36
#>     2     1     1     1     1     1
```

23.3.3 节点出入度分布

下面简化这个网络，仅考虑贡献者也是维护者的情况，就是说网络中所有节点既是维护者也是贡献者，这会过滤掉组织机构、大量没有在 CRAN 发过 R 包的贡献者、从没给其它维护者做贡献的维护者。简化后，网络节点的出度、入度的分布图如下。

```
# Maintainer 的入度
pdb_authors_net_indegree <- pdb_authors_dt[Authors %in% Maintainer, ][, .(in_degree = length(Authors)),
# Authors 的出度
pdb_authors_net_outdegree <- pdb_authors_dt[Authors %in% Maintainer, ][, .(out_degree = length(Maintaine

ggplot(pdb_authors_net_indegree, aes(x = in_degree)) +
  geom_histogram(binwidth = 1) +
  geom_freqpoly(binwidth = 1) +
  theme_classic()
ggplot(pdb_authors_net_outdegree, aes(x = out_degree)) +
  geom_histogram(binwidth = 1) +
  geom_freqpoly(binwidth = 1) +
  theme_classic()
```

23.3.4 可视化协作网络

节点的大小以维护者维护的 R 包数量来表示，边的大小以维护者之间协作次数来表示。为了美观起见，更为了突出重点，仅保留协作次数大于 1 的边。

```
# 边
pdb_authors_net_edge <- pdb_authors_dt[Authors %in% Maintainer, ][, .(edge_cnt = .N), by = c("Authors",
pdb_authors_net_edge[order(edge_cnt, decreasing = TRUE),]
```

©黄湘云

(a) 入度的分布 (b) 出度的分布

图 23.5: 节点的入度和出度的分布

```
#>                      Authors            Maintainer edge_cnt
#>   1:              Jim Hester          Gábor Csárdi       10
#>   2:          Hadley Wickham          Lionel Henry       10
#>   3:               Joe Cheng        Carson Sievert        9
#>   4:          Hadley Wickham        Jennifer Bryan        8
#>   5: Steven Andrew Culpepper James Joseph Balamuta        8
#>  ---
#> 526:             Aaron Wolen     Scott Chamberlain        2
#> 527:               Bob Rudis         Simon Garnier        2
#> 528:           Marco Sciaini         Simon Garnier        2
#> 529:          Carlos Morales           Martin Chan        2
#> 530:               Md Yeasin     Ranjit Kumar Paul        2
```

```
# 顶点
pdb_authors_net_vertex <- pdb_authors_dt[, .(vertex_cnt = length(unique(Package))), by = "Maintaine
pdb_authors_net_vertex[order(vertex_cnt, decreasing = TRUE),]
```

```
#>                Maintainer vertex_cnt
#>   1:       Hadley Wickham         43
#>   2:         Gábor Csárdi         33
#>   3:          Jeroen Ooms         28
#>   4:    Scott Chamberlain         28
#>   5:            Yihui Xie         21
#>  ---
#> 579:    Katriona Goldmann          1
#> 580:        Carlo Pacioni          1
```

```
#> 581:       Michael Scholz          1
#> 582: Javier Roca-Pardinas          1
#> 583:         Xianying Tan          1
```

这是一个有向图，其各个字段含义如下。

- Maintainer 维护者（代表流 to）
- Authors 贡献者（代表源 from）
- `edge_cnt` 边的大小表示维护者 Maintainer 和贡献者 Authors 的协作次数
- `vertex_cnt` 顶点大小表示维护者 Maintainer 维护的 R 包数量

下面先考虑用 igraph 包可视化这个复杂的有向带权网络。`pdb_authors_net_edge` 和 `pdb_authors_net_vertex` 都是数据框，首先调用 igraph 包的函数 `graph_from_data_frame()` 将其转化为网络类型 `igraph` ，然后使用函数 `plot()` 绘制网络图。

图 23.6: 开发者的协作关系网络

协作关系弱的开发者占大部分，构成一个「月亮」的造型，其中，不乏维护多个 R 包的开发者，这些人

©黄湘云

要么单干，要么在专业小领域、小组织内协作。与之相对应的是协作关系较强的开发者，人数虽少，影响力却大，构成一个「太阳」的造型。协作得多往往意味着维护的 R 包也不少，甚至同属于一个组织，因此，高产的开发者、影响力大的组织聚集在一起，如 R Core Team、RStudio、rOpenSci 等。

```
eb <- cluster_edge_betweenness(pdb_authors_graph)
eb
```

```
#> IGRAPH clustering edge betweenness, groups: 181, mod: 0.88
#> + groups:
#>   $`1`
#>   [1] "Matt Nunes"           "Daniel Grose"         "Guy Nason"
#>   [4] "Rebecca Killick"      "Idris Eckley"         "Alessandro Cardinali"
#>
#>   $`2`
#>   [1] "Jin Zhu"    "Shiyun Lin"
#>
#>   $`3`
#>    [1] "Julio Trecenti"       "Henrik Bengtsson"     "Morgane Pierre-Jean"
#>    [4] "Zhian N. Kamvar"      "Pierre Neuvial"       "Michal Bojanowski"
#>   + ... omitted several groups/vertices
```

igraph 包提供多种社区探测的算法，上面简单使用函数 `cluster_edge_betweenness()` 来探测，结果显示有 181 个社区。社区 1 包含的成员如下：

```
eb$names[eb$membership == 1]
```

```
#> [1] "Matt Nunes"           "Daniel Grose"         "Guy Nason"
#> [4] "Rebecca Killick"      "Idris Eckley"         "Alessandro Cardinali"
```

社区 3、14、21、34、46、52、75 的成员是比较多的。其中，社区 3 是以 RStudio 为核心的大社区，社区 14 是以 CRAN 为核心的大社区。

```
# RStudio 为核心的大社区
eb$names[eb$membership == 3]
```

```
#>  [1] "Julio Trecenti"       "Henrik Bengtsson"     "Morgane Pierre-Jean"
#>  [4] "Zhian N. Kamvar"      "Pierre Neuvial"       "Michal Bojanowski"
#>  [7] "Ian Lyttle"           "Thomas Lin Pedersen"  "Yihui Xie"
#> [10] "Dirk Schumacher"      "Jeroen Ooms"          "Gábor Csárdi"
#> [13] "Sean Kross"           "Carl Boettiger"       "Neal Richardson"
#> [16] "Ryan Hafen"           "Matthew Fidler"       "Hadley Wickham"
#> [19] "Mark Edmondson"       "Kirill Müller"        "Richard Iannone"
#> [22] "Carson Sievert"       "Winston Chang"        "Lionel Henry"
#> [25] "Jennifer Bryan"       "Michael Sumner"       "Scott Chamberlain"
#> [28] "Garrick Aden-Buie"    "Daniel Falbel"        "Matthew B. Jones"
```

©黄湘云

```
#> [31] "Hiroaki Yutani"       "Taiyun Wei"           "Jim Hester"
#> [34] "Romain François"      "Greg Freedman Ellis"  "Rhian Davies"
#> [37] "Bryce Mecum"          "Steph Locke"          "Christophe Dervieux"
#> [40] "Jonathan Keane"       "Thibaut Jombart"      "Dewey Dunnington"
#> [43] "Anne Cori"            "Bill Denney"          "Jared Huling"
#> [46] "Wush Wu"              "Atsushi Yasumoto"     "Barret Schloerke"
#> [49] "Yuan Tang"            "Duncan Garmonsway"    "Edzer Pebesma"
#> [52] "Sebastian Meyer"      "Derek Burk"           "Tim Taylor"
#> [55] "Alicia Schep"         "Tomasz Kalinowski"    "Michael Rustler"
#> [58] "Joe Cheng"            "Bhaskar Karambelkar"  "Sebastian Kreutzer"
#> [61] "JJ Allaire"           "JooYoung Seo"         "Zachary Foster"
#> [64] "Malcolm Barrett"      "Aaron Wolen"          "Bruno Tremblay"
#> [67] "Justin Wilkins"       "Yixuan Qiu"           "Johannes Friedrich"
#> [70] "Kevin Ushey"          "Steven M. Mortimer"   "Karthik Ram"
#> [73] "Jorrit Poelen"        "Maëlle Salmon"        "Aron Atkins"
#> [76] "Ramnath Vaidyanathan" "Thomas Leeper"        "Dirk Eddelbuettel"
#> [79] "Xianying Tan"
```

```
# CRAN 为核心的大社区
eb$names[eb$membership == 14]
```

```
#>  [1] "Achim Zeileis"        "Michael Hahsler"      "Michel Lang"
#>  [4] "Nikolaus Umlauf"      "Vincent Dorie"        "Bettina Gruen"
#>  [7] "Bernd Bischl"         "Ben Bolker"           "Marc Becker"
#> [10] "Friedrich Leisch"     "Brian Ripley"         "Michael Friendly"
#> [13] "John Fox"             "Kurt Hornik"          "Patrick Schratz"
#> [16] "Volodymyr Melnykov"   "Martin Maechler"      "George Ostrouchov"
#> [19] "Drew Schmidt"         "Georgi N. Boshnakov"  "Wei-Chen Chen"
#> [22] "Stefan Theussl"       "David Meyer"          "Jakob Bossek"
#> [25] "Francois Michonneau"  "Marius Hofert"        "Florian Schwendinger"
#> [28] "Felix Zimmer"         "Martin Binder"        "Phil Chalmers"
#> [31] "Lukas Sablica"        "Sebastian Fischer"    "Lennart Schneider"
#> [34] "Jakob Richter"        "Florian Wickelmaier"  "Rudolf Debelak"
#> [37] "Duncan Murdoch"       "Alexander Brenning"   "Ingo Feinerer"
```

同时，在 RStudio 这个大社区下，有一些与之紧密相关的小社区，比如 Rob Hyndman 等人的时间序列社区、Roger Bivand 等人的空间统计社区。

```
# 时间序列 Rob Hyndman
eb$names[eb$membership == 52]
```

```
#>  [1] "Asael Alonzo Matamoros"   "Nicholas Tierney"
#>  [3] "Sevvandi Kandanaarachchi" "Rob Hyndman"
```

```
#>  [5] "Di Cook"                "Mitchell O'Hara-Wild"
#>  [7] "Han Lin Shang"          "Sayani Gupta"
#>  [9] "Earo Wang"              "Christoph Bergmeir"
```

```
# 空间统计 Roger Bivand
eb$names[eb$membership == 75]
```

```
#> [1] "Sebastian Jeworutzki" "Roger Bivand"         "Colin Rundel"
#> [4] "Angela Li"            "Gianfranco Piras"     "Patrick Giraudoux"
#> [7] "Giovanni Millo"
```

结合前面的图 23.6 ，知道有很多小圈圈，这些放一边，重点关注那些大的圈圈，见下图。

Hadley Wickham
Scott Chamberlain
Joe Cheng
Karl Ottlie
Earl Brettiger
Roger Bivand
Bob Gentleman
Adrian Baddeley
Gert Janssenswillen
Bob Gentleman
Pedro Aphalo
Tyler Rinker
Evan Boehmke
Timothy Girard
Matt Dunnes
Guy Mason
Klaus Nothausen
Balasubramanian Narasimhan
Nicolas Greebeau
Maurice Vargas
Henrik Bengtsson
Tomasz Kalinowski
Matthew Fidler
Michael Sumner
Mark Bonham
Robin Lovelace
Rob Hyndman
Christoph Bergmeir
Mitchell O'Hara-Wild
Mark Hornsby
Sebastian Kurtzer
Andy Teucher
Kevin Ushey
Dean Attali
Daniel Sparks
Dominik Makowski
Brenton Wiernik
Jason Vaver Heaton
Matthew Tenney
Andrew Altomare
Jakub Nowosad
Tomas Kalinowski
Mikkel Meyer Andersen
Christopher Paciorek
Silverio Vega-Soriano
Danielle Ingram
Ann Mamarson
Kenneth Benoit
Paul Murrell
Yanqing Zeng
Thomas Buckheim
Tom Elliott
Frank Schumacher
Steffen Mazzig
Henrik Bengtsson
Colin Fay
Matthias Stigler
Stephen Rochette
Daniel Falbel
Tomasz Kalinowski
Benjamin Gorman
Michael Schlep
Stefan Geiger
Florian Schwendinger
Martin Maechler
Nicolas Langbein
Benoit Pleurmel
Veronique Bachelier
Victor Perrier
Drew Schmidt
Wei-Chen Chen
Georgi N. Boshnakov
Michael Friendly
Brian Ripley
Marc Becker
Alex Lang
Peter Ruckdeschel
Matthias Kohl
Karline Soetaert
Mark van der Loo
Nan Xiao
Daniel Doerfler
Dean Attali
Jahuto Bartal
Diego Hernando
Timothy Girard
Gregory Jefferis
Manuel Spitzer
Jackson Hoffart
Jason Sam Sellers
Georgi Feinstein
Thiyanga S. Talagala
Mark Chan
Christopher Paciorek
Silverio Vega-Soriano
Danielle Ingram
Ann Mamarson
Kenneth Benoit
James Joseph Balamuta
Gert Janssenswillen
Barbara Lerner
Emily Goose
Edgar Ruiz
Colin Fay
Matthias Stigler
Stephen Rochette
Daniel Falbel
Maurice Vargas
Bernhard Pfaff
Paul Murrell
Yanqing Zeng
Thomas Buckheim
Tom Elliott
Frank Schumacher
Steffen Mazzig
Henrik Bengtsson
Nicolas Greebeau
Thomas Nagler
Paul Murrell
Ben Goodrich
Jonathan Mattingly
Przemyslaw Biecek
Matt Dunnes
Dave Grosse
David Dooley
Mark Clark
Michael Boehmke
Brian Ripley
Martin Maechler
Nicolas Langbein
Benoit Pleurmel
Veronique Bachelier
Victor Perrier
Daniel Falbel
Tomasz Kalinowski
Matthew Fidler
Michael Sumner
Mark Bonham
Robin Lovelace
Rob Hyndman
Christoph Bergmeir
Mitchell O'Hara-Wild
Mark Hornsby
Sebastian Kurtzer
Andy Teucher
Kevin Ushey
Dean Attali
Daniel Sparks
Dominik Makowski
Brenton Wiernik
Jason Vaver Heaton
Matthew Tenney
Andrew Altomare
Jakub Nowosad
Tomas Kalinowski
Mikkel Meyer Andersen
Christopher Paciorek
Silverio Vega-Soriano
Danielle Ingram
Ann Mamarson
Kenneth Benoit
Paul Murrell
Yanqing Zeng
Thomas Buckheim
Tom Elliott
Frank Schumacher
Steffen Mazzig
Henrik Bengtsson

的影响力（或者重要性）。最后，借助 **ggraph** 包绘制维护者之间的贡献网络，节点的大小代表维护者影响力的强弱。

图 23.8: 开发者的影响力网络

前面两个网络图基于同一份数据、同样的网络布局算法，得到非常类似的结果。静态图上的标签相互重叠，影响细节的观察和探索，比如连接 CRAN 和 RStudio 两大阵营的通道。下面使用 **visNetwork** 包绘制交互式网络图，这样可以在图上使用鼠标放大、拖拽。可以发现在 CRAN 社区的 Achim Zeileis 和 RStudio 社区的 Max Kuhn 之间是由 Andri Signorell 牵线搭桥。

23.4 扩展阅读

R 语言网络分析方面的著作有 Erick Kolaczyk 的书籍《Statistical Analysis of Network Data with R》(Kolaczyk 和 Csárdi 2020)，网络可视化方面，推荐 Hadley Wickham 的著作《ggplot2: Elegant Graphics for Data Analysis》(H. Wickham, Navarro, 和 Pedersen 2024) 的第七章，Sam Tyner 等人的文章《Network Visualization with ggplot2》(Tyner, Briatte, 和 Hofmann 2017) 也值得一看。此外，读者若有兴趣，可以用 visNetwork 包制作交互式网络图形，它是 JS 库 vis-network 的 R 语言接口。而 Richard Iannone 开发的 DiagrammeR 包可以制作静态的矢量网页图形。

© 黄湘云

图 23.9: 开发者的影响力网络（visNetwork）

© 清华出版社

可视化网络的 R 包有 **igraph** 、**ggraph** 等，专业的软件有 Gephi 等。igraph 是非常流行和核心的网络分析 C 语言库，它提供有多种语言的接口，其 R 语言接口 igraph 包也被很多其它做网络分析的 R 包所引用。开源的 Gephi 软件适合处理中等规模的网络分析和可视化。大规模图计算可以用 Apache Spark 的 GraphX。R 语言这层，主要还是对应数据分析和数据产品，用于在内部咨询和商业分析。

企业级的图存储和计算框架，比较有名（可能是最有名的）的是 Neo4j ，它有以 AGPL 协议发布的开源版本，还有商业版本。Nebula Graph 分布式开源图数据库，高扩展性和高可用性，支持千亿节点、万亿条边、毫秒级查询，有中文文档，有企业应用案例，美团图数据库平台建设及业务实践。阿里研发的 GraphScope 提供一站式大规模图计算系统，支持图神经网络计算。HugeGraph 图数据库应用于金融反欺诈实践。

23.5 习题

1. 类似开发者协作关系的分析，可以统计 R 包被多少 R 包依赖，依赖数量的分布。统计 R 包被依赖的深度（若 R 包 A 被 R 包 B 依赖，R 包 B 被 R 包 C 依赖，以此类推）。进而，构建、分析、可视化依赖关系网络，分析 R 包的影响力。
2. 本文基于 2022 年 12 月 31 日的 R 包元数据进行分析，请与 2023 年 12 月 31 日的数据比较。

© 黄湘云

第二十四章　文本分析

R 语言任务视图中以自然语言处理（Natural Language Processing）涵盖文本分析（Text Analysis）的内容。R 语言社区中有两本文本分析相关的著作，分别是《Text Mining with R》(Silge 和 Robinson 2017) 和《Supervised Machine Learning for Text Analysis in R》(Hvitfeldt 和 Silge 2021)。

本文有两个目的：其一分析谢益辉 10 多年日志，挖掘写作主题及其变化；其二挖掘湘云的日志主题，计算与益辉日志的风格相似度。事实上，益辉在个人主页中是明确说了自己的兴趣范围的，本文借用文本挖掘中的主题建模方法，不过是一点实证，熟悉文本建模的操作过程。

从谢益辉公开的日志中，探索成功人士的经历，从中汲取一些经验、教训。最近才知道他有 300 多万字的日志，数字惊讶到我了，遂决定抽取最近 10 年的日志数据进行分析。中英文分开，首先处理、分析中文日志。文本操作和分析的内容有分词、词频、词云、主题建模、相似度度量等。对作者来说，感兴趣的主题与写作的内容有直接的关系。益辉的日志都是没有分类标签的，主题挖掘可以洞察作者的兴趣。

```
library(jiebaRD)  # 词库
library(jiebaR)   # 分词
library(ggplot2)  # 绘图
library(ggrepel)
library(ggwordcloud) # 词云
library(text2vec)    # LDA 算法
```

24.1　基本概念

24.1.1　中文分词

本文采用 jiebaR 作为中文分词工具，相比于 Rwordseg 包，不仅效果好，而且没有 rJava 包 (JAVA) 环境依赖。

```
library(jiebaRD)
library(jiebaR)
jieba_seg <- worker()
segment("齐白石对花鸟虫鱼都有研究呢", jieba_seg)
```

```
#> [1] "齐白石"   "对"       "花鸟虫鱼" "都"       "有"       "研究"     "呢"
```

©黄源 文稿

24.1.2 英文分词

英文环境下，词与词之间本身就是以空格分开的。

```
segment("Hello world!", jieba_seg)
```

```
#> [1] "Hello" "world"
```

24.1.3 停止词

数字、标点符号、空格不参与分词，会被抹掉。

```
segment("国画大师齐白石对花鸟虫鱼都有研究呢！", jieba_seg)
```

```
#> [1] "国画"     "大师"     "齐白石"   "对"       "花鸟虫鱼" "都"       "有"
#> [8] "研究"     "呢"
```

还可以添加停止词，将「花鸟虫鱼」作为停止词添加到停止词库中。

```
jieba_seg <- worker(stop_word = "data/text/stop_word.txt")
segment("国画大师齐白石对花鸟虫鱼都有研究呢", jieba_seg)
```

```
#> [1] "国画"   "大师"   "齐白石" "对"     "都"     "有"     "研究"   "呢"
```

24.1.4 分词与词性

中文环境没有英文的词干化（lemmatization）过程（比如 applied / applies -> apply），只考虑分词、词性。spacyr 包通过 reticulate 包调用 Python 社区的自然语言处理模块 spacy ，支持多种语言的分词、词性识别等。配置好 Python 环境，下载中文环境下的语言模型。

```
library(spacyr)
# 下载语言模型
# spacy_download_langmodel("zh_core_web_sm")
# 初始化语言模型
spacy_initialize(model = "zh_core_web_sm")
```

```
#> successfully initialized (spaCy Version: 3.7.2, language model: zh_core_web_sm)
```

```
# Token 化是分词
spacy_tokenize("国画大师齐白石对花鸟虫鱼都有研究呢")
```

```
#> $text1
#>  [1] "国画"   "大师"   "齐白石" "对"     "花鸟"   "虫鱼"   "都"     "有"
#>  [9] "研究"   "呢"
```

```
# 示例
txt <- c(d1 = "国画大师齐白石对花鸟虫鱼都有研究呢",
```

©黄湘云

```
        d2 = "中国人民银行很行",
        d3 = "张大千在研究国画")
# 解析
parsed_txt <- spacy_parse(txt)
```

```
#> Warning in spacy_parse.character(txt): lemmatization may not work properly in
#> model 'zh_core_web_sm'
```

```
# 结果
parsed_txt
```

```
#>    doc_id sentence_id token_id  token lemma   pos   entity
#> 1      d1           1        1   国画        NOUN
#> 2      d1           1        2   大师        NOUN
#> 3      d1           1        3 齐白石       PROPN PERSON_B
#> 4      d1           1        4     对         ADP
#> 5      d1           1        5   花鸟        NOUN
#> 6      d1           1        6   虫鱼        NOUN
#> 7      d1           1        7     都         ADV
#> 8      d1           1        8     有        VERB
#> 9      d1           1        9   研究        NOUN
#> 10     d1           1       10     呢        PART
#> 11     d2           1        1   中国       PROPN    ORG_B
#> 12     d2           1        2   人民        NOUN    ORG_I
#> 13     d2           1        3   银行        NOUN    ORG_I
#> 14     d2           1        4     很         ADV
#> 15     d2           1        5     行        VERB
#> 16     d3           1        1 张大千       PROPN PERSON_B
#> 17     d3           1        2     在         ADP
#> 18     d3           1        3   研究        VERB
#> 19     d3           1        4   国画        NOUN
```

人名都是名词，如齐白石、张大千等，「研究」既可做动词，也可做名词，「都」和「很」是副词。「中国人民银行很行」的「行」应当是形容词，但被识别为动词。

24.1.5 词频统计

输入字符串向量，计算每个字符串出现的次数。jiebaR 包的函数 `freq()` 可以统计词出现的频次。

```
freq(c("a", "a", "c"))
```

```
#>   char freq
#> 1    c    1
```

©黄湘云

```
#> 2    a    2
```

24.1.6 TF-IDF

输入一个列表，列表中每个元素是字符串向量，每个字符串向量代表一个文档。jiebaR 包的函数 `get_idf()` 可以计算 TF-IDF（Document Frequency - Inverse Document Frequency）值。

```
get_idf(list(c("abc", "def"), c("abc", " ")))
```

```
#>   name     count
#> 1  def 0.6931472
#> 2  abc 0.0000000
```

24.2 益辉的日志

下载益辉的日志数据

```
git clone git@github.com:yihui/yihui.org.git
```

经过整理后，打包成 Rdata 数据供 R 软件使用。

```
# 加载益辉的日志数据
load(file = "data/text/yihui.Rdata")
```

24.2.1 整体概况

2006 年获得中国人民大学学士学位，2009 和 2013 年分别获得中国人民大学硕士和爱荷华州立大学博士学位，在校期间，日志数量持续增加，又陆续创立统计之都，举办中国 R 语言大会。在毕业那年需要完成毕业论文，因此，日志数量明显减少。2013 -2016 年，每年都有书籍出版，期间，有博士毕业、找工作、安家等重要事情，因此，日志数量持续处于低位。稳定后，2017-2018 年除了正常出两本书以外，写了大量的日志，迎来第二个高峰，2018 年，中英文日志数量超过 300 篇。2019-2020 年集中精力在写一本食谱。2021 年第一本中文书《现代统计图形》在 10 年后出版，这主要是 2007-2011 年的工作。2021-2023 年日志数量（2023 年中文日志未发布）处于较低水平。

24.2.2 数据清洗

以 2001 年的一篇日志为例，展开数据清洗的过程。移除文章的 YAML 元数据，对于文本分析来说，主要是没啥信息含量。

```
remove_yaml <- function(x) {
  x[(max(which(x == "---")) + 1):length(x)]
```

©黄湘云

图 24.1: 益辉每年发布的日志数量

©黄湘云

```
}
x <- remove_yaml(x)
```

移除「我」「是」「你」「的」「了」「也」等高频的人称、助词、虚词。这些词出现的规律对表现个人风格很重要，且看红楼梦关于后 40 回作者归属的研究，通过比较一些助词、虚词的出现规律，从而看出作者的习惯、文风。这种东西是在长期的潜移默化中形成的，对作者自己来说，都可能是无意识的。

```
library(jiebaR)
# jieba_seg <- worker(stop_word = "data/text/stop_word.txt")
jieba_seg <- worker(stop_word = "data/text/cn_stopwords.txt")
```

添加新词，比如「歪贼」、「谢益辉」等，主要是人名、外号等实体。

```
new_words <- readLines(file("data/text/new_word.txt"))
new_user_word(worker = jieba_seg, words = new_words)
```

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

```
# 分词
x_seg <- segment(x, jieba_seg)
```

分词后，再移除数字和英文

```
remove_number_english <- function(x) {
  x <- x[!grepl("\\d{1,}", x)]
  x[!grepl("[a-zA-Z]", x)]
}
xx <- remove_number_english(x = x_seg)
```

词频统计

```
tmp <- freq(x = xx)
tmp <- tmp[order(tmp$freq, decreasing = T), ]
head(tmp)
```

```
#>       char freq
#> 285   一个    7
#> 397   同学    5
#> 178   一篇    4
#> 356   没有    4
#> 67    鼻音    3
#> 106 田春雨    3
```

ggwordcloud 包绘制词云图可视化词频统计的结果。

```
library(ggwordcloud)
head(tmp, 150) |>
  ggplot(aes(label = char, size = freq)) +
```

© 黄湘云


```
geom_text_wordcloud(seed = 2022, grid_size = 8, max_grid_size = 24) +
scale_size_area(max_size = 10)
```

图 24.2: 词云可视化词频结果

计算 TF-IDF 值

```
# tmp = get_idf(x = list(xx))
get_idf(x = list(xx)) |> head()
```

```
#>     name count
#> 1   黄新     0
#> 2   邹瑜     0
#> 3   张恒     0
#> 4 蒋前程     0
#> 5   肖攀     0
#> 6 刘浩然     0
```

©谢益辉

24.2.3 主题建模

益辉的日志是没有分类和标签的，所以，先聚类，接着逐个分析每个类代表的实际含义。然后，将聚类的结果作为结果标签，再应用多分类回归模型，最后联合聚类、分类模型，从无监督转化到有监督模型。

topicmodels (Grün 和 Hornik 2011) 基于 tm (Feinerer, Hornik, 和 Meyer 2008) Latent Dirichlet Allocation (LDA) 和 Correlated Topics Models (CTM) 文本主题建模，这一套工具比较适合英文文本分词、向量化和建模。

text2vec 包支持多个统计模型，如潜在狄利克雷分配等，可用于分类、回归、聚类等任务。更多详情见 https://text2vec.org。

```
library(text2vec)
```

首先将所有日志分词、向量化，构建文档-词矩阵 document-term matrix (DTM)

```
# 移除链接
remove_links <- function(x) {
  gsub(pattern = "(<http.*?>)|(\\(http.*?\\))|(<www.*?>)|(\\(www.*?>\\))", replacement = "", x)
}
# 清理、分词、清理
file_list1 <- lapply(file_list, remove_yaml)
file_list1 <- lapply(file_list1, remove_links)
file_list1 <- lapply(file_list1, segment, jiebar = jieba_seg)
file_list1 <- lapply(file_list1, remove_number_english)
```

去掉没啥实际意义的词（比如单个字），极高频词和极低频词。

```
# Token 化
it <- itoken(file_list1, ids = 1:length(file_list1), progressbar = FALSE)
v <- create_vocabulary(it)
# 去掉单个字 减少 3K
v <- v[nchar(v$term) > 1,]
# 去掉极高频词和极低频词 减少 1.4W
v <- prune_vocabulary(v, term_count_min = 10, doc_proportion_max = 0.2)
```

采用 LDA（Latent Dirichlet Allocation）算法建模

```
# 词向量化
vectorizer <- vocab_vectorizer(v)
# 文档-词矩阵 DTM
dtm <- create_dtm(it, vectorizer, type = "dgTMatrix")
#  10 个主题
lda_model <- LDA$new(n_topics = 10, doc_topic_prior = 0.1, topic_word_prior = 0.01)
# 训练模型
doc_topic_distr <- lda_model$fit_transform(
```

©黄湘云

```
    x = dtm, n_iter = 1000, convergence_tol = 0.001,
    n_check_convergence = 25, progressbar = FALSE
  )
#> INFO  [09:37:56.530] early stopping at 175 iteration
#> INFO  [09:37:57.446] early stopping at 75 iteration
```

下图展示主题的分布，各个主题及其所占比例。

```
barplot(
  doc_topic_distr[1, ], xlab = "主题", ylab = "比例",
  ylim = c(0, 1), names.arg = 1:ncol(doc_topic_distr)
)
```

图 24.3: 主题分布

将 10 个主题的 Top 12 词分别打印出来。

```
lda_model$get_top_words(n = 12, topic_number = 2L:10L, lambda = 0.3)
#>       [,1]   [,2]     [,3]     [,4]     [,5]   [,6]     [,7]     [,8]   [,9]
#>  [1,] "人生" "网站"   "统计"   "记得"   "吱吱" "小时候" "工作"   "代码" "社会"
#>  [2,] "生活" "域名"   "数据"   "老师"   "一看" "歌词"   "项目"   "文档" "观点"
#>  [3,] "读书" "服务器" "模型"   "同学"   "居然" "一首"   "编辑"   "文件" "教育"
#>  [4,] "力量" "注册"   "变量"   "学校"   "小子" "阿姨"   "免费"   "功能" "时代"
```

© 黄湘云

```
#>  [5,] "文字" "博客"   "计算"   "一次"   "满意" "好吃"   "社区"   "字体" "人类"
#>  [6,] "自然" "服务"   "图形"   "英语"   "整理" "照片"   "文章"   "用户" "文化"
#>  [7,] "一句" "系统"   "分布"   "高中"   "中间" "渴望"   "意味着" "生成" "成功"
#>  [8,] "一生" "广告"   "统计学" "第一次" "一遍" "听到"   "精力"   "图片" "认为"
#>  [9,] "娱乐" "访问"   "方法"   "北京"   "一天" "我家"   "最大"   "元素" "价值"
#> [10,] "作品" "搜索"   "矩阵"   "小学"   "打开" "地上"   "想法"   "编译" "创造"
#> [11,] "风格" "网址"   "回归"   "美帝"   "飞机" "黄瓜"   "写作"   "安装" "美国"
#> [12,] "文中" "网络"   "数学"   "回家"   "全都" "记忆"   "显著"   "函数" "本书"
```

结果有点意思，说明益辉喜欢统计图形（主题 4）、代码编程（主题 9）、倒腾网站（主题 3）、回忆青春（主题 5）、读书写作（主题 2、8、10），但是诗词歌赋、做菜吃饭，不是很明显。

24.2.4 日志相似性

我与益辉日志的相似性度量 https://text2vec.org/similarity.html

24.3 习题

1. **text2vec** 包内置的电影评论数据集 `movie_review` 中 sentiment（表示正面或负面评价）列作为响应变量，构建二分类模型，对用户的一段评论分类。（提示：词向量化后，采用 **glmnet** 包做交叉验证调整参数、模型）
2. 根据 CRAN 上发布的 R 包元数据分析 R 包的描述字段，实现 R 包主题分类。
3. 接习题 2，根据任务视图对 R 包的标记，建立有监督的多分类模型，评估模型的分类效果，并对尚未标记的 R 包分类。（提示：一个 R 包可能同时属于多个任务视图，考虑使用 xgboost 包）

©黄湘云

第二十五章　生存分析

> The fact that some people murder doesn't mean we should copy them. And murdering data, though not as serious, should also be avoided.
>
> — Frank E. Harrell [1]

```
library(survival)  # survfit
library(ggplot2)
library(ggfortify) # autoplot
library(ggsurvfit) # survfit2 and ggsurvfit
library(glmnet)    # Cox Models
library(VGAM)      # R >= 4.4.0
library(INLA)
```

生存分析可以用于用户流失分析，注册、激活、活跃。分析次日留存、7 日留存、15 日留存。有学生来上体验课，多久来付费上课。有一个人医院看病之后，多久办理住院。最早，生存分析用于研究飞机出去之后，有多少返回的。还是要回归到原始文献去了解基本概念，及其背后的思考和应用

以一个问题提出本章主题，讲述和展示一个数据集。建立模型，拟合模型，结果解释。

25.1　问题背景

急性粒细胞白血病生存数据

```
library(survival)
data(cancer, package = "survival")
str(aml)
```

```
'data.frame':   23 obs. of  3 variables:
 $ time  : num  9 13 13 18 23 28 31 34 45 48 ...
 $ status: num  1 1 0 1 1 0 1 1 0 1 ...
 $ x     : Factor w/ 2 levels "Maintained","Nonmaintained": 1 1 1 1 1 1 1 1 1 1 ...
```

数据的分布情况如下

[1] https://stat.ethz.ch/pipermail/r-help/2005-July/075649.html

©黄湘云

```
ggplot(data = aml, aes(x = time, y = status, color = x)) +
  geom_jitter(height = 0.2) +
  theme_minimal()
```

图 25.1: 急性粒细胞白血病

在垂直方向添加了抖动，不影响时间项 time ，可以对数据的分布看得更加清楚。

25.2 模型拟合

Cox 比例风险回归模型与 Box-Cox 变换 (Box 和 Cox 1964)

- `survival::coxph()` Cox 比例风险回归模型
- `MASS::boxcox()` Box-Cox 变换
- `glmnet::glmnet(family = "cox")`
- INLA 包的函数 `inla()` 与 `inla.surv()` 一起拟合，链接
- survstan Stan 与生存分析
- rstanarm 包的函数 `stan_jm()` 使用说明 Estimating Joint Models for Longitudinal and Time-to-Event Data with rstanarm 链接
- rstanarm 包的生存分析分支

25.2.1 survival

R 软件内置了 survival 包，它是实现生存分析的核心 R 包 (Terry M. Therneau 和 Patricia M. Grambsch 2000)，其函数 `survfit()` 拟合模型。

©黄源文


```
aml_survival <- survfit(Surv(time, status) ~ x, data = aml)
summary(aml_survival)
```

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

                x=Maintained
 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    9     11       1    0.909  0.0867       0.7541        1.000
   13     10       1    0.818  0.1163       0.6192        1.000
   18      8       1    0.716  0.1397       0.4884        1.000
   23      7       1    0.614  0.1526       0.3769        0.999
   31      5       1    0.491  0.1642       0.2549        0.946
   34      4       1    0.368  0.1627       0.1549        0.875
   48      2       1    0.184  0.1535       0.0359        0.944

                x=Nonmaintained
 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    5     12       2   0.8333  0.1076       0.6470        1.000
    8     10       2   0.6667  0.1361       0.4468        0.995
   12      8       1   0.5833  0.1423       0.3616        0.941
   23      6       1   0.4861  0.1481       0.2675        0.883
   27      5       1   0.3889  0.1470       0.1854        0.816
   30      4       1   0.2917  0.1387       0.1148        0.741
   33      3       1   0.1944  0.1219       0.0569        0.664
   43      2       1   0.0972  0.0919       0.0153        0.620
   45      1       1   0.0000     NaN           NA           NA
```

拟合 Cox 比例风险回归模型（Cox Proportional Hazards Regression Model）

```
aml_coxph <- coxph(Surv(time, status) ~ 1 + x, data = aml)
summary(aml_coxph)
```

```
Call:
coxph(formula = Surv(time, status) ~ 1 + x, data = aml)

  n= 23, number of events= 18

                 coef exp(coef) se(coef)     z Pr(>|z|)
xNonmaintained 0.9155    2.4981   0.5119 1.788   0.0737 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

©南京大学出版社

```
                exp(coef) exp(-coef) lower .95 upper .95
xNonmaintained      2.498     0.4003    0.9159     6.813

Concordance= 0.619  (se = 0.063 )
Likelihood ratio test= 3.38  on 1 df,   p=0.07
Wald test            = 3.2  on 1 df,   p=0.07
Score (logrank) test = 3.42  on 1 df,   p=0.06
```

展示拟合结果。可以绘制生存分析的图的 R 包有很多，比如 ggfortify 包、ggsurvfit 包和 survminer 包等。ggfortify 包可以直接针对函数 `survfit()` 的返回对象绘图，ggsurvfit 包提供新函数 `survfit2()` 拟合模型、函数 `ggsurvfit()` 绘制图形，画面内容更加丰富，而 survminer 包依赖很多。

```
library(ggplot2)
library(ggfortify)
autoplot(aml_survival, data = aml) +
  theme_minimal()
```

图 25.2: 急性粒细胞白血病生存数据

参数化的生存分析模型（参数模型，相对于非参数模型而言）

```
aml_surv_reg <- survreg(Surv(time, status) ~ x, data = aml, dist = "weibull")
summary(aml_surv_reg)
```

```
Call:
survreg(formula = Surv(time, status) ~ x, data = aml, dist = "weibull")
                Value Std. Error     z      p
(Intercept)     4.109      0.300 13.70 <2e-16
xNonmaintained -0.929      0.383 -2.43  0.015
```

©生信宝典

```
Log(scale)     -0.235      0.178 -1.32  0.188

Scale= 0.791

Weibull distribution
Loglik(model)= -80.5   Loglik(intercept only)= -83.2
    Chisq= 5.31 on 1 degrees of freedom, p= 0.021
Number of Newton-Raphson Iterations: 5
n= 23
```

下面 ggsurvfit 包再次拟合模型，并展示模型结果。

25.2.2 ggsurvfit

```r
library(ggsurvfit)
```

拟合模型需要函数 survfit2()

```r
aml_ggsurvfit <- survfit2(Surv(time, status) ~ x, data = aml)
```

模型输出

```r
aml_ggsurvfit
```

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

                 n events median 0.95LCL 0.95UCL
x=Maintained    11      7     31      18      NA
x=Nonmaintained 12     11     23       8      NA
```

```r
ggsurvfit(aml_ggsurvfit, linewidth = 1) +
  add_confidence_interval() +
  add_risktable() +
  add_quantile(y_value = 0.6, color = "gray50", linewidth = 0.75) +
  scale_y_continuous(label = scales::percent_format()) +
  labs(y = "生存百分比", title = "从手术到随机化的复发时间")
```

```
Warning in ggplot2::geom_blank(): All aesthetics have length 1, but the data has 22 rows.
i Did you mean to use `annotate()`?
All aesthetics have length 1, but the data has 22 rows.
i Did you mean to use `annotate()`?
All aesthetics have length 1, but the data has 22 rows.
i Did you mean to use `annotate()`?
All aesthetics have length 1, but the data has 22 rows.
i Did you mean to use `annotate()`?
```

©黄湘云


```
All aesthetics have length 1, but the data has 22 rows.
i Did you mean to use `annotate()`?
```

图 25.3: 比例风险回归

25.2.3 glmnet

glmnet 包拟合 Cox 比例风险回归模型 (Simon 等 2011) 适合需要多变量筛选的情况。

```
library(glmnet)
# alpha = 1 lasso
aml_glmnet <- glmnet(x = aml$x, y = Surv(aml$time, aml$status), family = "cox", alpha = 1)
aml_glmnet_cv <- cv.glmnet(x = aml$x, y = Surv(aml$time, aml$status), family = "cox", alpha = 1)
```

25.2.4 INLA

INLA 包拟合 Cox 比例风险回归模型 (Gómez-Rubio 2020) 采用近似贝叶斯推断。

```
library(INLA)
inla.setOption(short.summary = TRUE)
```

©李东风

```
aml_inla <- inla(inla.surv(time, status) ~ x, data = aml, family = "exponential.surv", num.threads
summary(aml_inla)
```

```
Fixed effects:
                 mean    sd 0.025quant 0.5quant 0.975quant   mode kld
(Intercept)    -4.173 0.378     -4.913   -4.173     -3.432 -4.173   0
xNonmaintained  0.984 0.483      0.036    0.984      1.931  0.984   0

 is computed
```

25.3 Tobit 回归

Tobit (Tobin's Probit) regression 起源于计量经济学中的 Tobit 模型，James Tobin 提出的，用于截尾数据，生存分析中的一种加速失效模型 (accelerated failure model) (J. Tobin 1958)。

- 逻辑回归，响应变量是无序的分类变量，假定服从二项、多项分布，拟合函数 `glm()` 和 `nnet::multinom()`
- Probit 回归，响应变量是有序的分类变量，拟合函数 `MASS::polr()`
- Tobit 回归，响应变量是有删失/截尾的，VGAM 包依赖少，稳定，推荐使用。VGAM 包括了广义线性模型

```
library(VGAM)
with(aml, SurvS4(time, status))
```

```
      time status
 [1,]    9      1
 [2,]   13      1
 [3,]   13      0
 [4,]   18      1
 [5,]   23      1
 [6,]   28      0
 [7,]   31      1
 [8,]   34      1
 [9,]   45      0
[10,]   48      1
[11,]  161      0
[12,]    5      1
[13,]    5      1
[14,]    8      1
[15,]    8      1
[16,]   12      1
```

```
[17,]   16      0
[18,]   23      1
[19,]   27      1
[20,]   30      1
[21,]   33      1
[22,]   43      1
[23,]   45      1
attr(,"type")
[1] "right"
attr(,"class")
[1] "SurvS4"
```

©黄湘云

第二十六章　时序分析

预测是非常古老的话题，几乎人人都想拥有预测未来的能力，唐朝袁天罡和李淳风的故事至今还广为流传。事实上，古时候只有至高无上的皇帝才可以去问钦天监了解星辰大海和国运命脉。时间序列数据的分析，以及根据分析得到的一般规律进行预测是经久不衰的命题。预测既包含一般规律指向的确定性，又有无法预知的不确定性，且**同时**包含认知局限带来的不确定性，后者往往更大。无休止地渴求往往伴随着巨大的挑战，而更大的挑战则是预测效果常常不能满足期待。

```
library(quantmod)    # 获取数据
library(ggplot2)     # 可视化
library(ggfortify)   # 静态展示
library(lmtest)      # 格兰杰因果检验
library(dygraphs)    # 交互展示
```

本章主要从以下几个方面展开：数据获取、数据探索、平稳性诊断、时间序列分解、模型拟合和预测。

26.1　数据获取

Joshua M. Ulrich 开发维护的 **quantmod** 包可以下载国内外股票市场的数据。本节主要以美团股价数据为例，美团自 2018-09-20 在香港挂牌上市，股票代码 3690.HK。首先用 **quantmod** 包 (Ryan 和 Ulrich 2022) 获取美团上市至 2023-11-24 每天的股价数据，包含 Open 开盘价、High 最高价、Low 最低价、Close 闭市价、Adjusted 调整价和 Volume 成交量数据。

```
library(quantmod)
# 美团股票代码 3690
meituan <- getSymbols("3690.HK", auto.assign = FALSE, src = "yahoo")
```

先来看数据的类型，数据类型颇为复杂，是由 `xts` 和 `zoo` 两种类型复合而成，`xts` 类型是继承自 `zoo` 类型的。

```
class(meituan)
```

```
[1] "xts" "zoo"
```

```
str(meituan)
```

```
An xts object on 2018-09-20 / 2023-11-24 containing:
```

©黄文彬

```
 Data:    double [1275, 6]
Columns: 3690.HK.Open, 3690.HK.High, 3690.HK.Low, 3690.HK.Close, 3690.HK.Volume ... with 1 more column
 Index:   Date [1275] (TZ: "UTC")
 xts Attributes:
   $ src    : chr "yahoo"
   $ updated: POSIXct[1:1], format: "2023-11-27 06:31:12"
```

数据集 meituan 是一个 xts 类型的时间序列数据对象，时间范围是 2018-09-20 至 2023-11-24，包含 4 个成分，分别如下

- Data 部分显示为 906 行 6 列的双精度浮点存储的数值。
- Columns 部分显示列名，依次是 3690.HK.Open、3690.HK.High、3690.HK.Low 和 3690.HK.Close 等，当列数很多时，显示时会省略。
- Index 部分表示索引列，有序是时间序列数据的本质特点。示例中索引存储数据点产生的先后顺序，索引是用日期来表示的，日期所在的时区是 “UTC”。
- xts 部分是数据类型的一些属性（元数据），说明数据集的来源，什么时候制作的数据。示例中数据是从雅虎财经下载的，下载时间是 2023-11-27 14:31:12。

与时间序列数据相关的数据类型有很多，比如 Base R 提供的 Date 和 POSIX 等，扩展包 **timeDate** 和 **chron** 也都有自己的一套数据类型及处理方法。**xts** 包是处理时间序列数据的主要工具之一，xts 是 eXtensible Time Series 的缩写。为了进一步了解用法，下面举个例子，使用该 R 包的函数 xts() 构造时间序列对象。

```r
xts(x = NULL,
    order.by = index(x),
    frequency = NULL,
    unique = TRUE,
    tzone = Sys.getenv("TZ"),
    ...)
```

- 参数 x 表示数据。
- 参数 order.by 表示索引数据。
- 参数 frequency 表示频率。
- 参数 unique 表示唯一。
- 参数 tzone 表示时区。

```r
library(zoo)
library(xts)
# 数据矩阵
x <- matrix(1:4, ncol = 2, nrow = 2)
# 日期索引
idx <- as.Date(c("2018-01-01", "2019-12-12"))
# xts = matrix + index
```

©黄湘云

```
xts(x, order.by = idx)
```

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

26.2 数据探索

26.2.1 zoo

zoo 包提供 S3 范型函数 `autoplot.zoo()` 专门可视化 `zoo` 类型的数据，它接受一个 `zoo` 类型的数据对象，返回一个 `ggplot2` 数据对象，然后用户可以添加自定义的绘图设置，更多详情见帮助文档 `?autoplot.zoo()` 。

```
# xts 包需要先加载，否则 Index 不是日期类型而是数值类型
library(ggplot2)
autoplot(meituan[, "3690.HK.Adjusted"]) +
  theme_classic() +
  labs(x = "日期", y = "股价")
```

图 26.1: 美团在香港上市以来的股价走势

zoo 包还提供另一个范型函数 `fortify()` 将 `zoo` 数据对象转化为 `data.frame` ，这可以方便使用 **ggplot2** 包来展示数据。参数 `melt = TRUE` 意味着重塑原数据集，将数据从宽格式转长格式。参数 `names = c(Index = "Date")` 表示将 Index 列重命名为 date 列。

统计学习与建模 ©

```
meituan_df <- fortify(
  meituan[, c("3690.HK.Adjusted", "3690.HK.High")],
  melt = TRUE, names = c(Index = "Date")
)
```

数据集 meituan_df 中的 Series 列是因子型的，将其标签 3690.HK.Adjusted 、3690.HK.High 调整为调整价、最高价。根据日期字段 Date 提取年份字段 year 和一年中的第几天的字段 day_of_year。

```
meituan_df <- within(meituan_df, {
  # 调整 Series 的标签
  Series <- factor(Series, labels = c("调整价", "最高价"))
  # 日期字段 Date 获取年份
  year <- format(Date, "%Y")
  # 日期字段 Date 一年中的第几天
  day_of_year <- as.integer(format(Date, "%j"))
})
```

调用 **ggplot2** 包绘制分面、分组时间序列图，以 day_of_year 为横轴，股价 Value 为纵轴，按 year 分组，按 Series 分面。

```
ggplot(data = meituan_df, aes(x = day_of_year, y = Value)) +
  geom_line(aes(color = year)) +
  facet_wrap(~Series, ncol = 1) +
  theme_classic() +
  labs(x = "一年中的第几天", y = "调整的股价", color = "年份")
```

图 26.2: 美团调整的股价逐年走势

© 王建湘

2019 年底开始出现疫情，2020 年整年陆续有疫情，美团股价一路狂飙突进，因疫情，利好外卖业务，市场看好外卖业务。2021 年政府去杠杆，互联网监管趋严，又监又管，受外部大环境，逆全球化趋势影响，整年股价一路走低。进入 2022 年，股价在 200 附近徘徊。

26.2.2 xts

```
library(xts)
```

xts 包提供 S3 泛型函数 `plot.xts()` 专门用来可视化 `xts` 类型的时间序列数据

```
plot(meituan[, "3690.HK.Adjusted"], main = "调整的股价")
```

图 26.3: 美团在香港上市以来的股价走势

还可以任意选择一个时间窗口，展示相关数据

```
plot(meituan[, "3690.HK.Adjusted"],
  subset = "2022-01-01/2022-12-31", main = "调整的股价"
)
```

© 黄湘云

图 26.4: 美团 2021 年的股价走势

元旦节三天不开市，所以假期没有数据。

26.2.3 ggfortify

ggfortify (Tang, Horikoshi, 和 Li 2016) 支持快速地可视化 `ts`、`timeSeries` 、`stl` 等多种类型的时序数据，**ggplot2** 做数据探索会有一些帮助。

```
library(ggfortify)
autoplot(meituan[, "3690.HK.Adjusted"], ts.geom = "line") +
  scale_x_date(
    date_breaks = "1 year",
    date_minor_breaks = "6 months",
    date_labels = "%b\n%Y"
  ) +
  theme_classic()
```

©黄湘云

图 26.5: 美团股价走势

26.2.4 dygraphs

dygraphs 包专门绘制交互式时间序列图形，它封装了时序数据可视化库 dygraphs，更多情况见 https://dygraphs.com/。下面以美团股价为例，展示时间窗口筛选、坐标轴名称、刻度标签、注释、事件标注、缩放等功能。

```
library(dygraphs)
# 缩放
dyUnzoom <- function(dygraph) {
  dyPlugin(
    dygraph = dygraph,
    name = "Unzoom",
    path = system.file("plugins/unzoom.js", package = "dygraphs")
  )
}

# 年月
getYearMonth <- '
  function(d) {
    var monthNames = ["01", "02", "03", "04", "05", "06","07", "08", "09", "10", "11", "12"];
    date = new Date(d);
    return date.getFullYear() + "-" + monthNames[date.getMonth()];
  }'
```

©曹志梁

```
# 绘图
dygraph(meituan[, "3690.HK.Adjusted"], main = "美团股价走势") |>
  dyRangeSelector(dateWindow = c("2023-01-01", "2023-11-24")) |>
  dyAxis(name = "x", axisLabelFormatter = getYearMonth) |>
  dyAxis("y", valueRange = c(0, 500), label = "美团股价") |>
  dyEvent("2020-01-23", "武汉封城", labelLoc = "bottom") |>
  dyShading(from = "2020-01-23", to = "2020-04-08", color = "#FFE6E6") |>
  dyAnnotation("2020-01-23", text = "武汉封城", tooltip = "武汉封城", width = 60) |>
  dyAnnotation("2020-04-08", text = "武汉解封", tooltip = "武汉解封", width = 60) |>
  dyHighlight(highlightSeriesOpts = list(strokeWidth = 2)) |>
  dySeries(label = "调整股价") |>
  dyLegend(show = "follow", hideOnMouseOut = FALSE) |>
  dyOptions(fillGraph = TRUE, drawGrid = FALSE, gridLineColor = "lightblue") |>
  dyUnzoom()
```

图 26.6: 美团股价变化趋势

上图默认展示 YTD 数据，在一个动态的时间窗口内显示数据，假如今天是 2023-07-15，则展示 2023-01-01 至 2023-07-15 的股价数据。在函数 `dyRangeSelector()` 中设定时间窗口参数 `dateWindow`，实现数据范围的筛选。

© 黄湘云

26.3 平稳性诊断

26.3.1 自相关图

```
autoplot(acf(AirPassengers, plot = FALSE)) +
  theme_classic()
```

图 26.7: 乘客数量自相关图

26.3.2 偏自相关图

```
autoplot(pacf(AirPassengers, plot = FALSE)) +
  theme_classic()
```

©黄湘云

图 26.8: 乘客数量偏自相关图

26.3.3 延迟算子

```
# 原始序列
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
```

```
# 延迟 1 期
lag(AirPassengers, k = 1)
```

© 统计出版社


```
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1948                                             112
1949 118 132 129 121 135 148 148 136 119 104 118 115
1950 126 141 135 125 149 170 170 158 133 114 140 145
1951 150 178 163 172 178 199 199 184 162 146 166 171
1952 180 193 181 183 218 230 242 209 191 172 194 196
1953 196 236 235 229 243 264 272 237 211 180 201 204
1954 188 235 227 234 264 302 293 259 229 203 229 242
1955 233 267 269 270 315 364 347 312 274 237 278 284
1956 277 317 313 318 374 413 405 355 306 271 306 315
1957 301 356 348 355 422 465 467 404 347 305 336 340
1958 318 362 348 363 435 491 505 404 359 310 337 360
1959 342 406 396 420 472 548 559 463 407 362 405 417
1960 391 419 461 472 535 622 606 508 461 390 432
```

26.3.4 差分算子

函数 `diff()` 实现差分算子，默认参数 `lag = 1` ，`differences = 1` 表示延迟期数为 1 的一阶差分。

```r
# 延迟 1 期 1 阶差分
diff(AirPassengers, lag = 1, differences = 1)
```

```
      Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
1949         6   14   -3   -8   14   13    0  -12  -17  -15   14
1950   -3   11   15   -6  -10   24   21    0  -12  -25  -19   26
1951    5    5   28  -15    9    6   21    0  -15  -22  -16   20
1952    5    9   13  -12    2   35   12   12  -33  -18  -19   22
1953    2    0   40   -1   -6   14   21    8  -35  -26  -31   21
1954    3  -16   47   -8    7   30   38   -9  -34  -30  -26   26
1955   13   -9   34    2    1   45   49  -17  -35  -38  -37   41
1956    6   -7   40   -4    5   56   39   -8  -50  -49  -35   35
1957    9  -14   55   -8    7   67   43    2  -63  -57  -42   31
1958    4  -22   44  -14   15   72   56   14 -101  -45  -49   27
1959   23  -18   64  -10   24   52   76   11  -96  -56  -45   43
1960   12  -26   28   42   11   63   87  -16  -98  -47  -71   42
```

© 基础统计

26.3.5 单位根检验

26.3.6 格兰杰因果检验

1969 年 Clive Granger 提出格兰杰因果检验，R 语言中 **lmtest** 包的函数 `grangertest()` 可以检验序列中变量之间的时间落差的相关性。

26.4 指数平滑模型

26.4.1 指数平滑

首先来回答何为指数平滑？用历史数据的线性组合预测下一个时期的值，线性组合的权重随距离变远而按指数衰减。不妨设观测序列数据为 $\{x_i\}$ ，预测序列数据为 $\{y_i\}$，用数学公式表达，如下：

$$y_h(1) = wx_h + w^2x_{h-1} + \cdots = \sum_{j=1}^{\infty} w^j x_{h+1-j}$$

其中，权重 $0 < w < 1$ ，权重越小表示距离远的历史数据对当前预测的贡献越小。线性组合的权重之和等于 1，所以

$$\sum_{j=1}^{\infty} w^j = \frac{w}{1-w}$$

则第 j 个权重应为

$$\frac{w^j}{\frac{w}{1-w}} = (1-w)w^{j-1}, j = 1, 2, \ldots$$

则根据历史的 h 期数据预测未来的 1 期数据 $y_h(1)$ 如下：

$$y_h(1) = (1-w)(x_h + wx_{h-1} + w^2x_{h-2} + \cdots) = (1-w)\sum_{j=0}^{\infty} w^j x_{h-j}$$

以上就是指数平滑（exponential smoothing），在早期应用中，权重 w 的选取主要依靠经验。适用于没有明显趋势性、季节性、周期性的时间序列数据。

26.4.2 函数 `filter()`

函数 `filter()` 实现一元时间序列的线性过滤，或者对多元时间序列的单个序列分别做线性变换，它只是根据既定的平滑模型变换数据，没有拟合数据。函数 `filter()` 实现递归过滤和卷积过滤两种数据变换方式，分别对应自回归和移动平均两种时间序列平滑模型。

©黄湘云

- 递归过滤（自回归）

$$y_i = x_i + f_1 y_{i-1} + \cdots + f_p y_{i-p} \tag{26.1}$$

- 卷积过滤（移动平均）

$$y_i = f_1 x_{i+o} + \cdots + f_p x_{i+o-(p-1)} \tag{26.2}$$

其中，p 代表模型的阶数，o 代表漂移项，O 表示英文单词 offset 的首字母。下面举个具体的例子来说明函数 `filter()` 的作用，设输入序列 $\{x_i\}$ 是从 1 至 10 的整数。首先考虑自回归的情况，代码如下：

```
x <- 1:10
# 自回归
filter(x, filter = c(2 / 3, 1 / 6, 1 / 6), method = "recursive")
```

```
Time Series:
Start = 1
End = 10
Frequency = 1
 [1]  1.000000  2.666667  4.944444  7.907407 11.540123 15.835391 20.798182
 [8] 26.428041 32.724289 39.687230
```

参数 `x` 指定输入的时间序列 $\{x_i\}$，参数 `method` 指定平滑的方法，`method = "recursive"` 表示使用自回归方法，参数 `filter` 表示自回归的系数，系数向量的长度代表模型方程式 26.1 中的 p ，`filter = c(2 / 3, 1 / 6, 1 / 6)` 对应的模型如下：

$$\begin{aligned}
y_1 &= x_1 \\
y_2 &= x_2 + \frac{2}{3} y_1 \\
y_3 &= x_3 + \frac{2}{3} y_2 + \frac{1}{6} y_1 \\
y_i &= x_i + \frac{2}{3} y_{i-1} + \frac{1}{6} y_{i-2} + \frac{1}{6} y_{i-3}, \quad i \geq 4
\end{aligned}$$

其中，序列 $\{y_i\}$ 表示函数 `filter()` 的输出结果，由上述方程不难看出自回归模型的递归的特点。为了理解自回归和递归的过程，下面依次计算 y_1 至 y_4 。

```
# y1
1
```

```
[1] 1
```

```
# y2
2 + 2/3 * 1
```

```
[1] 2.666667
```

©黄湘云

```
# y3
3 + 2/3 * (2 + 2/3 * 1) + 1/6 * 1
```

```
[1] 4.944444
```

```
# y4
4 + 2/3 * (3 + 2/3 * (2 + 2/3 * 1) + 1/6 * 1) + 1/6 *(2 + 2/3 * 1) + 1/6 * 1
```

```
[1] 7.907407
```

接下来，考虑移动平均的情况，代码如下：

```
# 移动平均
filter(x, filter = c(2 / 3, 1 / 6, 1 / 6), method = "convolution", sides = 1)
```

```
Time Series:
Start = 1
End = 10
Frequency = 1
 [1]  NA  NA 2.5 3.5 4.5 5.5 6.5 7.5 8.5 9.5
```

参数 `method = "convolution"` 表示使用移动平均。参数 `sides` 仅适用于卷积过滤，`sides = 1` 表示系数都是作用于过去的值。为了对比自回归和移动平均，不妨设移动平均的系数同自回归的系数，则移动平均模型如下：

$$
\begin{aligned}
&y_1 \quad \text{不存在} \\
&y_2 \quad \text{不存在} \\
&y_3 = \frac{2}{3}x_3 + \frac{1}{6}x_2 + \frac{1}{6}x_1 \\
&y_i = \frac{2}{3}x_i + \frac{1}{6}x_{i-1} + \frac{1}{6}x_{i-2}, \quad i \geq 3
\end{aligned}
$$

比照模型方程式 26.2 ，漂移项参数 o 为 0，也就是没有漂移，移动平均作用于过去的 3 期数据，也就是 $p = 3$ 。因输出序列 $\{y_i\}$ 中 y_1, y_2 不存在，下面仅计算 y_3, y_4 。

```
# y3
2/3 * 3 + 1/6 * 2 + 1/6 * 1
```

```
[1] 2.5
```

```
# y4
2/3 * 4 + 1/6 * 3 + 1/6 * 2
```

```
[1] 3.5
```

TTR 包提供许多移动平均的计算函数，比如 `SMA()` ，下面计算过去 3 个观察值的算术平均。

```
library(TTR)
SMA(x, n = 3)
```

```
 [1] NA NA  2  3  4  5  6  7  8  9
```

26.4.3 简单指数平滑

当时间序列不含趋势和季节性成分的时候，可以用简单指数平滑模型来拟合和预测。简单指数平滑模型如下：

$$
\begin{aligned}
\hat{y}_{t+h} &= a_t + h \times b_t + s_{t-p+1+(h-1) \mod p} \\
a_t &= \alpha(y_t - s_{t-p}) + (1-\alpha)(a_{t-1} + b_{t-1}) \\
b_t &= b_{t-1} \\
s_t &= s_{t-p}
\end{aligned}
$$

其中，周期 p

```
air_passengers_exp <- HoltWinters(AirPassengers, gamma = FALSE, beta = FALSE)
air_passengers_exp
```

```
Holt-Winters exponential smoothing without trend and without seasonal component.

Call:
HoltWinters(x = AirPassengers, beta = FALSE, gamma = FALSE)

Smoothing parameters:
 alpha: 0.9999339
 beta : FALSE
 gamma: FALSE

Coefficients:
      [,1]
a 431.9972
```

预测的残差平方和 SSE sum-of-squared-errors

```
air_passengers_exp$SSE
```

```
[1] 162510.6
```

```
# plot(air_passengers_exp)
autoplot(air_passengers_exp) +
  theme_classic()
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_line()`).
```

© 黄湘云

图 26.9: 简单指数平滑模型

向前预测 5 期

```
air_passengers_pred <- predict(air_passengers_exp, n.ahead = 10, prediction.interval = TRUE)
```

预测值及其预测区间

```
plot(air_passengers_exp, air_passengers_pred)
```

©黄湘云

图 26.10: 简单指数平滑模型预测

26.4.4 Holt 指数平滑

当时间序列不含季节性成分，可以用 Holt 指数平滑模型拟合和预测 (Holt 2004) 。

$$
\begin{aligned}
\hat{y}_{t+h} &= a_t + h \times b_t + s_{t-p+1+(h-1) \mod p} \\
a_t &= \alpha(y_t - s_{t-p}) + (1-\alpha)(a_{t-1} + b_{t-1}) \\
b_t &= \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1} \\
s_t &= s_{t-p}
\end{aligned}
$$

```
air_passengers_holt <- HoltWinters(AirPassengers, gamma = FALSE)
air_passengers_holt
```

```
Holt-Winters exponential smoothing with trend and without seasonal component.

Call:
HoltWinters(x = AirPassengers, gamma = FALSE)

Smoothing parameters:
```

统计学习基础 ©

```
alpha: 1
beta : 0.003218516
gamma: FALSE

Coefficients:
        [,1]
a 432.000000
b   4.597605
```

可知，$\alpha = 1, \beta = 0.0032$

```
plot(air_passengers_holt)
```

图 26.11: holt 指数平滑模型

26.4.5 Holt-Winters 指数平滑

时间序列同时含有趋势成分、季节性成分、随机成分，可以用 Holt-Winters 平滑模型来拟合和预测。根据趋势和季节性的关系，Holt-Winters 平滑模型分为可加 Holt-Winters 平滑和可乘 Holt-Winters 平滑。R 提供函数 `HoltWinters()` 拟合 Holt-Winters 平滑模型 (Holt 2004; Winters 1960)。

可加 Holt-Winters 平滑模型如下：

©李漫漫

$$
\begin{aligned}
\hat{y}_{t+h} &= a_t + h \times b_t + s_{t-p+1+(h-1) \mod p} \\
a_t &= \alpha(y_t - s_{t-p}) + (1-\alpha)(a_{t-1} + b_{t-1}) \\
b_t &= \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1} \\
s_t &= \gamma(y_t - a_t) + (1-\gamma)s_{t-p}
\end{aligned}
$$

可乘 Holt-Winters 平滑模型如下：

$$
\begin{aligned}
\hat{y}_{t+h} &= (a_t + h \times b_t) \times s_{t-p+1+(h-1) \mod p} \\
a_t &= \alpha(y_t / s_{t-p}) + (1-\alpha)(a_{t-1} + b_{t-1}) \\
b_t &= \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1} \\
s_t &= \gamma(y_t / a_t) + (1-\gamma)s_{t-p}
\end{aligned}
$$

其中 α, β, γ 是参数，p 为周期长度，a_t, b_t, s_t 分别代表水平、趋势和季节性成分。

```
air_passengers_add <- HoltWinters(AirPassengers, seasonal = "additive")
air_passengers_add
```

```
Holt-Winters exponential smoothing with trend and additive seasonal component.

Call:
HoltWinters(x = AirPassengers, seasonal = "additive")

Smoothing parameters:
 alpha: 0.2479595
 beta : 0.03453373
 gamma: 1

Coefficients:
          [,1]
a   477.827781
b     3.127627
s1  -27.457685
s2  -54.692464
s3  -20.174608
s4   12.919120
s5   18.873607
s6   75.294426
s7  152.888368
s8  134.613464
s9   33.778349
```

```
s10 -18.379060
s11 -87.772408
s12 -45.827781
```

可知，$\alpha = 0.248, \beta = 0.0345, \gamma = 1$

```r
autoplot(air_passengers_add) +
  theme_classic()
```

```
Warning: Removed 12 rows containing missing values or values outside the scale range
(`geom_line()`).
```

图 26.12: 可加 Holt-Winters 平滑模型拟合

```r
air_passengers_mult <- HoltWinters(AirPassengers, seasonal = "mult")
```

```r
autoplot(air_passengers_mult) +
  theme_classic()
```

```
Warning: Removed 12 rows containing missing values or values outside the scale range
(`geom_line()`).
```

© 黄湘云

图 26.13: 可乘 Holt-Winters 平滑模型拟合

做一个 Shiny 应用展示参数 α, β, γ 对 Holt-Winters 平滑预测的影响。

26.5 时间序列分解

- 可加模型

$$y_t = T_t + S_t + e_t$$

- 可乘模型

$$y_t = T_t \times S_t \times e_t$$

对时间序列 $\{y_t\}$ 分解，趋势性成分 T_t、季节性成分 S_t、剩余成分 e_t

26.5.1 函数 decompose()

函数 decompose() 分解

```
air_decomp_add <- decompose(x = AirPassengers, type = "additive")
```

函数返回一个列表，包含 6 个元素，分别是 x 原始序列，seasonal 季节性成分，figure 估计的季节图，trend 趋势成分，random 剩余成分，type 分解方法。

```
# plot(air_decomp_add)
autoplot(air_decomp_add) +
  theme_classic()
```

```
Warning: Removed 24 rows containing missing values or values outside the scale range
(`geom_line()`).
```

图 26.14: 变化趋势的分解

去掉季节性部分

```
AirPassengers_adjusted <- AirPassengers - air_decomp_add$seasonal
plot(AirPassengers_adjusted)
```

© 黄湘云

图 26.15: 季节性调整

26.5.2 函数 stl()

函数 stl() 将时间序列分解为趋势性成分、季节性成分（周期性）、剩余成分。

```
air_stl <- stl(x = AirPassengers, s.window = 12)
```

```
autoplot(air_stl) +
  theme_classic()
```

©黄湘云

图 26.16: 变化趋势的分解

剩余成分不是平稳序列，是异方差的。

xts 包的 `periodicity()` 函数可以检测时间序列数据的周期，但时序数据对象最好是在 **xts** 框架内。

```
xts::periodicity(AirPassengers)
```

```
Monthly periodicity from Jan 1949 to Dec 1960
```

26.6 经典时间序列模型

26.6.1 自回归模型

函数 `ar()` 拟合 AR 模型

```
ar(AirPassengers, order.max = 3)
```

```
Call:
ar(x = AirPassengers, order.max = 3)

Coefficients:
      1        2
 1.1656  -0.2294

Order selected 2  sigma^2 estimated as  1399
```

26.6.2 移动平均模型

将自回归的阶设为 0，函数 arima() 也可以用来拟合 MA 模型。

```
arima(AirPassengers, order = c(0, 1, 3))
```

```
Call:
arima(x = AirPassengers, order = c(0, 1, 3))

Coefficients:
         ma1     ma2      ma3
      0.1309  -0.359  -0.3599
s.e.  0.0741   0.090   0.0907

sigma^2 estimated as 949.5:  log likelihood = -693.45,  aic = 1394.91
```

26.6.3 自回归移动平均模型

函数 arima() 拟合 ARIMA 模型

```
arima(AirPassengers, order = c(1, 1, 3))
```

```
Call:
arima(x = AirPassengers, order = c(1, 1, 3))

Coefficients:
         ar1      ma1      ma2      ma3
      0.5227  -0.2906  -0.3884  -0.1219
s.e.  0.1291   0.1284   0.1445   0.1322

sigma^2 estimated as 886:  log likelihood = -688.45,  aic = 1386.89
```

forecast 包提供函数 `auto.arima()` 自动选择合适的自回归、差分和移动平均的阶来拟合数据。

```
forecast::auto.arima(AirPassengers)
```

```
Series: AirPassengers
ARIMA(2,1,1)(0,1,0)[12]

Coefficients:
         ar1     ar2      ma1
      0.5960  0.2143  -0.9819
s.e.  0.0888  0.0880   0.0292

sigma^2 = 132.3:  log likelihood = -504.92
AIC=1017.85   AICc=1018.17   BIC=1029.35
```

26.7 总结

方法没有好坏，只有适合与否。Holt-Winter 适合预警任务，算法简单，可以及时出预测结果，仅需要一步预测，不需要给出多步预测，要求快，以便迅速作出反应。Prophet 实现的贝叶斯结构可加模型适合短期预测任务，只要在可容许的时间范围内出结果即可，可以迅速出结果当然更好，需要给出多步预测结果，且结果需要强解释性，以便提前做一些商家供给、平台资源的分配。商分模型常常需要比较强的可解释性，算法策略模型重在预测精准度，对可解释性要求不高。

在时间序列数据的可视化方面，除了 Base R 提供的绘图方法外，静态的时序图 **lattice** 和 **ggplot2** 都不错，而交互式图形推荐使用 **plotly** 和 **dygraphs**。

PortfolioAnalytics 包做投资组合优化，均值-方差，收益和风险权衡。Rmetrics 提供系列时间序列数据分析和建模的 R 包，包括投资组合优化 **fPortfolio**、多元分析 **fMultivar**、自回归条件异方差模型 **fGarch**、二元相依结构的 Copulae 分析 **fCopulae** 、市场和基础统计 **fBasics** 。

fable 一元到多元时间序列预测问题，提供 ETS、ARIMA、TSLM 等模型，并有书籍时间序列预测原则。值得一提，**forecast** 包开发者 Rob J Hyndman 称已不再开发新的功能，推荐大家使用 **fable** 包。**feasts** 包辅助特征抽取、序列分解、汇总统计和绘制图形等，插件包 **fable.prophet** 接入 Prophet 的预测能力。**timetk** 时间序列数据处理、分析、预测和可视化工具箱，提供一致的操作方式，试图形成完成的解决方案。The Rmetrics Association 开发了一系列 R 包专门处理金融时间序列数据，比如 **fGarch** 包提供条件自回归异方差模型。

从时间序列中寻找规律，这样才是真的数据建模，从数据到模型，而不是相反 Finding Patterns in Time Series，识别金融时间序列的模式和统计规律。

©黄湘云

第二十七章　空间点模式分析

本章以斐济地震数据集 quakes 为例介绍空间点模式数据的操作、探索和分析。

```
library(spatstat)
library(sf)
library(ggplot2)
```

spatstat 是一个伞包，囊括 8 个子包，构成一套完整的空间点模式分析工具。

1. spatstat.utils 基础的辅助分析函数
2. spatstat.data 点模式分析用到的示例数据集
3. spatstat.sparse 稀疏数组
4. spatstat.geom 空间数据类和几何操作
5. spatstat.random 生成空间随机模式
6. spatstat.explore 空间数据的探索分析
7. spatstat.model 空间数据的参数建模和推理
8. spatstat.linnet 线性网络上的空间分析

sf 包是一个专门用于空间矢量数据操作的 R 包。**ggplot2** 包提供的几何图层函数 `geom_sf()` 和坐标参考系图层函数 `coord_sf()` 支持可视化空间点模式数据。

27.1　数据操作

27.1.1　类型转化

先对斐济地震数据 quakes 数据集做一些数据类型转化，从 data.frame 转 Simple feature 对象。

```
library(sf)
quakes_sf <- st_as_sf(quakes, coords = c("long", "lat"), crs = st_crs(4326))
quakes_sf
```

```
#> Simple feature collection with 1000 features and 3 fields
#> Geometry type: POINT
#> Dimension:     XY
```

©黄湘云

```
#> Bounding box:  xmin: 165.67 ymin: -38.59 xmax: 188.13 ymax: -10.72
#> Geodetic CRS:  WGS 84
#> First 10 features:
#>    depth mag stations              geometry
#> 1    562 4.8       41 POINT (181.62 -20.42)
#> 2    650 4.2       15 POINT (181.03 -20.62)
#> 3     42 5.4       43     POINT (184.1 -26)
#> 4    626 4.1       19 POINT (181.66 -17.97)
#> 5    649 4.0       11 POINT (181.96 -20.42)
#> 6    195 4.0       12 POINT (184.31 -19.68)
#> 7     82 4.8       43   POINT (166.1 -11.7)
#> 8    194 4.4       15 POINT (181.93 -28.11)
#> 9    211 4.7       35 POINT (181.74 -28.74)
#> 10   622 4.3       19 POINT (179.59 -17.47)
```

27.1.2 坐标转化

如果知道两个投影坐标系的 EPSG 代码，输入坐标就可以完成转化。如将坐标系 `EPSG:4326` 下的坐标 $(2, 49)$ 投影到另一个坐标系 `EPSG:3857` 。

```
st_transform(
  x = st_sfc(st_point(x = c(2, 49)), crs = 4326), crs = 3857
)
```

```
#> Geometry set for 1 feature
#> Geometry type: POINT
#> Dimension:     XY
#> Bounding box:  xmin: 222639 ymin: 6274861 xmax: 222639 ymax: 6274861
#> Projected CRS: WGS 84 / Pseudo-Mercator
```

```
#> POINT (222639 6274861)
```

名称	EPSG	赤道半径	半轴	发明者
GRS80	3857	a=6378137.0	rf=298.257222101	GRS 1980(IUGG, 1980)
WGS84	4326	a=6378137.0	rf=298.257223563	WGS 84

函数 `st_crs()` 查看坐标参考系的信息，比如 EPSG 代码为 4326 对应的坐标参考系统信息。我们也可以通过网站查询 EPSG 代码对应的坐标参考系统的详细介绍。

```
st_crs("EPSG:4326")
```

```
#> Coordinate Reference System:
```

©黄湘云

```
#>   User input: EPSG:4326
#>   wkt:
#> GEOGCRS["WGS 84",
#>     ENSEMBLE["World Geodetic System 1984 ensemble",
#>         MEMBER["World Geodetic System 1984 (Transit)"],
#>         MEMBER["World Geodetic System 1984 (G730)"],
#>         MEMBER["World Geodetic System 1984 (G873)"],
#>         MEMBER["World Geodetic System 1984 (G1150)"],
#>         MEMBER["World Geodetic System 1984 (G1674)"],
#>         MEMBER["World Geodetic System 1984 (G1762)"],
#>         MEMBER["World Geodetic System 1984 (G2139)"],
#>         ELLIPSOID["WGS 84",6378137,298.257223563,
#>             LENGTHUNIT["metre",1]],
#>         ENSEMBLEACCURACY[2.0]],
#>     PRIMEM["Greenwich",0,
#>         ANGLEUNIT["degree",0.0174532925199433]],
#>     CS[ellipsoidal,2],
#>         AXIS["geodetic latitude (Lat)",north,
#>             ORDER[1],
#>             ANGLEUNIT["degree",0.0174532925199433]],
#>         AXIS["geodetic longitude (Lon)",east,
#>             ORDER[2],
#>             ANGLEUNIT["degree",0.0174532925199433]],
#>     USAGE[
#>         SCOPE["Horizontal component of 3D system."],
#>         AREA["World."],
#>         BBOX[-90,-180,90,180]],
#>     ID["EPSG",4326]]
```

地球看作一个椭球体 ELLIPSOID，长半轴 6378137 米，短半轴 298.257223563 米，椭圆形的两个轴，纬度单位 0.0174532925199433，经度单位 0.0174532925199433 。

地球是一个不规则的球体，不同的坐标参考系对地球的抽象简化不同，会体现在坐标原点、长半轴、短半轴等属性上。为了方便在平面上展示地理信息，需要将地球表面投影到平面上，墨卡托投影是其中非常重要的一种投影方式，墨卡托投影的详细介绍见 PROJ 网站 。WGS 84 / Pseudo-Mercator 投影主要用于网页上的地理可视化，UTM 是 Universal Transverse Mercator 的缩写。360 度对应全球 60 个时区，每个时区横跨 6 经度。

```
st_transform(
  x = st_sfc(st_point(x = c(2, 49)), crs = 4326),
  crs = st_crs("+proj=utm +zone=32 +ellps=GRS80")
```

©黄湘云

```
)
#> Geometry set for 1 feature
#> Geometry type: POINT
#> Dimension:     XY
#> Bounding box:  xmin: -11818.95 ymin: 5451107 xmax: -11818.95 ymax: 5451107
#> Projected CRS: +proj=utm +zone=32 +ellps=GRS80

#> POINT (-11818.95 5451107)
```

快速简单绘图，可采用图层 `geom_sf()`，它相当于统计图层 `stat_sf()` 和坐标映射图层 `coord_sf()` 的叠加，`geom_sf()` 支持点、线和多边形等数据数据对象，可以混合叠加。`coord_sf()` 有几个重要的参数：

1. `crs`：在绘图前将各个 `geom_sf()` 图层中的**数据**映射到该坐标参考系。
2. `default_crs`：将非 sf 图层（没有携带 CRS 信息）的数据映射到该坐标参考系，默认使用 `crs` 参数的值，常用设置 `default_crs = sf::st_crs(4326)` 将非 sf 图层中的横纵坐标转化为经纬度，采用 World Geodetic System 1984 (WGS84)。
3. `datum`：经纬网线的坐标参考系，默认值 `sf::st_crs(4326)`。

下图的右子图将 quakes_sf 数据集投影到坐标参考系统EPSG:3460。

```r
library(ggplot2)
ggplot() +
  geom_sf(data = quakes_sf, aes(color = mag))
ggplot() +
  geom_sf(data = quakes_sf, aes(color = mag)) +
  coord_sf(crs = 3460)
```

数据集 quakes_sf 已经准备了坐标参考系统，此时，`coord_sf()` 就会采用数据集相应的坐标参考系统，即 `sf::st_crs(4326)`。上图的左子图相当于：

```r
ggplot() +
  geom_sf(data = quakes_sf, aes(color = mag)) +
  coord_sf(
    crs = 4326, datum = sf::st_crs(4326),
    default_crs = sf::st_crs(4326)
  )
```

27.1.3 凸包操作

```r
quakes_sf <- st_transform(quakes_sf, crs = 3460)
# 组合 POINT 构造 POLYGON
quakes_sfp <- st_cast(st_combine(st_geometry(quakes_sf)), "POLYGON")
```

©黄湘云

(a) 坐标参考系 4326（默认）

(b) 坐标参考系 3460

图 27.1: 斐济地震的空间分布

```
# 构造 POLYGON 的凸包
quakes_sfp_hull <- st_convex_hull(st_geometry(quakes_sfp))

# 绘制点及其包络
plot(st_geometry(quakes_sf))
# 添加凸包曲线
plot(quakes_sfp_hull, add = TRUE)

ggplot() +
  geom_sf(data = quakes_sf) +
  geom_sf(data = quakes_sfp_hull, fill = NA) +
  coord_sf(crs = 3460, xlim = c(569061, 3008322), ylim = c(1603260, 4665206))
```

27.2 数据探索

27.2.1 核密度估计

给定边界内的核密度估计与绘制热力图

```
# spatial point pattern ppp 类型
quakes_ppp <- spatstat.geom::as.ppp(quakes_sf)
```

©黄湘云

(a) 凸包（base R）

(b) 凸包（ggplot2）

图 27.2: 凸包

```
#> Warning in as.ppp.sf(quakes_sf): only first attribute column is used for marks
```

```
# 限制散点在给定的窗口边界内平滑
spatstat.geom::Window(quakes_ppp) <- spatstat.geom::as.owin(quakes_sfp_hull)
```

```
#> Warning: point-in-polygon test had difficulty with 1 point (total score not 0
#> or 1)
```

```
# 密度估计
density_spatstat <- spatstat.explore::density.ppp(quakes_ppp, dimyx = 256)
# 转化为 stars 对象 栅格数据
density_stars <- stars::st_as_stars(density_spatstat)
# 设置坐标参考系
density_sf <- st_set_crs(st_as_sf(density_stars), 3460)
```

27.2.2 绘制热力图

```
ggplot() +
  geom_sf(data = density_sf, aes(fill = v), col = NA) +
  scale_fill_viridis_c() +
```

© 统计出版社


```
  geom_sf(data = st_boundary(quakes_sfp_hull))

ggplot() +
  geom_sf(data = density_sf, aes(fill = v), col = NA) +
  scale_fill_viridis_c() +
  geom_sf(data = st_boundary(quakes_sfp_hull)) +
  geom_sf(data = quakes_sf, size = 1, col = "black")
```

(a) 核密度估计　　(b) 核密度估计（原始数据）

图 27.3: 热力图

© 黄湘云

第二十八章　空间点参考数据分析

本章内容属于空间分析的范畴，空间分析的内容十分广泛，主要分三大块，分别是空间点参考数据分析、空间点模式分析和空间区域数据分析。本章仅以一个模型和一个数据简略介绍空间点参考数据分析。一个模型是空间广义线性混合效应模型，空间广义线性混合效应模型在流行病学、生态学、环境学等领域有广泛的应用，如预测某地区内的疟疾流行度分布，预测某地区 PM 2.5 污染物浓度分布等。一个数据来自生态学领域，数据集所含样本量不大，但每个样本收集成本不小，采集样本前也都有实验设计，数据采集的地点是预先设定的。下面将对真实数据分析和建模，任务是预测核辐射强度在朗格拉普岛上的分布。

28.1　数据说明

在第二次世界大战的吉尔伯特及马绍尔群岛战斗中，美国占领了马绍尔群岛。战后，美国在该群岛的比基尼环礁中陆续进行了许多氢弹核试验，对该群岛造成无法弥补的环境损害。位于南太平洋的朗格拉普环礁是马绍尔群岛的一部分，其中，朗格拉普岛是朗格拉普环礁的主岛，修建有机场，在太平洋战争中是重要的军事基地。朗格拉普岛距离核爆炸的位置较近，因而被放射性尘埃笼罩了，受到严重的核辐射影响，从度假胜地变成人间炼狱，居民出现上吐下泻、皮肤灼烧、脱发等症状。即便是 1985 年以后，那里仍然无人居住，居民担心核辐射对身体健康的影响。又几十年后，一批科学家来到该岛研究生态恢复情况，评估当地居民重返家园的可行性。实际上，该岛目前仍然不适合人类居住，只有经批准的科学研究人员才能登岛。

Ole F. Christensen 和 Paulo J. Ribeiro Jr 将 `rongelap` 数据集存放在 **geoRglm**(Christensen 和 Ribeiro Jr. 2002) 包内，后来，**geoRglm** 不维护，已从 CRAN 移除了，笔者从他们主页下载了数据。数据集 `rongelap` 记录了 157 个测量点的伽马射线强度，即在时间间隔 `time` （秒）内放射的粒子数目 `counts` (个)，测量点的横纵坐标分别为 `cX` （米）和 `cY` （米），下表格 28.1 展示部分朗格拉普岛核辐射检测数据及海岸线坐标数据。

坐标原点在岛的东北，下图 28.2a 右上角的位置。采样点的编号见下图 28.2b，基本上按照从下（南）到上（北），从左（西）到右（东）的顺序依次测量。

© 黄湘云

图 28.1: 朗格拉普环礁和朗格拉普岛

表格 28.1: 朗格拉普岛核辐射检测数据及海岸线坐标数据

(a) 核辐射检测数据

cX 横坐标	cY 纵坐标	counts 数目	time 时间
-6050	-3270	75	300
-6050	-3165	371	300
-5925	-3320	1931	300
-5925	-3165	4357	300
-5800	-3350	2114	300
-5800	-3165	2318	300

(b) 海岸线坐标数据

cX 横坐标	cY 纵坐标
-5509.236	-3577.438
-5544.821	-3582.250
-5561.604	-3576.926
-5580.780	-3574.535
-5599.687	-3564.288
-5605.922	-3560.910

© 黄湘云

(a) 采样分布

(b) 采样顺序

图 28.2: 采样点在岛上的分布

© 黄湘云

28.2 数据探索

朗格拉普岛呈月牙形，有数千米长，但仅几百米宽，十分狭长。采样点在岛上的分布如图 28.2 所示，主网格以约 200 米的间隔采样，在岛屿的东北和西南方各有两个密集采样区，每个网格采样区是 5×5 方式排列的，上下左右间隔均为 40 米。朗格拉普岛上各个检测站点的核辐射强度如图 28.3 所示，越亮表示核辐射越强，四个检测区的采样阵列非常密集，通过局部放大展示了最左侧的一个检测区，它将作为后续模型比较的参照区域。

图 28.3: 岛上各采样点的核辐射强度

ggplot2 包只能在二维平面上展示数据，对于空间数据，立体图形更加符合数据产生背景。如图 28.4 所示，以三维图形展示朗格拉普岛上采样点的位置及检测到的辐射强度。**lattice** 包的函数 `cloud()` 可以绘制三维的散点图，将自定义的面板函数 `panel.3dcoastline()` 传递给参数 `panel.3d.cloud` 绘制岛屿海岸线。组合点和线两种绘图元素构造出射线，线的长短表示放射性的强弱，以射线表示粒子辐射现象更加贴切。

28.3 数据建模

28.3.1 广义线性模型

核辐射是由放射元素衰变产生的，通常用单位时间释放出来的粒子数目表示辐射强度，因此，建立如下泊松型广义线性模型来拟合核辐射强度。

$$
\begin{aligned}
\log(\lambda_i) &= \beta \\
y_i &\sim \text{Poisson}(t_i \lambda_i)
\end{aligned}
$$

©黄湘云

图 28.4: 岛上各采样点的辐射强度

其中，λ_i 表示核辐射强度，β 表示未知的截距，y_i 表示观测到的粒子数目，t_i 表示相应的观测时间，$i = 1, \ldots, 157$ 表示采样点的位置编号。R 软件内置的 **stats** 包有函数 `glm()` 可以拟合上述广义线性模型，代码如下。

```
fit_rongelap_poisson <- glm(counts ~ 1,
  family = poisson(link = "log"), offset = log(time), data = rongelap
)
summary(fit_rongelap_poisson)
```

```
#>
#> Call:
#> glm(formula = counts ~ 1, family = poisson(link = "log"), data = rongelap,
#>     offset = log(time))
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept) 2.013954   0.001454    1385   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for poisson family taken to be 1)
#>
#>     Null deviance: 61567  on 156  degrees of freedom
```

©黄湘云

```
#> Residual deviance: 61567  on 156  degrees of freedom
#> AIC: 63089
#>
#> Number of Fisher Scoring iterations: 4
```

当 family = poisson(link = "log") 时，响应变量只能是正整数，所以不能放 counts / time。泊松广义线性模型是对辐射强度建模，辐射强度与位置 cX 和 cY 有关。当响应变量为放射出来的粒子数目 counts 时，为了表示辐射强度，需要设置参数 offset，表示与放射粒子数目对应的时间间隔 time。联系函数是对数函数，因此时间间隔需要取对数。

从辐射强度的拟合残差的空间分布图 28.5 不难看出，颜色深和颜色浅的点分别聚集在一起，且与周围点的颜色呈现层次变化，拟合残差存在明显的空间相关性。如果将位置变量 cX 和 cY 加入广义线性模型，也会达到统计意义上的显著。

```
rongelap$poisson_residuals <- residuals(fit_rongelap_poisson)
ggplot(rongelap, aes(x = cX, y = cY)) +
  geom_point(aes(colour = poisson_residuals / time), size = 0.2) +
  scale_color_viridis_c(option = "C") +
  theme_bw() +
  labs(x = "横坐标（米）", y = "纵坐标（米）", color = "残差")
```

图 28.5: 残差的空间分布

图 28.6 描述残差的分布，从图 28.6a 发现残差存在一定的线性趋势，岛屿的东南方，残差基本为正，而在岛屿的西北方，残差基本为负，说明有一定的异方差性。从图 28.6b 发现残差在水平方向上的分布像个哑铃，说明异方差现象明显。从图 28.6c 发现残差在垂直方向上的分布像棵松树，也说明异方差现象明显。

© 王湘黄

```
ggplot(rongelap, aes(x = 1:157, y = poisson_residuals / time)) +
  geom_point(size = 1) +
  theme_bw() +
  labs(x = "编号", y = "残差")

ggplot(rongelap, aes(x = cX, y = poisson_residuals / time)) +
  geom_point(size = 1) +
  theme_bw() +
  labs(x = "横坐标", y = "残差")

ggplot(rongelap, aes(x = cY, y = poisson_residuals / time)) +
  geom_point(size = 1) +
  theme_bw() +
  labs(x = "纵坐标", y = "残差")
```

(a) 残差与编号的关系

(b) 残差与横坐标的关系

(c) 残差与纵坐标的关系

图 28.6: 残差分布图

© 黄湘云

28.3.2 空间线性混合效应模型

从实际场景出发，也不难理解，位置信息是非常关键的。进一步，充分利用位置信息，精细建模是很有必要的。相邻位置的核辐射强度是相关的，离得近的比离得远的更相关。下面对辐射强度建模，假定随机效应之间存在相关性结构，去掉随机效应相互独立的假设，这更符合位置效应存在相互影响的实际情况。

$$\log\big(\lambda(x_i)\big) = \beta + S(x_i) + Z_i \tag{28.1}$$

其中，β 表示截距，相当于平均水平，$\lambda(x_i)$ 表示位置 x_i 处的辐射强度，$S(x_i)$ 表示位置 x_i 处的空间效应，$S(x), x \in \mathcal{D} \subset \mathbb{R}^2$ 是二维平稳空间高斯过程 $\mathcal{S}$ 的具体实现。$\mathcal{D}$ 表示研究区域，可以理解为朗格拉普岛，它是二维实平面 $\mathbb{R}^2$ 的子集。Z_i 之间相互独立同正态分布 $\mathcal{N}(0,\tau^2)$ ，Z_i 表示非空间的随机效应，在空间统计中，常称之为块金效应，可以理解为测量误差、空间变差或背景辐射。值得注意，此时，块金效应和模型残差是合并在一起的。

28.3.2.1 自协方差函数

随机过程 $S(x)$ 的自协方差函数常用的有指数型、幂二次指数型（高斯型）和梅隆型，形式如下：

$$\begin{aligned}
\mathsf{Cov}\{S(x_i),S(x_j)\} &= \sigma^2 \exp\big(-\frac{\|x_i-x_j\|_2}{\phi}\big)\\
\mathsf{Cov}\{S(x_i),S(x_j)\} &= \sigma^2 \exp\big(-\frac{\|x_i-x_j\|_2^2}{2\phi^2}\big)\\
\mathsf{Cov}\{S(x_i),S(x_j)\} &= \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}\frac{\|x_i-x_j\|_2}{\phi}\right)^{\nu} K_\nu\left(\sqrt{2\nu}\frac{\|x_i-x_j\|_2}{\phi}\right)\\
K_\nu(x) &= \int_0^\infty \exp(-x\cosh t)\cosh(\nu t)\mathrm{dt}
\end{aligned} \tag{28.2}$$

其中，K_ν 表示阶数为 ν 的修正的第二类贝塞尔函数，$\Gamma(\cdot)$ 表示伽马函数，当 $\nu = 1/2$ ，梅隆型将简化为指数型，当 $\nu = \infty$ 时，梅隆型将简化为幂二次指数型。

$$\mathsf{Cov}\{S(x_i),S(x_j)\} = \sigma^2\rho(u_{ij})$$

其中，$\rho(u_{ij})$ 表示自相关函数。u_{ij} 表示位置 x_i 与 x_j 之间的距离，常用的有欧氏距离。梅隆型自相关函数图像如图 28.7 所示，不难看出，ν 影响自相关函数的平滑性，控制点与点之间相关性的变化，ν 越大相关性越迅速地递减。ϕ 控制自相关函数的范围，ϕ 越大相关性辐射距离越远。对模型来说，它们都是超参数。

28.3.2.2 nlme 包的自相关函数

nlme 包中带块金效应的指数型自相关函数设定如下：

©黄湘云

图 28.7: 梅隆型自相关函数曲线

$$\rho(u;\phi,\tau_{rel}^2) = \tau_{rel}^2 + (1-\tau_{rel}^2)(1-\exp(-\frac{u}{\phi}))$$

为了方便参数估计，**nlme** 包对参数做了一些重参数化的操作。

$$\begin{aligned} \tau_{rel}^2 &= \frac{\tau^2}{\tau^2+\sigma^2} \\ \sigma_{tol}^2 &= \tau^2+\sigma^2 \end{aligned} \tag{28.3}$$

当 u 趋于 0 时，$\rho(u;\phi,\tau_{rel}^2) = \tau_{rel}^2$ 。另外，ϕ 取值为正，τ_{rel}^2 取值介于 0-1 之间，在默认设置下，ϕ 的初始值为 $0.1 \times \max_{i,j\in A} u_{ij}$，即所有点之间距离的最大值的 10%，$\tau_{rel}^2$ 为 0.1 ，这只是作为参考，用户可根据实际情况调整。

下面以一个简单示例理解自相关函数 `corExp()` 的作用，令 $\phi = 1.2, \tau_{rel}^2 = 0.2$，则由距离矩阵和自相关函数构造的自相关矩阵如下：

```
library(nlme)
spatDat <- data.frame(x = (1:4) / 4, y = (1:4) / 4)
cs3Exp <- corExp(c(1.2, 0.2), form = ~ x + y, nugget = TRUE)
cs3Exp <- Initialize(cs3Exp, spatDat)
corMatrix(cs3Exp)
```

```
#>           [,1]      [,2]      [,3]      [,4]
```

©黄湘云

```
#> [1,] 1.0000000 0.595847 0.443792 0.3305402
#> [2,] 0.5958470 1.000000 0.595847 0.4437920
#> [3,] 0.4437920 0.595847 1.000000 0.5958470
#> [4,] 0.3305402 0.443792 0.595847 1.0000000
```

自相关矩阵的初始化结果等价于如下矩阵：

```
diag(0.2, 4) + (1 - 0.2) * exp(-as.matrix(dist(spatDat)) / 1.2)
```

```
#>           1        2        3         4
#> 1 1.0000000 0.595847 0.443792 0.3305402
#> 2 0.5958470 1.000000 0.595847 0.4437920
#> 3 0.4437920 0.595847 1.000000 0.5958470
#> 4 0.3305402 0.443792 0.595847 1.0000000
```

除了函数 `corExp()`，**nlme** 包还有好些自相关函数，如高斯自相关函数 `corGaus()`，线性自相关函数 `corLin()`，有理自相关函数 `corRatio()`，球型自相关函数 `corSpher()` 等。它们的作用与函数 `corExp()` 类似，使用方式也一样，如下是高斯型自相关函数的示例，其他的不再一一举例。

```
cs3Gaus <- corGaus(c(1.2, 0.2), form = ~ x + y, nugget = TRUE)
cs3Gaus <- Initialize(cs3Gaus, spatDat)
corMatrix(cs3Gaus)
```

```
#>           [,1]      [,2]      [,3]      [,4]
#> [1,] 1.0000000 0.7334843 0.5653186 0.3662667
#> [2,] 0.7334843 1.0000000 0.7334843 0.5653186
#> [3,] 0.5653186 0.7334843 1.0000000 0.7334843
#> [4,] 0.3662667 0.5653186 0.7334843 1.0000000
```

```
# 等价于
diag(0.2, 4) + (1 - 0.2) * exp(-as.matrix(dist(spatDat))^2 / 1.2^2)
```

```
#>           1         2         3         4
#> 1 1.0000000 0.7334843 0.5653186 0.3662667
#> 2 0.7334843 1.0000000 0.7334843 0.5653186
#> 3 0.5653186 0.7334843 1.0000000 0.7334843
#> 4 0.3662667 0.5653186 0.7334843 1.0000000
```

28.3.2.3 nlme 包的拟合函数 gls()

nlme 包的函数 `gls()` 实现限制极大似然估计方法，可以拟合存在异方差的一般线性模型。所谓一般线性模型，即在简单线性模型的基础上，残差不再是独立同分布的，而是存在相关性。函数 `gls()` 可以拟合具有空间自相关性的残差结构。这种线性模型又可以看作是一种带空间自相关结构的线性混合效应模型，空间随机效应的结构可以看作异方差的结构。

```r
fit_rongelap_gls <- gls(
  log(counts / time) ~ 1, data = rongelap,
  correlation = corExp(
    value = c(200, 0.1), form = ~ cX + cY, nugget = TRUE
  )
)
summary(fit_rongelap_gls)
```

```
#> Generalized least squares fit by REML
#>   Model: log(counts/time) ~ 1
#>   Data: rongelap
#>        AIC      BIC    logLik
#>   184.4451 196.6446 -88.22257
#>
#> Correlation Structure: Exponential spatial correlation
#>  Formula: ~cX + cY
#>  Parameter estimate(s):
#>       range      nugget
#> 169.7472087   0.1092496
#>
#> Coefficients:
#>                Value Std.Error  t-value p-value
#> (Intercept) 1.812914 0.1088037 16.66224       0
#>
#> Standardized residuals:
#>         Min          Q1         Med          Q3         Max
#> -5.57385199 -0.06909454  0.34610011  0.73852188  1.57152087
#>
#> Residual standard error: 0.5739672
#> Degrees of freedom: 157 total; 156 residual
```

nlme 包给出截距项 β 、相对块金效应 τ^2_{rel} 、范围参数 ϕ 和残差标准差 σ_{tol} 的估计，

$$\beta = 1.812914, \quad \phi = 169.7472088$$
$$\tau^2_{rel} = 0.1092496, \quad \sigma_{tol} = 0.5739672$$

根据前面的方程式 28.3 ，可以得到 τ^2 和 σ^2 的估计。

$$\tau^2 = \tau^2_{rel} \times \sigma^2_{tol} = 0.1092496 \times 0.3294383 = 0.035991$$
$$\sigma^2 = \sigma^2_{tol} - \tau^2_{rel} \times \sigma^2_{tol} = 0.5739672^2 - 0.1092496 \times 0.3294383 = 0.2934473$$

© 王建平

28.3.2.4 经验半变差函数图

接下来用经验半变差函数图检查空间相关性。为方便表述起见，令 $T(x_i)$ 代表方程式 28.1 等号右侧的部分，即表示线性预测（Linear Predictor）。

$$T(x_i) = \beta + S(x_i) + Z_i$$

令 $\gamma(u_{ij}) = \frac{1}{2}\mathsf{Var}\{T(x_i) - T(x_j)\}$ 表示半变差函数（Semivariogram），这里 u_{ij} 表示采样点 x_i 与 x_j 之间的距离。考虑到

$$\gamma(u_{ij}) = \frac{1}{2}\mathsf{E}\{\left[T(x_i) - T(x_j)\right]^2\} = \tau^2 + \sigma^2(1 - \rho(u_{ij})) \tag{28.4}$$

上式第一个等号右侧期望可以用样本代入来计算，称之为经验半变差函数，第二个等号右侧为理论半变差函数。为了便于计算，将距离做一定划分，尽量使得各个距离区间的样本点对的数目接近。此时，第 i 个距离区间上经验半变差函数值 $\hat{\gamma}(h_i)$ 的计算公式如下：

$$\hat{\gamma}(h_i) = \frac{1}{2N(h_i)} \sum_{j=1}^{N(h_i)} (T(x_i) - T(x_i + h'))^2, \quad h_{i,0} \le h' < h_{i,1}$$

其中，$[h_{i,0}, h_{i,1}]$ 表示第 i 个距离区间，$N(h_i)$ 表示第 i 个距离区间内所有样本点对的数目，只要两个点之间的距离在这个区间内，就算是一对。`rongelap` 数据集包含 157 个采样点，两两配对，共有 $(157-1) \times 157/2 = 12246$ 对。下面举个例子说明函数 `Variogram()` 的作用。假设模型参数已经估计出来了，可以根据理论变差公式方程式 28.4 计算，设置为 $\phi = 200, \tau^2_{rel} = 0.1$ 。

```
0.1  + (1 - 0.1) * (1 - exp(- 40 / 200 ))
```

```
#> [1] 0.2631423
```

可知当距离为 40 时，半变差函数值为 0.2631423 ，当距离为 175.9570 时，半变差函数值为 0.6266151 。下面基于 **nlme** 包中自相关函数计算半变差函数值，将 rongelap 数据代入函数 `Variogram()` 可以计算每个距离对应的函数值，默认计算 50 个，如图 28.8 所示。

```
cs <- corExp(value = c(200, 0.1), form = ~ cX + cY, nugget = TRUE)
cs <- Initialize(cs, rongelap)
vario <- Variogram(cs)
head(vario)
```

```
#>      variog     dist
#> 1 0.2631423  40.0000
#> 2 0.6266152 175.9570
#> 3 0.8107963 311.9141
#> 4 0.9041256 447.8711
#> 5 0.9514180 583.8282
```

©黄湘云

```
#> 6 0.9753822 719.7852
```

可以看到，当距离为 40 时，计算的结果与上面是一致的，也知道了函数 `Variogram()` 的作用。

图 28.8: 理论半变差函数图

nlme 包的函数 `Variogram()` 根据函数 `gls()` 估计的参数值计算模型残差的经验半变差函数值：

```
fit_rongelap_vario <- Variogram(fit_rongelap_gls,
  form = ~ cX + cY, data = rongelap, resType = "response"
)
fit_rongelap_vario
```

```
#>        variog       dist n.pairs
#> 1  0.07006716   89.44272     510
#> 2  0.12719889  144.22205     601
#> 3  0.17289246  252.98221     581
#> 4  0.22384959  368.78178     622
#> 5  0.26006395  443.84682     592
#> 6  0.20694239  521.53619     616
#> 7  0.41861866  718.05292     611
#> 8  0.30180842 1028.20230     610
#> 9  0.16717617 1493.73690     612
#> 10 0.14683508 2150.08137     612
#> 11 0.15149089 2993.10009     612
#> 12 0.21048419 3896.79355     613
#> 13 0.21372840 4600.21330     618
```

© 黄湘云


```
#> 14 0.17302418 4889.68302     607
#> 15 0.20205496 5065.98460     618
#> 16 0.18620361 5195.76751     623
#> 17 0.18613578 5293.99660     596
#> 18 0.20560623 5451.82538     616
#> 19 0.46457193 5604.41790     608
#> 20 0.55063433 5979.95802     612
```

注释

请思考 `fit_rongelap_vario` 输出的 `n.pairs` 的总对数为什么是 12090 而不是 12246?

结果显示，距离在 0-89.44272 米之间的坐标点有 510 对，经验半变差函数值为 0.07006716。距离在 89.44272-144.22205 米之间的坐标点有 601 对，经验半变差函数值为 0.12719889，依此类推。将距离和计算的经验半变差函数值绘制出来，即得到经验半变差图，如图 28.9 所示。刚开始，半变差值很小，之后随距离增加而增大，一直到达一个平台。半变差反比于空间相关性的程度，随着距离增加，空间相关性减弱。这说明数据中确含有空间相关性，模型中添加指数型自相关空间结构是合理的。

图 28.9: 残差的经验半变差图

如果空间相关性提取得很充分，则标准化残差的半变差图中的数据点应是围绕标准差 1 上下波动，无明显趋势，拟合线几乎是一条水平线，从图 28.10 来看，存在一些非均匀的波动，是采样点在空间的分布不均匀所致，岛屿狭长的中部地带采样点稀疏。如前所述，刻画空间相关性，除了指数型，还可以用其它自相关结构来拟合，留待读者练习。

©黄湘云

图 28.10: 标准化残差的经验半变差图

28.3.3 空间广义线性混合效应模型

简单的广义线性模型并没有考虑距离相关性，它认为各个观测点的数据是相互独立的。因此，考虑采用广义线性混合效应模型，在广义线性模型的基础上添加位置相关的随机效应，用以刻画未能直接观测到的潜在影响。^{137}Cs 放出伽马射线，在 $n = 157$ 个采样点，分别以时间间隔 t_i 测量辐射量 $y(x_i)$，建立泊松型空间广义线性混合效应模型。

$$\begin{aligned} \log\{\lambda(x_i)\} &= \beta + S(x_i) + Z_i \\ y(x_i) &\sim \text{Poisson}\big(t_i\lambda(x_i)\big) \end{aligned} \tag{28.5}$$

模型中，放射粒子数 $y(x_i)$ 作为响应变量服从均值为 $t_i\lambda(x_i)$ 的泊松分布，其它模型成分的说明同前。简单起见，下面不添加块金效应，即。掉模型中的 Z_i 。此时，块金效应对模型预测效果的提升很有限，由于 τ^2 和 σ^2 之间存在的可识别性问题，会显著增加参数估计的复杂度。

nlme 包不能拟合空间广义线性混合效应模型，**spaMM** 包可以，它的使用语法与前面介绍的函数 `glm()` 、**nlme** 包都类似，函数 `fitme()` 可以拟合从线性模型到广义线性混合效应模型的一大类模型，且使用统一的语法，输出一个 `HLfit` 类型的数据对象。**spaMM** 包的函数 `Matern()` 实现了梅隆型自协方差函数，指数型和幂二次指数型是它的特例。当固定 $\nu = 0.5$ 时，梅隆型自协方差函数 `Matern()` 的形式退化为 $\sigma^2\exp(-\alpha u)$ ，其中，α 与范围参数关联，相当于前面出现的 $1/\phi$ 。

```
library(spaMM)
fit_rongelap_spamm <- fitme(
  formula = counts ~ 1 + Matern(1 | cX + cY) + offset(log(time)),
  family = poisson(link = "log"), data = rongelap,
```

©黄湘云


```
  fixed = list(nu = 0.5), method = "REML"
)
summary(fit_rongelap_spamm)
```

```
#> formula: counts ~ 1 + Matern(1 | cX + cY) + offset(log(time))
#> Estimation of corrPars and lambda by REML (p_bv approximation of restricted logL).
#> Estimation of fixed effects by ML (p_v approximation of logL).
#> Estimation of lambda by 'outer' REML, maximizing restricted logL.
#> family: poisson( link = log )
#>  ------------ Fixed effects (beta) ------------
#>             Estimate Cond. SE t-value
#> (Intercept)    1.829  0.08797   20.78
#>  --------------- Random effects ---------------
#> Family: gaussian( link = identity )
#>                     --- Correlation parameters:
#>        1.nu       1.rho
#> 0.500000000 0.009211761
#>            --- Variance parameters ('lambda'):
#> lambda = var(u) for u ~ Gaussian;
#>    cX + cY  :  0.3069
#> # of obs: 157; # of groups: cX + cY, 157
#>  ------------- Likelihood values  -------------
#>                         logLik
#> logL       (p_v(h)): -1318.010
#> Re.logL  (p_b,v(h)): -1319.522
```

从输出结果来看，模型固定效应的截距项 β 为 `1.829`，空间随机效应的方差 σ^2 为 `0.3069`，对比函数 `Matern()` 实现的指数型自协方差函数公式与方程式 28.2 ，将输出结果转化一下，则 $\phi = 1/0.00921 = 108.57$ ，表示在这个模型的设定下，空间相关性的最大影响距离约为 108.5 米。

28.4 模型预测

接下来，预测给定的边界（海岸线）内任意位置的核辐射强度，展示全岛的核辐射强度分布。先从点构造多边形数据，再将多边形做网格划分，继而将网格中心点作为模型输入获得核辐射强度的预测值。

28.4.1 海岸线数据

海岸线上取一些点，点的数量越多，对海岸线的刻画越精确，这在转弯处体现得非常明显。海岸线的数据是以成对的坐标构成，导入 R 语言中，是以数据框的形式存储，为了方便后续的操作，引入空间数据操作的 **sf** 包 (Pebesma 2018)，将核辐射数据和海岸线数据转化为 POINT 类型的空间点数据。

©南京大学出版社

```
library(sf)
rongelap_sf <- st_as_sf(rongelap, coords = c("cX", "cY"), dim = "XY")
rongelap_coastline_sf <- st_as_sf(rongelap_coastline, coords = c("cX", "cY"), dim = "XY")
```

sf 包提供了大量操作空间数据的函数，比如函数 `st_bbox()` 计算一组空间数据的矩形边界，获得左下和右上两个点的坐标 (xmin,ymin) 和 (xmax,ymax)，下面还会陆续涉及其它空间数据操作。

```
st_bbox(rongelap_coastline_sf)
```

```
#>        xmin        ymin        xmax        ymax
#> -6299.31201 -3582.25000    20.37916   103.54140
```

`rongelap_coastline_sf` 数据集是朗格拉普岛海岸线的采样点坐标，是一个 POINT 类型的数据，为了以海岸线为边界生成规则网格，首先连接点 POINT 构造多边形 POLYGON 对象。POINT 和 POLYGON 是 **sf** 包内建的基础的几何类型，其它复杂的空间类型是由它们衍生而来。函数 `st_geometry` 提取空间点数据中的几何元素，再用函数 `st_combine` 将点组合起来，最后用函数 `st_cast` 转换成 POLYGON 多边形类型。

```
rongelap_coastline_sfp <- st_cast(st_combine(st_geometry(rongelap_coastline_sf)), "POLYGON")
```

图 28.11 上下两个子图分别展示空间点集和多边形。上图是原始的采样点数据，下图是以点带线，串联 POINT 数据构造 POLYGON 数据后的多边形。后续的数据操作将围绕这个多边形展开。

```
# 点集
ggplot(rongelap_coastline_sf) +
  geom_sf(size = 0.5) +
  theme_void()
# 多边形
ggplot(rongelap_coastline_sfp) +
  geom_sf(fill = "white", linewidth = 0.5) +
  theme_void()
```

28.4.2 边界处理

为了确保覆盖整个岛，处理好边界问题，需要一点缓冲空间，就是说在给定的边界线外围再延伸一段距离，构造一个更大的多边形，这可以用函数 `st_buffer()` 实现，根据海岸线构造缓冲区，得到一个 POLYGON 类型的几何数据对象。考虑到朗格拉普岛的实际大小，缓冲距离选择 50 米。

```
rongelap_coastline_buffer <- st_buffer(rongelap_coastline_sfp, dist = 50)
```

缓冲区构造出来的效果如图 28.12 所示，为了便于与海岸线对比，图中将采样点、海岸线和缓冲区都展示出来了。

```
ggplot() +
  geom_sf(data = rongelap_sf, size = 0.2) +
  geom_sf(data = rongelap_coastline_sfp, fill = NA, color = "gray30") +
```

©黄湘云

(a) 点数据

(b) 多边形数据

图 28.11: 朗格拉普岛海岸线的表示

© 基础统计


```
  geom_sf(data = rongelap_coastline_buffer, fill = NA, color = "black") +
  theme_void()
```

图 28.12: 朗格拉普岛海岸线及其缓冲区

28.4.3 构造网格

接下来，利用函数 st_make_grid() 根据朗格拉普岛海岸缓冲线构造网格，朗格拉普岛是狭长的，因此，网格是 75×150 的，意味着水平方向 75 行，垂直方向 150 列。网格的疏密程度是可以调整的，网格越密，格点越多，核辐射强度分布越精确，计算也越耗时。

```
# 构造带边界约束的网格
rongelap_coastline_grid <- st_make_grid(rongelap_coastline_buffer, n = c(150, 75))
```

函数 st_make_grid() 根据 rongelap_coastline_buffer 的矩形边界网格化，效果如图 38.7 所示，依次添加了网格、海岸线和缓冲区。实际上，网格只需要覆盖朗格拉普岛即可，岛外的部分是大海，不需要覆盖，根据现有数据和模型对岛外区域预测核辐射强度也没有意义，因此，在后续的操作中，岛外的网格都要去掉。函数 st_make_grid() 除了支持方形网格划分，还支持六边形网格划分。

```
ggplot() +
  geom_sf(data = rongelap_coastline_grid, fill = NA, color = "gray") +
  geom_sf(data = rongelap_coastline_sfp, fill = NA, color = "gray30") +
  geom_sf(data = rongelap_coastline_buffer, fill = NA, color = "black") +
  theme_void()
```

© 黄湘云

图 28.13: 朗格拉普岛规则化网格操作

接下来，调用 **sf** 包函数 `st_intersects()` 将小网格落在缓冲区和岛内的筛选出来，一共 1612 个小网格，再用函数 `st_centroid()` 计算这些网格的中心点坐标。函数 `st_intersects()` 的作用是对多边形和网格取交集，包含与边界线交叉的网格，默认返回值是一个稀疏矩阵，与索引函数 `[.sf`（这是 **sf** 包扩展 `[` 函数的一个例子）搭配可以非常方便地过滤出目标网格。与之相关的函数 `st_crosses()` 可以获得与边界线交叉的网格。

```
# 将 sfc 类型转化为 sf 类型
rongelap_coastline_grid <- st_as_sf(rongelap_coastline_grid)
rongelap_coastline_buffer <- st_as_sf(rongelap_coastline_buffer)
rongelap_grid <- rongelap_coastline_grid[rongelap_coastline_buffer, op = st_intersects]
# 计算网格中心点坐标
rongelap_grid_centroid <- st_centroid(rongelap_grid)
```

过滤出来的网格如图 38.7 所示，全岛网格化后，图中将朗格拉普岛海岸线、网格都展示出来了。网格的中心点将作为新的坐标数据，后续要在这些新的坐标点上预测核辐射强度。

```
ggplot() +
  geom_sf(data = rongelap_coastline_sfp,
          fill = NA, color = "gray30", linewidth = 0.5) +
  geom_sf(data = rongelap_grid, fill = NA, color = "gray30") +
  theme_void()
```

©黄湘云

图 28.14: 朗格拉普岛规则网格划分结果

28.4.4 整理数据

函数 st_coordinates() 抽取网格中心点的坐标并用函数 as.data.frame() 转化为数据框类型，新数据的列名需要和训练数据保持一致，最后补充漂移项 time，以便输入模型中。漂移项并不影响核辐射强度，指定为 300 或 400 都可以。

```
rongelap_grid_df <- as.data.frame(st_coordinates(rongelap_grid_centroid))
colnames(rongelap_grid_df) <- c("cX", "cY")
rongelap_grid_df$time <- 1
```

将数据输入 **spaMM** 包拟合的模型对象 fit_rongelap_spamm，并将模型返回的结果整理成数据框，再与采样点数据合并。predict() 是一个泛型函数，**spaMM** 包为模型对象提供了相应的预测方法。

```
# 预测值
rongelap_grid_pred <- predict(fit_rongelap_spamm,
  newdata = rongelap_grid_df, type = "response"
)
rongelap_grid_df$pred_sp <- as.vector(rongelap_grid_pred)
# 线性预测的方差
rongelap_grid_var <- get_predVar(fit_rongelap_spamm,
  newdata = rongelap_grid_df, variances = list(predVar = TRUE), which = "predVar"
)
```

© 基础统计


```
#> Non-identity link: predVar is on linear-predictor scale.
```

```
rongelap_grid_df$var_sp <- as.vector(rongelap_grid_var)
```

在空间线性混合效应模型一节，截距 β ，方差 σ^2 ，块金效应 τ^2 和范围参数 ϕ 都估计出来了。在此基础上，采用简单克里金插值方法预测，对于未采样观测的位置 x_0，它的辐射强度的预测值 $\hat{\lambda}(x_0)$ 及其预测方差 $\mathsf{Var}\{\hat{\lambda}(x_0)\}$ 的计算公式如下。

$$\hat{\lambda}(x_0) = \beta + \boldsymbol{u}^\top (V + \tau^2 I)^{-1} (\boldsymbol{\lambda} - \mathbf{1}\beta)$$
$$\mathsf{Var}\{\hat{\lambda}(x_0)\} = \sigma^2 - \boldsymbol{u}^\top (V + \tau^2 I)^{-1} \boldsymbol{u}$$

其中，协方差矩阵 V 中第 i 行第 j 列的元素为 $\mathsf{Cov}\{S(x_i), S(x_j)\}$ ，列向量 $\boldsymbol{u}$ 的第 i 个元素为 $\mathsf{Cov}\{S(x_i), S(x_0)\}$ 。

```
# 截距
beta <- 1.812914
# 范围参数
phi <- 169.7472088
# 方差
sigma_sq <- 0.2934473
# 块金效应
tau_sq <- 0.035991
# 自协方差函数
cov_fun <- function(h) sigma_sq * exp(-h / phi)
# 观测距离矩阵
m_obs <- cov_fun(st_distance(x = rongelap_sf)) + diag(tau_sq, 157)
# 预测距离矩阵
m_pred <- cov_fun(st_distance(x = rongelap_sf, y = rongelap_grid_centroid))
# 简单克里金插值 Simple Kriging
mean_sk <- beta + t(m_pred) %*% solve(m_obs, log(rongelap_sf$counts / rongelap_sf$time) - beta)
# 辐射强度预测值
rongelap_grid_df$pred_sk <- exp(mean_sk)
# 辐射强度预测方差
rongelap_grid_df$var_sk <- sigma_sq - diag(t(m_pred) %*% solve(m_obs, m_pred))
```

28.4.5 展示结果

将预测结果以散点图的形式呈现到图上，见下图 28.15 ，由于散点非常多，紧挨在一起就连成片了。上子图是 **nlme** 包预测的结果，下子图是 **spaMM** 包预测的结果，前者图像看起来会稍微平滑一些。

从空间线性混合效应模型到空间广义线性混合效应模型的效果提升不多，差异不太明显。下图 28.16 展示核辐射强度预测方差的分布。越简单的模型，预测值的分布越平滑，越复杂的模型，捕捉到更多局部

© 黄湘云

图 28.15: 朗格拉普岛核辐射强度的分布

© 黄湘云

细节，因而，预测值的分布越曲折。

图 28.16: 核辐射强度预测方差的分布

考虑到核辐射在全岛的分布应当是连续性的，空间连续性也是这类模型的假设，接下来绘制热力图，先用 **stars** 包 (Pebesma 2022) 将预测数据按原网格化的精度转化成栅格对象，裁减超出朗格拉普岛海岸线以外的内容。

```
library(abind)
library(stars)
rongelap_grid_sf <- st_as_sf(rongelap_grid_df, coords = c("cX", "cY"), dim = "XY")
rongelap_grid_stars <- st_rasterize(rongelap_grid_sf, nx = 150, ny = 75)
rongelap_stars <- st_crop(x = rongelap_grid_stars, y = rongelap_coastline_sfp)
```

除了矢量栅格化函数 `st_rasterize()` 和栅格剪裁函数 `st_crop()` ，**stars** 包还提供栅格数据图层

© 黄湘云

`geom_stars()`，这可以和 **ggplot2** 内置的图层搭配使用。下图 28.17 是 **ggplot2** 包和 **grid** 包一起绘制的辐射强度的热力分布图，展示 **spaMM** 包的预测效果。图左侧一小一大两个虚线框是放大前后的部分区域，展示朗格拉普岛核辐射强度的局部变化。

图 28.17: 朗格拉普岛核辐射强度的分布

美国当年是在比基尼环礁做的氢弹核试验，试验地与朗格拉普岛相距 100 多英里。核辐射羽流受大气、海洋环流等影响，漂流到朗格拉普岛。又受朗格拉普岛周围水文、地理环境影响，核辐射强度在全岛的分布是不均匀的，图中越亮的地方表示受到的核辐射越严重。

©黄湘云

第二十九章　空间区域数据分析

29.1　苏格兰唇癌数据分析

> Everything is related to everything else, but near things are more related than distant things.
>
> — Waldo Tobler (Tobler 1970)

空间区域数据分析

空间区域数据的贝叶斯建模

- Bayesian spatial and spatio-temporal GLMMs with possible extremes glmmfields
- Bayesian spatial analysis geostan
- Spatial Models in Stan: Intrinsic Auto-Regressive Models for Areal Data
- Exact sparse CAR models in Stan 网页文档
- Spatial Models in Stan: Intrinsic Auto-Regressive Models for Areal Data 网页文档 原始数据和代码，接上面苏格兰唇癌数据分析，用 CmdStanR 更新后的代码
- Spatial modeling of areal data. Lip cancer in Scotland INLA 建模
- CAR models Scotland Lip cancer dataset Stan 建模
- 空间计量 区域数据分析 on-the-use-of-r-for-spatial-econometrics

响应变量服从泊松分布

- BYM-INLA (Blangiardo 等 2013; Moraga 2020)
- BYM-Stan (Morris 等 2019; Donegan 2022; Cabral, Bolin, 和 Rue 2022)

记录 1975-1986 年苏格兰 56 个地区的唇癌病例数，这是一个按地区汇总的数据。

```
library(sf)
```

```
Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

```
scotlips <- st_read('data/scotland/scotland.shp', crs = st_crs("EPSG:27700"))
```

```
Reading layer `scotland' from data source
    `/Users/runner/work/data-analysis-in-action/data-analysis-in-action/data/scotland/
scotland.shp'
```

```
  using driver `ESRI Shapefile'
Simple feature collection with 56 features and 9 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 7150.759 ymin: 529557.2 xmax: 468393.4 ymax: 1218479
Projected CRS: OSGB36 / British National Grid
```

```r
str(scotlips)
```

```
Classes 'sf' and 'data.frame':  56 obs. of  10 variables:
 $ SP_ID    : chr  "12" "13" "19" "02" ...
 $ NAME     : chr  "Sutherland" "Nairn" "Inverness" "Banff-Buchan" ...
 $ ID       : num  12 13 19 2 17 16 21 50 15 25 ...
 $ District : int  12 13 19 2 17 16 21 50 15 25 ...
 $ Observed : int  5 3 9 39 2 9 16 6 17 19 ...
 $ Expected : num  1.8 1.1 5.5 8.7 1.1 4.6 10.5 19.6 7.8 15.5 ...
 $ pcaff    : int  16 10 7 16 10 16 7 1 7 1 ...
 $ Latitude : num  58.1 57.5 57.2 57.6 57.1 ...
 $ Longitude: num  4.64 3.98 4.73 2.36 4.09 3 2.98 3.2 3.1 3.3 ...
 $ geometry :sfc_MULTIPOLYGON of length 56; first list element: List of 1
  ..$ :List of 1
  .. ..$ : num [1:73, 1:2] 254302 254442 253074 245057 259217 ...
  ..- attr(*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"
 - attr(*, "sf_column")= chr "geometry"
 - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA
  ..- attr(*, "names")= chr [1:9] "SP_ID" "NAME" "ID" "District" ...
```

```r
library(ggplot2)
ggplot() +
  geom_sf(data = scotlips, aes(fill = Observed)) +
  scale_fill_viridis_c() +
  theme_minimal()
```

© 黄湘云

图 29.1: 苏格兰各地区唇癌病例数分布

29.2 美国各州犯罪率分析

响应变量服从高斯分布的调查数据 (Bivand 2001)

数据集 USArrests 记录 1973 年美国各州每 10 万居民中因谋杀 Murder、袭击 Assault 和强奸 Rape 被警察逮捕的人数以及城市人口所占百分比（可以看作城市化率）。

表格 29.1: 数据集 USArrests（部分）

州名	区域划分	谋杀犯	袭击犯	城市化率	强奸犯
Alabama	South	13.2	236	58	21.2
Alaska	West	10.0	263	48	44.5
Arizona	West	8.1	294	80	31.0
Arkansas	South	8.8	190	50	19.5
California	West	9.0	276	91	40.6
Colorado	West	7.9	204	78	38.7

©黄湘云

表格 29.1: 数据集 USArrests（部分）

州名	区域划分	谋杀犯	袭击犯	城市化率	强奸犯

```
library(sf)
# 州数据
us_state_sf <- readRDS("data/us-state-map-2010.rds")
# 观测数据
us_state_df <- merge(x = us_state_sf, y = us_arrests,
  by.x = "NAME", by.y = "state_name", all.x = TRUE)

ggplot() +
  geom_sf(
    data = us_state_df, aes(fill = Assault), color = "gray80", lwd = 0.25) +
  scale_fill_viridis_c(option = "plasma", na.value = "white") +
  theme_void()
```

图 29.2: 因袭击被逮捕的人数分布

1973 年美国各州因袭击被逮捕的人数与城市化率的关系：相关分析

阿拉斯加州和夏威夷州与其它州都不相连，属于孤立的情况，下面在空间相关性的分析中排除这两个州。

```
# 州的中心
```

©黄湘云

图 29.3: 逮捕人数比例与城市化率的关系

```r
centers48 <- subset(
  x = data.frame(x = state.center$x, y = state.center$y),
  subset = !state.name %in% c("Alaska", "Hawaii")
)
# 观测数据
arrests48 <- subset(
  x = USArrests,
  subset = !rownames(USArrests) %in% c("Alaska", "Hawaii")
)
library(spData)
library(spdep)
# KNN
k4.48 <- knn2nb(knearneigh(as.matrix(centers48), k = 4))
# Moran I test
moran.test(x = arrests48$Assault, listw = nb2listw(k4.48))
```

```
    Moran I test under randomisation

data:  arrests48$Assault
weights: nb2listw(k4.48)

Moran I statistic standard deviate = 3.4216, p-value = 0.0003113
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
      0.294385644      -0.021276596       0.008511253
```

```r
# Permutation test for Moran's I statistic
moran.mc(x = arrests48$Assault, listw = nb2listw(k4.48), nsim = 499)
```

```
    Monte-Carlo simulation of Moran I

data:  arrests48$Assault
weights: nb2listw(k4.48)
number of simulations + 1: 500

statistic = 0.29439, observed rank = 500, p-value = 0.002
alternative hypothesis: greater
```

© 黄湘云

第六部分

优化建模

© 黄湘云

第三十章　统计计算

每一个统计模型的背后都有一个优化问题，统计计算的任务就是求解优化问题。

30.1　回归问题与优化问题

1996 年出现 Lasso（Least Absolute Selection and Shrinkage Operator，简称 Lasso）(Tibshirani 1996)，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，简称 LAR）算法 (Efron 等 2004) 的出现有力促进了 Lasso 在高维小样本数据中的应用。为了解决 Lasso 的有偏估计问题，自适应 Lasso、松弛 Lasso，SCAD （Smoothly Clipped Absolute Deviation，简称 SCAD）(Y. Kim, Choi, 和 Oh 2008)，MCP (Minimax Concave Penalty，简称 MCP) (Zhang 2010) 陆续出现。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。具体地，一个带 L1 正则项的线性回归模型，其对应的优化问题如下：

$$\arg\min_{\boldsymbol{\beta},\lambda} \quad \frac{1}{2}||\boldsymbol{y} - X\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1$$

其中，$X \in \mathbb{R}^{n\times k}$，$\boldsymbol{y} \in \mathbb{R}^n$，$\boldsymbol{\beta} \in \mathbb{R}^k$，$0 < \lambda \in \mathbb{R}$ 。下面以逻辑回归模型为例，介绍 R 语言中求解此类优化问题的方法。

30.2　对数似然与损失函数

30.2.1　Logistic 分布

在介绍逻辑回归之前，先了解一下 Logistic 分布。一个均值为 m ，方差为 $\frac{\pi^2}{3}s^2$ 的 Logistic 分布函数的形式为

$$F(x) = \frac{1}{1+\exp(-\frac{x-m}{s})}$$

© 王建平

密度函数的形式为

$$f(x) = \frac{\exp(-\frac{x-m}{s})}{s(1+\exp(-\frac{x-m}{s}))^2} = \frac{\exp(\frac{x-m}{s})}{s(1+\exp(\frac{x-m}{s}))^2}$$

密度函数与分布函数的关系如下：

$$\frac{dF(x)}{dx} = f(x) = sF(x)(1-F(x))$$

也就是说 Logistic 分布是上述微分方程的解。

(a) 概率密度函数　　(b) 概率分布函数

图 30.1: 逻辑斯谛分布

R 语言中分别表示逻辑斯谛分布的密度函数、分布函数、分位函数和随机数生成函数如下：

```
dlogis(x, location = 0, scale = 1, log = FALSE)
plogis(q, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
qlogis(p, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
rlogis(n, location = 0, scale = 1)
```

如果函数参数 `location` 或 `scale` 没有指定，则分别取默认值 0 和 1，就是标准的逻辑斯谛分布。位置参数（类似正态分布中的均值 μ）为 `location = m`，尺度参数（类似正态分布中的标准差 σ）为 `scale = s`，逻辑斯谛分布是一个长尾分布。

30.2.2 逻辑回归

响应变量 Y 服从伯努利分布 Bernoulli(p)，取值是 0 或 1，对线性预测 $X\boldsymbol{\beta}$ 做 Logistic 变换

$$\boldsymbol{p} = \mathsf{E}Y = \text{Logistic}(X\boldsymbol{\beta}) = \frac{1}{1+e^{-(\alpha+X\boldsymbol{\beta})}} = \frac{e^{\alpha+X\boldsymbol{\beta}}}{1+e^{\alpha+X\boldsymbol{\beta}}}$$

张饼 ©

Logistic 的逆变换

$$\text{Logistic}^{-1}(\boldsymbol{p}) = \ln\big(\frac{\boldsymbol{p}}{1-\boldsymbol{p}}\big) = \alpha + X\boldsymbol{\beta}$$

记数据矩阵 X 为

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1k} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{nk} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_1^\top \\ \boldsymbol{x}_2^\top \\ \vdots \\ \boldsymbol{x}_n^\top \end{bmatrix}$$

每一行表示一次观测，每一列表示一个变量的 n 次观测，记 $X = (X_1, X_2, \cdots, X_k)$ 是一个 $n \times k$ 数据矩阵，其中 $\boldsymbol{x}_i^\top$ 表示矩阵 X 的第 i 行，一共有 n 行，可以看作是 $1 \times k$ 的矩阵，$X_j, j = 1, 2, \cdots, k$ 表示矩阵 X 的第 j 列，一共有 k 列。类似地，$\boldsymbol{\beta} = (\beta_1, \beta_2, \cdots, \beta_k)^\top$ 是一个列向量，可以看作是 $k \times 1$ 的矩阵，β_j 表示第 j 个变量 X_j 的系数。对第 i 次观测

$$\text{Logistic}^{-1}(p_i) = \ln\big(\frac{p_i}{1-p_i}\big) = \alpha + \boldsymbol{x}_i^\top\boldsymbol{\beta}$$

关于参数 $\alpha, \boldsymbol{\beta}$ 的似然函数如下：

$$\begin{aligned} \mathcal{L}(\alpha, \boldsymbol{\beta}) &= \prod_{i=1}^n p_i^{y_i}(1-p_i)^{1-y_i} \\ &= \prod_{i=1}^n \Big(\frac{e^{\alpha+\boldsymbol{x}_i^\top\boldsymbol{\beta}}}{1+e^{\alpha+\boldsymbol{x}_i^\top\boldsymbol{\beta}}}\Big)^{y_i}\Big(\frac{1}{e^{\alpha+\boldsymbol{x}_i^\top\boldsymbol{\beta}}}\Big)^{1-y_i} \end{aligned} \tag{30.1}$$

关于参数 $\alpha, \boldsymbol{\beta}$ 的对数似然函数如下：

$$\begin{aligned} \ell(\alpha, \boldsymbol{\beta}) &= \log \mathcal{L}(\alpha, \boldsymbol{\beta}) \\ &= \sum_{i=1}^n \Big[y_i \log(p_i) + (1-y_i)\log(1-p_i)\Big] \\ &= \sum_{i=1}^n \Big[y_i \log\Big(\frac{e^{\alpha+\boldsymbol{x}_i\boldsymbol{\beta}}}{1+e^{\alpha+\boldsymbol{x}_i\boldsymbol{\beta}}}\Big) + (1-y_i)\log\Big(\frac{1}{e^{\alpha+\boldsymbol{x}_i\boldsymbol{\beta}}}\Big)\Big] \end{aligned} \tag{30.2}$$

对数似然函数 $\ell(\alpha, \boldsymbol{\beta})$ 关于参数 $\alpha, \boldsymbol{\beta}$ 的偏导数如下：

$$\begin{aligned} \frac{\partial \ell(\alpha, \boldsymbol{\beta})}{\partial \alpha} &= \sum_{i=1}^n \Big[\big(\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}\big)\frac{\partial p_i}{\partial \alpha}\Big] \\ \frac{\partial \ell(\alpha, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^n \Big[\big(\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}\big)\frac{\partial p_i}{\partial \beta}\Big] \\ &= \sum_{i=1}^n \Big[\big(\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}\big)p_i(1-p_i)\boldsymbol{x}_i^\top\Big] \end{aligned} \tag{30.3}$$

©黄湘云

其中，$p_i = \frac{e^{\alpha+\boldsymbol{x}_i\boldsymbol{\beta}}}{1+e^{\alpha+\boldsymbol{x}_i\boldsymbol{\beta}}}$，要使 $\ell(\alpha,\boldsymbol{\beta})$ 取极大值，一般通过迭代加权最小二乘算法（Iteratively (Re-)Weighted Least Squares，简称 IWLS）求解此优化问题，它可以看作拟牛顿法的一种特殊情况，在 R 语言中，函数 `glm()` 是求解此类问题的办法。

30.3 数值优化问题求解器

30.3.1 `optim()`

从一个逻辑回归模型模拟一组样本，共 2500 条记录，即 $n = 2500$，10 个观测变量，即 $k = 10$，其中，只有变量 X_1 和 X_2 的系数非零，参数设定为 $\alpha = 1, \beta_1 = 3, \beta_2 = -2$，而 $\beta_i = 0, i = 3, \cdots, 10$ 模拟数据的代码如下：

```
set.seed(2023)
n <- 2500
k <- 10
X <- matrix(rnorm(n * k), ncol = k)
y <- rbinom(n, size = 1, prob = plogis(1 + 3 * X[, 1] - 2 * X[, 2]))
```

模拟数据矩阵 X 与上述记号 X 是对应的，记号 $\boldsymbol{x}_i^\top$ 表示数据矩阵的第 i 行。α 是逻辑回归方程的截距，$\boldsymbol{\beta}$ 是 k 维列向量，X 是 $n \times k$ 维的矩阵且 $n > k$，y 是 n 维向量。极大化对数似然函数方程式 30.2 ，就是求解一个多维非线性无约束优化问题。方便起见，将 α 合并进 $\boldsymbol{\beta}$ 向量，另，函数 `optim()` 默认求极小，因此在对数似然函数前添加负号。

```
# 目标函数
log_logit_lik <- function(beta) {
  p <- plogis(cbind(1, X) %*% beta)
  -sum(y * log(p) + (1 - y) * log(1 - p))
}
```

高维情形下，没法绘制似然函数图形，退化到二维，如图 30.2 所示，二维情形下的逻辑回归模型的负对数似然函数曲面。

当用 Base R 函数 `optim()` 来求解时，发现 Nelder-Mead 算法收敛慢，易陷入局部最优解，即使迭代 10000 次，与真值仍然相去甚远。当用 SANN （模拟退火算法）求解此 11 维非线性无约束优化问题时，迭代 10000 次后，比较接近真值。

```
optim(
  par = rep(1, 11), # 初始值
  fn = log_logit_lik, # 目标函数
  method = "SANN",
  control = list(maxit = 10000)
)
```

©黄湘云

图 30.2: 二维情形下的逻辑回归模型的负对数似然函数曲面

©黄湘云

```
#> $par
#>  [1]  1.0755086156  3.2857327374 -2.1172404451 -0.0268567120  0.0184306330
#>  [6]  0.0304496968  0.0045154725  0.1283816433 -0.0746276329 -0.0624193044
#> [11] -0.0001349772
#>
#> $value
#> [1] 754.1838
#>
#> $counts
#> function gradient
#>    10000       NA
#>
#> $convergence
#> [1] 0
#>
#> $message
#> NULL
```

根据目标函数计算其梯度，有了梯度信息，可以使用迭代效率更高的 L-BFGS-B 算法。

```
# 梯度函数
log_logit_lik_grad <- function(beta) {
  p <- plogis(cbind(1, X) %*% beta)
  -t((y / p - (1 - y) / (1 - p)) * p * (1 - p)) %*% cbind(1, X)
}

optim(
  par = rep(1, 11), # 初始值
  fn = log_logit_lik, # 目标函数
  gr = log_logit_lik_grad, # 目标函数的梯度
  method = "L-BFGS-B"
)
```

```
#> $par
#>  [1]  1.00802641  3.11296713 -2.00955313  0.05855394 -0.02650585  0.01330428
#>  [7]  0.02171815  0.10213455 -0.02949774 -0.08633384  0.08098888
#>
#> $value
#> [1] 750.9724
#>
#> $counts
```

©梁汉明

```
#> function gradient
#>       13       13
#>
#> $convergence
#> [1] 0
#>
#> $message
#> [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

相比于函数 `optim()`，R 包 **nloptr** 不但可以提供类似的数值优化功能，而且可以处理各类非线性约束，能力更强。仍然基于上面的优化问题，调用 **nloptr** 包求解的代码如下：

```
library(nloptr)
nlp <- nloptr(
  x0 = rep(1, 11),
  eval_f = log_logit_lik,
  eval_grad_f = log_logit_lik_grad,
  opts = list(
    "algorithm" = "NLOPT_LD_LBFGS",
    "xtol_rel" = 1.0e-8
  )
)
nlp
```

```
#>
#> Call:
#>
#> nloptr(x0 = rep(1, 11), eval_f = log_logit_lik, eval_grad_f = log_logit_lik_grad,
#>     opts = list(algorithm = "NLOPT_LD_LBFGS", xtol_rel = 1e-08))
#>
#>
#> Minimization using NLopt version 2.7.1
#>
#> NLopt solver status: 3 ( NLOPT_FTOL_REACHED: Optimization stopped because
#> ftol_rel or ftol_abs (above) was reached. )
#>
#> Number of Iterations....: 23
#> Termination conditions:  xtol_rel: 1e-08
#> Number of inequality constraints:  0
#> Number of equality constraints:    0
#> Optimal value of objective function:  750.97235708148
```

©黄湘云

```
#> Optimal value of controls: 1.008028 3.112977 -2.009557 0.05854534 -0.02650855 0.01330416 0.02171839
#> 0.1021212 -0.02949994 -0.08632463 0.08098663
```

如果对数似然函数是多模态的，一般的求解器容易陷入局部最优解，推荐用 **nloptr** 包的全局优化求解器。

30.3.2 glm()

Base R 提供的函数 `glm()` 拟合模型，指定联系函数为 logit 变换。

```
fit_r <- glm(y ~ X, family = binomial(link = "logit"))
summary(fit_r)
```

```
#>
#> Call:
#> glm(formula = y ~ X, family = binomial(link = "logit"))
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  1.00803    0.07395  13.631   <2e-16 ***
#> X1           3.11298    0.13406  23.222   <2e-16 ***
#> X2          -2.00956    0.09952 -20.192   <2e-16 ***
#> X3           0.05855    0.06419   0.912    0.362
#> X4          -0.02651    0.06588  -0.402    0.687
#> X5           0.01330    0.06461   0.206    0.837
#> X6           0.02172    0.06496   0.334    0.738
#> X7           0.10212    0.06279   1.626    0.104
#> X8          -0.02950    0.06474  -0.456    0.649
#> X9          -0.08632    0.06482  -1.332    0.183
#> X10          0.08099    0.06385   1.268    0.205
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 3381.4  on 2499  degrees of freedom
#> Residual deviance: 1501.9  on 2489  degrees of freedom
#> AIC: 1523.9
#>
#> Number of Fisher Scoring iterations: 6
```

或者也可以用函数 `glm.fit()`，效果类似，使用方式不同罢了。

```
fit_r2 <- glm.fit(x = cbind(1, X), y = y, family = binomial(link = "logit"))
coef(fit_r2)
```

```
#>  [1]  1.00802820  3.11297679 -2.00955727  0.05854534 -0.02650855  0.01330416
#>  [7]  0.02171839  0.10212118 -0.02949994 -0.08632463  0.08098663
```

函数 glm() 的参数是一个公式，函数 glm.fit() 的参数是矩阵、向量，用函数 glm() 拟合模型，其内部调用的就是函数 glm.fit()。

30.3.3 glmnet 包

调用 **glmnet** 包的函数 glmnet() 拟合模型，指定指数族的具体形式为二项分布，伯努利分布是二项分布的特殊形式，也叫两点分布或 0-1 分布。

```
library(Matrix)
library(glmnet)
fit_glm <- glmnet(x = X, y = y, family = "binomial")
```

逻辑回归模型系数在 L1 正则下的迭代路径图

```
plot(fit_glm, ylab = "回归系数")
```

图 30.3: 回归系数的迭代路径

©黄湘云

从图可见，剩余两个系数是非零的，一个是 3，一个是 -2，其余都被压缩，而接近为 0 了。

```
plot(fit_glm$lambda,
  ylab = expression(lambda), xlab = "迭代次数",
  main = "惩罚系数的迭代路径"
)
```

图 30.4: 惩罚系数的迭代路径

随着迭代的进行，惩罚系数 λ 越来越小，接近于 0，这也是符合预期的，因为模型本来就是简单的逻辑回归，不带惩罚项。选择一个迭代趋于稳定时的 λ 比如 0.0005247159，此时各个参数的取值如下：

```
coef(fit_glm, s = 0.0005247159)
```

```
#> 11 x 1 sparse Matrix of class "dgCMatrix"
#>                       s1
#> (Intercept)  0.997741857
#> V1           3.076358149
#> V2          -1.984018387
#> V3           0.052633923
#> V4          -0.020195037
#> V5           0.008065018
```

统计学习 ©

```
#> V6          0.015936357
#> V7          0.095722046
#> V8         -0.023589159
#> V9         -0.080864640
#> V10         0.075234011
```

截距 (Intercept) 对应 $\alpha = 0.997741857$，而 $\beta_1 = 3.076358149$ 对应 V1，$\beta_2 = -1.984018387$ 对应 V2，以此类推。

30.4 评估模型的分类效果

逻辑回归模型是二分类模型，评估模型的分类效果，两个办法。

1. 可以用 AUC 指标或者 ROC 曲线，**pROC** 包和 **ROCR** 包都可以绘制 ROC 曲线。
2. 可以用 Wilcoxon 检验，越显著表示分类效果越好。

30.4.1 ROC 曲线和 AUC 值

ROC 是 Receiver Operating Characteristic 简写。随机抽取 2000 个样本作为训练集，余下的数据作为测试集。

```
dat <- cbind.data.frame(X, y)
set.seed(20232023)
idx <- sample(x = 1:nrow(dat), size = 2000, replace = F)
# 训练集
dat_train <- dat[idx, ]
# 测试集
dat_test <- dat[-idx, ]
```

函数 `glm()` 拟合训练集数据

```
fit_binom <- glm(y ~ ., data = dat_train, family = binomial(link = "logit"))
```

将训练好的模型用于测试集，调用函数 `predict()` 进行预测，`type = "response"` 获得预测概率值，它是对数几率，比值比的对数。

```
dat_test$pred <- predict(fit_binom, newdata = dat_test, type = "response")
```

返回值介于 0 - 1 之间，表示预测概率。在测试集上绘制 ROC 曲线。

```
pROC::plot.roc(
  y ~ pred, data = dat_test,
  col = "dodgerblue", print.auc = TRUE,
  auc.polygon = TRUE, auc.polygon.col = "#f6f6f6",
```

© 黄湘云


```
  xlab = "FPR", ylab = "TPR", main = "预测 ROC 曲线"
)
```

```
#> Setting levels: control = 0, case = 1
```

```
#> Setting direction: controls < cases
```

图 30.5: ROC 曲线

ROC 曲线越往左上角拱，表示预测效果越好。FPR 是 False Positive Rate 的缩写，TPR 是 True Positive Rate 的缩写。

```
# 计算 AUC 值
pROC::auc(y ~ pred, data = dat_test)
```

```
#> Setting levels: control = 0, case = 1
```

```
#> Setting direction: controls < cases
```

```
#> Area under the curve: 0.9487
```

AUC 是 area under curve 的缩写，表示 ROC 曲线下的面积，所以 AUC 指标越接近 1 越好。

30.4.2 Wilcoxon 检验

对每个标签的预测概率指定服从均匀分布，相当于随机猜测，所以最后 ROC 会接近对角线，而且样本量越大越接近，AUC 会越来越接近 0.5。如果预测结果比随机猜测要好，Wilcoxon 检验会显著，预测

效果越好检验会越显著，表示预测 pred 和观测 y 越接近。

```
wilcox.test(pred ~ y, data = dat_test)
#>
#>  Wilcoxon rank sum test with continuity correction
#>
#> data:  pred by y
#> W = 3140, p-value < 2.2e-16
#> alternative hypothesis: true location shift is not equal to 0
```

©黄湘云

第三十一章　数值优化

本章亮点

1. 比较全面地展示各类优化问题的 R 语言实现，其覆盖面之广，远超市面上同类 R 语言书籍。从线性优化到凸优化（包含凸二次优化和凸锥优化），从简单整数线性优化到混合整数非线性优化，再到一般的非线性优化，触及最前沿的热门话题。
2. 对每类优化问题都给出示例及 R 语言实现，涉及 10 余个各类优化器。参考 Lingo 和 1stOpt 等国内外商业优化建模软件的官方示例，也参考开源软件 Octave（语法大量兼容 Matlab 的科学计算软件）的非线性优化示例，给出 R 语言实现。经过对比，发现 R 语言求解器的效果可以达到同类开源和商业软件的水平。
3. 对于 R 语言社区难以求解的复杂优化问题，也都给出了开源替代方案，并总结了实战经验。比如，混合整数非线性优化，通过 **rAMPL** 包 (Brandao 2023) 连接 AMPL 软件，调用开源的优化求解器 Couenne 求解。R 语言社区的优化建模扩展包相比于商业软件的最大优势是免费易获取，可以随时查找相关 R 包的论文和源码深入研究，了解优化算法的理论和实现过程。

本章介绍五类典型的优化问题及 R 语言求解过程，按从易到难的顺序分别是线性优化、凸二次优化、凸锥优化、非线性优化、整数优化。学习完本章内容，读者可以灵活运用各类软件工具包解决各类标准优化问题。本章内容不涉及优化算法理论，对理论感兴趣的读者可以寻着 R 包参考文献或者相关书籍进一步学习。

除了 R 软件内置一些数值优化求解器，R 语言社区还有大量数值优化方面的函数和 R 包。特别是 **ROI** 包 (Theußl, Schwendinger, 和 Hornik 2020)，它通过插件包对 20 多个 R 包提供统一的函数调用方式，相当于一个运筹优化方面的基础设施平台，极大地方便学习和使用。

ROI 包通过插件包来实际调用第三方做数值优化的 R 包。**Rglpk** 包 (Theussl 和 Hornik 2023) 可以求解大规模线性优化、整数线性优化和混合整数线性优化，**ROI** 包通过插件包 **ROI.plugin.glpk** 与之连接调用。**nloptr** 包 (Johnson 2023) 可以求解二次优化和非线性优化，**ROI** 包通过插件包 **ROI.plugin.nloptr** 与之连接调用。**scs** 包 (O'Donoghue 等 2016) 可以求解凸锥优化，**ROI** 包通过插件包 **ROI.plugin.scs** 与之连接调用。**ECOSolveR** 包 (Fu 和 Narasimhan 2023) 可以求解含整数变量约束的凸锥优化，**ROI** 包通过插件包 **ROI.plugin.ecos** 与之连接调用。**quadprog** 包 (S original by Berwin A. Turlach 2019) 可以求解凸二次优化，**ROI** 包通过插件包 **ROI.plugin.quadprog** 与之连接调用。本文不能一一概括，相信读者之后可以举一反三。

图 31.1: 数值优化扩展包、插件包和 ROI 包的关系图

```
library(ROI)
library(ROI.plugin.glpk)     # 线性和整数线性优化
library(ROI.plugin.nloptr)   # 非线性优化
library(ROI.plugin.scs)      # 凸锥优化
library(ROI.plugin.ecos)     # 可含整数变量的凸锥优化
library(ROI.plugin.quadprog) # 凸二次优化
library(lattice)
# 自定义作图用的调色板
custom_palette <- function(irr, ref, height, saturation = 0.9) {
  hsv(
    h = height, s = 1 - saturation * (1 - (1 - ref)^0.5),
    v = irr
  )
}
```

31.1 线性优化

线性优化是指目标函数和约束条件都是线性的优化问题。考虑如下线性优化问题：

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & -6x_1 - 5x_2 \\
\text{s.t.} \quad & \begin{cases} x_1 + 4x_2 \le 16 \\ 6x_1 + 4x_2 \le 28 \\ 2x_1 - 5x_2 \le 6 \end{cases}
\end{aligned}
$$

其中，目标函数是 $-6x_1 - 5x_2$，min 表示求目标函数的最小值，$\boldsymbol{x} = (x_1, x_2)^\top$ 表示决策变量，无特殊说明，决策变量都取实数。s.t. 是 subject to 的缩写，专指约束条件。上述线性优化问题写成矩阵形式，如下：

汪荣鹏©

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & \begin{bmatrix} -6 \\ -5 \end{bmatrix}^{\top} \boldsymbol{x} \\
\text{s.t.} \quad & \left\{ \begin{bmatrix} 1 & 4 \\ 6 & 4 \\ 2 & -5 \end{bmatrix} \boldsymbol{x} \leq \begin{bmatrix} 16 \\ 28 \\ 6 \end{bmatrix} \right.
\end{aligned}
$$

用 $\boldsymbol{d}$ 表示目标函数的系数向量，A 表示约束矩阵，$\boldsymbol{b}$ 表示右手边的向量。上述优化问题用矩阵表示，如下：

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & \boldsymbol{d}^{\top} \boldsymbol{x} \\
\text{s.t.} \quad & A\boldsymbol{x} \leq \boldsymbol{b}
\end{aligned}
$$

用 **ROI** 包提供的一套使用语法表示该线性优化问题，代码如下：

```
# 定义优化问题
op <- OP(
  objective = L_objective(L = c(-6, -5)),
  constraints = L_constraint(
    L = matrix(c(
      1, 4,
      6, 4,
      2, -5
    ), ncol = 2, byrow = TRUE),
    dir = c("<=", "<=", "<="),
    rhs = c(16, 28, 6)
  ),
  types = c("C", "C"),
  maximum = FALSE
)
# 优化问题描述
op
```

```
#> ROI Optimization Problem:
#>
#> Minimize a linear objective function of length 2 with
#> - 2 continuous objective variables,
#>
#> subject to
#> - 3 constraints of type linear.
#> - 0 lower and 0 upper non-standard variable bounds.
```

统计出版社 ©

```
# 求解优化问题
res <- ROI_solve(op, solver = "glpk")
# 最优解
res$solution

#> [1] 2.4 3.4

# 目标函数值
res$objval

#> [1] -31.4
```

函数 `OP()` 定义一个优化问题，参数如下：

- `objective` ： 指定目标函数，用函数 `L_objective()` 表示线性优化中的目标函数，函数名中 L 表示 Linear（线性），包含数值型向量。
- `constraints` ： 指定约束条件，用函数 `L_constraint()` 表示线性优化中的约束条件，函数名中 L 表示 Linear（线性），包含约束矩阵 A ，约束分量的方向可为 `>=` 、`<=` 或 `=` ，本例中为 `<=`，右手边的向量 b 。
- `types` ： 指定决策变量的类型，分三种情况，`B` 表示 0-1 变量，字母 B 是 binary 的意思，`I` 表示整型变量，字母 I 是 integer 的意思，`C` 表示数值型变量，字母 C 是 continuous 的意思。本例中，两个变量都是连续型的，`types = c("C", "C")` 。
- `maximum` ： 指定目标函数需要求极大还是极小，默认求极小，取值为逻辑值 `TRUE` 或 `FALSE`。

不同类型的目标函数和约束条件组合在一起可以构成非常丰富的优化问题。**ROI** 包支持的目标函数、约束条件及相应的代码见下表。后续将根据优化问题，逐个介绍用法。

表格 31.1: **ROI** 包可以表示的目标函数和约束条件

目标函数	代码	约束条件	代码
线性函数	`L_objective()`	无约束	留空
二次函数	`Q_objective()`	箱式约束	`V_bound()`
非线性函数	`F_objective()`	线性约束	`L_constraint()`
		二次约束	`Q_constraint()`
		锥约束	`C_constraint()`
		非线性约束	`F_constraint()`

31.2 凸二次优化

二次优化分严格凸二次和非严格凸二次优化问题，严格凸要求矩阵对称正定，非严格凸要求矩阵对称半正定。对于矩阵负定的情况，不是凸优化问题，暂不考虑。二次优化的一般形式如下：

©黄湘云

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \frac{1}{2}\boldsymbol{x}^\top D\boldsymbol{x} + \boldsymbol{d}^\top \boldsymbol{x} \\ \text{s.t.} \quad & A\boldsymbol{x} \le \boldsymbol{b} \end{aligned}$$

二次优化不都是凸优化，当且仅当矩阵 D 半正定时，上述二次优化是凸二次优化，当矩阵 D 正定时，上述二次优化是严格凸二次优化。下面举个严格凸二次优化的具体例子，令

$$D = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, \quad \boldsymbol{d} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}, \quad A = \begin{bmatrix} -1 & -1 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} -2 \\ 2 \\ 3 \end{bmatrix}$$

即目标函数

$$Q(x_1, x_2) = x_1^2 + x_2^2 - x_1x_2 + 3x_1 - 2x_2$$

二次优化中的数据矩阵和向量 $D, \boldsymbol{d}, A, \boldsymbol{b}$ 依次用 `Dmat`、`dvec`、`Amat`、`bvec` 表示出来。

```
Dmat <- matrix(c(2, -1, -1, 2), nrow = 2, byrow = TRUE)
dvec <- c(3, -2)
Amat <- matrix(c(-1, -1, 1, -1, 0, 1), ncol = 2, byrow = TRUE)
bvec <- c(-2, 2, 3)
```

同样，也是在函数 `OP()` 中传递目标函数，约束条件。在函数 `Q_objective()` 中定义二次优化的目标函数，字母 Q 是 Quadratic 的意思，表示二次部分，字母 L 是 Linear 的意思，表示线性部分。函数 `L_constraint()` 的使用同线性优化，不再赘述。根据 **ROI** 包的使用接口定义的参数，定义目标优化。

```
op <- OP(
  objective = Q_objective(Q = Dmat, L = dvec),
  constraints = L_constraint(L = Amat, dir = rep("<=", 3), rhs = bvec),
  maximum = FALSE
)
op
```

```
#> ROI Optimization Problem:
#>
#> Minimize a quadratic objective function of length 2 with
#> - 2 continuous objective variables,
#>
#> subject to
#> - 3 constraints of type linear.
#> - 0 lower and 0 upper non-standard variable bounds.
```

nloptr 包有许多优化求解器，可用于求解二次优化的也有好几个。对于一个目标优化，函数 `ROI_applicable_solvers()` 可以找到能够求解此优化问题的求解器。

```
ROI_applicable_solvers(op)
```

```
#> [1] "nloptr.cobyla" "nloptr.mma"    "nloptr.auglag" "nloptr.isres"
#> [5] "nloptr.slsqp"  "quadprog"
```

下面使用其中的 `nloptr.slsqp` 来求解。

```
nlp <- ROI_solve(op, solver = "nloptr.slsqp", start = c(1, 2))
nlp$objval
```

```
#> [1] -0.08333333
```

```
nlp$solution
```

```
#> [1] 0.1666667 1.8333333
```

作为对比，移除线性不等式约束，求解无约束优化问题。目标函数仍然是二次型，但是已经没有线性约束条件，所以不是二次优化问题，再用求解器 `nloptr.slsqp` 求解的结果已不是无约束优化的解。

```
op2 <- OP(
  objective = Q_objective(Q = Dmat, L = dvec),
  maximum = FALSE
)
op2
```

```
#> ROI Optimization Problem:
#>
#> Minimize a quadratic objective function of length 2 with
#> - 2 continuous objective variables,
#>
#> subject to
#> - 0 constraints
#> - 0 lower and 0 upper non-standard variable bounds.
```

```
nlp2 <- ROI_solve(op2, solver = "nloptr.slsqp", start = c(1, 2))
nlp2$objval
```

```
#> [1] -1
```

```
nlp2$solution
```

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

在可行域上画出等高线，标记目标解的位置，图 31.2 展示无约束和有约束条件下的解。图中橘黄色线围成的三角形区域是可行域，红点表示无约束下求解器 `nloptr.slsqp` 获得的解 $(0, 1)$ ，真正的无约束解是蓝点所在位置为 $(-4/3, 1/3)$ ，黄点表示线性约束下求解器 `nloptr.slsqp` 获得的解 $(1/6, 11/6)$ 。所以，不能用二次优化的求解器去求解无约束的二次优化问题。

quadprog 包在求解约束条件下的严格凸二次优化问题时，同时给出无约束条件下的解。这个包自定义

©黄湘云

图 31.2: 对比无约束和有约束条件下的解

了一套二次优化问题的符号，查看求解函数 `solve.QP()` 的说明，略作对应后，求解上述优化问题的代码如下。

```
library(quadprog)
sol <- solve.QP(
  Dmat = Dmat, dvec = -dvec, Amat = t(-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
```

其中，返回值的 `unconstrained.solution` 表示无约束下的解，与预期的解一致，这就没有疑惑了。可见，约束二次优化问题和无约束二次优化问题的求解器不同。

31.3 凸锥优化

31.3.1 锥与凸锥

二维平面上，圆盘和扇面是凸锥。三维空间中，球，圆锥、椭球、椭圆锥都是凸锥，如图 31.3 所示。

图 31.3: 常见的三维凸锥

锥定义在对称的矩阵上，凸锥要求矩阵正定。一个 2 阶对称矩阵 A 是正定的

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

意味着 $a_{11} > 0, a_{22} > 0, a_{12} = a_{21}, a_{11}a_{22} - a_{12}a_{21} > 0$ 。一般地，将 n 阶半正定的对称矩阵 A 构成的集合记为 $\mathcal{K}_+^n$ 。

$$\mathcal{K}_+^n = \{A \in \mathbb{R}^{n\times n} | \boldsymbol{x}^\top A\boldsymbol{x} \geq 0,\ \forall \boldsymbol{x} \in \mathbb{R}^n\}$$

目标函数为线性的凸锥优化的一般形式如下：

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{d}^\top \boldsymbol{x} \\ \text{s.t.} \quad & A\boldsymbol{x} + \boldsymbol{k} = \boldsymbol{b} \\ & \boldsymbol{k} \in \mathcal{K}. \end{aligned}$$

其中，集合 $\mathcal{K}$ 是一个非空的封闭凸锥。在一个凸锥里，寻求一个线性目标函数的最小值。专门求解此类问题的 **scs** 包也在 **ROI** 包的支持范围内，可以求解的锥优化包括零锥、线性锥、二阶锥、指数锥、幂锥和半正定锥。

下面举个例子说明凸锥，含参对称矩阵 $A(m_1, m_2, m_3)$ 如下：

© 基础工

$$A(m_1, m_2, m_3) = \begin{bmatrix} 1 & m_1 & m_2 \\ m_1 & 1 & m_3 \\ m_2 & m_3 & 1 \end{bmatrix}.$$

而 $\boldsymbol{k} = \boldsymbol{b} - A\boldsymbol{x}$ 是非空封闭凸锥集合 $\mathcal{K}$ 中的元素。半正定矩阵 A 生成的集合（凸锥）K 如下：

$$K = \{(m_1, m_2, m_3) \in \mathbb{R}^3 \mid A(m_1, m_2, m_3) \in \mathcal{K}_+^3\},$$

集合 K 是有界半正定的，要求含参矩阵 A 的行列式大于等于 0。矩阵 A 的行列式如下：

$$\det(A(m_1, m_2, m_3)) = -(m_1^2 + m_2^2 + m_3^2 - 2m_1m_2m_3 - 1)$$

集合 K 的边界可表示为如下方程的解：

$$m_1^2 + m_2^2 + m_3^2 - 2m_1m_2m_3 = 1$$

或等价地表示为如下矩阵形式：

$$\begin{bmatrix} m_1 \\ m_2 \end{bmatrix}^\top \begin{bmatrix} 1 & -m_3 \\ -m_3 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} = 1 - m_3^2.$$

当 $m_3 = 0$ 时，集合 K 的边界表示平面上的一个单位圆，当 $m_3 \in [-1, 1]$ ，集合 K 的边界表示一个椭圆。为了获得一个直观的印象，将集合 K 的边界绘制出来，如图 31.3 所示，边界是一个三维曲面，曲面及其内部构成一个凸锥。

31.3.2 零锥

零锥的定义如下：

$$\mathcal{K}_{zero} = \{0\}$$

常用于表示线性等式约束。

31.3.3 线性锥

线性锥（Linear Cone）的定义如下：

$$\mathcal{K}_{lin} = \{x \in \mathbb{R} | x \geq 0\}$$

© 黄湘云

图 31.4: 锥

©黄湘云

常用于表示线性不等式约束。

31.3.4 二阶锥

二阶锥（Second-order Cone）的定义如下：

$$\mathcal{K}_{soc}^n = \{(t,x) \in \mathbb{R}^n | x \in \mathbb{R}^{n-1}, t \in \mathbb{R}, \|x\|_2 \le t\}$$

常用于凸二次优化问题。考虑如下二阶锥优化 SOCP 问题：

$$\begin{aligned} \max_{(\boldsymbol{y},t)} \quad & y_1 + y_2 \\ \text{s.t.} \quad & \sqrt{(2+3y_1)^2 + (4+5y_2)^2} \le 6 + 7t \\ & y_1, y_2 \in \mathbb{R}, \quad t \in (-\infty, 9]. \end{aligned}$$

令 $\boldsymbol{x} = (y_1, y_2, t)^\top$ ， $\boldsymbol{b} = (b_1, b_2, b_3)^\top$

$$A = \begin{bmatrix} \boldsymbol{a_1}^\top \\ \boldsymbol{a_2}^\top \\ \boldsymbol{a_3}^\top \end{bmatrix}$$

上述 SOCP 问题的非线性不等式约束等价于

$$\sqrt{(b_2 - \boldsymbol{a_2}^\top \boldsymbol{x})^2 + (b_3 - \boldsymbol{a_3}^\top \boldsymbol{x})^2} \le b_1 - \boldsymbol{a_1}^\top \boldsymbol{x}$$

其中，

$$A = \begin{bmatrix} 0 & 0 & -7 \\ -3 & 0 & 0 \\ 0 & -5 & 0 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 6 \\ 2 \\ 4 \end{bmatrix}$$

scs 包不能求解此类优化问题，下面调用 **ECOSolveR** 包求解。

```
library(ROI.plugin.ecos)
op <- OP(
  objective = c(1, 1, 0),
  constraints = C_constraint(
    L = rbind(
      c(0, 0, -7),
      c(-3, 0, 0),
      c(0, -5, 0)
```

```
    ),
    cones = K_soc(3), rhs = c(6, 2, 4)
  ), maximum = TRUE,
  bounds = V_bound(ld = -Inf, ui = 3, ub = 9, nobj = 3)
)
sol <- ROI_solve(op, solver = "ecos")
# 最优解
sol$solution

#> [1] 19.055671  6.300041  9.000000

# 目标函数值
sol$objval

#> [1] 25.35571
```

对决策变量 y_1 添加整数约束，则只有 **ECOSolveR** 包可以求解。

```
op <- OP(
  objective = c(1, 1, 0),
  constraints = C_constraint(
    L = rbind(
      c(0, 0, -7),
      c(-3, 0, 0),
      c(0, -5, 0)
    ),
    cones = K_soc(3), rhs = c(6, 2, 4)
  ), maximum = TRUE,
  # 决策变量约束
  types = c("I", "C", "C"),
  bounds = V_bound(ld = -Inf, ui = 3, ub = 9, nobj = 3)
)
sol <- ROI_solve(op, solver = "ecos")
# 最优解
sol$solution

#> [1] 19.000000  6.355418  9.000000

# 目标函数值
sol$objval

#> [1] 25.35542
```

©黄海广

31.3.5 指数锥

指数锥（Exponential Cone）的定义如下：

$$\mathcal{K}_{\text{expp}} = \{(x_1, x_2, x_3) \in \mathbb{R}^3 | x_2 > 0, x_2 \exp\left(\frac{x_1}{x_2}\right) \leq x_3\} \cup \{(x_1, 0, x_3) \in \mathbb{R}^3 | x_1 \leq 0, x_3 \geq 0\}$$

它的对偶如下：

$$\mathcal{K}_{\text{expd}} = \{(x_1, x_2, x_3) \in \mathbb{R}^3 | x_1 < 0, -x_1 \exp\left(\frac{x_2}{x_1}\right) \leq \exp(1) x_3\} \cup \{(0, x_2, x_3) \in \mathbb{R}^3 | x_2, x_3 \geq 0\}$$

考虑一个锥优化问题

$$\begin{aligned} \max_{(\boldsymbol{x}, \boldsymbol{t})} \quad & x_1 + 2x_2 \\ \text{s.t.} \quad & \exp(7 + 3x_1 + 5x_2) \leq 9 + 11t_1 + 12t_2 \\ & x_1, x_2 \in (-\infty, 20],\ t_1, t_2 \in (-\infty, 50] \end{aligned}$$

约束条件 $\exp(7 + 3x_1 + 5x_2) \leq 9 + 11t_1 + 12t_2$ 可以用指数锥来表示

$$\begin{aligned} u &= 7 + 3y_1 + 5y_2 \\ v &= 1 \\ w &= 9 + 11t_1 + 12t_2 \end{aligned}$$

记 $\boldsymbol{x} = (y_1, y_2, t_1, t_2)^\top$ ，则线性约束矩阵 A 和约束向量 $\boldsymbol{b}$ 如下：

$$A = \begin{bmatrix} -3 & -5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -11 & -12 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 7 \\ 1 \\ 9 \end{bmatrix}$$

指数锥用函数 `K_expp()` 表示，锥优化问题的代码如下：

```
# 目标优化
op <- OP(
  objective = c(1, 2, 0, 0),
  # 锥约束
  constraints = C_constraint(L = rbind(
    c(-3, -5, 0, 0),
    c(0, 0, 0, 0),
    c(0, 0, -11, -12)
  ), cone = K_expp(1), rhs = c(7, 1, 9)),
  bounds = V_bound(ld = -Inf, ub = c(20, 20, 50, 50)),
```

```
  maximum = TRUE
)
op

#> ROI Optimization Problem:
#>
#> Maximize a linear objective function of length 4 with
#> - 4 continuous objective variables,
#>
#> subject to
#> - 3 constraints of type conic.
#>   |- 3 conic constraints of type 'expp'
#> - 4 lower and 4 upper non-standard variable bounds.
```

对于锥优化，可以调用 **scs** 包来求解。

```
# 调用 scs 包
library(ROI.plugin.scs)
sol <- ROI_solve(op, solver = "scs")
# 最优解
sol$solution

#> [1] -33.3148  20.0000  50.0000  50.0000

# 目标函数值
sol$objval

#> [1] 6.685201
```

31.3.6 幂锥

一个三维幂锥（Power Cone）的定义如下：

$$\mathcal{K}_{\text{powp}}^{\alpha} = \{(x_1, x_2, x_3) \in \mathbb{R}^3 | x_1, x_2 \geq 0, x_1^{\alpha} x_2^{1-\alpha} \geq |x_3|\}, \alpha \in [0, 1]$$

它的对偶形式如下：

$$\mathcal{K}_{\text{powp}}^{\alpha} = \left\{(x_1, x_2, x_3) \in \mathbb{R}^3 | x_1, x_2 \geq 0, (\frac{x_1}{\alpha})^{\alpha} (\frac{x_2}{1-\alpha})^{1-\alpha} \geq |x_3|\right\}, \alpha \in [0, 1]$$

考虑如下锥优化问题

©清华大学出版社

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & 3x_1 + 5x_2 \\
\text{s.t.} \quad & 5 + x_1 \le (2 + x_2)^4 \\
& x_1 \ge 0,\ x_2 \ge 2
\end{aligned}
$$

约束条件 $5 + x_1 \le (2 + x_2)^4$ 可以重新表示为幂锥

$$
\begin{aligned}
u &= 5 + y_1 \\
v &= 1 \\
w &= 2 + y_2 \\
\alpha &= 1/4
\end{aligned}
$$

记 $\boldsymbol{x} = (y_1, y_2)^\top$ ，约束矩阵和约束向量如下

$$
A = \begin{bmatrix} -1 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 5 \\ 1 \\ 2 \end{bmatrix}
$$

幂锥用函数 `K_powp()` 表示，锥优化问题的代码如下：

```
A <- rbind(c(-1, 0), c(0, 0), c(0, -1))
cpowp <- C_constraint(L = A, cones = K_powp(1 / 4), rhs = c(5, 1, 2))
op <- OP(
  objective = c(3, 5),
  constraints = cpowp,
  bounds = V_bound(lb = c(0, 2))
)
op
```

```
#> ROI Optimization Problem:
#>
#> Minimize a linear objective function of length 2 with
#> - 2 continuous objective variables,
#>
#> subject to
#> - 3 constraints of type conic.
#>   |- 3 conic constraints of type 'powp'
#> - 1 lower and 0 upper non-standard variable bounds.
```

```
sol <- ROI_solve(op, solver = "scs", max_iter = 1e6)
# 最优解
sol$solution
```

©黄湘云

```
#> [1] 250.998234   2.000352
```

```
# 目标函数值
sol$objval
```

```
#> [1] 762.9965
```

31.3.7 半正定锥

如果矩阵 A 是半正定的，记为 $A \succeq 0$ ，如果矩阵 A 是正定的，记为 $A \succ 0$ 。记 n 阶实对称矩阵的集合为 $\mathcal{S}^n$ 。半正定锥（Positive Semi Definite Cone）的定义如下：

$$\mathcal{K}_{\text{psd}}^n = \{A | A \in \mathcal{S}^n, \boldsymbol{x}^\top A \boldsymbol{x} \geq 0, \forall \boldsymbol{x} \in \mathbb{R}^n\}$$

考虑如下锥优化问题

$$\begin{aligned}
\min_{\boldsymbol{x}} \quad & x_1 + x_2 - x_3 \\
\text{s.t.} \quad & x_1 \begin{bmatrix} 10 & 3 \\ 3 & 10 \end{bmatrix} + x_2 \begin{bmatrix} 6 & -4 \\ -4 & 10 \end{bmatrix} + x_3 \begin{bmatrix} 8 & 1 \\ 1 & 6 \end{bmatrix} \preceq \begin{bmatrix} 16 & -13 \\ -13 & 60 \end{bmatrix} \\
& x_1, x_2, x_3 \geq 0
\end{aligned}$$

函数 `K_psd()` 表示半正定锥，函数 `vech()` 将对称矩阵的上三角部分拉成一个向量。

```
(A <- toeplitz(x = 3:1))
```

```
#>      [,1] [,2] [,3]
#> [1,]    3    2    1
#> [2,]    2    3    2
#> [3,]    1    2    3
```

```
vech(A)
```

```
#>      [,1]
#> [1,]    3
#> [2,]    2
#> [3,]    1
#> [4,]    3
#> [5,]    2
#> [6,]    3
```

锥优化的表示如下

```
F1 <- rbind(c(10, 3), c(3, 10))
F2 <- rbind(c(6, -4), c(-4, 10))
```

统计出版社 ©

```
F3 <- rbind(c(8, 1), c(1, 6))
F0 <- rbind(c(16, -13), c(-13, 60))
# 目标优化
op <- OP(
  objective = L_objective(c(1, 1, -1)),
  constraints = C_constraint(
    L = vech(F1, F2, F3),
    cones = K_psd(3),
    rhs = vech(F0)
  )
)
op
```

```
#> ROI Optimization Problem:
#>
#> Minimize a linear objective function of length 3 with
#> - 3 continuous objective variables,
#>
#> subject to
#> - 3 constraints of type conic.
#>   |- 3 conic constraints of type 'psd'
#> - 0 lower and 0 upper non-standard variable bounds.
```

仍然调用 **scs** 包求解器。

```
sol <- ROI_solve(op, solver = "scs")
# 最优解
sol$solution
```

```
#> [1] 5.782736e-06 1.065260e-06 1.486444e+00
```

```
# 目标函数值
sol$objval
```

```
#> [1] -1.486437
```

31.4 非线性优化

非线性优化按是否带有约束，以及约束是线性还是非线性，分为无约束优化、箱式约束优化、线性约束优化和非线性约束优化。箱式约束可看作是线性约束的特殊情况。

© 黄湘云

表格 31.2: R 软件内置的非线性优化函数

	nlm()	nlminb()	constrOptim()	optim()
无约束	支持	支持	不支持	支持
箱式约束	不支持	支持	支持	支持
线性约束	不支持	不支持	支持	不支持

R 软件内置的 **stats** 包有 4 个数值优化方面的函数，函数 `nlm()` 可求解无约束优化问题，函数 `nlminb()` 可求解无约束、箱式约束优化问题，函数 `constrOptim()` 可求解箱式和线性约束优化。函数 `optim()` 是通用型求解器，包含多个优化算法，可求解无约束、箱式约束优化问题。尽管这些函数在 R 语言中长期存在，在统计中有广泛的使用，如非线性最小二乘 `stats::nls()`，极大似然估计 `stats4::mle()` 和广义最小二乘估计 `nlme::gls()` 等。但是，这些优化函数的求解能力有重合，使用语法不尽相同，对于非线性约束无能为力，下面仍然主要使用 **ROI** 包来求解多维非线性优化问题。

31.4.1 一元非线性优化

求如下一维分段非线性函数的最小值，其函数图像见图 31.5 ，这个函数是不连续的，更不光滑。

$$f(x) = \begin{cases} 10 & x \in (-\infty, -1] \\ \exp(-\frac{1}{|x-1|}) & x \in (-1, 4) \\ 10 & x \in [4, +\infty) \end{cases}$$

```
fn <- function(x) ifelse(x > -1, ifelse(x < 4, exp(-1 / abs(x - 1)), 10), 10)
```

图 31.5: 一维函数图像

©黄湘云

函数 `optimize()` 可以求解一元函数的极值问题，默认求极小值，参数 `f` 表示目标函数，参数 `interval` 表示搜索在此区间内最小值。函数返回一个列表，元素 `minimum` 表示极小值点，`objective` 表示极值点对应的目标函数值。

```
optimize(f = fn, interval = c(-4, 20), maximum = FALSE)
```

```
#> $minimum
#> [1] 19.99995
#>
#> $objective
#> [1] 10
```

```
optimize(f = fn, interval = c(-7, 20), maximum = FALSE)
```

```
#> $minimum
#> [1] 0.9992797
#>
#> $objective
#> [1] 0
```

值得注意，对于不连续的分段函数，在不同的区间内搜索极值，可能获得不同的结果，可以绘制函数图像帮助选择最小值。

31.4.2 多元隐函数优化

这个优化问题来自 1stOpt 软件的帮助文档，下面利用 R 语言来求该多元隐函数的极值。

$$\begin{aligned}\min_{\boldsymbol{x}} y = &\ \sin\left((yx_1-0.5)^2+2x_1x_2^2-\frac{y}{10}\right)\cdot \\ &\exp\left(-\left((x_1-0.5-\exp(-x_2+y))\right)^2+x_2^2-\frac{y}{5}+3\right)\right)\end{aligned}$$

其中，$x_1\in[-1,7], x_2\in[-2,2]$ 。

对于隐函数 $f(x_1,x_2,y)=0$ ，常规的做法是先计算隐函数的偏导数，并令偏导数为 0，再求解非线性方程组，得到各个驻点，最后，将驻点代入原方程，比较驻点处函数值，根据优化目标选择最大或最小值。

$$\begin{aligned}\frac{\partial f(x_1,x_2,y)}{\partial x_1}&=0\\ \frac{\partial f(x_1,x_2,y)}{\partial x_2}&=0\end{aligned}$$

如果目标函数很复杂，隐函数偏导数难以计算，可以考虑暴力网格搜索。先估计隐函数值 z 的大致范围，给定 x,y 时，计算一元非线性方程的根。

```
fn <- function(m) {
  subfun <- function(x) {
```

©黄湘云

```
    f1 <- (m[1] * x - 0.5)^2 + 2 * m[1] * m[2]^2 - x / 10
    f2 <- -((m[1] - 0.5 - exp(-m[2] + x))^2 + m[2]^2 - x / 5 + 3)
    x - sin(f1) * exp(f2)
  }
  uniroot(f = subfun, interval = c(-1, 1))$root
}
```

在位置 $(1, 2)$ 处函数值为 0.0007368468。

```
# 测试函数 fn
fn(m = c(1, 2))
```

```
#> [1] 0.0007368468
```

将目标区域网格化，通过一元非线性方程求根的方式获得每个格点处的函数值。

```
df <- expand.grid(
  x1 = seq(from = -1, to = 7, length.out = 81),
  x2 = seq(from = -2, to = 2, length.out = 41)
)
# 计算格点处的函数值
df$fn <- apply(df, 1, FUN = fn)
```

在此基础上，绘制隐函数图像，如图 31.6 所示，可以获得关于隐函数的大致情况。

最后，获得暴力网格搜索的结果，目标函数在 $(2.8, -0.9)$ 处取得最小值 -0.02159723。总的来说，这是一个近似结果，如果进一步缩小搜索区域，将网格划分得越细，搜索的结果将越接近全局最小值。

```
df[df$fn == min(df$fn), ]
```

```
#>      x1   x2          fn
#> 930 2.8 -0.9 -0.02159723
```

将求隐函数极值的问题转为含非线性等式约束的非线性优化问题。

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & y \\
\text{s.t.} \quad & f(x_1, x_2, y) = 0
\end{aligned}
$$

由于等式约束非常复杂，手动计算等式约束的雅可比矩阵不可行，可以用 **numDeriv** 包的函数 `jacobian()` 计算等式约束的雅可比矩阵。考虑到本例中仅含有一个等式约束，雅可比矩阵退化为梯度向量，这可以用 **numDeriv** 包的另一个函数 `grad()` 计算。

```
# 等式约束
heq <- function(x) {
  f1 <- (x[1] * x[3] - 0.5)^2 + 2 * x[1] * x[2]^2 - x[3] / 10
  f2 <- (x[1] - 0.5 - exp(-x[2] + x[3]))^2 + x[2]^2 - x[3] / 5 + 3
  x[3] - sin(f1) * exp(-f2)
```

©黄湘云

图 31.6: 隐函数图像

统计学习方法

```
}
# 等式约束的梯度
heq.jac <- function(x) {
  numDeriv::grad(func = heq, x = x)
}
```

函数 L_objective() 表示含 1 个决策变量的线性目标函数，函数 F_constraint() 表示非线性等式约束。

```
# 定义优化问题
op <- OP(
  objective = L_objective(L = c(0, 0, 1)),
  constraints = F_constraint(
    # 等式约束
    F = list(heq = heq),
    dir = "==",
    rhs = 0,
    # 等式约束的雅可比
    J = list(heq.jac = heq.jac)
  ),
  bounds = V_bound(
    ld = -Inf, ud = Inf,
    li = c(1, 2), ui = c(1, 2),
    lb = c(-1, -2), ub = c(7, 2),
    nobj = 3L
  ),
  maximum = FALSE # 求最小
)
op
```

```
#> ROI Optimization Problem:
#>
#> Minimize a linear objective function of length 3 with
#> - 3 continuous objective variables,
#>
#> subject to
#> - 1 constraint of type nonlinear.
#> - 3 lower and 2 upper non-standard variable bounds.
```

将网格搜索的结果作为初值，继续寻找更优的目标函数值。

```
nlp <- ROI_solve(op,
  solver = "nloptr.slsqp", start = c(2.8, -0.9, -0.02159723)
```

©黄湘云

```
)
# 最优解
nlp$solution
```

```
#> [1]  2.89826224 -0.85731584 -0.02335409
```

```
# 目标函数值
nlp$objval
```

```
#> [1] -0.02335409
```

可以发现，更优的目标函数值 -0.02335 在 $(2.898, -0.8573)$ 取得。

31.4.3 多元无约束优化

31.4.3.1 示例 1

Rastrigin 函数是一个 n 维优化问题测试函数。

$$\min_{\boldsymbol{x}} \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$$

计算函数值的 R 代码如下：

```
fn <- function(x) {
  sum(x^2 - 10 * cos(2 * pi * x) + 10)
}
```

绘制二维情形下的 Rastrigin 函数图像，如图 31.7 所示，这是一个多模态的函数，有许多局部极小值。如果采用 BFGS 算法寻优容易陷入局部极值点。

不失一般性，考虑函数维数 $n = 20$ ，决策变量 $x_i \in [-50, 50], i = 1, 2, \ldots, n$ 的情况。

```
op <- OP(
  objective = F_objective(fn, n = 20L),
  bounds = V_bound(ld = -50, ud = 50, nobj = 20L)
)
```

调全局优化器求解优化问题。

```
nlp <- ROI_solve(op, solver = "nloptr.directL")
# 最优解
nlp$solution
```

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

```
# 目标函数值
nlp$objval
```

©黄湘云

图 31.7: 二维 Rastrigin 函数图像

©黄湘云


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

31.4.3.2 示例 2

下面这个优化问题来自 1stOpt 软件帮助手册，是一个无约束非线性优化问题，它的目标函数非常复杂，一般的求解器都无法求解。最优解在 $(7.999982, 7.999982)$ 取得，目标函数值为 -7.978832。

$$\min_{\boldsymbol{x}} \ \cos(x_1)\cos(x_2) - \sum_{i=1}^{5}\Big((-1)^i \cdot i \cdot 2 \cdot \exp\big(-500 \cdot ((x_1 - i \cdot 2)^2 + (x_2 - i \cdot 2)^2)\big)\Big)$$

目标函数分两步计算，先计算累加部分的通项，然后代入计算目标函数。

```
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)))
}
```

直观起见，绘制目标函数在区域 $[-50, 50] \times [-50, 50]$ 内的图像，如图 31.8a 所示，可以看到几乎没有变化的梯度，给寻优过程带来很大困难。再将区域 $[0, 12] \times [0, 12]$ 上的三维图像绘制出来，如图 31.8b 所示，可见，有不少局部陷阱，且分布在 $x_2 = x_1$ 的直线上。

不失一般性，下面考虑 $x_1, x_2 \in [-50, 50]$ ，面对如此复杂的函数，调用全局优化器 `nloptr.directL` 寻优。

```
op <- OP(
  objective = F_objective(fn, n = 2L),
  bounds = V_bound(ld = -50, ud = 50, nobj = 2L)
)
nlp <- ROI_solve(op, solver = "nloptr.directL")
nlp$solution
```

```
#> [1] 22.22222  0.00000
```

```
nlp$objval
```

```
#> [1] -0.9734211
```

结果还是陷入局部最优解。运筹优化方面的商业软件，著名的有 Lingo 和 Matlab，下面采用 Lingo 20 求解，Lingo 代码如下：

```
SETS:
P/1..5/;
Endsets
```

©黄湘云

(a) 区域 $[-50,50]\times[-50,50]$ 内的函数图像

(b) 区域 $[0,12]\times[0,12]$ 内的函数图像

图 31.8: 局部放大前后的函数图像

© 基础统计


```
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);
```

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。

在这种情况下，数值优化算法遇到瓶颈，可以采用一些全局随机优化算法，比如 **GA** 包 (Scrucca 2013) 实现的遗传算法。经过对参数的一些调优，可以获得与商业软件几乎一样的结果。

```r
nlp <- GA::ga(
  type = "real-valued",
  fitness = function(x) -fn(x),
  lower = c(0, 0), upper = c(12, 12),
  popSize = 500, maxiter = 100,
  monitor = FALSE, seed = 20232023
)
# 最优解
nlp@solution
```

```
#>             x1       x2
#> [1,] 7.999982 7.999981
```

```r
# 目标函数值
nlp@fitnessValue
```

```
#> [1] 7.978832
```

其中，参数 `type` 指定决策变量的类型，`type = "real-valued"` 表示目标函数中的决策变量是实值连续的，参数 `fitness` 是目标函数，函数 `ga()` 对目标函数求极大，所以，对当前优化问题，添加了一个负号。参数 `popSize` 控制种群大小，值越大，运行时间越长，搜索范围越广，获得的全局优化解越好。对于复杂的优化问题，可以不断增加种群大小来寻优，直至增加种群大小也不能获得更好的解。参数 `maxiter` 控制种群进化的次数，值越大，搜索次数可以越多，获得的解越好。参数 `popSize` 的影响大于参数 `maxiter` ，减少陷入局部最优解（陷阱）的可能。根据已知条件尽可能缩小可行域，以减少种群数量，进而缩短算法迭代时间。

31.4.4 多元箱式约束优化

有如下带箱式约束的多元非线性优化问题，该示例来自函数 `nlminb()` 的帮助文档，如果没有箱式约束，全局极小值点在 $(1, 1, \cdots, 1)$ 处取得。

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & (x_1 - 1)^2 + 4\sum_{i=1}^{n-1}(x_{i+1} - x_i^2)^2 \\
\text{s.t.} \quad & 2 \le x_1, x_2, \cdots, x_n \le 4
\end{aligned}
$$

©黄湘云

R 语言编码的函数代码如下：

```
fn <- function(x) {
  n <- length(x)
  sum(c(1, rep(4, n - 1)) * (x - c(1, x[-n])^2)^2)
}
```

在二维的情形下，可以绘制目标函数的三维图像，见图 31.9 ，函数曲面和香蕉函数有些相似。

图 31.9: 类香蕉函数的曲面图

Base R 有 3 个函数可以求解这个优化问题，分别是 `nlminb()` 、`constrOptim()` 和 `optim()` ，因此，不妨在这个示例上，用这 3 个函数分别求解该优化问题，介绍它们的用法，最后，介绍 **ROI** 包实现的方法。这个优化问题的目标函数是 n 维非线性的，不失一般性，又不让问题变得太过简单，下面考虑 25 维的情况，

31.4.4.1 `nlminb()`

函数 `nlminb()` 参数 `start` 指定迭代初始值，参数 `objective` 指定目标函数，参数 `lower` 和 `upper` 分别指定箱式约束中的下界和上界。给定初值 $(3, 3, \cdots, 3)$，下界 $(2, 2, \cdots, 2)$ 和上界 $(4, 4, \cdots, 4)$ 。`nlminb()` 帮助文档说该函数出于历史兼容性的原因尚且存在，一般来说，这个函数会一直维护下去的。

©王清平


```
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)"
```

从返回结果来看，求解过程成功收敛，最优解的前 23 个决策变量取值为 2，在箱式约束的边界上，第 24 个分量没有边界上，而在内部，第 25 个决策变量取值为 4，也在边界上。目标函数值为 368.1059。

31.4.4.2 constrOptim()

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：

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ 2 \\ -4 \\ -4 \end{bmatrix}$$

©李东风 统计学

```r
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
```

返回结果中 `convergence = 1` 表示迭代次数到达默认的极限 `maxit = 500` 。参考函数 `nlminb()` 的求解结果，可知还没有收敛。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。

```r
constrOptim(
  theta = rep(3, 25), # 初始值
  f = fn, # 目标函数
  method = "Nelder-Mead",
  control = list(maxit = 1000),
```

©黄湘云

```
  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
```

结果有改善，目标函数值从 378.4208 减小到 370.8601，但还是没有收敛，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 BFGS 优化算法，这得先计算出来目标函数的梯度。

```
# 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(
```

©梁俊文

```
  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
```

从结果来看，虽然已经收敛，但相比于 Nelder-Mead 方法，目标函数值变大了，可见已陷入局部最优解。

31.4.4.3 optim()

下面再使用函数 `optim()` 提供的 `L-BFGS-B` 算法求解优化问题。

```
optim(
  par = rep(3, 25), fn = fn, gr = NULL, method = "L-BFGS-B",
```

©黄湘云

```
  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"
```

发现结果和函数 `nlminb()` 的结果差不多了。

```
optim(
  par = rep(3, 25), fn = fn, gr = gr, method = "L-BFGS-B",
  lower = rep(2, 25), upper = rep(4, 25)
)

#> $par
#>  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3
#>
#> $value
#> [1] 373
#>
#> $counts
#> function gradient
#>        2        2
#>
#> $convergence
#> [1] 0
#>
```

```
#> $message
#> [1] "CONVERGENCE: NORM OF PROJECTED GRADIENT <= PGTOL"
```

然而，当在函数 `optim()` 里提供梯度信息的时候，虽然目标函数及梯度的计算次数变少了，求解速度提升了，但是最优解反而变差了，最优解和在函数 `constrOptim()` 中设置 `method = "BFGS"` 算法基本一致。

31.4.4.4 ROI 包

下面通过 **ROI** 包，分别调用求解器 `nloptr.lbfgs` 和 `nloptr.directL` ，发现前者同样陷入局部最优解，而后者可以获得与 `nlminb()` 函数一致的结果。

```
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))
# 目标函数值
nlp$objval

#> [1] 373

# 最优解
nlp$solution

#>  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3
```

调全局优化算法。

```
nlp <- ROI_solve(op, solver = "nloptr.directL")
# 目标函数值
nlp$objval

#> [1] 368.106

# 最优解
nlp$solution

#>  [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
```

©黄湘云

31.4.5 多元线性约束优化

对于带线性约束的多元非线性优化问题，Base R 提供函数 `constrOptim()` 来求解，下面的示例来自其帮助文档，这是一个带线性约束的二次规划问题。

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & -\begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}^{\top} \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^{\top}\boldsymbol{x} \\
\text{s.t.} \quad & \begin{bmatrix} -4 & 2 & 0 \\ -3 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}^{\top} \boldsymbol{x} \geq \begin{bmatrix} -8 \\ 2 \\ 0 \end{bmatrix}
\end{aligned}
$$

```
fQP <- function(x) {
  -sum(c(0, 5, 0) * x) + 0.5 * sum(x * x)
}
Amat <- matrix(c(-4, -3, 0, 2, 1, 0, 0, -2, 1),
  ncol = 3, nrow = 3, byrow = FALSE
)
bvec <- c(-8, 2, 0)
# 目标函数的梯度
gQP <- function(x) {
  -c(0, 5, 0) + x
}
constrOptim(
  theta = c(2, -1, -1),
  f = fQP, g = gQP,
  ui = t(Amat), ci = bvec
)
```

```
#> $par
#> [1] 0.4761908 1.0476188 2.0952376
#>
#> $value
#> [1] -2.380952
#>
#> $counts
#> function gradient
#>      406       81
#>
#> $convergence
#> [1] 0
```

```
#>
#> $message
#> NULL
#>
#> $outer.iterations
#> [1] 3
#>
#> $barrier.value
#> [1] -0.0006243894
```

在上一节，箱式约束可以看作线性约束的一种特殊情况，**ROI** 包是支持箱式、线性、二次、锥和非线性约束的。因此，下面给出调用 **ROI** 包求解上述优化问题的代码。

```
Dmat <- diag(rep(1,3))
dvec <- c(0, 5, 0)
op <- OP(
  objective = Q_objective(Q = Dmat, L = -dvec),
  constraints = L_constraint(L = t(Amat), dir = rep(">=", 3), rhs = bvec),
  maximum = FALSE
)
nlp <- ROI_solve(op, solver = "nloptr.slsqp", start = c(0, 1, 2))
# 最优解
nlp$solution
```

```
#> [1] 0.4761905 1.0476190 2.0952381
```

```
# 目标函数值
nlp$objval
```

```
#> [1] -2.380952
```

可见输出结果与函数 `constrOptim()` 是一致的。

31.4.6 多元非线性约束优化

nloptr 包的非线性优化能力覆盖开源优化软件 Octave 和 Ipopt 。通过插件包 **ROI.plugin.nloptr**，**ROI** 包可以调用 **nloptr** 包内置的所有求解器，常用的求解器见下表。表中从优化器类型（局部还是全局优化器），支持的约束条件类型（箱式还是非线性），是否需要提供目标函数的梯度、黑塞和约束条件的雅可比矩阵信息等方面归纳各个求解器的能力。

©黄湘云

表格 31.3: 常用的非线性优化求解器

求解器	类型	约束	梯度	黑塞	雅可比
`nloptr.lbfgs`	局部	箱式	需要	不需要	不需要
`nloptr.slsqp`	局部	非线性	需要	不需要	需要
`nloptr.auglag`	局部	非线性	需要	不需要	需要
`nloptr.directL`	全局	箱式	不需要	不需要	不需要
`nloptr.isres`	全局	非线性	不需要	不需要	不需要

31.4.6.1 非线性等式约束

下面这个示例来自 Octave 软件的非线性优化帮助文档，Octave 中的函数 `sqp()` 使用序列二次优化求解器（successive quadratic programming solver）求解非线性优化问题，示例中该优化问题包含多个非线性等式约束。

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & \exp\left(\prod_{i=1}^{5} x_i\right) - \frac{1}{2}(x_1^3 + x_2^3 + 1)^2 \\
\text{s.t.} \quad & \begin{cases} \sum_{i=1}^{5} x_i^2 - 10 = 0 \\ x_2 x_3 - 5 x_4 x_5 = 0 \\ x_1^3 + x_2^3 + 1 = 0 \end{cases}
\end{aligned}
$$

目标函数是非线性的，有 5 个变量，约束条件也是非线性的，有 3 个等式约束。先手动计算目标函数的梯度，等式约束的雅可比矩阵。

```
# 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
  )
}
# 等式约束的雅可比矩阵
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
  )
}
```

在 OP() 函数里定义目标优化的各个成分。

```
# 定义目标优化
op <- OP(
  # 5 个决策变量
  objective = F_objective(F = fn, n = 5L, G = gr),
  constraints = F_constraint(
    F = list(heq = heq),
    dir = "==",
    rhs = 0,
    # 等式约束的雅可比矩阵
    J = list(heq.jac = heq.jac)
  ),
  bounds = V_bound(ld = -Inf, ud = Inf, nobj = 5L),
  maximum = FALSE # 求最小
)
op

#> ROI Optimization Problem:
#>
#> Minimize a nonlinear objective function of length 5 with
#> - 5 continuous objective variables,
#>
#> subject to
#> - 1 constraint of type nonlinear.
#> - 5 lower and 0 upper non-standard variable bounds.
```

©黄湘云

调用 SQP（序列二次优化）求解器 `nloptr.slsqp` 。

```
nlp <- ROI_solve(op,
  solver = "nloptr.slsqp",
  start = c(-1.8, 1.7, 1.9, -0.8, -0.8)
)
# 最优解
nlp$solution
```

```
#> [1] -1.7171435  1.5957096  1.8272458 -0.7636431 -0.7636431
```

```
# 目标函数值
nlp$objval
```

```
#> [1] 0.05394985
```

计算结果和 Octave 的示例一致。

31.4.6.2 多种非线性约束

- 非线性等式约束
- 非线性不等式约束，不等式约束包含等号
- 箱式约束

此优化问题来源于 **Ipopt** 官网的帮助文档，约束条件比较复杂。提供的初始值为 $x_0 = (1, 5, 5, 1)$，最优解为 $x_\star = (1.00000000, 4.74299963, 3.82114998, 1.37940829)$。优化问题的具体内容如下：

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & x_1 x_4 (x_1 + x_2 + x_3) + x_3 \\
\text{s.t.} \quad & \begin{cases} x_1^2 + x_2^2 + x_3^2 + x_4^2 = 40 \\ x_1 x_2 x_3 x_4 \geq 25 \\ 1 \leq x_1, x_2, x_3, x_4 \leq 5 \end{cases}
\end{aligned}
$$

下面用 **ROI** 调 **nloptr** 包求解，看结果是否和例子一致，**nloptr** 支持箱式约束且支持不等式约束包含等号。

```
# 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]))
}
# 定义目标优化
op <- OP(
  objective = F_objective(F = fn, n = 4L, G = gr), # 4 个决策变量
  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
)
```

作为对比参考，先计算目标函数的初始值和最优值。

```
# 目标函数初始值
fn(c(1, 5, 5, 1))
```

```
#> [1] 16
```

```
# 目标函数最优值
fn(c(1.00000000, 4.74299963, 3.82114998, 1.37940829))
```

```
#> [1] 17.01402
```

统计学习 ©

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。
- 函数 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] 1.191201 4.360952 4.267227 1.162977
```

```
nlp$objval
```

```
#> [1] 17.87039
```

可以看出，**nloptr** 提供的优化能力可以覆盖 Ipopt 求解器，从以上求解的情况来看，推荐使用 nloptr.slsqp 求解器，这也是 Octave 的选择。

31.5 整数优化

整数优化情况有很多，篇幅所限，仅考虑以下几类常见情形：

1. 目标函数和约束条件为线性，变量取值都为整数的整数优化。
2. 目标函数和约束条件为线性，变量取值为 0 或 1 的 0-1 整数优化。
3. 目标函数和约束条件为线性，部分变量带有整数约束的混合整数线性优化。
4. 目标函数为凸二次、约束条件为线性，部分变量是整数的混合整数二次优化。
5. 目标函数和约束条件为非线性，部分变量是整数的混合整数非线性优化。

©李漫天

31.5.1 纯整数线性优化

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & -2x_1 - x_2 - 4x_3 - 3x_4 - x_5 \\
\text{s.t.} \quad & \begin{cases} 2x_2 + x_3 + 4x_4 + 2x_5 < 54 \\ 3x_1 + 4x_2 + 5x_3 - x_4 - x_5 < 62 \\ x_1, x_2 \in [0, 100] \quad x_3 \in [3, 100] \\ x_4 \in [0, 100] \quad x_5 \in [2, 100] \\ x_i \in \mathbb{Z},\ i = 1, 2, \cdots, 5. \end{cases}
\end{aligned}
$$

求解器 `glpk` 还可以求解一些整数优化问题。

```
op <- OP(
  objective = L_objective(c(-2, -1, -4, -3, -1)),
  types = rep("I", 5),
  constraints = L_constraint(
    L = matrix(c(
      0, 2, 1, 4, 2,
      3, 4, 5, -1, -1
    ), ncol = 5, byrow = TRUE),
    dir = c("<", "<"),
    rhs = c(54, 62)
  ),
  # 添加约束
  bounds = V_bound(
    li = 1:5, ui = 1:5,
    lb = c(0, 0, 3, 0, 2), ub = rep(100, 5), nobj = 5
  ),
  maximum = FALSE
)
op
```

```
#> ROI Optimization Problem:
#>
#> Minimize a linear objective function of length 5 with
#> - 5 integer objective variables,
#>
#> subject to
#> - 2 constraints of type linear.
#> - 2 lower and 5 upper non-standard variable bounds.
```

```
# 求解
```

©黄湘云


```
res <- ROI_solve(op, solver = "glpk")
# 最优解
res$solution
```

```
#> [1] 15  0  6 11  2
```

```
# 目标函数值
res$objval
```

```
#> [1] -89
```

可知，最优解在 $(15, 0, 6, 11, 2)$ 处取得，目标函数值为 -89 。

注意：还有一组最优解 $(19, 0, 4, 10, 5)$ ，目标函数值也为 -89 ，但是 `glpk` 求解器未能给出。

31.5.2 0-1 整数线性优化

目标函数是线性的，决策变量的取值要么是 0 要么是 1。指派问题属于典型的 0-1 整数优化问题。有 n 个人需要去完成 n 项任务，每个人完成一项任务，每项任务只由一个人完成，每个人单独完成各项任务所需花费（时间、费用）不同。要求设计一个方案，人和任务之间建立一一对应的关系，使得总花费最少。

设第 i 个人完成第 j 项任务的花费为 d_{ij} ，当安排第 i 个人完成第 j 项任务时，记为 $x_{ij} = 1$ ，否则，记为 $x_{ij} = 0$ ，指派问题的数学模型如下：

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \\
\text{s.t.} \quad & \begin{cases} \sum_{i=1}^{n} x_{ij} = 1, & j = 1, 2, \ldots, n \\ \sum_{j=1}^{n} x_{ij} = 1, & i = 1, 2, \ldots, n \\ x_{ij} = 0 \ \text{或} \ 1 \end{cases}
\end{aligned}
$$

指派问题在 **lpSolve** 包 (Berkelaar 等 2023) 做了很好的封装，只需提供花费矩阵，即可调用求解器求解该问题。

```
# 花费矩阵 D
D <- matrix(c(
  2, 7, 7, 2,
  7, 7, 3, 2,
  7, 2, 8, 10,
  1, 9, 8, 2
), nrow = 4, ncol = 4, byrow = F)
# 加载 lpSolve 包
library(lpSolve)
# 调用指派问题求解器
```

统计学习方法 ©李航

```
sol <- lp.assign(D)
# 最优解
sol$solution
```

```
#>      [,1] [,2] [,3] [,4]
#> [1,]    0    0    0    1
#> [2,]    0    0    1    0
#> [3,]    0    1    0    0
#> [4,]    1    0    0    0
```

```
# 总花费
sol$objval
```

```
#> [1] 8
```

可以使总花费最少的指派计划是第 1 个人完成第 4 项任务，第 2 个人完成第 3 项任务，第 3 个人完成第 2 项任务，第 4 个人完成第 1 项任务，总花费为 8。

31.5.3 混合整数线性优化

目标函数是线性的，一部分决策变量是整数。

$$
\begin{aligned}
\max_{\boldsymbol{x}} \quad & 3x_1 + 7x_2 - 12x_3 \\
\text{s.t.} \quad & \begin{cases} 5x_1 + 7x_2 + 2x_3 \leq 61 \\ 3x_1 + 2x_2 - 9x_3 \leq 35 \\ x_1 + 3x_2 + x_3 \leq 31 \\ x_1, x_2 \geq 0, \quad x_2, x_3 \in \mathbb{Z}, \quad x_3 \in [-10, 10] \end{cases}
\end{aligned}
$$

矩阵形式如下

$$
\begin{aligned}
\max_{\boldsymbol{x}} \quad & \begin{bmatrix} 3 \\ 7 \\ -12 \end{bmatrix}^{\top} \boldsymbol{x} \\
\text{s.t.} \quad & \begin{cases} \begin{bmatrix} 5 & 7 & 2 \\ 3 & 2 & -9 \\ 1 & 3 & 1 \end{bmatrix} \boldsymbol{x} \leq \begin{bmatrix} 61 \\ 35 \\ 31 \end{bmatrix} \end{cases}
\end{aligned}
$$

第 1 个变量是连续值，第 2、3 个变量是整数，第 3 个变量的下、上界分别是 -10 和 10。

```
op <- OP(
  objective = L_objective(c(3, 7, -12)),
  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
  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.
```

```
# 求解
res <- ROI_solve(op, solver = "glpk")
# 最优解
res$solution
```

```
#> [1]  0.3333333  8.0000000 -2.0000000
```

```
res$objval
```

```
#> [1] 81
```

31.5.4 混合整数二次优化

目标函数是二次的，一部分决策变量是整数。

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & x_1^2 + x_2^2 - x_1 x_2 + 3x_1 - 2x_2 \\
\text{s.t.} \quad & \begin{cases} -x_1 - x_2 <= -2 \\ x_1 - x_2 <= 2 \\ x_2 <= 3. \\ x_1 \in \mathbb{Z} \end{cases}
\end{aligned}
$$

在二次优化的基础上，对变量添加整型约束，即变成混合整数二次优化（Mixed Integer Quadratic Programming，简称 MIQP）。

```
# D
Dmat <- matrix(c(2, -1, -1, 2), nrow = 2, byrow = TRUE)
# d
dvec <- c(3, -2)
# A
Amat <- matrix(c(
  -1, -1,
  1, -1,
  0, 1
), ncol = 2, byrow = TRUE)
# b
bvec <- c(-2, 2, 3)
# 目标优化
op <- OP(
  objective = Q_objective(Q = Dmat, L = dvec),
  constraints = L_constraint(Amat, rep("<=", 3), bvec),
  types = c("I", "C"),
  maximum = FALSE # 求最小
)
# 查看可用于该优化问题的求解器
ROI_applicable_solvers(op)
```

```
#> NULL
```

目前，**ROI** 包支持的开源求解器都不能处理 MIQP 问题。**ECOSolveR** 包可以求解凸二阶锥优化，部分变量可以是整数。因此，先将凸二次优化转化为凸锥优化问题，再连接 **ECOSolveR** 包提供 `ecos` 求解器，最后，调 `ecos` 求解器求解。

©黄海广

$$
\begin{aligned}
\min_{(t,\boldsymbol{x})} \quad & t \\
\text{s.t.} \quad & \begin{cases} x_1^2 + x_2^2 - x_1x_2 + 3x_1 - 2x_2 \le t \\ -x_1 - x_2 <= -2 \\ x_1 - x_2 <= 2 \\ x_2 <= 3. \\ x_1 \in \mathbb{Z} \end{cases}
\end{aligned}
$$

引入新的变量 t ，原目标函数化为线性，约束条件增加一个二次型。

$$
\begin{aligned}
\min_{(t,\boldsymbol{x})} \quad & t \\
\text{s.t.} \quad & \begin{cases} \boldsymbol{x}^\top D\boldsymbol{x} + 2\boldsymbol{d}^\top \boldsymbol{x} \le t \\ A\boldsymbol{x} \le \boldsymbol{b} \\ x_1 \in \mathbb{Z} \end{cases}
\end{aligned}
$$

其中，

$$
D = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, \quad \boldsymbol{d} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}, \quad A = \begin{bmatrix} -1 & -1 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} -2 \\ 2 \\ 3 \end{bmatrix}
$$

最后，凸二次优化转为二阶锥优化 SOCP，形式如下：

$$
\begin{aligned}
\min_{(t^\star,\boldsymbol{x})} \quad & t^\star \\
\text{s.t.} \quad & \begin{cases} \|D^{1/2}\boldsymbol{x} + D^{-1/2}\boldsymbol{d}\|_2 \le t^\star \\ A\boldsymbol{x} \le \boldsymbol{b} \\ x_1 \in \mathbb{Z} \end{cases}
\end{aligned}
$$

代码如下

因二次优化的目标函数是二次连续可微的，而且是凸函数，求解器 Bonmin 可以获得最优解。

```
var x1 integer;
var x2;
minimize z: x1^2 + x2^2 - x1 * x2 + 3 * x1 - 2 * x2;
subject to A_limit: -x1 - x2 <= -2;
subject to B_limit: x1 - x2 <= 2;
subject to C_limit: x2 <= 3;
```

```
library(rAMPL)
# 配置 AMPL 安装路径
env <- new(Environment, "/opt/AMPL/ampl.macos64")
```

©南京大学出版社

```
ampl <- new(AMPL, env)
# 加载混合整数二次优化模型文件
ampl$read("code/MIQP.mod")
# 设置 MIQP 求解器 Bonmin
ampl$setOption("solver", "bonmin")
# 求解问题
ampl$solve()
# 最优解
ampl$getData("x1")
ampl$getData("x2")
# 目标函数值
ampl$getData("z")
```

最优解在 $(0,2)$ 处获得，最优值为 0。

31.5.5 混合整数非线性优化

在 R 语言社区的官方仓库中还没有开源的 R 包可以求解此类问题，开源社区中 Bonmin 项目专门求解混合整数非线性优化 MINLP（Mixed Integer Non-Linear Programming）问题。数学优化软件 AMPL 封装了 Bonmin 软件，并提供 R 语言接口 rAMPL。AMPL 社区版可以免费使用打包的开源求解器。

- 线性优化求解器 HiGHS。
- 混合整数线性优化求解器 cbc。
- 混合整数非线性优化求解器 Bonmin 和 Couenne。
- 非线性优化求解器 Ipopt。

安装 AMPL 社区版软件后，再安装 **rAMPL** 包，它依赖 **Rcpp** 包，所以需要一并安装。

```
install.packages("Rcpp", type = "source")
# 从 AMPL 官网安装 rAMPL 包
install.packages("https://ampl.com/dl/API/rAMPL.tar.gz", repos = NULL,
  INSTALL_opts = c("--no-multiarch", "--no-staged-install")
)
```

下面求解如下混合整数非线性优化问题。

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & 1.5(x_1-\sin(x_1-x_2))^2+0.5x_2^2+x_3^2-x_1x_2-2x_1+x_2x_3 \\
\text{s.t.} \quad & \begin{cases} x_1,x_2\in\mathbb{R} \quad x_3\in\mathbb{Z} \\ x_1,x_2\in[-20,20] \quad x_3\in[-10,10]. \end{cases}
\end{aligned}
$$

AMPL 模型代码如下：

©李东风

```
var X1;
var X2;
var X3 integer;
minimize z: 1.5 * (X1 - sin(X1 - X2))^2 + 0.5 * X2^2 + X3^2 - X1 * X2 - 2 * X1 + X2 * X3;
subject to A_limit: -20 <= X1 <= 20;
subject to B_limit: -20 <= X2 <= 20;
subject to C_limit: -10 <= X3 <= 10;
```

将代码保存到文件 `code/MINLP.mod` ，下面加载 **rAMPL** 包，调用求解器 Bonmin 求解该优化问题。

```r
library(rAMPL)
# 配置 AMPL 安装路径
env <- new(Environment, "/opt/AMPL/ampl.macos64")
ampl <- new(AMPL, env)
# 加载混合整数非线性优化模型文件
ampl$read("code/MINLP.mod")
# 设置 MINLP 求解器 Bonmin
ampl$setOption("solver", "bonmin")
# 求解问题
ampl$solve()
# 最优解
ampl$getData("X1")
ampl$getData("X2")
ampl$getData("X3")
# 目标函数值
ampl$getData("z")
```

如果使用 Bonmin 求解器，该优化问题的最优解在 $(2.892556, 1.702552, -1)$ 处获得，相应的目标函数值为 -4.176012 。如果使用求解器 Couenne ，它可以找到非凸混合整数非线性优化问题的全局最优解，Couenne 好于 Bonmin 求解器。

```r
# 调用 couenne 求解器
ampl$setOption("solver", "couenne")
# 求解问题
ampl$solve()
```

最优解在 $x_1 = 4.999633, x_2 = 9.734148, x_3 = -5$ 处取得，最优值为 -10.96182 。下面将两个最优解代入目标函数，验证一下最优值。

```r
fun <- 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]
}
```

```
# 局部最优解
fun(x = c(2.892556, 1.702552, -1))

#> [1] -4.176012

# 全局最优解
fun(x = c(4.999633, 9.734148, -5))

#> [1] -10.96182
```

31.6 总结

对大部分常规优化问题，都可以纳入 **ROI** 包的框架内。对少量复杂的优化问题，目前，必须借助开源社区的第三方求解器。

- 对于含整型变量的凸锥优化问题，**scs** 包不能求解，**ECOSolveR** 包可以，它还可以求解可转化为凸二阶锥优化问题的混合整数二次优化问题。
- 对于特定问题，比如 0-1 整数线性优化中的指派问题，相比于 **ROI** 包的大一统调用方式，**lpSolve** 包给出非常简明的使用语法。对凸二次优化问题，给出 **quadprog** 包的使用语法，补充说明 **nloptr** 包的结果，以及与 **ROI** 包调用语法的差异。
- 对于凸的混合整数二次优化和非凸的混合整数非线性优化问题，借助 **rAMPL** 包分别调用开源的求解器 Bonmin 和 Couenne 求解。
- 对于复杂的非线性优化问题，因其具有非凸、多模态等特点，求解非常困难。需要引入随机优化算法，比如采用 **GA** 包的遗传算法求解，效果可以达到商业软件的水平。
- 对于凸优化问题，可以求解得又快又好，而对于非凸优化问题，要么只能获得局部最优解，要么可以搜索全局最优解，但不给保证，而且运行时间长。

优化建模是一个具有基础性和支柱性的任务，几乎每个统计模型和机器学习算法背后都有一个优化问题。在 R 语言社区的优化任务视图 (Schwendinger 和 Borchers 2023) 中，可以看到数以百计的扩展包。非常广阔的应用场景催生了非常丰富的理论。根据目标函数和约束条件的情况，可以从不同的角度划分，如线性和非线性优化，连续和离散优化，确定性和随机优化，凸优化和非凸优化等。相关的理论著作非常多，感兴趣的读者可以根据自身情况找本教材系统性地学习。本章结构是按照优化问题分类组织的，主要涉及确定性的数值优化，因部分优化问题比较复杂，因此，也涉及少量的随机优化方法。

优化建模是一个具有重要商业价值的领域，相关的开源和商业软件有很多，比较流行的有 Python 社区的 Pyomo (Hart, Watson, 和 Woodruff 2011)，Julia 社区的 JuMP (Dunning, Huchette, 和 Lubin 2017)。比较著名的商业软件有 Lingo、Mosek、Gurobi 等，而 AMPL 一个软件平台，对 20 个开源和商业求解器提供一套统一的建模语言，且提供 R、Python 等编程语言接口。

相比于 Python 和 Julia 社区，R 语言社区在整合开源的优化建模软件方面，还有较长的路要走，**ROI** 包的出现意味着向整合的路迈出坚实的一步。优化建模的场景具有复杂性和多样性，算法实现更是五花八门，仅线性和整数线性优化方面，就至少有 **lpSolve**、**Rglpk** 和 **highs** (Schwendinger 和 Schumacher 2023) 等包，更别提非线性优化方面。这就又出现一个问题，对一个优化问题，采用何种算法及算法实

© 高等教育出版社

现具有最好的效果，满足可用性、可靠性。尽管涉及数学和统计，但高质量的软件工具更是一个工程问题。

从数据分析的角度来说，无论是 Python，还是 Julia，甚至于底层的 C++ 库，都不过是软件工具，首要问题是将实际问题转化为统计或数学模型，这需要抓住主要问题的关键因素，只有先做好建模的工作才能实现工具到商业价值的转化。

31.7 习题

1. 求解线性优化和整数线性优化的 R 包有很多，从使用语法、可求解的问题规模和问题类型比较 **lpSolve**、**Rglpk** 和 **highs** 等 R 包。

2. 求解非线性优化问题的 R 包有很多，其中有一些通过 **Rcpp** 包打包、调用 C++ 库，比如 **RcppEnsmallen**、**RcppNumerical** 等包，还有的 C++ 库提供头文件，可以在 C++ 环境中直接调用，比如 optim 库。通过 R 和 C++ 混合编程，一则引入更加庞大的开源社区，二则扩展求解非线性优化问题的规模和性能。请从求解问题类型、规模和性能等方面比较 5 个比较流行的 C++ 库。

3. 回顾凸二次优化一节，当矩阵 D 为半正定矩阵时，二次优化是非严格凸二次优化。调整示例里目标函数中的矩阵 D 使其行列式等于 0，其它条件不变。使用 **ROI** 包调用合适的优化求解器求解此类问题。

4. 求解如下 2 维非线性无约束优化问题。

$$\min_{\boldsymbol{x}} \quad 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

5. 求解如下 n 维非线性箱式约束优化问题。

$$\min_{\boldsymbol{x}} \quad \exp\big(-\sum_{i=1}^{n}(\frac{x_i}{\beta})^{2m}\big) - 2\exp(-\sum_{i=1}^{n} x_i^2)\prod_{i=1}^{n}\cos^2(x_i)$$

其中，$\beta. = 15, m = 3$ ，$x_i \in [-20, 20], i = 1, 2, \ldots, n$ 。请读者分别考虑 $n = 2$ 和 $n = 4$ 的情况。（全局最优解在 $x_i = 0, i = 1, 2, \ldots, n$ 处取得，最优值为 -1 。）

6. 求解如下非线性约束优化问题。

$$\begin{aligned}
\min_{\boldsymbol{x}} \quad & \exp(\sin(50x_1)) + \sin(60\exp(x_2)) + \sin(70\sin(x_1)) \\
& + \sin(\sin(80x_2)) - \sin(10(x_1 + x_2)) + \frac{(x_1^2 + x_2^2)^{\sin(x_2)}}{4} \\
\text{s.t.} \quad & \begin{cases} x_1 - \big((\cos(x_2))^{x_1} - x_1\big)^{x_2} \le 0 \\ -50 \le x_1, x_2 \le 50 \end{cases}
\end{aligned}$$

© 黄湘云

目标函数是不连续的，其函数图像如图 31.10 所示。（提示：容错能力低的求解器一般无法求解。Lingo 给出一个局部最优解 $(-46.14402, -0.8879601)$，目标函数值为 -2.645518，仅供参考。）

图 31.10: 目标函数的曲面图

7. 求解如下非线性约束优化问题。

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & x_1^2 \sin(x_2) + x_2^2 \cos(x_1) \\
\text{s.t.} \quad & \begin{cases} 1 \le 3x_1 - x_2 \le 3 \\ x_1 + x_2 \ge 2 \\ x_1 x_2 = 2 \\ \sin(x_1)\cos(x_2) \le 0.6 \\ x_1, x_2 \in (-100, 100). \end{cases}
\end{aligned}
$$

© 王建湖

第三十二章　优化问题

```
library(ROI)
library(ROI.plugin.glpk)
library(ROI.plugin.nloptr)
library(ROI.plugin.scs)
library(ROI.plugin.quadprog)
library(lattice)
# 自定义调色板
custom_palette <- function(irr, ref, height, saturation = 0.9) {
  hsv(
    h = height, s = 1 - saturation * (1 - (1 - ref)^0.5),
    v = irr
  )
}
```

32.1　旅行商问题

旅行商问题 The Traveling Salesman Problem 是一个混合整数线性规划问题，**TSP** 包 (Hahsler 和 Hornik 2007) 是求解此问题的最佳工具包。一般地，旅行商问题作如下定义。已知 n 个城市之间的距离，以矩阵 D 表示各个城市之间的距离，其元素 d_{ij} 表示城市 i 到城市 j 之间的距离，其对角元素 $d_{ii}=0$，其中 $i,j=1,2,\cdots,n$ 。一个旅行路线可以用 $\{1,2,\ldots,n\}$ 的循环排列 π 表示，$\pi(i)$ 表示在旅行线路中跟在城市 i 之后的城市。旅行商问题就是找一个排列 π 使得如下旅行线路最短。

$$\sum_{i=1}^{n} d_{i\pi(i)}$$

每个城市必须走到，且只能走一次。等价于如下整数规划问题，也是一个指派问题。

©黄湘云

$$
\begin{aligned}
\min \ & \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \\
\text{s.t.} \ & \sum_{i=1}^{n} x_{ij} = 1,\ j = 1, 2, \ldots, n, \\
& \sum_{j=1}^{n} x_{ij} = 1,\ i = 1, 2, \ldots, n, \\
& x_{ij} = 0 \text{ or } 1
\end{aligned}
$$

某人要去美国 10 个城市旅行，分别是亚特兰大 Atlanta、芝加哥 Chicago、丹佛 Denver 、休斯顿 Houston、洛杉矶 Los Angeles、迈阿密 Miami、纽约 New York、旧金山 San Francisco、西雅图 Seattle、华盛顿特区 Washington DC。10 个城市的分布如图 32.1 所示。从洛杉矶出发，最后回到洛杉矶，如何规划旅行线路使得总行程最短？行程最短的路径是什么？

图 32.1: 10 个城市的分布图

简单起见，这 10 个城市之间的距离以直线距离代替，R 内置的数据集 `UScitiesD` 已经记录了这 10 个城市之间的直线距离。`UScitiesD` 是一个 dist 类型的数据，可以用函数 `as.matrix()` 将其转化为矩阵类型。

```
data(UScitiesD)
D <- as.matrix(UScitiesD)
library(TSP)
D_tsp <- as.TSP(D)
# 出发城市洛杉矶
```

©黄湘云


```
tour_sol <- solve_TSP(x = D_tsp, method = "nearest_insertion", start = 5)
tour_sol
```

```
#> object of class 'TOUR'
#> result of method 'nearest_insertion' for 10 cities
#> tour length: 7373
```

途经 10 个城市的最短路程为 7373 。因采用启发式的随机优化算法，每次求解的结果可能会有所不同，建议运行多次，比较结果，选择最优的方法。

```
# 旅行最短路程
tour_length(tour_sol)
```

```
#> [1] 7373
```

```
# 旅行线路方案
as.integer(tour_sol)
```

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

```
labels(D_tsp)[as.integer(tour_sol)]
```

```
#>  [1] "LosAngeles"    "Houston"       "Miami"         "Atlanta"
#>  [5] "Washington.DC" "NewYork"       "Chicago"       "Denver"
#>  [9] "Seattle"       "SanFrancisco"
```

求解结果对应的旅行方案，如图 32.2 所示，依次走过的城市是：洛杉矶、旧金山、西雅图、丹佛、芝加哥、纽约、华盛顿特区、亚特兰大、迈阿密、休斯顿。

32.2 投资组合问题

作为一个理性的投资者，希望回报最大而风险最小，给定投资和回报的约束条件下，选择风险最小的组合。一个简单的马科维茨投资组合优化问题如下：

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \boldsymbol{w}^\top \hat{\Sigma} \boldsymbol{w} \\ \text{s.t.} \quad & A\boldsymbol{w}^\top \leq \boldsymbol{b} \end{aligned}$$

其中，$\boldsymbol{w}$ 是权重向量，每个分量代表对投资对象的投资比例，$\hat{\Sigma}$ 是关于投资对象的协方差矩阵，约束条件中包含两个部分，一个是权重之和为 1，一个是投资组合的收益率达到预期值。下面基于 12 个科技公司公开的股价数据介绍此组合优化问题。

首先利用 **quantmod** 包获取微软、谷歌、亚马逊、惠普、甲骨文、英特尔、威瑞森、eBay、AT&T、Apple、Adobe 和 IBM 等 12 支股票的历史股价数据。根据 2022-11-01 至 2022-12-01 期间的股票调整价，计算各支股票天粒度的收益率。收益率可以看作一个随机变量，收益率的波动变化，即随机变量的方差，可以看作风险。

©黄湘云

图 32.2: 10 个城市的路线图

```
# 12 支股票的收益率
tech_stock_return <- readRDS(file = "data/tech_stock_return.rds")
DD <- 100 * tech_stock_return
# 平均收益率
r <- mean(DD)
r
```

```
#> [1] 0.3476413
```

```
# 目标函数
foo <- Q_objective(Q = cov(DD), L = rep(0, ncol(DD)))
# 投资约束
full_invest <- L_constraint(rep(1, ncol(DD)), "==", 1)
# 回报约束
target_return <- L_constraint(apply(DD, 2, mean), "==", r)
# 目标规划
op <- OP(objective = foo, constraints = rbind(full_invest, target_return))
op
```

```
#> ROI Optimization Problem:
#>
#> Minimize a quadratic objective function of length 12 with
#> - 12 continuous objective variables,
```

©黄湘云

```
#>
#> subject to
#> - 2 constraints of type linear.
#> - 0 lower and 0 upper non-standard variable bounds.
```

求解器 nloptr.slsqp 需要给初值和等式约束的梯度，而求解器 quadprog 不需要给初值。下面使用 quadprog 来求解组合优化问题。

```
library(ROI.plugin.quadprog)
sol <- ROI_solve(op, solver = "quadprog")
# 最优解：投资组合
w <- sol$solution
# 保留 4 位小数
round(w, 4)
```

```
#>  [1] 0.0000 0.0000 0.0000 0.0000 0.3358 0.0000 0.0000 0.0000 0.3740 0.0000
#> [11] 0.0000 0.2902
```

```
# 目标函数值：投资风险
sqrt(t(w) %*% cov(DD) %*% w)
```

```
#>           [,1]
#> [1,] 0.9860861
```

求解出来的投资组合是甲骨文、AT&T 和 IBM，投资比例分别是 33.58% 、37.40% 和 29.02% 。以上 12 支股票都属于科技公司，收益率具有非常高的相关性，因此，最终选出来 3 支。

与给定预期回报而风险最小的组合优化问题相对应的是另一个问题：给定风险的约束条件下，获得预期回报最大的组合。即求解如下组合优化问题：

$$
\begin{aligned}
\max_{\boldsymbol{w}} \quad & \boldsymbol{w}^\top \hat{\boldsymbol{\mu}} \\
\text{s.t.} \quad & A\boldsymbol{w} \leq \boldsymbol{b} \\
& \boldsymbol{w}^\top \hat{\Sigma} \boldsymbol{w} \leq \sigma
\end{aligned}
$$

其中，目标函数中 $\hat{\boldsymbol{\mu}}$ 表示根据历史数据获得的投资对象的收益率，约束条件中 σ 表示投资者可以接受的投资风险，其他符号的含义同前。在给定风险约束 σ 下，求取回报最大的组合。线性约束也可以用函数 Q_constraint() 来表示，这样线性约束和二次约束可以整合在一起，代码如下：

```
# 风险阈值
sigma <- sqrt(t(w) %*% cov(DD) %*% w)
sigma
```

```
#>           [,1]
#> [1,] 0.9860861
```

统计学习 ©

```r
# 12 阶的全 0 矩阵
zero_mat <- diag(x = rep(0, ncol(DD)))
# 目标函数
foo <- Q_objective(Q = zero_mat, L = colMeans(DD))
# 线性和二次约束
maxret_constr <- Q_constraint(
  Q = list(cov(DD), NULL),
  L = rbind(
    rep(0, ncol(DD)),
    rep(1, ncol(DD))
  ),
 dir = c("<=", "=="), rhs = c(1/2 * sigma^2, 1)
)
# 目标规划
op <- OP(objective = foo, constraints = maxret_constr, maximum = TRUE)
op
```

```
#> ROI Optimization Problem:
#>
#> Maximize a quadratic objective function of length 12 with
#> - 12 continuous objective variables,
#>
#> subject to
#> - 2 constraints of type quadratic.
#> - 0 lower and 0 upper non-standard variable bounds.
```

函数 ROI_applicable_solvers() 识别规划问题类型，给出可求解此规划问题的求解器。

```r
ROI_applicable_solvers(op)
```

```
#> [1] "nloptr.cobyla" "nloptr.mma"    "nloptr.auglag" "nloptr.isres"
#> [5] "nloptr.slsqp"
```

quadprog 求解器不能求解该问题，尝试求解器 nloptr.slsqp ，12 支股票同等看待，所以，权重的初始值都设置为 $\frac{1}{12}$ 。

```r
# 求解规划问题
nlp <- ROI_solve(op, solver = "nloptr.slsqp", start = rep(1/12, 12))
# 投资组合
w <- nlp$solution
# 保留 4 位小数
round(w, 4)
```

```
#>  [1] 0.0000 0.0000 0.0000 0.0000 0.3358 0.0000 0.0000 0.0000 0.3740 0.0000
```

©黄湘云

```
#> [11] 0.0000 0.2902
```

```
# 投资组合的预期收益
w %*% colMeans(DD)
```

```
#>           [,1]
#> [1,] 0.3476413
```

结果显示，投资组合是甲骨文、AT&T 和 IBM，投资比例分别是 33.58% 、37.40% 和 29.02% 。

值得注意，当约束条件比较复杂，比如包含一些非线性的等式或不等式约束，可以用函数 `F_constraint()` 来表示，这更加的灵活，但需要传递（非）线性约束的雅可比向量或矩阵。用函数 `F_constraint()` 表示的代码如下，求解结果是一样的。

```
# x 是一个表示权重的列向量
# 等式约束
# 权重之和为 1 的约束
heq <- function(x) {
  sum(x)
}
# 等式约束的雅可比
heq.jac <- function(x) {
  rep(1, length(x))
}
# 不等式约束
# 二次的风险约束
hin <- function(x){
  1/2 * t(x) %*% cov(DD) %*% x
}
# 不等式约束的雅可比
hin.jac <- function(x){
  cov(DD) %*% x
}
# 目标规划
op <- OP(
  objective = L_objective(L = colMeans(DD)), # 12 个目标变量
  constraints = F_constraint(
    # 等式和不等式约束
    F = list(heq = heq, hin = hin),
    dir = c("==", "<="),
    rhs = c(1, 1/2 * sigma^2),
    # 等式和不等式约束的雅可比
    J = list(heq.jac = heq.jac, hin.jac = hin.jac)
```

内部版本

```
  ),
  # 目标变量的取值范围
  bounds = V_bound(ld = 0, ud = 1, nobj = 12L),
  maximum = TRUE # 最大回报
)
op
```

```
#> ROI Optimization Problem:
#>
#> Maximize a linear objective function of length 12 with
#> - 12 continuous objective variables,
#>
#> subject to
#> - 2 constraints of type nonlinear.
#> - 0 lower and 12 upper non-standard variable bounds.
```

```
# 求解规划问题
nlp <- ROI_solve(op, solver = "nloptr.slsqp", start = rep(1/12, 12))
# 投资组合
w <- nlp$solution
round(w, 4)
```

```
#>  [1] 0.0000 0.0000 0.0000 0.0000 0.3358 0.0000 0.0000 0.0000 0.3740 0.0000
#> [11] 0.0000 0.2902
```

```
# 投资组合的预期收益
w %*% colMeans(DD)
```

```
#>           [,1]
#> [1,] 0.3476413
```

32.3 高斯过程回归

高斯过程回归模型如下：

$$\boldsymbol{y}(x) = D\boldsymbol{\beta} + S(x)$$

其中，$\boldsymbol{\beta}$ 是一个 $p \times 1$ 维列向量，随机过程 $S(x)$ 是均值为零，协方差为 $V_{\boldsymbol{\theta}}$ 的平稳高斯过程，协方差矩阵 $V_{\boldsymbol{\theta}}$ 的元素如下：

$$\mathsf{Cov}\{S(x_i), S(x_j)\} = \sigma^2 \exp(-\|x_i - x_j\|/\phi)$$

©黄湘云

其中，$\boldsymbol{\theta} = (\sigma^2, \phi)$ 表示与协方差矩阵相关的参数，随机过程 $S(x)$ 的一个实现服从多元正态分布 $\mathrm{MVN}(\mathbf{0}, V_{\boldsymbol{\theta}})$ ，则 $\boldsymbol{y}(x)$ 也服从多元正态分布 $\mathrm{MVN}(D\boldsymbol{\beta}, V_{\boldsymbol{\theta}})$ 。参数 $\boldsymbol{\beta}$ 的广义最小二乘估计为 $\hat{\boldsymbol{\beta}}(\boldsymbol{\theta}) = (D^\top V_{\boldsymbol{\theta}}^{-1} D)^{-1} D^\top V_{\boldsymbol{\theta}}^{-1} \boldsymbol{y}$ ，关于参数 $\boldsymbol{\theta}$ 的剖面对数似然函数如下：

$$\log \mathcal{L}(\boldsymbol{\theta}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log(\det V_{\boldsymbol{\theta}}) - \frac{1}{2}\boldsymbol{y}^\top V_{\boldsymbol{\theta}}^{-1}(I - D(D^\top V_{\boldsymbol{\theta}}^{-1} D)^{-1} D^\top V_{\boldsymbol{\theta}}^{-1})\boldsymbol{y}$$

下面考虑一个来自 **MASS** 包真实数据 `topo`。topo 数据集最初来自 John C. Davis （1973 年）所著的书《Statistics and Data Analysis in Geology》。后来，J. J. Warnes 和 B. D. Ripley （1987 年）以该数据集为例指出空间高斯过程的协方差函数的似然估计中存在的问题 (Warnes 和 Ripley 1987)，并将其作为数据集 `topo` 放在 **MASS** 包里。Paulo J. Ribeiro Jr 和 Peter J. Diggle （2001 年）将该数据集打包成自定义的 `geodata` 数据类型，放在 **geoR** 包里，并在他俩合著的书《Model-based Geostatistics》中多次出现。`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。

```
library(MASS)
data(topo)
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 ...
```

根据 `topo` 数据集，$D = \mathbf{1}$ 是一个 52×1 的列向量，$\boldsymbol{\beta} = \beta$ 是一个截距项。设置参数初值 $(\sigma, \phi) = (65, 2)$ 。为了与 Ripley 的论文中的图比较，下面扔掉了对数似然函数中常数项，用 R 语言编码的似然函数如下：

```
log_lik <- function(x) {
  n <- nrow(topo)
  D <- t(t(rep(1, n)))
  Sigma <- x[1]^2 * exp(-as.matrix(dist(topo[, c("x", "y")])) / x[2])
  inv_Sigma <- solve(Sigma)
  P <- diag(1, n) - D %*% solve(t(D) %*% solve(Sigma, D), t(D)) %*% inv_Sigma
  as.vector(-1 / 2 * log(det(Sigma)) - 1 / 2 * t(topo[, "z"]) %*% inv_Sigma %*% P %*% topo[, "z"])
}
log_lik(x = c(65, 2))
```

```
#> [1] -207.1364
```

关于参数的偏导计算复杂，就不计算梯度了，下面调用 R 软件内置的 `nlminb` 优化器。发现，对不同的初始值，收敛到不同的位置，目标函数值非常接近。

```
op <- OP(
  objective = F_objective(log_lik, n = 2L),
```

```
  bounds = V_bound(lb = c(55, 5), ub = c(75, 8)),
  maximum = TRUE
)
nlp <- ROI_solve(op, solver = "nlminb", start = c(65, 2))
nlp$solution
```

```
#> [1] 65  5
```

```
nlp$objval
```

```
#> [1] -197.4197
```

如果初始值靠近局部极值点，则就近收敛到该极值点，比如初值 $(65, 7)$ ，$(70, 7.5)$ 。

```
nlp <- ROI_solve(op, solver = "nlminb", start = c(65, 7))
nlp$solution
```

```
#> [1] 65  7
```

```
nlp$objval
```

```
#> [1] -196.9407
```

```
nlp <- ROI_solve(op, solver = "nlminb", start = c(70, 7.5))
nlp$solution
```

```
#> [1] 70.0  7.5
```

```
nlp$objval
```

```
#> [1] -196.8441
```

尝试调用来自 **nloptr** 包的全局优化求解器 `nloptr.directL` ，大大小小的坑都跳过去了，结果还是比较满意的。

```
nlp <- ROI_solve(op, solver = "nloptr.directL")
nlp$solution
```

```
#> [1] 63.934407  6.121375
```

```
nlp$objval
```

```
#> [1] -196.8158
```

目标区域网格化，计算格点处的似然函数值，然后绘制似然函数图像。

```
dat <- expand.grid(
  sigma = seq(from = 55, to = 75, length.out = 41),
  phi = seq(from = 5, to = 8, length.out = 31)
)
dat$fn <- apply(dat, 1, log_lik)
```

©黄湘云

似然函数关于参数 (σ, ϕ) 的三维曲面见图 32.3 。

图 32.3: 对数似然函数的曲面图

等高线图呈现一道非常长且平滑的山岭 long flat ridge，山岭上布满许多局部极大值，普通的数值优化求解器常常陷入其中，只有全局优化求解器才可能找到全局极大值点。高斯过程回归模型的对数似然函数是非凸的，多模态的。

上图中没有看到许多局部极小值，与作者论文中的图 1 似乎不符。原因是什么？似然函数中涉及到的矩阵运算不精确，应该设计精度更高的运算方式？**lattice** 包绘图引擎无法展示更加细微的差异？还有一种解释，上图是对的，算法迭代时，对不同的初值，常常收敛到不同的结果，而这些不同的结果都位于岭上不同位置，对应的对数似然值却又几乎一样。

作为验证，下面调用 **nlme** 包的 `gls()` 函数拟合数据，参数的极大似然估计结果与全局优化求解器的结果比较一致。参数估计结果 $(\sigma, \phi) = (63.93429, 6.121352)$ ，对数似然函数值为 -244.6006 ，自编的似然函数 `log_lik()` 在最优解处的值为 -196.8158，再加上之前扔掉的常数项 `-52 / 2 * log(2 * pi)` ，就是 -244.6006 ，丝毫不差。

```
library(nlme)
fit_topo_ml <- gls(z ~ 1,
  data = topo, method = "ML",
  correlation = corExp(value = 65, form = ~ x + y)
```

©黄湘云

图 32.4: 对数似然函数的等高线图

```
)
summary(fit_topo_ml)

#> Generalized least squares fit by maximum likelihood
#>   Model: z ~ 1
#>   Data: topo
#>        AIC     BIC    logLik
#>   495.2012 501.055 -244.6006
#>
#> Correlation Structure: Exponential spatial correlation
#>  Formula: ~x + y
#>  Parameter estimate(s):
#>    range
#> 6.121352
#>
#> Coefficients:
#>               Value Std.Error  t-value p-value
#> (Intercept) 863.708  45.49859 18.98318       0
#>
#> Standardized residuals:
#>        Min         Q1        Med         Q3        Max
#> -2.7169766 -1.1919732 -0.5272282  0.1453374  1.5061096
```

```
#>
#> Residual standard error: 63.93429
#> Degrees of freedom: 52 total; 51 residual
```

如果使用限制极大似然估计，会发现参数估计结果与之相距甚远，而对数似然函数值相差无几。参数估计结果 $(\sigma, \phi) = (128.8275, 25.47324)$ 。

```
fit_topo_reml <- gls(z ~ 1,
  data = topo, method = "REML",
  correlation = corExp(value = 65, form = ~ x + y)
)
summary(fit_topo_reml)
```

```
#> Generalized least squares fit by REML
#>   Model: z ~ 1
#>   Data: topo
#>        AIC      BIC    logLik
#>   485.1558 490.9513 -239.5779
#>
#> Correlation Structure: Exponential spatial correlation
#>  Formula: ~x + y
#>  Parameter estimate(s):
#>    range
#> 25.47324
#>
#> Coefficients:
#>                Value Std.Error  t-value p-value
#> (Intercept) 877.8956  116.7163 7.521619       0
#>
#> Standardized residuals:
#>         Min          Q1         Med          Q3         Max
#> -1.45850507 -0.70167923 -0.37178079 -0.03800119  0.63732032
#>
#> Residual standard error: 128.8275
#> Degrees of freedom: 52 total; 51 residual
```

32.4 泊松混合分布

有限混合模型（Finite Mixtures of Distributions）的应用非常广泛，本节参考 **BB** 包 (Varadhan 和 Gilbert 2009) 的帮助手册，以泊松混合分布为例，介绍其参数的极大似然估计。更多详细的理论和算法介绍从略，感兴趣的读者可以查阅相关文献 (Hasselblad 1969)。**BB** 包比内置函数 `optim()` 功能更强，

©黄湘云

可以求解大规模非线性方程组，也可以求解带简单约束的非线性优化问题，还可以从多个初始值出发寻找全局最优解。

两个泊松分布以一定比例 p 混合，以概率 p 服从泊松分布 $\text{Poisson}(\lambda_1)$ ，而以概率 $1-p$ 服从泊松分布 $\text{Poisson}(\lambda_1)$ 。

$$p \times \text{Poisson}(\lambda_1) + (1-p) \times \text{Poisson}(\lambda_2)$$

泊松混合分布的概率密度函数 $f(x; p, \lambda_1, \lambda_2)$ 如下：

$$f(x; p, \lambda_1, \lambda_2) = p \times \frac{\lambda_1^x \exp(-\lambda_1)}{x!} + (1-p) \times \frac{\lambda_2^x \exp(-\lambda_2)}{x!}$$

随机变量 X 服从参数为 p 的伯努利分布 $X \sim \text{Bernoulli}(1, p)$ ，随机变量 Y 服从泊松混合分布，在伯努利分布的基础上，泊松混合分布也可作如下定义：

$$Y \sim \begin{cases} \text{Poisson}(\lambda_1), & \text{当 } X = 1 \text{ 时}, \\ \text{Poisson}(\lambda_2), & \text{当 } X = 0 \text{ 时}. \end{cases}$$

对数似然函数如下：

$$\ell(p, \lambda_1, \lambda_2) = \sum_{i=0}^{n} y_i \log\left(p \times \exp(-\lambda_1) \times \frac{\lambda_1^{x_i}}{x_i!} + (1-p) \times \exp(-\lambda_2) \times \frac{\lambda_2^{x_i}}{x_i!}\right)$$

下表格 32.1 数据来自 1947 年 Walter Schilling 发表在 JASA 的一篇文章 (Schilling 1947)。连续三年搜集伦敦《泰晤士报》刊登的死亡告示，每天的告示发布 80 岁及以上女性死亡人数。经过汇总统计，发现，在三年里，没有人死亡的告示出现 162 次，死亡 1 人的告示出现 267 次。

表格 32.1: 死亡人数的统计

死亡人数	0	1	2	3	4	5	6	7	8	9
发生频次	162	267	271	185	111	61	27	8	3	1

考虑到夏季和冬季对老人死亡率的影响是不同的，因此，引入泊松混合分布来对数据建模。

```
# 对数似然函数
# p 是一个长度为 3 的向量
# y 是观测数据向量
poissmix_loglik <- function(p, y) {
  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)
```

©黄湘云

```
}
# lgamma(i + 1) 表示整数 i 的阶乘的对数
# 参数的下限
lo <- c(0, 0, 0)
# 参数的上限
hi <- c(1, Inf, Inf)
# 随机生成一组参数初始值
p0 <- runif(3, c(0.2, 1, 1), c(0.8, 5, 8))
# 汇总统计出来的死亡人数的频次分布
y <- c(162, 267, 271, 185, 111, 61, 27, 8, 3, 1)
```

调用 **BB** 包的函数 `BBoptim()` 求解多元非线性箱式约束优化问题。

```
library(BB)
# 参数估计
ans <- BBoptim(
  par = p0, fn = poissmix_loglik, y = y,
  lower = lo, upper = hi,
  control = list(maximize = TRUE)
)
```

```
#> iter:  0  f-value:  -2914.957  pgrad:  55.03288
#> iter:  10  f-value:  -2008.118  pgrad:  5.18969
#> iter:  20  f-value:  -1997.529  pgrad:  2.464499
#> iter:  30  f-value:  -1989.96  pgrad:  0.2905813
#> iter:  40  f-value:  -1989.946  pgrad:  0.002119123
#>   Successful convergence.
```

```
ans
```

```
#> $par
#> [1] 0.359883 1.256091 2.663401
#>
#> $value
#> [1] -1989.946
#>
#> $gradient
#> [1] 2.273737e-06
#>
#> $fn.reduction
#> [1] -925.0115
#>
#> $iter
```

©梁文清

```
#> [1] 49
#>
#> $feval
#> [1] 51
#>
#> $convergence
#> [1] 0
#>
#> $message
#> [1] "Successful convergence"
#>
#> $cpar
#> method      M
#>      2     50
```

numDeriv::hessian 计算极大似然点的黑塞矩阵，然后计算参数估计的标准差。

```
# 黑塞矩阵
hess <- numDeriv::hessian(x = ans$par, func = poissmix_loglik, y = y)
hess
```

```
#>           [,1]       [,2]       [,3]
#> [1,] -907.1103  270.22857  341.25428
#> [2,]  270.2286 -113.47939  -61.68192
#> [3,]  341.2543  -61.68192 -192.78212
```

```
# 标准差
se <- sqrt(diag(solve(-hess)))
se
```

```
#> [1] 0.1946833 0.3500299 0.2504766
```

multiStart 从不同初始值出发寻找全局最大值，先找一系列局部极大值，通过比较获得全局最大值。

```
# 随机生成 10 组初始值
p0 <- matrix(runif(30, c(0.2, 1, 1), c(0.8, 8, 8)),
             nrow = 10, ncol = 3, byrow = TRUE)
ans <- multiStart(
  par = p0, fn = poissmix_loglik, action = "optimize",
  y = y, lower = lo, upper = hi, quiet = TRUE,
  control = list(maximize = TRUE, trace = FALSE)
)
# 筛选出迭代收敛的解
pmat <- round(cbind(ans$fvalue[ans$conv], ans$par[ans$conv, ]), 4)
```

© 基础统计


```
dimnames(pmat) <- list(NULL, c("fvalue", "parameter 1",
                               "parameter 2", "parameter 3"))
# 去掉结果一样的重复解
pmat[!duplicated(pmat), ]
```

```
#>         fvalue parameter 1 parameter 2 parameter 3
#> [1,] -2000.833      0.3638      2.2914      2.0734
#> [2,] -1989.946      0.3599      1.2561      2.6634
#> [3,] -2000.011      0.0108      6.5471      2.1165
#> [4,] -1989.946      0.6401      2.6634      1.2561
#> [5,] -1989.946      0.3599      1.2560      2.6634
```

32.5 极大似然估计

一元函数最优化问题和求根问题是相关的。在统计应用中，二项分布的比例参数的置信区间估计涉及求根，伽马分布的参数的极大似然估计涉及求根。下面介绍求根在估计伽马分布的参数中的应用。

形状参数为 α 和尺度参数为 σ 的伽马分布的概率密度函数 $f(x;\alpha,\sigma)$ 如下：

$$f(x;\alpha,\sigma)=\frac{1}{\sigma^{\alpha}\Gamma(\alpha)}x^{\alpha-1}\exp(-\frac{x}{\sigma}),\quad \alpha\geq 0,\sigma>0$$

其中，$\Gamma(\cdot)$ 表示伽马函数，伽马分布的均值为 $\alpha\sigma$ ，方差为 $\alpha\sigma^2$ 。下图 32.5 展示两个伽马分布的概率密度函数，形状参数分别为 5 和 9，尺度参数均为 1，即伽马分布 $f(x;5,1)$ 和 $f(x;9,1)$ 。

给定一组来自伽马分布的样本 $x_1,x_2,\ldots,x_n$ ，关于参数 α 和 σ 的似然函数如下：

$$\mathcal{L}(\alpha,\sigma)=(\frac{1}{\sigma^{\alpha}\Gamma(\alpha)})^n(\prod_{i=1}^n x_i)^{\alpha-1}\exp(-\frac{\sum_{i=1}^n x_i}{\sigma})$$

则，其对数似然函数如下：

$$\ell(\alpha,\sigma)=-n\big(\alpha\log(\sigma)+\log\Gamma(\alpha)\big)+(\alpha-1)\sum_{i=1}^n\log(x_i)-\frac{\sum_{i=1}^n x_i}{\sigma}$$

对数似然函数关于参数 α 和 σ 的偏导数如下：

$$\begin{aligned}\frac{\partial\ell(\alpha,\sigma)}{\partial\alpha}&=-n\Big(\log(\sigma)+\big(\log\Gamma(\alpha)\big)'\Big)+\sum_{i=1}^n\log(x_i)=0\\ \frac{\partial\ell(\alpha,\sigma)}{\partial\sigma}&=-\frac{n\alpha}{\sigma}+\frac{\sum_{i=1}^n x_i}{\sigma^2}=0\end{aligned}$$

根据第二个式子可得 $\sigma=\frac{1}{n\alpha}\sum_{i=1}^n x_i$ ，将其代入第一个式子可得

©黄湘云

图 32.5: 伽马分布的概率密度函数

$$\log(\alpha) - \big(\log\Gamma(\alpha)\big)' = \log\big(\frac{1}{n}\sum_{i=1}^{n} x_i\big) - \frac{1}{n}\sum_{i=1}^{n}\log(x_i)$$

```
set.seed(20232023)
x <- rgamma(1000, shape = 1.5, scale = 2)
# 形状参数和尺度参数的矩估计
c(mean(x)^2 /var(x), var(x)/mean(x))
```

```
#> [1] 1.636030 1.902239
```

```
# 极大似然估计
# 常量
cc <- log(mean(x)) - mean(log(x))
# 方程
fun <- function(alpha){
  log(alpha) - digamma(alpha) - cc
}
# 找根
uniroot(f = fun, interval = c(1, 3))
```

```
#> $root
#> [1] 1.610272
#>
```

© 统计出版社


```
#> $f.root
#> [1] 2.825244e-09
#>
#> $iter
#> [1] 6
#>
#> $init.it
#> [1] NA
#>
#> $estim.prec
#> [1] 6.103516e-05
```

求得形状参数的估计 $\alpha = 1.610272$ ，进而，可得尺度参数的估计 $\sigma = 1.932667$ 。

函数 `uniroot()` 只能找到方程的一个根，**rootSolve** 包采用牛顿-拉弗森（Newton-Raphson ）算法找一元非线性方程（组）的根，特别适合有多个根的情况。

```
library(rootSolve)
# 非线性方程（组）的根
multiroot(f = fun, start = 1.2)
```

```
#> $root
#> [1] 1.610272
#>
#> $f.root
#> [1] 3.121097e-10
#>
#> $iter
#> [1] 5
#>
#> $estim.precis
#> [1] 3.121097e-10
```

```
# 搜索一个方程在区间内所有的根
uniroot.all(f = fun, interval = c(1, 3))
```

```
#> [1] 1.610339
```

32.6 习题

1. 某人要周游美国各州，从纽约出发，走遍 50 个州的行政中心，最后回到纽约。规划旅行线路使得总行程最短。Base R 内置的 R 包 **datasets** 包含美国 50 个州的地理中心数据 `state.center` 。

© 王湘君

2. 有限混合模型也常用 EM 算法来估计参数，美国黄石公园老忠实间歇泉的喷发规律近似为二维高斯混合分布，请读者以 R 软件内置的数据集 `faithful` 为基础，采用 EM 算法估计参数。

3. 获取百度、阿里、腾讯、京东、美团、滴滴、字节、360、网易、新浪等 10 支股票的历史股价数据。根据 2021-12-01 至 2022-12-01 股票的调整价计算 12 个月的股价收益率，根据月度股价收益率和波动率数据，设置投资组合，使得月度收益率不低于 2%。股票代码以数字编码和 HK 结尾的为港股代码，有的公司在美股和港股上都有。可以用 **quantmod** 包下载各个公司的股价数据，下载拼多多股价数据的代码如下：

```
quantmod::getSymbols("PDD", auto.assign = FALSE, src = "yahoo")
```

表格 32.2: 一些互联网公司及股票代码

公司	美团	阿里巴巴	京东	百度	腾讯	拼多多	京东	阿里巴巴
股票代码	3690.HK	9988.HK	9618.HK	9888.HK	0700.HK	PDD	JD	BABA

© 黄湘云

第七部分

贝叶斯建模

© 贝叶斯

第三十三章　概率推理框架

本章的目的是让读者快速熟悉和上手，主要分为以下几个部分。

1. Stan 的概览，介绍 Stan 是什么，怎么样。
2. Stan 的入门，以推理一个正态分布均值参数为例，从基础语法、类型声明和代码结构三个方面介绍 Stan 的使用。
3. 选择先验分布，先验分布在贝叶斯推理中的重要性不言而喻，本节以一个简单的广义线性模型为例，介绍常见的几个先验分布对模型的影响。
4. 选择推理算法，接上一节的例子，围绕怎么用、效果如何介绍 Stan 内置的几个推理算法。

33.1　Stan 概览

Stan 是一个贝叶斯统计建模和计算的概率推理框架，也是一门用于贝叶斯推断和优化的概率编程语言 (Gelman, Lee, 和 Guo 2015; Carpenter 等 2017)。它使用汉密尔顿蒙特卡罗算法（Hamiltonian Monte Carlo algorithm ，简称 HMC 算法）抽样，内置一种可以自适应调整采样步长的 No-U-Turn sampler（简称 NUTS 采样器）。Stan 还提供自动微分变分推断（Automatic Differentiation Variational Inference algorithm 简称 ADVI 算法）算法做近似贝叶斯推断获取参数的后验分布，以及拟牛顿法（the limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm 简称 L-BFGS 算法）优化算法获取参数的惩罚极大似然估计。

经过 10 多年的发展，Stan 已经形成一个相对成熟的生态，它提供统计建模、数据分析和预测能力，广泛应用于社会、生物、物理、工程、商业等领域，在学术界和工业界的影响力也不小。下图 33.1 是 Stan 生态中各组件依赖架构图，math 库 (Carpenter 等 2015) 是 Stan 框架最核心的组件，它基于 Boost 、Eigen 、OpenCL 、SUNDIALS 和 oneTBB 等诸多 C++ 库，提供概率推理、自动微分、矩阵计算、并行计算、GPU 计算和求解代数微分方程等功能。

CmdStan 是 Stan 的命令行接口，可在 MacOS / Linux 的终端软件，Windows 的命令行窗口或 PowerShell 软件中使用。**CmdStanR** (Gabry, Češnovar, 和 Johnson 2023)、CmdStanPy 和 MathematicaStan 分别是 CmdStan 的 R 语言、Python 语言和 Mathematica 语言接口。每次当 Stan 发布新版本时，CmdStan 也会随之发布新版，只需指定新的 CmdStan 安装路径，**CmdStanR** 就可

© 黄湘云

图 33.1: Stan、CmdStan 和 CmdStanR 等的依赖关系图

以使用上，**CmdStanR** 包与 Stan 是相互独立的更新机制。**CmdStanR** 负责处理 CmdStan 运行的结果，而编译代码，生成模型和模拟采样等都是由 **CmdStan 完成**。入门 **CmdStanR** 后，可以快速转入对 Stan 底层原理的学习，有利于编码符合实际需要的复杂模型，有利于掌握常用的炼丹技巧，提高科研和工作的效率。

此外，**bayesplot** 包 (Gabry 等 2019) 针对 cmdstanr 包生成的拟合模型对象提供一系列可视化图形，用于诊断采样过程、展示后验分布等。**loo** 包 (Vehtari, Gelman, 和 Gabry 2017) 计算 LOOIC （留一交叉验证信息准则）和 WAIC （通用信息准则）等指标，用于模型评估与比较。**posterior** 包 (Vehtari 等 2021) 对采样数据提供统一的操作方法和类型转化，计算常用的后验分布的统计量等。**projpred** 包 (Piironen 和 Vehtari 2017a; Piironen, Paasiniemi, 和 Vehtari 2020) 实现投影预测推断用于模型预测和特征选择。

rstan 包 (Stan Development Team 2023a) 是 Stan 的 R 语言接口，该接口依赖 **Rcpp** (Eddelbuettel 和 François 2011; Eddelbuettel 和 Balamuta 2018)、**RcppEigen** (D. Bates 和 Eddelbuettel 2013)、**BH** (Eddelbuettel, Emerson, 和 Kane 2023)、**RcppParallel** (Allaire 等 2023) 和 **StanHeaders** (Stan Development Team 2023b) 等 R 包，由于存在众多上游 R 包依赖和兼容性问题，尤其在 Windows 系统环境中，因此，**RStan 的**安装、更新都比较麻烦。**RStan** 的更新通常严重滞后于 Stan 的更新，不利于及时地使用最新的学术研究成果。而相比于 **rstan** 包，**CmdStanR** 更加轻量，可以更快地将 CmdStan 的新功能融入进来，而且 **cmdstanr** 和 CmdStan 是分离的，方便用户升级和维护。

rstanarm (Goodrich 等 2023) 和 **brms** (Bürkner 2017) 是 **RStan** 的扩展包，各自提供了一套用于表示统计模型的公式语法。它们都支持丰富的统计模型，比如线性模型、广义线性模型、线性混合效应模

©曹志浩

型、广义线性混合效应模型等。相比于 **rstan**，它们使用起来更加方便，因为它内置了大量统计模型的 Stan 实现，即将公式语法翻译成 Stan 编码的模型，然后调用 **rstan** 或 **cmdstanr** 翻译成 C++，最后编译成动态链接库。除了依赖 **rstan** 包，**rstanarm** 和 **brms** 还依赖大量其它 R 包。

顺便一提，类似的用于概率推理和统计分析的框架，还有 Python 社区的 PyMC (Abril-Pla O 2023) 和 TensorFlow Probability (Joshua V. Dillon 2017)，它们采用的 MCMC 采样算法也是基于 NUTS 的 HMC 算法。

33.2 Stan 入门

33.2.1 Stan 的基础语法

下面以一个简单示例介绍 Stan 的用法，包括 Stan 的基本用法、变量类型、代码结构等，

考虑一个已知方差的正态分布，设 $-3,-2,-1,0,1,2,3$ 是取自正态分布 $\mathcal{N}(\mu,1)$ 的一个样本，也是取自该正态分布的一组随机数。现在的问题是估计该正态分布的均值参数 μ 。Stan 编码的正态分布模型如下：

```
transformed data {
  vector[7] y = [-3, -2, -1, 0, 1, 2, 3]';
}
parameters {
  real mu;
}
model {
  y ~ normal(mu, 1);
}
```

- `transformed data` 代码块是一组已知的数据，这部分数据是不需要从外部传递进来的。这个样本是以向量存储的，需要声明向量的长度和类型（默认类型是实数），每一行以分号结尾，这与 C++ 的语法一样。
- `parameters` 代码块是未知的参数，需要声明各个参数的类型。这里只有一个参数，且只是一个未知的实数，声明类型即可。
- `model` 代码块是抽样语句表示的模型结构，符号 ~ 表示服从的意思，函数 `y ~ normal(mu, 1)` 是正态分布的抽样语句。

接下来，编译 Stan 代码，准备参数初值，配置采样的参数。首先加载 **cmdstanr** 包，设置 2 条迭代链，给每条链设置相同的参数初始值。代码编译后，生成一个模型对象 `mod_gaussian`，接着，调用方法 `sample()` ，传递迭代初值 `init`，初始化阶段的迭代次数 `iter_warmup` ，采样阶段的迭代次数 `iter_sampling`，采样的链条数 `chains` 及并行时分配的 CPU 核心数 `parallel_chains` ，随机数种子 `seed` 。

©黄湘云

```r
library(cmdstanr)
nchains <- 2 # 2 条迭代链
# 给每条链设置相同的参数初始值
inits_data_gaussian <- lapply(1:nchains, function(i) {
  list(
    mu = 1
  )
})

fit_gaussian <- mod_gaussian$sample(
  init = inits_data_gaussian,   # 迭代初值
  iter_warmup = 200,            # 每条链初始化迭代次数
  iter_sampling = 200,          # 每条链采样迭代次数
  chains = nchains,         # 马尔科夫链的数目
  parallel_chains = nchains,# 指定 CPU 核心数，可以给每条链分配一个
  seed = 20232023           # 设置随机数种子，不要使用 set.seed() 函数
)
```

```
#> Running MCMC with 2 parallel chains...
#>
#> Chain 1 Iteration:   1 / 400 [  0%]  (Warmup)
#> Chain 1 Iteration: 100 / 400 [ 25%]  (Warmup)
#> Chain 1 Iteration: 200 / 400 [ 50%]  (Warmup)
#> Chain 1 Iteration: 201 / 400 [ 50%]  (Sampling)
#> Chain 1 Iteration: 300 / 400 [ 75%]  (Sampling)
#> Chain 1 Iteration: 400 / 400 [100%]  (Sampling)
#> Chain 2 Iteration:   1 / 400 [  0%]  (Warmup)
#> Chain 2 Iteration: 100 / 400 [ 25%]  (Warmup)
#> Chain 2 Iteration: 200 / 400 [ 50%]  (Warmup)
#> Chain 2 Iteration: 201 / 400 [ 50%]  (Sampling)
#> Chain 2 Iteration: 300 / 400 [ 75%]  (Sampling)
#> Chain 2 Iteration: 400 / 400 [100%]  (Sampling)
#> Chain 1 finished in 0.0 seconds.
#> Chain 2 finished in 0.0 seconds.
#>
#> Both chains finished successfully.
#> Mean chain execution time: 0.0 seconds.
#> Total execution time: 0.4 seconds.
```

默认情况下，采样过程中会输出一些信息，以上是 2 条链并行采样的过程，给出百分比进度及时间消耗。采样完成后，调用方法 summary() 汇总和展示采样结果。

```
fit_gaussian$summary()
```

```
#> # A tibble: 2 x 10
#>   variable     mean   median    sd   mad      q5     q95  rhat ess_bulk ess_tail
#>   <chr>       <dbl>    <dbl> <dbl> <dbl>   <dbl>   <dbl> <dbl>    <dbl>    <dbl>
#> 1 lp__     -14.4    -14.2    0.708 0.212 -15.6   -14.0    1.01     177.     220.
#> 2 mu         0.0365   0.0307 0.348 0.298  -0.511   0.572  1.01     229.     173.
```

输出模型中各个参数的后验分布的一些统计量，如均值（mean）、中位数（median）、标准差（sd），0.05 分位点（q5），0.95 分位点（q95）等。此外，还有 `lp__` 后验对数概率密度值，每个模型都会有该值。`summary()` 方法有一些参数可以控制数字的显示方式和精度。下面展示的是保留 4 位有效数字的结果。

```
fit_gaussian$summary(.num_args = list(sigfig = 4, notation = "dec"))
```

```
#> # A tibble: 2 x 10
#>   variable      mean    median     sd    mad       q5      q95  rhat ess_bulk
#>   <chr>        <dbl>     <dbl>  <dbl>  <dbl>    <dbl>    <dbl> <dbl>    <dbl>
#> 1 lp__     -14.43    -14.15    0.7083 0.2118 -15.60   -14.00   1.007    177.2
#> 2 mu         0.03647   0.03073 0.3476 0.2978  -0.5115   0.5722 1.007    229.0
#> # i 1 more variable: ess_tail <dbl>
```

接下来，要介绍 Stan 代码中的保留字 target 的含义，因为它在 Stan 代码中很常见，与输出结果中的 `lp__` 一行紧密相关。

- `lp__` 表示后验概率密度函数的对数。
- target 累加一些和参数无关的数不影响参数的估计，但影响 `lp__` 的值。
- 抽样语句表示模型会扔掉后验概率密度函数的对数的常数项。

```
library(ggplot2)
library(bayesplot)
mcmc_hist(fit_gaussian$draws("lp__")) +
  theme_classic()
```

©黄湘云

图 33.2: lp___ 的后验分布

为此，不妨在之前的 Stan 代码的基础上添加两行，新的 Stan 代码如下：

```
transformed data {
  vector[7] y = [-3, -2, -1, 0, 1, 2, 3]';
}
parameters {
  real mu;
}
model {
  y ~ normal(mu, 1);
  target += 12345;
  target += mean(exp(y));
}
```

接着，再次编译代码、采样，为了节约篇幅，设置两个参数 `show_messages` 和 `refresh` ，不显示中间过程和采样进度。其它参数设置不变，代码如下：

```
fit_gaussian <- mod_gaussian_target$sample(
  init = inits_data_gaussian,
  iter_warmup = 200,
  iter_sampling = 200,
  chains = nchains,
```

©草稿 仅供

```
  parallel_chains = nchains,
  show_messages = FALSE,    # 不显示中间过程
  refresh = 0,              # 不显示采样进度
  seed = 20232023
)
fit_gaussian$summary(.num_args = list(sigfig = 4, notation = "dec"))
```

```
#> # A tibble: 2 x 10
#>   variable        mean      median     sd    mad         q5        q95  rhat
#>   <chr>          <dbl>       <dbl>  <dbl>  <dbl>      <dbl>      <dbl> <dbl>
#> 1 lp__     12335.      12335.      0.7074 0.1483 12334.     12336.     1.008
#> 2 mu           0.03647     0.03073 0.3476 0.2978    -0.5115     0.5722 1.007
#> # i 2 more variables: ess_bulk <dbl>, ess_tail <dbl>
```

可以清楚地看到 `lp__` 的值发生了变化，而参数 `mu` 的值没有变化。这是因为抽样语句 `y ~ normal(mu, 1);` 隐含一个 `lp__` ，target 指代 `lp__` 的值，符号 `+=` 表示累加。两次累加后得到 12335.09。

```
model {
  y ~ normal(mu, 1);
  target += 12345;
  target += mean(exp(y));
}
```

```
y <- c(-3, -2, -1, 0, 1, 2, 3)
12345 + mean(exp(y)) - 14.45
```

```
#> [1] 12335.09
```

下面从概率密度函数出发，用 R 语言来计算逐点对数似然函数值。一般地，不妨设 $x_1, x_2, \cdots, x_n$ 是来自正态总体 $\mathcal{N}(\mu, 1)$ 的一个样本。则正态分布的概率密度函数 $f(x)$ 的对数如下：

$$\log f(x) = \log \frac{1}{\sqrt{2\pi}} - \frac{(x-\mu)^2}{2}$$

已知参数 μ 是一个非常接近 0 的数，不妨将 $\mu = 0$ 代入计算。

```
sum(dnorm(x = y, mean = 0, sd = 1, log = TRUE))
```

```
#> [1] -20.43257
```

去掉常数项后，计算概率密度函数值的对数和。

```
# 扔掉常数
f <- function(y, mu) {
  return(-0.5 * (y - mu)^2)
}
sum(f(-3:3, 0))
```

```
#> [1] -14
```

这就比较接近原 lp__ 的值了，所以，lp__ 表示后验概率密度函数的对数，扔掉了与参数无关的常数项。若以概率密度函数的对数 normal_lpdf 替代抽样语句，则常数项是保留的。normal_lpdf 是 Stan 内置的函数，输入值为随机变量的取值 y 、位置参数 mu 和尺度参数 sigma，返回值为 real 实数。

```
real normal_lpdf(reals y | reals mu, reals sigma)
```

```
transformed data {
  vector[7] y = [-3, -2, -1, 0, 1, 2, 3]';
}
parameters {
  real mu;
}
model {
  target += normal_lpdf(y | mu, 1);
}
```

接着，编译上述代码以及重复采样的步骤，参数设置也一样。

```
fit_gaussian <- mod_gaussian_lpdf$sample(
  init = inits_data_gaussian,
  iter_warmup = 200,
  iter_sampling = 200,
  chains = nchains,
  parallel_chains = nchains,
  show_messages = FALSE,
  refresh = 0,
  seed = 20232023
)
fit_gaussian$summary(.num_args = list(sigfig = 4, notation = "dec"))
```

```
#> # A tibble: 2 x 10
#>   variable      mean    median     sd    mad       q5      q95  rhat ess_bulk
#>   <chr>        <dbl>     <dbl>  <dbl>  <dbl>    <dbl>    <dbl> <dbl>    <dbl>
#> 1 lp__     -20.86    -20.58    0.7083 0.2119 -22.03   -20.43   1.007    176.7
#> 2 mu         0.03647   0.03073 0.3476 0.2978  -0.5115   0.5722 1.007    229.0
#> # i 1 more variable: ess_tail <dbl>
```

可以看到，此时 lp__ 的值包含常数项，两种表示方式对参数的计算结果没有影响。

©黄湘云

33.2.2 Stan 的变量类型

Stan 语言和 C/C++ 语言比较类似，变量需要先声明再使用，函数需要用 `return` 返回值，总而言之，类型声明比较严格。变量的声明没有太多的内涵，就是 C++ 和 Stan 定义的语法，比如整型用 `int` 声明。建模过程中，时常需要将 R 语言环境中的数据传递给 Stan 代码编译出来的模型，而 Stan 是基于 C++ 语言，在变量类型方面有继承有发展。下表给出 Stan 与 R 语言中的变量类型对应关系。值得注意，R 语言的类型检查是不严格的，使用变量也不需要提前声明和初始化。Stan 语言中向量、矩阵的类型都是实数，下标也从 1 开始，元组类型和 R 语言中的列表类似，所有向量默认都是列向量。

下表第一列表示 Stan 语言的变量类型，第二列给出使用该变量的声明示例，第三列给出 R 语言中构造该类型变量的示例。

表格 33.1: Stan 变量类型和 R 语言中的对应

类型	Stan 语言	R 语言
整型	`int x = 1;`	`x = 1L`
实数	`real x = 3.14;`	`x = 3.14`
向量	`vector[3] x = [1, 2, 3]';`	`x = c(1, 2, 3)`
矩阵	`matrix[3,1] x;`	`matrix(data = c(1, 2, 3), nrow = 3)`
数组	`array[3] int x;`	`array(data = c(1L, 2L, 3L), dim = c(3, 1, 1))`
元组	`tuple(vector[3],vector[3]) x;`	`list(x = c(1, 2, 3), y = c(4, 5, 6))`

33.2.3 Stan 的代码结构

Stan 代码文件最多有如下 7 块内容，模拟、拟合和预测模型会用到其中的一部分或全部。

```
functions {
  // ... function declarations and definitions ...
}
data {
  // ... declarations ...
}
transformed data {
   // ... declarations ... statements ...
}
parameters {
   // ... declarations ...
}
transformed parameters {
   // ... declarations ... statements ...
}
```

```
model {
   // ... declarations ... statements ...
}
generated quantities {
   // ... declarations ... statements ...
}
```

33.2.4 Stan 的函数使用

Stan 有大量的内建函数，然而，有时候，Stan 内建的函数不足以满足需求，需要自己创建函数。下面以函数 `cholesky_decompose` 为例介绍 Stan 内置/一般函数的调用，在该函数的基础上自定义函数 `cholesky_decompose2` ，这不过是对它改个名字，其它内容只要符合 Stan 语言即可，不甚重要。

根据 Stan 官网函数 `cholesky_decompose` 帮助文档，Cholesky 分解的形式（Cholesky 分解有多种形式）如下：

$$M = LL^\top$$

M 是一个对称正定的矩阵，而 L 是一个下三角矩阵。函数 `cholesky_decompose` 有一个参数 A，A 需要传递一个对称正定的矩阵。不妨设这个对称正定的矩阵为

$$M = \begin{bmatrix} 4 & 1 \\ 1 & 1 \end{bmatrix}$$

```
# 准备函数
stan_file <- write_stan_file("
functions {
 matrix cholesky_decompose2(matrix A) {
   return cholesky_decompose(A);
 }
}
parameters {
 real x;
}
model {
 x ~ std_normal();
}
")
```

接着，将以上 Stan 代码编译

```
mod_cholesky_decompose <- cmdstan_model(stan_file = stan_file, compile = TRUE)
```

准备测试数据，只要是一个对称正定的矩阵都可以做 cholesky 分解。

```
# 测试矩阵
M <- rbind(c(4, 1), c(1, 1))
```

cmdstanr 包导出函数的方法将以上 Stan 代码中的函数部分独立导出。

```
# 编译独立的函数
mod_cholesky_decompose$expose_functions()
```

现在，可以直接调用导出的函数 `cholesky_decompose2` 。

```
# cholesky 分解
mod_cholesky_decompose$functions$cholesky_decompose2(A = M)
```

```
#>      [,1]      [,2]
#> [1,]  2.0 0.0000000
#> [2,]  0.5 0.8660254
```

最后，将 Stan 函数计算的结果与 R 语言内置的 cholesky 分解函数的结果比较。发现，函数 `chol()` 的结果正好是 `cholesky_decompose2` 的转置。

```
chol(M)
```

```
#>      [,1]      [,2]
#> [1,]    2 0.5000000
#> [2,]    0 0.8660254
```

查看帮助文档，可知 R 软件对 Cholesky 分解的定义如下：

$$M = L^\top L$$

根据数学表达式，感觉上都是差不多的，但还是有差异。R 与 Stan 混合编程就需要注意这些表达上不同的，不然，排错会很麻烦。

提示

StanHeaders 可以编译和调用 Stan 的内置的数学函数，比如 Cholesky 分解函数 `cholesky_decompose` 。

```
library(StanHeaders)
stanFunction("cholesky_decompose", A = M)
#>      [,1]      [,2]
#> [1,]  2.0 0.0000000
#> [2,]  0.5 0.8660254
```

可以看到，结果和前面一样。

©黄湘云

33.3 先验分布

考虑一个响应变量服从伯努利分布的广义线性模型。

$$\boldsymbol{y} \sim \text{Bernoulli}(\boldsymbol{p})$$
$$\text{logit}(\boldsymbol{p}) = \log(\frac{\boldsymbol{p}}{1-\boldsymbol{p}}) = \alpha + X\boldsymbol{\beta}$$

下面模拟生成 2500 个样本，其中 10 个正态协变量，非 0 的回归系数是截距 $\alpha = 1$ 和向量 $\boldsymbol{\beta}$ 中的 $\beta_1 = 3, \beta_2 = -2$ 。对模型实际有用的是 3 个变量，采用贝叶斯建模，其它变量应该被收缩掉。贝叶斯收缩（Bayesian shrinkage）与变量选择（Variable selection）是有关系的，先验分布影响收缩的力度。

```
set.seed(2023)
n <- 2500
k <- 10
X <- matrix(rnorm(n * k), ncol = k)
y <- rbinom(n, size = 1, prob = plogis(1 + 3 * X[, 1] - 2 * X[, 2]))
# 准备数据
mdata <- list(k = k, n = n, y = y, X = X)
```

在贝叶斯先验分布中，有几个常用的概率分布，分别是正态分布、拉普拉斯分布（双指数分布）、柯西分布，下图集中展示了这几个的标准分布。

```
#> Warning: A numeric `legend.position` argument in `theme()` was deprecated in ggplot2
#> 3.5.0.
#> i Please use the `legend.position.inside` argument of `theme()` instead.
```

接下来，考虑几种常见的先验设置。

33.3.1 正态先验

指定回归系数 α, β 的先验分布如下

$$\alpha \sim \mathcal{N}(0, 1000)$$
$$\beta \sim \mathcal{N}(0, 1000)$$

正态分布中设置相当大的方差意味着分布相当扁平，α, β 的取值在区间 $(-\infty, +\infty)$ 上比较均匀。

```
data {
  int<lower=1> k;
  int<lower=0> n;
  matrix[n, k] X;
  array[n] int<lower=0, upper=1> y;
}
```

©黄湘云

图 33.3: 几个常用的概率分布

```
parameters {
  vector[k] beta;
  real alpha;
}
model {
  target += normal_lpdf(beta | 0, 1000);
  target += normal_lpdf(alpha | 0, 1000);
  target += bernoulli_logit_glm_lpmf(y | X, alpha, beta);
}
generated quantities {
  vector[n] log_lik;
  for (i in 1 : n) {
    log_lik[i] = bernoulli_logit_lpmf(y[i] | alpha + X[i] * beta);
  }
}
```

```
mod_logit_normal <- cmdstan_model(
  stan_file = "code/bernoulli_logit_glm_normal.stan",
  compile = TRUE, cpp_options = list(stan_threads = TRUE)
)

fit_logit_normal <- mod_logit_normal$sample(
  data = mdata,
```

© 基础统计


```
  chains = 2,
  parallel_chains = 2,
  iter_warmup = 1000,
  iter_sampling = 1000,
  threads_per_chain = 2,
  seed = 20232023,
  show_messages = FALSE,
  refresh = 0
)

# 输出结果
fit_logit_normal$summary(c("alpha", "beta", "lp__"))
```

```
#> # A tibble: 12 x 10
#>    variable      mean    median     sd    mad         q5      q95  rhat ess_bulk
#>    <chr>        <dbl>     <dbl>  <dbl>  <dbl>      <dbl>    <dbl> <dbl>    <dbl>
#>  1 alpha       1.02      1.02   0.0715 0.0728    0.899    1.13e+0 1.00     3082.
#>  2 beta[1]     3.14      3.13   0.133  0.135     2.93     3.36e+0 1.00     2103.
#>  3 beta[2]    -2.02     -2.02   0.0985 0.0999   -2.19    -1.86e+0 1.00     2192.
#>  4 beta[3]     0.0590    0.0586 0.0641 0.0644   -0.0456   1.65e-1 1.00     4654.
#>  5 beta[4]    -0.0270   -0.0252 0.0669 0.0640   -0.140    8.03e-2 1.00     4460.
#>  6 beta[5]     0.0122    0.0130 0.0682 0.0691   -0.100    1.25e-1 1.00     4532.
#>  7 beta[6]     0.0220    0.0205 0.0697 0.0683   -0.0938   1.36e-1 1.00     4304.
#>  8 beta[7]     0.103     0.102  0.0629 0.0619    0.00356  2.06e-1 0.999    4095.
#>  9 beta[8]    -0.0297   -0.0303 0.0667 0.0663   -0.143    7.91e-2 1.00     3671.
#> 10 beta[9]    -0.0870   -0.0856 0.0635 0.0615   -0.193    1.54e-2 1.00     4099.
#> 11 beta[10]    0.0818    0.0831 0.0639 0.0679   -0.0239   1.86e-1 1.00     4417.
#> 12 lp__     -843.     -842.     2.41   2.23   -847.      -8.39e+2 1.00      760.
#> # i 1 more variable: ess_tail <dbl>
```

33.3.2 Lasso 先验

指定回归系数 α, β 的先验分布如下

$$
\begin{aligned}
\lambda &\sim \text{Half_Cauchy}(0, 0.01) \\
\alpha &\sim \text{Double_exponential}(0, \lambda) \\
\beta &\sim \text{Double_exponential}(0, \lambda)
\end{aligned}
$$

其中，α, β 服从双指数分布，惩罚因子 λ 服从柯西分布。顺便一提，若把双指数分布改为正态分布，则 Lasso 先验变为岭先验。相比于岭先验，Lasso 先验有意将回归系数往 0 上收缩，这非常类似于频率派

©黄湘云

中的岭回归与 Lasso 回归的关系 (Bhadra 等 2019)。

```
data {
  int<lower=1> k;
  int<lower=0> n;
  matrix[n, k] X;
  array[n] int<lower=0, upper=1> y;
}
parameters {
  vector[k] beta;
  real alpha;
  real<lower=0> lambda;
}
model {
  target += double_exponential_lpdf(beta | 0, lambda);
  target += double_exponential_lpdf(alpha | 0, lambda);
  target += cauchy_lpdf(lambda | 0, 0.01);
  target += bernoulli_logit_glm_lpmf(y | X, alpha, beta);
}
generated quantities {
  vector[n] log_lik;
  for (i in 1 : n) {
    log_lik[i] = bernoulli_logit_lpmf(y[i] | alpha + X[i] * beta);
  }
}
```

```
mod_logit_lasso <- cmdstan_model(
  stan_file = "code/bernoulli_logit_glm_lasso.stan",
  compile = TRUE, cpp_options = list(stan_threads = TRUE)
)

fit_logit_lasso <- mod_logit_lasso$sample(
  data = mdata,
  chains = 2,
  parallel_chains = 2,
  iter_warmup = 1000,
  iter_sampling = 1000,
  threads_per_chain = 2,
  seed = 20232023,
  show_messages = FALSE,
  refresh = 0
```

©黄湘云

```
)

# 输出结果
fit_logit_lasso$summary(c("alpha", "beta", "lambda", "lp__"))
```

```
#> # A tibble: 13 x 10
#>    variable      mean    median     sd    mad         q5      q95  rhat ess_bulk
#>    <chr>        <dbl>     <dbl>  <dbl>  <dbl>      <dbl>    <dbl> <dbl>    <dbl>
#>  1 alpha       0.993     0.989  0.0738 0.0741    0.873    1.12e+0  1.00    2427.
#>  2 beta[1]     3.08      3.08   0.137  0.133     2.86     3.31e+0  1.00    1707.
#>  3 beta[2]    -1.99     -1.98   0.100  0.102    -2.15    -1.82e+0  1.00    2014.
#>  4 beta[3]     0.0533    0.0517 0.0614 0.0586   -0.0463   1.60e-1  1.00    4035.
#>  5 beta[4]    -0.0232   -0.0222 0.0600 0.0571   -0.121    7.16e-2  1.00    4184.
#>  6 beta[5]     0.0118    0.0128 0.0628 0.0602   -0.0959   1.15e-1  1.00    5750.
#>  7 beta[6]     0.0180    0.0163 0.0612 0.0603   -0.0810   1.23e-1  1.00    4271.
#>  8 beta[7]     0.0942    0.0940 0.0615 0.0607   -0.00725  1.95e-1  1.00    3869.
#>  9 beta[8]    -0.0268   -0.0252 0.0607 0.0612   -0.125    7.30e-2  1.00    3632.
#> 10 beta[9]    -0.0815   -0.0817 0.0603 0.0589   -0.182    1.73e-2  1.00    3377.
#> 11 beta[10]    0.0747    0.0742 0.0637 0.0605   -0.0305   1.80e-1  1.00    3815.
#> 12 lambda      0.598     0.566  0.189  0.153     0.361    9.32e-1  1.00    3142.
#> 13 lp__     -775.     -775.     2.58   2.42   -780.      -7.71e+2  1.00     829.
#> # i 1 more variable: ess_tail <dbl>
```

计算 LOO-CV 比较正态先验和 Lasso 先验

```
fit_logit_normal_loo <- fit_logit_normal$loo(variables = "log_lik", cores = 1)
print(fit_logit_normal_loo)
```

```
#>
#> Computed from 2000 by 2500 log-likelihood matrix
#>
#>          Estimate   SE
#> elpd_loo   -762.3 27.2
#> p_loo        11.3  0.5
#> looic      1524.6 54.5
#> ------
#> Monte Carlo SE of elpd_loo is 0.1.
#>
#> All Pareto k estimates are good (k < 0.5).
#> See help('pareto-k-diagnostic') for details.
```

```
fit_logit_lasso_loo <- fit_logit_lasso$loo(variables = "log_lik", cores = 1)
print(fit_logit_lasso_loo)
```

©南京大学出版社

```
#>
#> Computed from 2000 by 2500 log-likelihood matrix
#>
#>          Estimate   SE
#> elpd_loo   -761.5 26.8
#> p_loo        10.3  0.5
#> looic      1522.9 53.6
#> ------
#> Monte Carlo SE of elpd_loo is 0.1.
#>
#> All Pareto k estimates are good (k < 0.5).
#> See help('pareto-k-diagnostic') for details.
```

loo 包的函数 `loo_compare()` 比较两个模型

```
loo::loo_compare(list(model0 = fit_logit_normal_loo,
                      model1 = fit_logit_lasso_loo))
```

```
#>        elpd_diff se_diff
#> model1  0.0       0.0
#> model0 -0.9       0.5
```

输出结果中最好的模型放在第一行。LOOIC 越小越好，所以，Lasso 先验更好。

33.3.3 Horseshoe 先验

Horseshoe 先验（Horse shoe）(Piironen 和 Vehtari 2017b) 指定回归系数 $\alpha, \boldsymbol{\beta}$ 的先验分布如下

$$
\begin{aligned}
\lambda_i &\sim \text{Half_Cauchy}(0,1) \\
\alpha|\lambda_0, \tau &\sim \mathcal{N}(0, \tau^2\lambda_0^2) \\
\beta_i|\lambda_i, \tau &\sim \mathcal{N}(0, \tau^2\lambda_i^2), \quad i = 1, 2, \cdots, 10
\end{aligned}
$$

其中，τ 称之为全局超参数，它将所有的回归系数朝着 0 收缩。而作用在局部超参数 λ_i 上的重尾柯西先验允许某些回归系数逃脱收缩。

```
data {
  int<lower=1> k;
  int<lower=0> n;
  matrix[n, k] X;
  array[n] int<lower=0, upper=1> y;
}
parameters {
  vector[k] beta_tilde;
```

©贾俊平

```
  real alpha;
  real<lower=0> tau;
  vector<lower=0>[k] lambda;
}
transformed parameters {
  vector[k] beta = beta_tilde .* lambda * tau;
}
model {
  target += normal_lpdf(beta_tilde | 0, lambda);
  target += normal_lpdf(alpha | 0, lambda);
  target += cauchy_lpdf(tau | 0, 1);
  target += cauchy_lpdf(lambda | 0, 1);
  target += bernoulli_logit_glm_lpmf(y | X, alpha, beta);
}
generated quantities {
  vector[n] log_lik;
  for (i in 1 : n) {
    log_lik[i] = bernoulli_logit_lpmf(y[i] | alpha + X[i] * beta);
  }
}

# horseshoe 先验
mod_logit_horseshoe <- cmdstan_model(
  stan_file = "code/bernoulli_logit_glm_horseshoe.stan",
  compile = TRUE, cpp_options = list(stan_threads = TRUE)
)

fit_logit_horseshoe <- mod_logit_horseshoe$sample(
  data = mdata,
  chains = 2,
  parallel_chains = 2,
  iter_warmup = 1000,
  iter_sampling = 1000,
  threads_per_chain = 2,
  seed = 20232023,
  show_messages = FALSE,
  refresh = 0
)

fit_logit_horseshoe$summary(c("alpha", "beta", "tau", "lambda", "lp__"))
```

```
#> # A tibble: 23 x 10
#>    variable     mean   median     sd    mad      q5     q95  rhat ess_bulk
#>    <chr>       <dbl>    <dbl>  <dbl>  <dbl>   <dbl>   <dbl> <dbl>    <dbl>
#>  1 alpha     0.933    0.932   0.0770 0.0741  0.809   1.07   1.00     1274.
#>  2 beta[1]   3.05     3.05    0.133  0.133   2.83    3.26   1.00     1382.
#>  3 beta[2]  -1.97    -1.96    0.0985 0.0975 -2.13   -1.80   1.00     1875.
#>  4 beta[3]   0.0272   0.0196  0.0485 0.0399 -0.0407  0.119  0.999    1697.
#>  5 beta[4]  -0.0112  -0.00764 0.0453 0.0383 -0.0905  0.0591 1.00     1777.
#>  6 beta[5]   0.00532  0.00429 0.0432 0.0347 -0.0684  0.0838 1.00     1920.
#>  7 beta[6]   0.00861  0.00554 0.0453 0.0386 -0.0625  0.0890 1.00     2330.
#>  8 beta[7]   0.0622   0.0572  0.0557 0.0595 -0.0135  0.164  1.00     1450.
#>  9 beta[8]  -0.0153  -0.0111  0.0455 0.0362 -0.0959  0.0543 1.00     1838.
#> 10 beta[9]  -0.0490  -0.0396  0.0546 0.0549 -0.152   0.0218 1.00     1638.
#> # i 13 more rows
#> # i 1 more variable: ess_tail <dbl>
```

可以看到回归系数小的压缩效果很明显，而回归系数大的几乎没有压缩。

```r
fit_logit_horseshoe_loo <- fit_logit_horseshoe$loo(variables = "log_lik", cores = 1)
print(fit_logit_horseshoe_loo)
```

```
#>
#> Computed from 2000 by 2500 log-likelihood matrix
#>
#>          Estimate   SE
#> elpd_loo   -760.0 26.5
#> p_loo         7.6  0.3
#> looic      1519.9 53.1
#> ------
#> Monte Carlo SE of elpd_loo is 0.1.
#>
#> All Pareto k estimates are good (k < 0.5).
#> See help('pareto-k-diagnostic') for details.
```

LOOIC 比之 Lasso 先验的情况更小了。

注释

```r
library(rstanarm)
# set up the prior, use hyperprior tau ~ half-Cauchy(0,tau0^2)
D <- ncol(X) # 10 变量
n <- nrow(X) # 2500 样本量
```

贝叶斯模型 ©

```r
p0 <- 5 # prior guess for the number of relevant variables
sigma <- 1 / sqrt(mean(y)*(1-mean(y))) # pseudo sigma
tau0 <- p0 / (D - p0) * sigma / sqrt(n)
# hs() 函数指定层次收缩先验 Hierarchical shrinkage
# 拟合模型
fit <- stan_glm(
  y ~ X, family = binomial(), data = data.frame(I(X), y),
  # horseshoe 先验
  prior = hs(df = 1, global_df = 1, global_scale = tau0)
)
# 输出结果
summary(fit, digits = 4)
```

模型输出如下：

```
Model Info:
 function:     stan_glm
 family:       binomial [logit]
 formula:      y ~ X
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 2500
 predictors:   11

Estimates:
              mean    sd      10%     50%     90%
(Intercept)  1.0016  0.0732  0.9080  1.0007  1.0947
X1           3.0921  0.1343  2.9219  3.0888  3.2660
X2          -1.9907  0.1002 -2.1200 -1.9903 -1.8631
X3           0.0205  0.0429 -0.0180  0.0084  0.0804
X4          -0.0069  0.0364 -0.0534 -0.0018  0.0297
X5           0.0045  0.0367 -0.0336  0.0008  0.0474
X6           0.0062  0.0364 -0.0323  0.0014  0.0519
X7           0.0469  0.0559 -0.0068  0.0327  0.1258
X8          -0.0082  0.0376 -0.0545 -0.0021  0.0296
X9          -0.0342  0.0492 -0.1042 -0.0196  0.0109
X10          0.0310  0.0472 -0.0125  0.0180  0.0971

Fit Diagnostics:
           mean    sd      10%     50%     90%
```

```
mean_PPD 0.5915 0.0087 0.5804 0.5916 0.6028

The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for deta

MCMC diagnostics
              mcse   Rhat   n_eff
(Intercept)   0.0010 0.9994 5422
X1            0.0021 0.9996 3994
X2            0.0016 1.0000 3817
X3            0.0006 0.9997 4531
X4            0.0005 0.9998 4652
X5            0.0005 0.9993 5052
X6            0.0005 0.9994 4795
X7            0.0010 1.0002 3045
X8            0.0006 1.0000 4397
X9            0.0009 1.0002 3034
X10           0.0008 1.0003 3292
mean_PPD      0.0001 0.9994 4742
log-posterior 0.1367 1.0012 1206

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample s
```

rstanarm 包可以获得与前面一致的结果，甚至收缩效果比手写的 Stan 代码好一点点。

33.3.4 SpikeSlab 先验

SpikeSlab 先验（Spike Slab）放在非 0 协变量的个数上，是离散的先验。回归系数的先验分布的有限混合，常用有限混合多元正态分布。参考文章 Discrete Mixture Models

注释

BoomSpikeSlab 包是 **Boom** 包的扩展，提供基于 SpikeSlab 先验的贝叶斯变量选择功能。

```r
set.seed(2023)
n <- 2500
k <- 10
X <- matrix(rnorm(n * k), ncol = k)
y <- rbinom(n, size = 1, prob = plogis(1 + 3 * X[, 1] - 2 * X[, 2]))
# 加载 BoomSpikeSlab
library(BoomSpikeSlab)
fit_logit_spike <- logit.spike(y ~ X, niter = 500)
```

© 贝叶斯


```
# 模型输出
summary(fit_logit_spike)

null log likelihood:            -1690.677
posterior mean log likelihood:  -766.5283
posterior max log likelihood:   -754.8686
mean deviance R-sq:             0.5466147

predicted vs observed success rates, by decile:
                  predicted    observed
(0.00596,0.0279] 0.01388670 0.008032129
(0.0279,0.108]   0.06371528 0.060000000
(0.108,0.273]    0.17839881 0.176000000
(0.273,0.496]    0.39146661 0.404000000
(0.496,0.734]    0.61807048 0.608000000
(0.734,0.865]    0.80694458 0.764000000
(0.865,0.942]    0.90690322 0.928000000
(0.942,0.979]    0.96436544 0.976000000
(0.979,0.992]    0.98636201 0.992000000
(0.992,0.996]    0.99441504 1.000000000

summary of coefficients:
             mean    sd mean.inc sd.inc inc.prob
(Intercept)  1.02 0.105     1.02  0.105     1.00
X2          -2.00 0.232    -2.02  0.118     0.99
X1           3.10 0.354     3.13  0.170     0.99
X10          0.00 0.000     0.00  0.000     0.00
X9           0.00 0.000     0.00  0.000     0.00
X8           0.00 0.000     0.00  0.000     0.00
X7           0.00 0.000     0.00  0.000     0.00
X6           0.00 0.000     0.00  0.000     0.00
X5           0.00 0.000     0.00  0.000     0.00
X4           0.00 0.000     0.00  0.000     0.00
X3           0.00 0.000     0.00  0.000     0.00
```

33.4 推理算法

开篇提及 Stan 内置了多种推理算法，不同的算法获得的结果是存在差异的。

©黄湘云

- full Bayesian statistical inference with MCMC sampling (NUTS, HMC)
- approximate Bayesian inference with variational inference (ADVI)
- penalized maximum likelihood estimation with optimization (L-BFGS)

33.4.1 惩罚极大似然算法

L-BFGS 算法拟合模型，速度非常快。

```
# L-BFGS 算法拟合模型
fit_optim_logit <- mod_logit_lasso$optimize(
  data = mdata, # 观测数据
  init = 0,     # 所有参数初值设为 0
  refresh = 0,  # 不显示迭代进程
  algorithm = "lbfgs", # 优化器
  threads = 1,    # 单线程
  seed = 20232023 # 随机数种子
)
```

```
#> Finished in  0.2 seconds.
```

```
fit_optim_logit$summary(c("alpha", "beta", "lambda", "lp__"))
```

```
#> # A tibble: 13 x 2
#>    variable   estimate
#>    <chr>         <dbl>
#>  1 alpha       0.981
#>  2 beta[1]     3.05
#>  3 beta[2]    -1.96
#>  4 beta[3]     0.0488
#>  5 beta[4]    -0.0166
#>  6 beta[5]     0.00528
#>  7 beta[6]     0.0126
#>  8 beta[7]     0.0923
#>  9 beta[8]    -0.0204
#> 10 beta[9]    -0.0777
#> 11 beta[10]    0.0721
#> 12 lambda      0.488
#> 13 lp__     -768.
```

33.4.2 变分近似推断算法

ADVI 算法拟合模型，可选的优化器有 `meanfield` 和 `fullrank` ，相比于 L-BFGS 稍慢

©黄湘云

```r
# ADVI 算法拟合模型
fit_advi_logit <- mod_logit_lasso$variational(
  data = mdata, # 观测数据
  init = 0,     # 所有参数初值设为 0
  refresh = 0,  # 不显示迭代进程
  algorithm = "meanfield", # 优化器
  threads = 1,    # 单线程
  seed = 20232023 # 随机数种子
)
```

```
#> Finished in  1.6 seconds.
```

```r
fit_advi_logit$summary(c("alpha", "beta", "lambda", "lp__"))
```

```
#> # A tibble: 13 x 7
#>    variable       mean     median     sd    mad         q5       q95
#>    <chr>         <dbl>      <dbl>  <dbl>  <dbl>      <dbl>     <dbl>
#>  1 alpha        1.02       1.02    0.0615 0.0630    0.914      1.11
#>  2 beta[1]      3.07       3.07    0.0899 0.0870    2.93       3.22
#>  3 beta[2]     -1.98      -1.98    0.0675 0.0666   -2.08      -1.86
#>  4 beta[3]      0.0161     0.0159  0.0678 0.0670   -0.0945     0.129
#>  5 beta[4]     -0.0199    -0.0221  0.0639 0.0621   -0.121      0.0857
#>  6 beta[5]     -0.00128   -0.00202 0.0722 0.0713   -0.116      0.121
#>  7 beta[6]     -0.0423    -0.0446  0.0705 0.0689   -0.156      0.0754
#>  8 beta[7]      0.0760     0.0750  0.0490 0.0489   -0.00517    0.152
#>  9 beta[8]     -0.0742    -0.0752  0.0659 0.0637   -0.181      0.0347
#> 10 beta[9]     -0.0495    -0.0501  0.0802 0.0818   -0.185      0.0805
#> 11 beta[10]     0.0444     0.0440  0.0520 0.0522   -0.0444     0.128
#> 12 lambda       0.698      0.662   0.243  0.228     0.379      1.13
#> 13 lp__      -777.      -777.      3.39   3.22   -783.      -773.
```

33.4.3 拉普拉斯近似算法

Stan 内置的 Laplace 近似算法是对后验分布的 Laplace 正态近似，再从近似的后验分布中采样获得样本，最后，对样本进行统计分析获得参数的后验估计。详见 Stan 语言参考手册的Laplace Approximation 一章。

```r
# Laplace 算法
fit_laplace_logit <- mod_logit_lasso$laplace(
  data = mdata, # 观测数据
  init = 0,     # 所有参数初值设为 0
  refresh = 0,  # 不显示迭代进程
```

©贾清泉

```
  threads = 1,    # 单线程
  seed = 20232023 # 随机数种子
)
```

```
#> Finished in  0.2 seconds.
#> Finished in  1.7 seconds.
```

```
fit_laplace_logit$summary(c("alpha", "beta", "lambda", "lp__"))
```

```
#> # A tibble: 13 x 7
#>    variable       mean     median     sd    mad        q5       q95
#>    <chr>         <dbl>      <dbl>  <dbl>  <dbl>     <dbl>     <dbl>
#>  1 alpha       0.983      0.981   0.0701 0.0719    0.874     1.10
#>  2 beta[1]     3.05       3.05    0.131  0.129     2.84      3.28
#>  3 beta[2]    -1.97      -1.97    0.0960 0.0977   -2.12     -1.80
#>  4 beta[3]     0.0516     0.0493  0.0620 0.0605   -0.0432    0.150
#>  5 beta[4]    -0.0200    -0.0216  0.0647 0.0602   -0.121     0.0904
#>  6 beta[5]     0.00546    0.00505 0.0639 0.0645   -0.0971    0.110
#>  7 beta[6]     0.0135     0.0138  0.0642 0.0629   -0.0929    0.116
#>  8 beta[7]     0.0920     0.0917  0.0638 0.0659   -0.0179    0.195
#>  9 beta[8]    -0.0231    -0.0217  0.0641 0.0669   -0.128     0.0867
#> 10 beta[9]    -0.0810    -0.0798  0.0646 0.0640   -0.194     0.0212
#> 11 beta[10]    0.0732     0.0745  0.0639 0.0614   -0.0328    0.176
#> 12 lambda      0.562      0.536   0.168  0.154     0.333     0.866
#> 13 lp__     -775.      -775.      2.63   2.46   -780.     -772.
```

33.4.4 探路者变分算法

探路者算法 Pathfinder 属于变分法，针对可微的对数目标密度函数，沿着逆牛顿优化算法的迭代路径，获得目标密度函数的正态近似。正态近似中的局部协方差的估计采用 LBFGS 计算的负逆 Hessian 矩阵。探路者算法的优势是可以极大地减少对数密度函数和梯度的计算次数，缓解迭代陷入局部最优点和鞍点（何为鞍点，一个可视化示例详见章节 32.3 ）。

```
# Pathfinder 算法
fit_pathfinder_logit <- mod_logit_lasso$pathfinder(
  data = mdata, # 观测数据
  init = 0,     # 所有参数初值设为 0
  refresh = 0,  # 不显示迭代进程
  num_threads = 1,    # 单线程
  seed = 20232023 # 随机数种子
)
```

```
#> Finished in  3.7 seconds.
```

©黄湘云

```
fit_pathfinder_logit$summary(c("alpha", "beta", "lambda", "lp__"))
```

```
#> # A tibble: 13 x 7
#>    variable      mean    median     sd    mad         q5       q95
#>    <chr>        <dbl>     <dbl>  <dbl>  <dbl>      <dbl>     <dbl>
#>  1 alpha       0.995     0.993  0.0765 0.0816    0.875      1.12
#>  2 beta[1]     3.08      3.07   0.136  0.130     2.88       3.32
#>  3 beta[2]    -1.98     -1.98   0.102  0.106    -2.15      -1.81
#>  4 beta[3]     0.0555    0.0594 0.0616 0.0586   -0.0509     0.158
#>  5 beta[4]    -0.0217   -0.0209 0.0627 0.0614   -0.124      0.0761
#>  6 beta[5]     0.0125    0.0132 0.0653 0.0631   -0.0885     0.118
#>  7 beta[6]     0.0202    0.0198 0.0620 0.0648   -0.0816     0.125
#>  8 beta[7]     0.0968    0.0969 0.0596 0.0617   -0.00574    0.193
#>  9 beta[8]    -0.0286   -0.0247 0.0600 0.0572   -0.139      0.0643
#> 10 beta[9]    -0.0795   -0.0748 0.0598 0.0623   -0.183      0.0102
#> 11 beta[10]    0.0748    0.0713 0.0618 0.0612   -0.0234     0.173
#> 12 lambda      0.598     0.581  0.171  0.163     0.361      0.914
#> 13 lp__     -775.     -775.     2.60   2.41   -780.      -771.
```

33.5 习题

1. 在章节 33.3 的基础上，比较 Stan 实现的贝叶斯 Lasso 和 R 包 **glmnet** 的结果，发现 **glmnet** 包是很有竞争力的。在选择 Lasso 先验的情况下，收缩效果比 Stan 还好，运行速度也很快。Stan 的优势在于不限于先验分布的选取，当选择 Horseshoe 先验时，Stan 的收缩又比 **glmnet** 包更好。Stan 的优势还在于不限于 **glmnet** 包支持的常见分布族，如高斯、二项、泊松、多项、Cox 等。本质上，这两点都是 Stan 作为一门概率编程语言的优势，只要知道概率分布的数学表达式，总是可以用 Stan 编码出来的。

```
library(glmnet)
# 10 折交叉验证 Lasso 回归
fit_lasso <- cv.glmnet(x = X, y = y, family = "binomial", alpha = 1, nfolds = 10)
# 回归系数
coef(fit_lasso, s = fit_lasso$lambda.min)
```

```
#> 11 x 1 sparse Matrix of class "dgCMatrix"
#>                      s1
#> (Intercept)  0.94875532
#> V1           2.89856744
#> V2          -1.86004844
#> V3           0.02317963
#> V4           .
```

©黄湘云

```
#> V5           .
#> V6           .
#> V7           0.06356056
#> V8           .
#> V9          -0.05280930
#> V10          0.04624153
```

2. 基于德国信用卡评分数据，建立逻辑回归模型，分析 20 个协变量对响应变量的贡献，采用合适的先验分布选择适当的变量数。

```r
german_credit_data <- readRDS(file = "data/german_credit_data.rds")
str(german_credit_data)
```

```
#> 'data.frame':    1000 obs. of  21 variables:
#>  $ checking: Factor w/ 4 levels "A11","A12","A13",..: 1 2 4 1 1 4 4 2 4 2 ...
#>  $ duration: num  6 48 12 42 24 36 24 36 12 30 ...
#>  $ history : Factor w/ 5 levels "A30","A31","A32",..: 5 3 5 3 4 3 3 3 3 5 ...
#>  $ purpose : Factor w/ 10 levels "A40","A41","A410",..: 5 5 8 4 1 8 4 2 5 1 ...
#>  $ amount  : num  1169 5951 2096 7882 4870 ...
#>  $ savings : Factor w/ 5 levels "A61","A62","A63",..: 5 1 1 1 1 5 3 1 4 1 ...
#>  $ employed: Factor w/ 5 levels "A71","A72","A73",..: 5 3 4 4 3 3 5 3 4 1 ...
#>  $ installp: Factor w/ 4 levels "1","2","3","4": 4 2 2 2 3 2 3 2 2 4 ...
#>  $ marital : Factor w/ 4 levels "A91","A92","A93",..: 3 2 3 3 3 3 3 3 1 4 ...
#>  $ coapp   : Factor w/ 3 levels "A101","A102",..: 1 1 1 3 1 1 1 1 1 1 ...
#>  $ resident: Factor w/ 4 levels "1","2","3","4": 4 2 3 4 4 4 4 2 4 2 ...
#>  $ property: Factor w/ 4 levels "A121","A122",..: 1 1 1 2 4 4 2 3 1 3 ...
#>  $ age     : num  67 22 49 45 53 35 53 35 61 28 ...
#>  $ other   : Factor w/ 3 levels "A141","A142",..: 3 3 3 3 3 3 3 3 3 3 ...
#>  $ housing : Factor w/ 3 levels "A151","A152",..: 2 2 2 3 3 3 2 1 2 2 ...
#>  $ existcr : num  2 1 1 1 2 1 1 1 1 2 ...
#>  $ job     : Factor w/ 4 levels "A171","A172",..: 3 3 2 3 3 2 3 4 2 4 ...
#>  $ depends : num  1 1 2 2 2 2 1 1 1 1 ...
#>  $ telephon: Factor w/ 2 levels "A191","A192": 2 1 1 1 1 2 1 2 1 1 ...
#>  $ foreign : Factor w/ 2 levels "A201","A202": 1 1 1 1 1 1 1 1 1 1 ...
#>  $ good_bad: Factor w/ 2 levels "1","2": 1 2 1 1 2 1 1 1 1 2 ...
```

3. 下图是美国黄石公园老忠实间歇泉喷发时间和等待时间的分布规律，请建立合适的正态混合模型，用 Stan 拟合模型，并对结果做出说明。(提示：参考 Stan 用户手册的有限混合章节)

```
data {
  int<lower=1> K;          // number of mixture components
  int<lower=1> N;          // number of data points
  array[N] real y;         // observations
```

© 黄湘云

图 33.4: 黄石公园老忠实间歇泉

```
}
parameters {
  simplex[K] theta;          // mixing proportions
  ordered[K] mu;             // locations of mixture components
  vector<lower=0>[K] sigma;  // scales of mixture components
}
model {
  vector[K] log_theta = log(theta);  // cache log calculation
  sigma ~ lognormal(0, 2);
  mu ~ normal(0, 10);
  for (n in 1:N) {
    vector[K] lps = log_theta;
    for (k in 1:K) {
      lps[k] += normal_lpdf(y[n] | mu[k], sigma[k]);
    }
    target += log_sum_exp(lps);
  }
}
```

4. 在章节 十二 的探索分析基础上，对美国黄石公园的老忠实泉喷发规律，建立二项分布和二维正态分布的混合模型，请用 Stan 编码估计模型中的参数。

$$f(\boldsymbol{x}; p, \boldsymbol{\mu_1}, \Sigma_1, \boldsymbol{\mu_2}, \Sigma_2) = p\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu_1}, \Sigma_1) + (1-p)\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu_2}, \Sigma_2)$$

其中，参数 p 是一个介于 0 到 1 之间的常数，参数 $\boldsymbol{\mu_1} = (\mu_{11}, \mu_{12})^\top, \boldsymbol{\mu_2} = (\mu_{21}, \mu_{22})^\top$ 是二维的列向量，参数 $\Sigma_1 = (\sigma_{ij}), \Sigma_2 = (\delta_{ij}), i = 1, 2, j = 1, 2$ 是二阶的协方差矩阵。(提示：因有限混合模

型存在可识别性问题，简单起见，考虑各个多元正态分布的协方差矩阵相同的情况。）

```
data {
  int<lower=1> K;  // number of mixture components
  int<lower=1> N;  // number of observations
  int<lower=1> D;  // dimension of observations
  array[N] vector[D] y; // observations: a list of N vectors (each has D elements)
}
transformed data {
  vector[D] mu0 = rep_vector(0, D);
  matrix[D, D] Sigma0 = diag_matrix(rep_vector(1, D));
}
parameters {
  simplex[K] theta;       // mixing proportions
  array[K] positive_ordered[D] mu; // locations of mixture components
  // scales of mixture components
  array[K] cholesky_factor_corr[D] Lcorr; // cholesky factor (L_u matrix for R)
}
model {
  for(i in 1:K){
    mu[i] ~ multi_normal(mu0, Sigma0); // prior for mu
    Lcorr[i] ~ lkj_corr_cholesky(2.0); // prior for cholesky factor of a correlation matrix
  }

  vector[K] log_theta = log(theta);  // cache log calculation

  for (n in 1:N) {
    vector[K] lps = log_theta;
    for (k in 1:K) {
      lps[k] += multi_normal_cholesky_lpdf(y[n] | mu[k], Lcorr[k]);
    }
    target += log_sum_exp(lps);
  }
}
```

©黄湘云

第三十四章　广义线性模型

34.1　生成模拟数据

先介绍泊松广义线性模型，包括模拟和计算，并和 Stan 实现的结果比较。

泊松广义线性模型如下：

$$\log(\lambda) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$
$$Y \sim \text{Poisson}(u\lambda)$$

设定参数向量 $\beta = (\beta_0, \beta_1, \beta_2) = (0.5, 0.3, 0.2)$，观测变量 X_1 和 X_2 的均值都为 0，协方差矩阵 Σ 为

$$\begin{bmatrix} 1.0 & 0.8 \\ 0.8 & 1.0 \end{bmatrix}$$

模拟观测到的响应变量值和协变量值，添加漂移项

```
set.seed(2023)
n <- 2500 # 样本量
beta <- c(0.5, 0.3, 0.2)
X <- MASS::mvrnorm(n, mu = rep(0, 2), Sigma = matrix(c(1, 0.8, 0.8, 1), 2))
u <- rep(c(2, 4), each = n / 2)
lambda <- u * exp(cbind(1, X) %*% beta)
y <- rpois(n, lambda = lambda)
```

34.2　拟合泊松模型

拟合泊松回归模型

```
fit_poisson_glm <- glm(y ~ X, family = poisson(link = "log"), offset = log(u))
summary(fit_poisson_glm)
```

©贾俊平

```
#>
#> Call:
#> glm(formula = y ~ X, family = poisson(link = "log"), offset = log(u))
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept) 0.488932   0.009427   51.86   <2e-16 ***
#> X1          0.289984   0.014298   20.28   <2e-16 ***
#> X2          0.214846   0.014420   14.90   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for poisson family taken to be 1)
#>
#>     Null deviance: 6052.9  on 2499  degrees of freedom
#> Residual deviance: 2675.5  on 2497  degrees of freedom
#> AIC: 10773
#>
#> Number of Fisher Scoring iterations: 4
```

```r
# 对数似然函数值
log_poisson_lik <- logLik(fit_poisson_glm)
# 计算 AIC AIC(fit_poisson_glm)
-2 * c(log_poisson_lik) + 2 * attr(log_poisson_lik, "df")
```

```
#> [1] 10772.79
```

下面用 Stan 编码泊松回归模型，模型代码如下：

```
data {
  int<lower=1> k;
  int<lower=0> n;
  matrix[n, k] X;
  array[n] int<lower=0> y;
  vector[n] log_offset;
}
parameters {
  vector[k] beta;
  real alpha;
}
model {
  target += std_normal_lpdf(beta);
```

©黄湘云

```
  target += std_normal_lpdf(alpha);
  target += poisson_log_glm_lpmf(y | X, alpha + log_offset, beta);
}
generated quantities {
  vector[n] log_lik; // pointwise log-likelihood for LOO
  vector[n] y_rep;   // replications from posterior predictive dist
  for (i in 1 : n) {
    real y_hat_i = alpha + X[i] * beta + log_offset[i];
    log_lik[i] = poisson_log_lpmf(y[i] | y_hat_i);
    y_rep[i] = poisson_log_rng(y_hat_i);
  }
}
```

Stan 代码主要分三部分：

1. 数据部分 `data`：声明模型的输入数据，数据类型、大小、约束。
2. 参数部分 `parameters`：类似数据部分，声明模型的参数，参数类型、大小。
3. 模型部分 `model`：指定模型参数的先验分布。
4. 生成量 `generated quantities`：拟合模型获得参数估计值后，计算一些统计量。

下面准备数据

```
nchains <- 4 # 4 条迭代链
# 给每条链设置不同的参数初始值
inits_data <- lapply(1:nchains, function(i) {
  list(
    alpha = runif(1, 0, 1),
    beta = runif(2, 1, 10)
  )
})

# 准备数据
poisson_d <- list(
  n = 2500, # 观测记录的条数
  k = 2, # 协变量个数
  X = X, # N x 2 矩阵
  y = y, # N 向量
  log_offset = log(u)
)
```

编译模型，抽样获取参数的后验分布

©贾俊平

```r
# 加载 cmdstanr 包
library(cmdstanr)
# 编译模型
mod_poisson <- cmdstan_model(
  stan_file = "code/poisson_log_glm.stan",
  compile = TRUE,
  cpp_options = list(stan_threads = TRUE)
)
# 采样拟合模型
fit_poisson_stan <- mod_poisson$sample(
  data = poisson_d, # 观测数据
  init = inits_data, # 迭代初值
  iter_warmup = 1000, # 每条链预处理迭代次数
  iter_sampling = 2000, # 每条链总迭代次数
  chains = nchains, # 马尔科夫链的数目
  parallel_chains = 1, # 指定 CPU 核心数，可以给每条链分配一个
  threads_per_chain = 1, # 每条链设置一个线程
  show_messages = FALSE, # 不显示迭代的中间过程
  refresh = 0, # 不显示采样的进度
  seed = 20222022 # 设置随机数种子，不要使用 set.seed() 函数
)
# 迭代诊断
fit_poisson_stan$diagnostic_summary()

#> $num_divergent
#> [1] 0 0 0 0
#>
#> $num_max_treedepth
#> [1] 0 0 0 0
#>
#> $ebfmi
#> [1] 1.015694 1.132531 1.041293 1.084272

# 输出结果
fit_poisson_stan$summary(c("alpha", "beta", "lp__"))

#> # A tibble: 4 x 10
#>   variable      mean    median      sd     mad        q5      q95  rhat ess_bulk
#>   <chr>        <dbl>     <dbl>   <dbl>   <dbl>     <dbl>    <dbl> <dbl>    <dbl>
#> 1 alpha        0.489     0.489 0.00964 0.00966     0.473  5.05e-1  1.00    4416.
#> 2 beta[1]      0.290     0.290 0.0144  0.0143      0.266  3.14e-1  1.00    3093.
```

```
#> 3 beta[2]      0.215     0.215 0.0145  0.0145      0.191  2.39e-1  1.00    3168.
#> 4 lp__     -5388.    -5388.    1.23    1.05    -5390.    -5.39e+3  1.00    3469.
#> # i 1 more variable: ess_tail <dbl>
```

34.3 参数后验分布

加载 **bayesplot** 包，bayesplot 包提供一系列描述数据分布的绘图函数，比如绘制散点图 `mcmc_scatter()`。β_1 和 β_2 的联合分布

```
library(ggplot2)
library(bayesplot)
mcmc_scatter(fit_poisson_stan$draws(c("beta[1]", "beta[2]")), size = 1) +
  theme_classic() +
  labs(x = expression(beta[1]), y = expression(beta[2]))
```

图 34.1: β_1 和 β_2 的联合分布

如果提取采样的数据，也可使用 ggplot2 包绘图，不局限于 bayesplot 设定的风格。

```
beta_df <- fit_poisson_stan$draws(c("beta[1]", "beta[2]"), format = "draws_df")
ggplot(data = beta_df, aes(x = `beta[1]`, y = `beta[2]`)) +
  geom_density_2d_filled() +
  facet_wrap(~.chain, ncol = 2) +
```

© 草稿

```
  theme_classic() +
  labs(x = expression(beta[1]), y = expression(beta[2]))
```

图 34.2: β_1 和 β_2 的联合分布

β_1 和 β_2 的热力图

```
mcmc_hex(fit_poisson_stan$draws(c("beta[1]", "beta[2]"))) +
  theme_classic() +
  labs(x = expression(beta[1]), y = expression(beta[2]))
```

© 黄湘云

图 34.3: β_1 和 β_2 的热力图

各个参数的轨迹图

```
mcmc_trace(fit_poisson_stan$draws(c("beta[1]", "beta[2]")),
  facet_args = list(
    labeller = ggplot2::label_parsed, strip.position = "top", ncol = 1
  )
) +
  theme_classic()
```

© 黄湘云

图 34.4: 各个参数的轨迹图

可以将模型参数的后验分布图展示出来

```
mcmc_dens(fit_poisson_stan$draws(c("beta[1]", "beta[2]")),
  facet_args = list(
    labeller = ggplot2::label_parsed, strip.position = "top", ncol = 1
  )
) +
  theme_classic()
```

© 黄湘云

图 34.5: 各个参数的分布图（密度图）

后验分布的中位数、80% 区间

```
mcmc_areas(fit_poisson_stan$draws(c("beta[1]", "beta[2]")), prob = 0.8) +
  scale_y_discrete(labels = scales::parse_format()) +
  theme_classic()
```

© 黄湘云

图 34.6: 各个参数的分布图（岭线图）

岭线图就是将各个参数的后验分布图放在一起。

```
mcmc_areas_ridges(x = fit_poisson_stan$draws(), pars = c("beta[1]", "beta[2]")) +
  scale_y_discrete(labels = scales::parse_format()) +
  theme_classic()
```

© 黄湘云

图 34.7: 各个参数的分布图（岭线图）

参数的 $\hat{R}$ 潜在尺度收缩因子

```
bayesplot::rhat(fit_poisson_stan, pars = "alpha")
```

```
#>    alpha
#> 1.000374
```

后验预测诊断的想法是检查根据拟合模型生成的随机数 y^{rep} 与真实观测数据 y 的接近程度。为直观起见，可以用一系列描述数据分布的图来可视化检验。

```
# mcmc_scatter(fit_poisson_stan$draws(),
#   pars = c("beta[1]", "beta[2]"),
#   np = nuts_params(fit_poisson_stan)
# )

mcmc_nuts_energy(x = nuts_params(fit_poisson_stan), binwidth = 1) +
  ggtitle(label = "NUTS Energy Diagnostic")
```

© 黄湘云

图 34.8: NUTS 能量诊断图

y 是真实数据，yrep 是根据贝叶斯拟合模型生成的数据。下图是真实数据的密度图和 50 组生成数据的密度图。

```
# 抽取 yrep 数据
yrep <- fit_poisson_stan$draws(variables = "y_rep", format = "draws_matrix")
pp_check(y, yrep = yrep[1:50, ], fun = ppc_dens_overlay) +
  theme_classic()
```

© 黄湘云

图 34.9: 后验预测诊断图（密度图）

观察后验预测区间与真实数据的覆盖情况，不妨取前 50 次观测的数据，即 `y[1:50]` 与第 2 个自变量 `X[1:50, 2]` ，基于后验分布的 500 次采样数据绘制 50% 后验置信区间。

```
ppc_intervals(y[1:50], yrep = yrep[1:1000, 1:50], x = X[1:50, 2], prob = 0.5)
```

©黄湘云

图 34.10: 后验预测诊断图（区间图）

34.4 模型评估指标

loo 包可以计算 WAIC

```
fit_poisson_waic <- loo::waic(fit_poisson_stan$draws(variables = "log_lik"))
print(fit_poisson_waic)
```

```
#>
#> Computed from 8000 by 2500 log-likelihood matrix
#>
#>           Estimate   SE
#> elpd_waic  -5386.4 37.7
#> p_waic         3.0  0.1
#> waic       10772.9 75.5
```

loo 包推荐使用 LOO-CV ，它还提供诊断信息、有效样本量和蒙特卡罗估计。

```
fit_poisson_loo <- fit_poisson_stan$loo(variables = "log_lik", cores = 2)
print(fit_poisson_loo)
```

```
#>
#> Computed from 8000 by 2500 log-likelihood matrix
```

©黄湘云

```
#>
#>          Estimate   SE
#> elpd_loo  -5386.4 37.7
#> p_loo         3.0  0.1
#> looic     10772.9 75.5
#> ------
#> Monte Carlo SE of elpd_loo is 0.0.
#>
#> All Pareto k estimates are good (k < 0.5).
#> See help('pareto-k-diagnostic') for details.
```

34.5 可选替代实现

对于常见的统计模型，rstanarm 和 **brms** 包都内置了预编译的 Stan 程序，下面用 **brms** 包的函数 `brm()` 拟合带上述漂移项的泊松广义线性模型，参数估计结果和 Base R 函数 `glm()` 的几乎一致，因编译和抽样的过程比较花费时间，速度不及 Base R。

```
# brms
dat <- data.frame(y = y, X = X, u = u)
colnames(dat) <- c("y", "x1", "x2", "u")
fit_poisson_brm <- brms::brm(y ~ x1 + x2 + offset(log(u)),
  data = dat, family = poisson(link = "log")
)
fit_poisson_brm
```

```
 Family: poisson
  Links: mu = log
Formula: y ~ x1 + x2 + offset(log(u))
   Data: dat (Number of observations: 2500)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept     0.49      0.01     0.47     0.51 1.00     2509     2171
x1            0.29      0.01     0.26     0.32 1.00     1771     1645
x2            0.21      0.01     0.19     0.24 1.00     1727     1847

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
```

```
scale reduction factor on split chains (at convergence, Rhat = 1).
```

调用函数 `brm()` 拟合模型后返回一个 brmsfit 对象 `fit_poisson_brm`，**brms** 包提供很多函数处理该数据对象，比如 `brms::loo()` 计算 LOO-CV

```
brms::loo(fit_poisson_brm)
```

```
Computed from 4000 by 2500 log-likelihood matrix

         Estimate   SE
elpd_loo  -5386.3 37.8
p_loo         2.9  0.1
looic     10772.6 75.5
------
Monte Carlo SE of elpd_loo is 0.0.

All Pareto k estimates are good (k < 0.5).
See help('pareto-k-diagnostic') for details.
```

输出结果中，LOO IC 信息准则 Loo information criterion，looic 指标的作用类似频率派模型中的 AIC 指标，所以也几乎相同的。

```
# 后验预测检查
brms::pp_check(fit_poisson_brm)
```

34.6 案例：吸烟喝酒和食道癌的关系

本例数据集 esoph 来自 Base R 内置的 datasets 包，是法国伊勒-维莱讷食道癌研究数据，研究吸烟、喝酒与食道癌的关系，量化酒精、烟草、酒精和烟草的交互作用。部分数据集见表格 34.1 ，年龄组 agegp、酒精量 alcgp 和烟草量 tobgp 为有序的分类变量，正常来说，年龄越大，吸烟、喝酒对食道癌影响越大。

表格 34.1: 食道癌研究数据（部分）

年龄组	酒精量	烟草量	实验组	控制组
25-34	0-39g/day	0-9g/day	0	40
25-34	0-39g/day	10-19	0	10
25-34	0-39g/day	20-29	0	6
25-34	0-39g/day	30+	0	5
25-34	40-79	0-9g/day	0	27
25-34	40-79	10-19	0	7

统计学基础 ©

34.6.1 描述分析

先来简单统计一下各年龄组、酒精量组的食道癌发病人数

```
xtabs(data = esoph, cbind(ncases, ncontrols) ~ agegp + alcgp)
```

```
#> , ,  = ncases
#>
#>         alcgp
#> agegp   0-39g/day 40-79 80-119 120+
#>   25-34         0     0      0    1
#>   35-44         1     4      0    4
#>   45-54         1    20     12   13
#>   55-64        12    22     24   18
#>   65-74        11    25     13    6
#>   75+           4     4      2    3
#>
#> , ,  = ncontrols
#>
#>         alcgp
#> agegp   0-39g/day 40-79 80-119 120+
#>   25-34        61    45      5    4
#>   35-44        88    76     20    6
#>   45-54        77    61     27    2
#>   55-64        77    62     19    8
#>   65-74        60    28     16    2
#>   75+          23     8      0    0
```

图 34.11 描述食道癌发病率与年龄组、酒精量的关系

```
library(ggplot2)
aggregate(cbind(ncases, ncontrols) ~ agegp + alcgp, data = esoph, sum) |>
  ggplot(aes(x = agegp, y = alcgp, fill = ncases / (ncases + ncontrols))) +
  scale_fill_viridis_c(labels = scales::percent_format()) +
  geom_tile() +
  labs(x = "年龄组", y = "酒精量", fill = "发病率")
```

©黄湘云

图 34.11: 食道癌发病率与年龄组、酒精量的关系

34.6.2 拟合模型

响应变量服从二项分布，自变量包含年龄分组 agegp、酒精量 alcgp、烟草量 tobgp 和酒精量与烟草量的交互作用，建立广义线性模型。

```
fit_glm_esoph <- glm(cbind(ncases, ncontrols) ~ agegp + tobgp * alcgp,
  data = esoph, family = binomial(link = "logit")
)
```

模型输出

```
summary(fit_glm_esoph)
```

```
#>
#> Call:
#> glm(formula = cbind(ncases, ncontrols) ~ agegp + tobgp * alcgp,
#>     family = binomial(link = "logit"), data = esoph)
#>
#> Coefficients:
#>                 Estimate Std. Error z value Pr(>|z|)
#> (Intercept)     -1.16933    0.20767  -5.631 1.79e-08 ***
#> agegp.L          3.97135    0.69286   5.732 9.94e-09 ***
#> agegp.Q         -1.58715    0.61943  -2.562   0.0104 *
#> agegp.C          0.09866    0.47331   0.208   0.8349
```

© 基础统计


```
#> agegp^4          0.09950    0.32816   0.303   0.7617
#> agegp^5         -0.27067    0.21516  -1.258   0.2084
#> tobgp.L          1.10809    0.27042   4.098 4.17e-05 ***
#> tobgp.Q          0.26586    0.25419   1.046   0.2956
#> tobgp.C          0.29394    0.24026   1.223   0.2212
#> alcgp.L          2.42627    0.28829   8.416  < 2e-16 ***
#> alcgp.Q          0.12999    0.25418   0.511   0.6091
#> alcgp.C          0.36600    0.22252   1.645   0.1000
#> tobgp.L:alcgp.L -0.42942    0.58589  -0.733   0.4636
#> tobgp.Q:alcgp.L  0.33676    0.56764   0.593   0.5530
#> tobgp.C:alcgp.L -0.15742    0.54313  -0.290   0.7719
#> tobgp.L:alcgp.Q  0.04169    0.53027   0.079   0.9373
#> tobgp.Q:alcgp.Q -0.62384    0.50922  -1.225   0.2205
#> tobgp.C:alcgp.Q -0.06700    0.48120  -0.139   0.8893
#> tobgp.L:alcgp.C -0.25088    0.47211  -0.531   0.5951
#> tobgp.Q:alcgp.C  0.02303    0.44197   0.052   0.9584
#> tobgp.C:alcgp.C -0.17340    0.40908  -0.424   0.6717
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 367.953  on 87  degrees of freedom
#> Residual deviance:  76.886  on 67  degrees of freedom
#> AIC: 233.94
#>
#> Number of Fisher Scoring iterations: 6
```

整理模型输出后，见表格 34.2

表格 34.2: 广义线性模型各个参数的估计结果

term	estimate	std.error	statistic	p.value
(Intercept)	-1.1693289	0.2076675	-5.6307761	0.0000000
agegp.L	3.9713482	0.6928602	5.7318175	0.0000000
agegp.Q	-1.5871516	0.6194310	-2.5622734	0.0103989
agegp.C	0.0986649	0.4733144	0.2084553	0.8348735
agegp^4	0.0995027	0.3281577	0.3032161	0.7617251
agegp^5	-0.2706732	0.2151596	-1.2580111	0.2083877
tobgp.L	1.1080898	0.2704214	4.0976401	0.0000417
tobgp.Q	0.2658557	0.2541932	1.0458805	0.2956162

©黄湘云

表格 34.2: 广义线性模型各个参数的估计结果

term	estimate	std.error	statistic	p.value
tobgp.C	0.2939369	0.2402602	1.2234109	0.2211745
alcgp.L	2.4262714	0.2882861	8.4161937	0.0000000
alcgp.Q	0.1299883	0.2541785	0.5114056	0.6090671
alcgp.C	0.3659976	0.2225178	1.6448016	0.1000107
tobgp.L:alcgp.L	-0.4294231	0.5858868	-0.7329456	0.4635916
tobgp.Q:alcgp.L	0.3367616	0.5676428	0.5932632	0.5530050
tobgp.C:alcgp.L	-0.1574229	0.5431330	-0.2898423	0.7719369
tobgp.L:alcgp.Q	0.0416850	0.5302708	0.0786108	0.9373422
tobgp.Q:alcgp.Q	-0.6238362	0.5092212	-1.2250790	0.2205454
tobgp.C:alcgp.Q	-0.0670047	0.4811987	-0.1392454	0.8892562
tobgp.L:alcgp.C	-0.2508767	0.4721073	-0.5313976	0.5951433
tobgp.Q:alcgp.C	0.0230305	0.4419683	0.0521088	0.9584420
tobgp.C:alcgp.C	-0.1733950	0.4090805	-0.4238652	0.6716641

34.6.3 与 brms 比较

下面从贝叶斯的视角分析和建模，使用 **brms** 包对该数据拟合，同样是广义线性模型。

```
fit_brm_esoph <- brm(ncases | trials(ncases + ncontrols) ~ agegp + tobgp * alcgp,
                     data = esoph, family = binomial(link = "logit"))
```

```
 Family: binomial
  Links: mu = logit
Formula: ncases | trials(ncases + ncontrols) ~ agegp + tobgp * alcgp
   Data: esoph (Number of observations: 88)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Population-Level Effects:
                Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept          -1.91      0.25    -2.49    -1.51        735 1.01
agegp.L             3.39      0.86     2.13     5.45        674 1.01
agegp.Q            -1.68      0.78    -3.58    -0.50        658 1.01
agegp.C             0.31      0.57    -0.59     1.63        709 1.00
agegpE4            -0.01      0.36    -0.80     0.65        907 1.01
agegpE5            -0.20      0.21    -0.59     0.22       1970 1.00
tobgp.L             0.63      0.20     0.24     1.03       4654 1.00
```

©黄湘云

```
tobgp.Q             0.03      0.20    -0.38     0.42       3469 1.00
tobgp.C             0.17      0.20    -0.21     0.57       3892 1.00
alcgp.L             1.41      0.22     0.99     1.84       4067 1.00
alcgp.Q            -0.16      0.20    -0.56     0.24       3335 1.00
alcgp.C             0.25      0.19    -0.12     0.62       3870 1.00
tobgp.L:alcgp.L    -0.69      0.42    -1.51     0.16       3878 1.00
tobgp.Q:alcgp.L     0.13      0.43    -0.75     0.97       4249 1.00
tobgp.C:alcgp.L    -0.30      0.44    -1.15     0.58       5149 1.00
tobgp.L:alcgp.Q     0.13      0.41    -0.67     0.94       3127 1.00
tobgp.Q:alcgp.Q    -0.46      0.41    -1.24     0.34       4037 1.00
tobgp.C:alcgp.Q    -0.05      0.40    -0.82     0.74       4490 1.00
tobgp.L:alcgp.C    -0.15      0.38    -0.89     0.58       3507 1.00
tobgp.Q:alcgp.C     0.04      0.37    -0.69     0.75       3274 1.00
tobgp.C:alcgp.C    -0.17      0.36    -0.88     0.54       3773 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

输出结果和 `glm()` 有不少差别的。

34.7 案例：哥本哈根住房状况调查

数据集 housing 哥本哈根住房状况调查中的次数分布表，`Sat` 住户对目前居住环境的满意程度，是一个有序的因子变量，`Infl` 住户对物业管理的感知影响程度，`Type` 租赁住宿类型，如塔楼、中庭、公寓、露台，`Cont` 联系居民可与其他居民联系 (低、高)，`Freq` 每个类中的居民人数，调查的人数。

```r
data("housing", package = "MASS")
str(housing)
```

```
#> 'data.frame':    72 obs. of  5 variables:
#>  $ Sat : Ord.factor w/ 3 levels "Low"<"Medium"<..: 1 2 3 1 2 3 1 2 3 1 ...
#>  $ Infl: Factor w/ 3 levels "Low","Medium",..: 1 1 1 2 2 2 3 3 3 1 ...
#>  $ Type: Factor w/ 4 levels "Tower","Apartment",..: 1 1 1 1 1 1 1 1 1 2 ...
#>  $ Cont: Factor w/ 2 levels "Low","High": 1 1 1 1 1 1 1 1 1 1 ...
#>  $ Freq: int  21 21 28 34 22 36 10 11 36 61 ...
```

响应变量是居民对居住环境满意度 Sat ，分三个等级，且存在强弱，等级，大小之分。

```r
# 因子变量的处理
options(contrasts = c("contr.treatment", "contr.poly"))
# 有序逻辑回归
```

```r
housing_mass <- MASS::polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing, Hess = TRUE)
summary(housing_mass)
```

```
#> Call:
#> MASS::polr(formula = Sat ~ Infl + Type + Cont, data = housing,
#>     weights = Freq, Hess = TRUE)
#>
#> Coefficients:
#>                 Value Std. Error t value
#> InflMedium     0.5664    0.10465   5.412
#> InflHigh       1.2888    0.12716  10.136
#> TypeApartment -0.5724    0.11924  -4.800
#> TypeAtrium    -0.3662    0.15517  -2.360
#> TypeTerrace   -1.0910    0.15149  -7.202
#> ContHigh       0.3603    0.09554   3.771
#>
#> Intercepts:
#>             Value   Std. Error t value
#> Low|Medium  -0.4961  0.1248    -3.9739
#> Medium|High  0.6907  0.1255     5.5049
#>
#> Residual Deviance: 3479.149
#> AIC: 3495.149
```

计算置信区间

```r
# 剖面
confint(profile(housing_mass), level = 0.95)
```

```
#>                    2.5 %      97.5 %
#> InflMedium     0.3616415  0.77195375
#> InflHigh       1.0409701  1.53958138
#> TypeApartment -0.8069590 -0.33940432
#> TypeAtrium    -0.6705862 -0.06204495
#> TypeTerrace   -1.3893863 -0.79533958
#> ContHigh       0.1733589  0.54792854
```

34.8 习题

1. 分析挑战者号航天飞机 O 型环数据。**DAAG** 包的 orings 数据集记录美国挑战者号航天飞机 O 型环在不同温度下发生 Erosion 腐蚀和 Blowby 串气的失效数量。图 34.12 展示航天飞机 O 型环

© 黄湘云

在不同温度下失效的分布图（条件密度图）：随着温度升高，O 型环越来越不容易失效。请分别用 Base R 函数 `glm()` 和 **cmdstanr** 包建模分析 O 型环数据。

图 34.12: 航天飞机 O 型环在不同温度下失效的条件密度图

2. 基于数据集 infert 分析自然流产和人工流产后的不育情况，
3. 根据章节 二十八 的数据，建立贝叶斯空间广义线性混合模型，用 Stan 预测核辐射强度的分布。

©黄湘云

第三十五章　分层正态模型

> This is a bit like asking how should I tweak my sailboat so I can explore the ocean floor.
>
> — Roger Koenker [1]

乔治 · 博克斯说，所有的模型都是错的，但有些是有用的。在真实的数据面前，尽我们所能，结果发现没有最好的模型，只有更好的模型。总是需要自己去构造符合自己需求的模型及其实现，只有自己能够实现，才能在模型的海洋中畅快地遨游。

介绍分层正态模型的定义、结构、估计，分层正态模型与曲线生长模型的关系，分层正态模型与潜变量模型的关系，分层正态模型与线性混合效应的关系。以 **rstan** 包和 **nlme** 包拟合分层正态模型，说明 **rstan** 包的一些用法，比较贝叶斯和频率派方法拟合的结果，给出结果的解释。再对比 16 个不同的 R 包实现，总结一般地使用经验，也体会不同 R 包的独特性。

```r
library(StanHeaders)
library(ggplot2)
library(rstan)
# 将编译的 Stan 模型与代码文件放在一起
rstan_options(auto_write = TRUE)
# 如果CPU和内存足够，设置成与马尔科夫链一样多
options(mc.cores = 2)
# 调色板
custom_colors <- c(
  "#4285f4", # GoogleBlue
  "#34A853", # GoogleGreen
  "#FBBC05", # GoogleYellow
  "#EA4335"  # GoogleRed
)
rstan_ggtheme_options(
  panel.background = element_rect(fill = "white"),
  legend.position = "top"
)
rstan_gg_options(
```

[1] https://stat.ethz.ch/pipermail/r-help/2013-May/354311.html

©黄湘云


```
  fill = "#4285f4", color = "white",
  pt_color = "#EA4335", chain_colors = custom_colors
)
library(bayesplot)
```

35.1 rstan 包

本节以 8schools 数据为例介绍分层正态模型及 **rstan** 包实现，8schools 数据最早来自 Rubin (1981)，分层正态模型如下：

$$
\begin{aligned}
y_j \sim \mathcal{N}(\theta_j, \sigma_j^2) \quad \theta_j = \mu + \tau \times \eta_j \\
\theta_j \sim \mathcal{N}(\mu, \tau^2) \quad \eta_j \sim \mathcal{N}(0,1) \\
\mu \sim \mathcal{N}(0, 100^2) \quad \tau \sim \text{half_normal}(0, 100^2)
\end{aligned}
$$

其中，y_j, σ_j 是已知的观测数据，θ_j 是模型参数，η_j 是服从标准正态分布的潜变量，μ, τ 是超参数，分别服从正态分布（将方差设置为很大的数，则变成弱信息先验或无信息均匀先验）和半正态分布（随机变量限制为正值）。

35.1.1 拟合模型

用 **rstan** 包来拟合模型，下面采用非中心的参数化表示，降低参数的相关性，减少发散的迭代次数，提高采样效率。

```
# 编译模型
eight_schools_fit <- stan(
  model_name = "eight_schools",
  # file = "code/eight_schools.stan",
  model_code = "
  // saved as eight_schools.stan
  data {
    int<lower=0> J;                // number of schools
    array[J] real y;               // estimated treatment effects
    array[J] real <lower=0> sigma; // standard error of effect estimates
  }
  parameters {
    real mu;                // population treatment effect
    real<lower=0> tau;      // standard deviation in treatment effects
    vector[J] eta;          // unscaled deviation from mu by school
  }
```

©贾俊平

```
  transformed parameters {
    vector[J] theta = mu + tau * eta;        // school treatment effects
  }
  model {
    target += normal_lpdf(mu | 0, 100);
    target += normal_lpdf(tau | 0, 100);
    target += normal_lpdf(eta | 0, 1);  // prior log-density
    target += normal_lpdf(y | theta, sigma); // log-likelihood
  }
  ",
  data = list( # 观测数据
    J = 8,
    y = c(28, 8, -3, 7, -1, 1, 18, 12),
    sigma = c(15, 10, 16, 11, 9, 11, 10, 18)
  ),
  warmup = 1000, # 每条链预处理迭代次数
  iter = 2000,   # 每条链总迭代次数
  chains = 2,    # 马尔科夫链的数目
  cores = 2,     # 指定 CPU 核心数，可以给每条链分配一个
  verbose = FALSE, # 不显示迭代的中间过程
  refresh = 0,     # 不显示采样的进度
  seed = 20232023  # 设置随机数种子，不要使用 set.seed() 函数
)
```

35.1.2 模型输出

用函数 print() 打印输出结果，保留 2 位小数。

```
print(eight_schools_fit, digits = 2)
```

```
#> Inference for Stan model: eight_schools.
#> 2 chains, each with iter=2000; warmup=1000; thin=1;
#> post-warmup draws per chain=1000, total post-warmup draws=2000.
#>
#>            mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
#> mu         7.90    0.16 5.00  -1.79   4.60   7.71  11.05  18.37   988    1
#> tau        6.43    0.20 5.35   0.22   2.49   5.23   8.94  19.73   733    1
#> eta[1]     0.40    0.02 0.96  -1.57  -0.21   0.41   1.06   2.27  2252    1
#> eta[2]     0.00    0.02 0.88  -1.74  -0.59  -0.01   0.57   1.77  1977    1
#> eta[3]    -0.17    0.02 0.92  -1.93  -0.78  -0.18   0.41   1.71  2307    1
#> eta[4]    -0.04    0.02 0.91  -1.90  -0.64  -0.01   0.55   1.76  2053    1
```

© 贾俊平


```
#> eta[5]    -0.34    0.02 0.89  -2.03  -0.97  -0.38   0.25   1.43  1769    1
#> eta[6]    -0.22    0.02 0.87  -1.89  -0.81  -0.23   0.36   1.52  1959    1
#> eta[7]     0.33    0.02 0.86  -1.30  -0.26   0.31   0.90   2.00  2020    1
#> eta[8]     0.05    0.02 0.96  -1.89  -0.59   0.05   0.69   1.94  2597    1
#> theta[1]  11.29    0.20 8.11  -1.69   5.88  10.14  15.34  31.08  1728    1
#> theta[2]   7.80    0.13 6.25  -4.68   3.88   7.88  11.62  19.78  2362    1
#> theta[3]   6.30    0.17 7.57 -10.90   2.15   6.63  10.87  20.04  1902    1
#> theta[4]   7.72    0.14 6.53  -5.33   3.78   7.52  11.66  21.22  2249    1
#> theta[5]   5.10    0.14 6.48  -9.20   1.04   5.69   9.50  16.71  2030    1
#> theta[6]   6.06    0.16 6.88  -8.61   2.07   6.39  10.33  19.01  1766    1
#> theta[7]  10.41    0.14 6.42  -0.18   6.03   9.59  13.98  24.97  2057    1
#> theta[8]   8.45    0.19 8.00  -7.45   4.02   8.13  12.69  26.56  1728    1
#> lp__     -50.67    0.11 2.64 -56.69 -52.25 -50.40 -48.78 -46.34   584    1
#>
#> Samples were drawn using NUTS(diag_e) at Tue Jan 23 09:56:08 2024.
#> For each parameter, n_eff is a crude measure of effective sample size,
#> and Rhat is the potential scale reduction factor on split chains (at
#> convergence, Rhat=1).
```

值得一提，数据有限而且规律不明确，数据隐含的信息不是很多，则先验分布的情况将会对参数估计结果产生很大影响。Stan 默认采用无信息的先验分布，当使用非常弱的信息先验时，结果就非常不同了。提取任意一个参数的结果，如查看参数 τ 的 95% 置信区间。

```r
print(eight_schools_fit, pars = "tau", probs = c(0.025, 0.975))
```

```
#> Inference for Stan model: eight_schools.
#> 2 chains, each with iter=2000; warmup=1000; thin=1;
#> post-warmup draws per chain=1000, total post-warmup draws=2000.
#>
#>     mean se_mean   sd 2.5% 97.5% n_eff Rhat
#> tau 6.43     0.2 5.35 0.22 19.73   733    1
#>
#> Samples were drawn using NUTS(diag_e) at Tue Jan 23 09:56:08 2024.
#> For each parameter, n_eff is a crude measure of effective sample size,
#> and Rhat is the potential scale reduction factor on split chains (at
#> convergence, Rhat=1).
```

从迭代抽样数据获得与 `print(fit)` 一样的结果。以便后续对原始采样数据做任意的进一步分析。**rstan** 包扩展泛型函数 `summary()` 以支持对 stanfit 数据对象汇总，输出各个参数分链条和合并链条的后验分布结果。

©贾俊平

35.1.3 操作数据

抽取数据对象 eight_schools_fit 中的采样数据，合并几条马氏链的结果，返回的结果是一个列表。

```
eight_schools_sim <- extract(eight_schools_fit, permuted = TRUE)
```

返回列表中的每个元素是一个数组，标量参数对应一维数组，向量参数对应二维数组。

```
str(eight_schools_sim)
```

```
#> List of 5
#>  $ mu   : num [1:2000(1d)] 9.808 0.202 13.798 13.456 1.051 ...
#>   ..- attr(*, "dimnames")=List of 1
#>   .. ..$ iterations: NULL
#>  $ tau  : num [1:2000(1d)] 4.52 7.81 6.35 2.3 4.48 ...
#>   ..- attr(*, "dimnames")=List of 1
#>   .. ..$ iterations: NULL
#>  $ eta  : num [1:2000, 1:8] -0.2148 1.3054 0.261 -0.0981 1.2931 ...
#>   ..- attr(*, "dimnames")=List of 2
#>   .. ..$ iterations: NULL
#>   .. ..$           : NULL
#>  $ theta: num [1:2000, 1:8] 8.84 10.39 15.46 13.23 6.84 ...
#>   ..- attr(*, "dimnames")=List of 2
#>   .. ..$ iterations: NULL
#>   .. ..$           : NULL
#>  $ lp__ : num [1:2000(1d)] -50.9 -49.5 -49 -52.1 -49.8 ...
#>   ..- attr(*, "dimnames")=List of 1
#>   .. ..$ iterations: NULL
```

对于列表，适合用函数 lapply() 配合算术函数计算 μ, τ 等参数的均值。

```
fun_mean <- function(x) {
  if (length(dim(x)) > 1) {
    apply(x, 2, mean)
  } else {
    mean(x)
  }
}
lapply(eight_schools_sim, FUN = fun_mean)
```

```
#> $mu
#> [1] 7.896911
#>
#> $tau
```

©黄湘云

```
#> [1] 6.427487
#>
#> $eta
#> [1]  0.39815800 -0.00403665 -0.17091492 -0.03835530 -0.34447579 -0.21592391
#> [7]  0.33375651  0.04527884
#>
#> $theta
#> [1] 11.293515  7.796730  6.300619  7.722628  5.100476  6.059899 10.411386
#> [8]  8.451163
#>
#> $lp__
#> [1] -50.66637
```

类似地，计算 μ, τ 等参数的分位点。

```
fun_quantile <- function(x, probs) {
  if (length(dim(x)) > 1) {
    t(apply(x, 2, quantile, probs = probs))
  } else {
    quantile(x, probs = probs)
  }
}
lapply(eight_schools_sim, fun_quantile, probs = c(2.5, 25, 50, 75, 97.5) / 100)
```

```
#> $mu
#>      2.5%       25%       50%       75%     97.5%
#> -1.787025  4.603868  7.706957 11.054801 18.370903
#>
#> $tau
#>       2.5%        25%        50%        75%      97.5%
#>  0.2221582  2.4933088  5.2289974  8.9369194 19.7296019
#>
#> $eta
#>
#>             2.5%        25%          50%       75%    97.5%
#>   [1,] -1.571814 -0.2051970  0.412478990 1.0592655 2.267580
#>   [2,] -1.740860 -0.5924433 -0.009587292 0.5740810 1.768344
#>   [3,] -1.933032 -0.7815080 -0.181728735 0.4102823 1.709490
#>   [4,] -1.896206 -0.6383988 -0.008634091 0.5491973 1.758581
#>   [5,] -2.029051 -0.9674769 -0.376669547 0.2505514 1.425229
#>   [6,] -1.890733 -0.8146685 -0.227713876 0.3642582 1.524509
```

©黄湘云

```
#>   [7,] -1.295196 -0.2552945  0.310815774 0.9046577 1.995717
#>   [8,] -1.894366 -0.5853674  0.053723000 0.6912584 1.940257
#>
#> $theta
#>
#>               2.5%      25%       50%       75%    97.5%
#>   [1,]  -1.6909041 5.879186 10.136104 15.343330 31.08250
#>   [2,]  -4.6774314 3.880805  7.876314 11.616239 19.77814
#>   [3,] -10.9015495 2.149377  6.629543 10.872316 20.03882
#>   [4,]  -5.3329649 3.779338  7.521149 11.663011 21.22172
#>   [5,]  -9.2028941 1.035711  5.692159  9.501092 16.70603
#>   [6,]  -8.6129198 2.068728  6.393843 10.333108 19.00727
#>   [7,]  -0.1804168 6.033491  9.594300 13.980905 24.96505
#>   [8,]  -7.4463975 4.015728  8.129632 12.694195 26.55720
#>
#> $lp__
#>      2.5%       25%       50%       75%     97.5%
#> -56.69300 -52.24867 -50.39536 -48.78482 -46.34192
```

同理，可以计算最大值 `max()`、最小值 `min()` 和中位数 `median()` 等。

35.1.4 采样诊断

获取马尔科夫链迭代点列数据

```
eight_schools_sim <- extract(eight_schools_fit, permuted = FALSE)
```

`eight_schools_sim` 是一个三维数组，1000（次迭代）* 2 （条链）* 19（个参数）。如果 `permuted = TRUE` 则会合并马氏链的迭代结果，变成一个列表。

```
# 数据类型
class(eight_schools_sim)
```

```
#> [1] "array"
```

```
# 1000（次迭代）* 2 （条链）* 19（个参数）
str(eight_schools_sim)
```

```
#>  num [1:1000, 1:2, 1:19] 14.18 2.19 12.29 12.51 5.02 ...
#>  - attr(*, "dimnames")=List of 3
#>   ..$ iterations: NULL
#>   ..$ chains    : chr [1:2] "chain:1" "chain:2"
#>   ..$ parameters: chr [1:19] "mu" "tau" "eta[1]" "eta[2]" ...
```

提取参数 μ 的迭代点列，绘制迭代轨迹。

© 统计出版社


```
eight_schools_mu_sim <- eight_schools_sim[, , "mu"]
matplot(
  eight_schools_mu_sim, xlab = "迭代次数", ylab = expression(mu),
  type = "l", lty = "solid", col = custom_colors
)
abline(h = apply(eight_schools_mu_sim, 2, mean), col = custom_colors)
legend(
  "topleft", legend = paste("chain", 1:2), box.col = "white",
  inset = 0.01, lty = "solid", horiz = TRUE, col = custom_colors
)
```

图 35.1: Base R 绘制参数 μ 的迭代轨迹

也可以使用 **rstan** 包提供的函数 `traceplot()` 或者 `stan_trace()` 绘制参数的迭代轨迹图。

```
stan_trace(eight_schools_fit, pars = "mu") +
  labs(x = "迭代次数", y = expression(mu))
```

© 黄湘云

图 35.2: rstan 绘制参数 μ 的迭代轨迹

35.1.5 后验分布

可以用函数 stan_hist() 或 stan_dens() 绘制后验分布图。下图分别展示参数 μ、τ 的直方图，以及二者的散点图，参数 μ 的后验概率密度分布图。

```
p1 <- stan_hist(eight_schools_fit, pars = c("mu","tau"), bins = 30)
p2 <- stan_scat(eight_schools_fit, pars = c("mu","tau"), size = 1) +
  labs(x = expression(mu), y = expression(tau))
p3 <- stan_dens(eight_schools_fit, pars = "mu") + labs(x = expression(mu))
library(patchwork)
p1 / (p2 + p3)
```

© 黄湘云

图 35.3: rstan 包绘制后验分布图

相比于 **rstan** 包，**bayesplot** 包可视化能力更强，支持对特定的参数做变换。**bayesplot** 包的函数 `mcmc_pairs()` 以矩阵图展示多个参数的分布，下图展示参数 μ，$\log(\tau)$ 后验分布图。但是，这些函数都固定了一些标题，不能修改。

```
bayesplot::mcmc_pairs(
  eight_schools_fit, pars = c("mu", "tau"), transform = list(tau = "log")
)
```

©黄湘云

图 35.4: bayesplot 包绘制后验分布图

35.2 其它 R 包

35.2.1 nlme

接下来，用 **nlme** 包拟合模型。

```
# 成绩
y <- c(28, 8, -3, 7, -1, 1, 18, 12)
# 标准差
sigma <- c(15, 10, 16, 11, 9, 11, 10, 18)
# 学校编号
```

©汪荣鑫

```
g <- 1:8
```

首先，调用 **nlme** 包的函数 `lme()` 拟合模型。

```
library(nlme)
fit_lme <- lme(y ~ 1, random = ~ 1 | g, weights = varFixed(~ sigma^2), method = "REML")
summary(fit_lme)
```

```
#> Linear mixed-effects model fit by REML
#>   Data: NULL
#>        AIC      BIC    logLik
#>   60.21091 60.04864 -27.10546
#>
#> Random effects:
#>  Formula: ~1 | g
#>         (Intercept) Residual
#> StdDev:    2.917988 0.780826
#>
#> Variance function:
#>  Structure: fixed weights
#>  Formula: ~sigma^2
#> Fixed effects:  y ~ 1
#>                Value Std.Error DF  t-value p-value
#> (Intercept) 7.785729  3.368082  8 2.311621  0.0496
#>
#> Standardized Within-Group Residuals:
#>         Min          Q1         Med          Q3         Max
#> -1.06635035 -0.73588511 -0.02896764  0.50254917  1.62502386
#>
#> Number of Observations: 8
#> Number of Groups: 8
```

随机效应的标准差 2.917988 ，随机效应部分的估计

```
ranef(fit_lme)
```

```
#>   (Intercept)
#> 1  1.18135690
#> 2  0.02625714
#> 3 -0.55795543
#> 4 -0.08130333
#> 5 -1.29202240
#> 6 -0.70215328
```

©黄湘云

```
#> 7  1.25167648
#> 8  0.17414393
```

类比 Stan 输出结果中的 θ 向量，每个学校的成绩估计

```
7.785729 + 2.917988 * ranef(fit_lme)
```

```
#>   (Intercept)
#> 1   11.232914
#> 2    7.862347
#> 3    6.157622
#> 4    7.548487
#> 5    4.015623
#> 6    5.736854
#> 7   11.438106
#> 8    8.293879
```

35.2.2 lme4

接着，采用 **lme4** 包拟合模型，发现 **lme4** 包获得与 **nlme** 包一样的结果。

```
control <- lme4::lmerControl(
  check.conv.singular = "ignore",
  check.nobs.vs.nRE = "ignore",
  check.nobs.vs.nlev = "ignore"
)
fit_lme4 <- lme4::lmer(y ~ 1 + (1 | g), weights = 1 / sigma^2, control = control, REML = TRUE)
summary(fit_lme4)
```

```
#> Linear mixed model fit by REML ['lmerMod']
#> Formula: y ~ 1 + (1 | g)
#> Weights: 1/sigma^2
#> Control: control
#>
#> REML criterion at convergence: 54.2
#>
#> Scaled residuals:
#>      Min       1Q   Median       3Q      Max
#> -1.06635 -0.73589 -0.02897  0.50255  1.62502
#>
#> Random effects:
#>  Groups   Name        Variance Std.Dev.
#>  g        (Intercept) 8.5145   2.9180
```

```
#>  Residual             0.6097   0.7808
#> Number of obs: 8, groups:  g, 8
#>
#> Fixed effects:
#>             Estimate Std. Error t value
#> (Intercept)    7.786      3.368   2.312
```

35.2.3 blme

下面使用 **blme** 包 (Chung 等 2013)，**blme** 包基于 **lme4** 包，参数估计结果完全一致。

```
# the mode should be at the boundary of the space.

fit_blme <- blme::blmer(
  y ~ 1 + (1 | g), control = control, REML = TRUE,
  cov.prior = NULL, weights = 1 / sigma^2
)
summary(fit_blme)
```

```
#> Prior dev  : 0
#>
#> Linear mixed model fit by REML ['blmerMod']
#> Formula: y ~ 1 + (1 | g)
#> Weights: 1/sigma^2
#> Control: control
#>
#> REML criterion at convergence: 54.2
#>
#> Scaled residuals:
#>      Min       1Q   Median       3Q      Max
#> -1.06635 -0.73589 -0.02897  0.50255  1.62502
#>
#> Random effects:
#>  Groups   Name        Variance Std.Dev.
#>  g        (Intercept) 8.5145   2.9180
#>  Residual             0.6097   0.7808
#> Number of obs: 8, groups:  g, 8
#>
#> Fixed effects:
#>             Estimate Std. Error t value
#> (Intercept)    7.786      3.368   2.312
```

非商业版本

35.2.4 MCMCglmm

MCMCglmm 包 (Hadfield 2010) 采用 MCMC 算法拟合数据。

```
schools <- data.frame(y = y, sigma = sigma, g = g)
schools$g <- as.factor(schools$g)
# inverse-gamma prior with scale and shape equal to 0.001
prior1 <- list(
  R = list(V = diag(schools$sigma^2), fix = 1),
  G = list(G1 = list(V = 1, nu = 0.002))
)
# 为可重复
set.seed(20232023)
# 拟合模型
fit_mcmc <- MCMCglmm::MCMCglmm(
  y ~ 1, random = ~g, rcov = ~ idh(g):units,
  data = schools, prior = prior1, verbose = FALSE
)
# 输出结果
summary(fit_mcmc)
```

```
#>
#>  Iterations = 3001:12991
#>  Thinning interval  = 10
#>  Sample size  = 1000
#>
#>  DIC: -98.07615
#>
#>  G-structure:  ~g
#>
#>   post.mean  l-95% CI u-95% CI eff.samp
#> g     11.23 0.0004247    68.57    361.3
#>
#>  R-structure:  ~idh(g):units
#>
#>          post.mean l-95% CI u-95% CI eff.samp
#> g1.units       225      225      225        0
#> g2.units       100      100      100        0
#> g3.units       256      256      256        0
#> g4.units       121      121      121        0
#> g5.units        81       81       81        0
```

```
#> g6.units       121      121      121        0
#> g7.units       100      100      100        0
#> g8.units       324      324      324        0
#>
#>  Location effects: y ~ 1
#>
#>             post.mean l-95% CI u-95% CI eff.samp pMCMC
#> (Intercept)    7.6938  -0.5149  15.3275     1023 0.062 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

R-structure 表示残差方差，这是已知的参数。G-structure 表示随机截距的方差，Location effects 表示固定效应的截距。截距和 **nlme** 包的结果很接近。

35.2.5 cmdstanr

一般地，**rstan** 包使用的 stan 框架版本低于 **cmdstanr** 包，从 **rstan** 包切换到 **cmdstanr** 包，需要注意语法、函数的变化。**rstan** 和 **cmdstanr** 使用的 Stan 版本不同导致参数估计结果不同，结果可重复的条件非常苛刻，详见 Stan 参考手册。在都是较新的版本时，Stan 代码不需要做改动，如下：

```
data {
  int<lower=0> J; // 学校数目
  array[J] real y; // 测试效果的预测值
  array[J] real <lower=0> sigma; // 测试效果的标准差
}
parameters {
  real mu;
  real<lower=0> tau;
  vector[J] eta;
}
transformed parameters {
  vector[J] theta;
  theta = mu + tau * eta;
}
model {
  target += normal_lpdf(mu | 0, 100);
  target += normal_lpdf(tau | 0, 100);
  target += normal_lpdf(eta | 0, 1);
  target += normal_lpdf(y | theta, sigma);
}
```

©黄湘云

此处，给参数 μ, τ 添加了非常弱（模糊）的先验，结果将出现较大不同。

```
eight_schools_dat <- list(
  J = 8,
  y = c(28, 8, -3, 7, -1, 1, 18, 12),
  sigma = c(15, 10, 16, 11, 9, 11, 10, 18)
)
library(cmdstanr)
mod_eight_schools <- cmdstan_model(
  stan_file = "code/eight_schools.stan",
  compile = TRUE, cpp_options = list(stan_threads = TRUE)
)
fit_eight_schools <- mod_eight_schools$sample(
  data = eight_schools_dat, # 数据
  chains = 2,             # 总链条数
  parallel_chains = 2,    # 并行数目
  iter_warmup = 1000,     # 每条链预处理的迭代次数
  iter_sampling = 1000,   # 每条链采样的迭代次数
  threads_per_chain = 2, # 每条链设置 2 个线程
  seed = 20232023,        # 随机数种子
  show_messages = FALSE, # 不显示消息
  refresh = 0 # 不显示采样迭代的进度
)
```

结果保留 3 位有效数字，模型输出如下：

```
fit_eight_schools$summary(.num_args = list(sigfig = 3, notation = "dec"))
```

```
#> # A tibble: 19 x 10
#>    variable      mean    median    sd   mad      q5    q95  rhat ess_bulk
#>    <chr>        <dbl>     <dbl> <dbl> <dbl>   <dbl>  <dbl> <dbl>    <dbl>
#>  1 lp__      -50.6     -50.4     2.59  2.49  -55.4   -46.7   1.00     720.
#>  2 mu          7.79      7.81    5.05  4.76   -0.652  15.9   1.00    1519.
#>  3 tau         6.28      4.97    5.36  4.66    0.384  16.6   1.00     833.
#>  4 eta[1]      0.402     0.387   0.963 0.974  -1.18    1.99  1.00    3034.
#>  5 eta[2]      0.00672   0.00898 0.877 0.841  -1.45    1.46  1.00    2404.
#>  6 eta[3]     -0.169    -0.178   0.918 0.916  -1.68    1.38  1.00    2580.
#>  7 eta[4]     -0.0109   -0.0144  0.878 0.876  -1.43    1.45  1.00    2492.
#>  8 eta[5]     -0.340    -0.359   0.840 0.811  -1.69    1.06  1.00    1952.
#>  9 eta[6]     -0.186    -0.178   0.887 0.854  -1.64    1.31  1.00    2559.
#> 10 eta[7]      0.339     0.362   0.910 0.890  -1.25    1.82  1.00    2322.
#> 11 eta[8]      0.0687    0.0622  0.893 0.897  -1.35    1.53  1.00    2924.
```

```
#> 12 theta[1]  11.3       9.95    8.07  6.45    0.450  26.8   1.00    2421.
#> 13 theta[2]   7.75      7.64    6.06  5.45   -1.99   18.0   1.00    2510.
#> 14 theta[3]   6.27      6.91    7.44  6.28   -6.72   17.4   1.00    2381.
#> 15 theta[4]   7.83      7.81    6.75  5.98   -2.99   18.7   1.00    2551.
#> 16 theta[5]   5.25      5.59    6.02  5.67   -5.50   14.3   1.00    1958.
#> 17 theta[6]   6.29      6.58    6.65  6.16   -5.62   16.7   1.00    2506.
#> 18 theta[7]  10.6      10.0     6.76  6.22    0.486  22.6   1.00    2242.
#> 19 theta[8]   8.45      8.11    7.52  6.29   -2.99   21.2   1.00    2472.
#> # i 1 more variable: ess_tail <dbl>
```

模型采样过程的诊断结果如下：

```
fit_eight_schools$diagnostic_summary()
```

```
#> $num_divergent
#> [1] 0 0
#>
#> $num_max_treedepth
#> [1] 0 0
#>
#> $ebfmi
#> [1] 0.7979763 0.8891174
```

分层模型的参数 $\mu, \log(\tau)$ 的后验联合分布呈现经典的漏斗状。

```
bayesplot::mcmc_scatter(
  fit_eight_schools$draws(), pars = c("mu", "tau"),
  transform = list(tau = "log"), size = 2
) + labs(x = "$\\mu$", y = "$\\log(\\tau)$")
```

©黄湘云

图 35.5: 参数 $\mu, \log(\tau)$ 的联合分布

对于调用 **cmdstanr** 包拟合的模型，适合用 **bayesplot** 包来可视化后验分布和诊断采样。

35.3 案例：rats 数据

rats 数据最早来自 Gelfand 等 (1990)，记录 30 只小鼠每隔一周的重量，一共进行了 5 周。第一次记录是小鼠第 8 天的时候，第二次测量记录是第 15 天的时候，一直持续到第 36 天。下面在 R 环境中准备数据。

```
# 总共 30 只老鼠
N <- 30
# 总共进行 5 周
T <- 5
# 小鼠重量
y <- structure(c(
  151, 145, 147, 155, 135, 159, 141, 159, 177, 134,
  160, 143, 154, 171, 163, 160, 142, 156, 157, 152, 154, 139, 146,
  157, 132, 160, 169, 157, 137, 153, 199, 199, 214, 200, 188, 210,
  189, 201, 236, 182, 208, 188, 200, 221, 216, 207, 187, 203, 212,
  203, 205, 190, 191, 211, 185, 207, 216, 205, 180, 200, 246, 249,
  263, 237, 230, 252, 231, 248, 285, 220, 261, 220, 244, 270, 242,
```

©黄湘云

```
  248, 234, 243, 259, 246, 253, 225, 229, 250, 237, 257, 261, 248,
  219, 244, 283, 293, 312, 272, 280, 298, 275, 297, 350, 260, 313,
  273, 289, 326, 281, 288, 280, 283, 307, 286, 298, 267, 272, 285,
  286, 303, 295, 289, 258, 286, 320, 354, 328, 297, 323, 331, 305,
  338, 376, 296, 352, 314, 325, 358, 312, 324, 316, 317, 336, 321,
  334, 302, 302, 323, 331, 345, 333, 316, 291, 324
), .Dim = c(30, 5))
# 第几天
x <- c(8.0, 15.0, 22.0, 29.0, 36.0)
xbar <- 22.0
```

重复测量的小鼠重量数据 rats 如下表格 35.1 所示。

表格 35.1: 小鼠重量数据（部分）

	第 8 天	第 15 天	第 22 天	第 29 天	第 36 天
1	151	199	246	283	320
2	145	199	249	293	354
3	147	214	263	312	328
4	155	200	237	272	297
5	135	188	230	280	323
6	159	210	252	298	331

小鼠重量数据的分布和变化情况见下图，由图可以假定 30 只小鼠的重量服从正态分布，而 30 只小鼠的重量呈现一种线性增长趋势。

35.4 频率派方法

35.4.1 nlme

nlme 包适合长格式的数据，因此，先将小鼠数据整理成长格式。

```
rats_data <- data.frame(
  weight = as.vector(y),
  rats = rep(1:30, times = 5),
  days = rep(c(8, 15, 22, 29, 36), each = 30)
)
```

将 30 只小鼠的重量变化及回归曲线画出来，发现各只小鼠的回归线的斜率几乎一样，截距略有不同。不同小鼠的出生重量是不同，前面 Stan 采用变截距变斜率的混合效应模型拟合数据。

© 黄湘云

(a) 小鼠重量的分布 (b) 小鼠重量的变化

图 35.6: 30 只小鼠 5 次测量的数据

```
ggplot(data = rats_data, aes(x = days, y = weight)) +
  geom_point() +
  geom_smooth(formula = "y ~ x", method = "lm", se = FALSE) +
  theme_bw() +
  facet_wrap(facets = ~rats, labeller = "label_both", ncol = 6) +
  labs(x = "第几天", y = "重量")
```

© 黄湘云

图 35.7: 小鼠重量变化曲线

小鼠的重量随时间增长，不同小鼠的情况又会有所不同。作为一个参照，首先考虑变截距的随机效应模型。

$$y_{ij} = \beta_0 + \beta_1 * x_j + \alpha_i + \epsilon_{ij}, \quad i = 1, 2, \ldots, 30. \quad j = 1, 2, 3, 4, 5$$

其中，y_{ij} 表示第 i 只小鼠在第 j 次测量的重量，一共 30 只小鼠，共测量了 5 次。固定效应部分是 β_0 和 β_1 ，分别表示截距和斜率。随机效应部分是 α_i 和 ϵ_{ij} ，分别服从正态分布 $\alpha_i \sim \mathcal{N}(0, \sigma_\alpha^2)$ 和

$\epsilon_{ij} \sim \mathcal{N}(0, \sigma_\epsilon^2)$ 。σ_α^2 和 σ_ϵ^2 分别表示组间方差（group level）和组内方差（individual level）。

```
library(nlme)
rats_lme0 <- lme(data = rats_data, fixed = weight ~ days, random = ~ 1 | rats)
summary(rats_lme0)
```

```
#> Linear mixed-effects model fit by REML
#>   Data: rats_data
#>        AIC     BIC    logLik
#>   1145.302 1157.29 -568.6508
#>
#> Random effects:
#>  Formula: ~1 | rats
#>         (Intercept) Residual
#> StdDev:    14.03351 8.203811
#>
#> Fixed effects:  weight ~ days
#>                 Value Std.Error  DF  t-value p-value
#> (Intercept) 106.56762 3.0379720 119 35.07854       0
#> days          6.18571 0.0676639 119 91.41824       0
#>  Correlation:
#>      (Intr)
#> days -0.49
#>
#> Standardized Within-Group Residuals:
#>        Min         Q1        Med         Q3        Max
#> -2.7388198 -0.4770046  0.1261342  0.5634904  2.9981636
#>
#> Number of Observations: 150
#> Number of Groups: 30
```

当然，若考虑不同小鼠的生长速度不同（变化不是很大），可用变截距和变斜率的随机效应模型表示生长曲线模型，下面加载 **nlme** 包调用函数 `lme()` 拟合该模型。

```
library(nlme)
rats_lme <- lme(data = rats_data, fixed = weight ~ days, random = ~ days | rats)
summary(rats_lme)
```

```
#> Linear mixed-effects model fit by REML
#>   Data: rats_data
#>        AIC      BIC    logLik
#>   1107.373 1125.357 -547.6867
#>
```

©黄湘云


```
#> Random effects:
#>  Formula: ~days | rats
#>  Structure: General positive-definite, Log-Cholesky parametrization
#>             StdDev     Corr
#> (Intercept) 10.7425835 (Intr)
#> days         0.5105447 -0.159
#> Residual     6.0146608
#>
#> Fixed effects:  weight ~ days
#>                 Value Std.Error  DF  t-value p-value
#> (Intercept) 106.56762 2.2976183 119 46.38178       0
#> days          6.18571 0.1055912 119 58.58174       0
#>  Correlation:
#>      (Intr)
#> days -0.343
#>
#> Standardized Within-Group Residuals:
#>        Min         Q1        Med         Q3        Max
#> -2.6370825 -0.5394956  0.1187658  0.4927200  2.6090652
#>
#> Number of Observations: 150
#> Number of Groups: 30
```

模型输出结果中，固定效应中的截距项 (Intercept) 对应 106.56762，斜率 days 对应 6.18571。Stan 模型中截距参数 alpha0 的后验估计是 106.332，斜率参数 beta_c 的后验估计是 6.188。对比 Stan 和 **nlme** 包的拟合结果，可以发现贝叶斯和频率方法的结果是非常接近的。截距参数 alpha0 可以看作小鼠的初始（出生）重量，斜率参数 beta_c 可以看作小鼠的生长率 growth rate。

函数 lme() 的输出结果中，随机效应的随机截距标准差 10.7425835，对应 tau_alpha，表示每个小鼠的截距偏移量的波动。而随机斜率的标准差为 0.5105447，对应 tau_beta，相对随机截距标准差来说很小。残差标准差为 6.0146608，对应 tau_c，表示与小鼠无关的剩余量的波动，比如测量误差。总之，和 Stan 的结果有所不同，但相去不远。主要是前面的 Stan 模型没有考虑随机截距和随机斜率之间的相关性，这可以进一步调整 (Sorensen, Hohenstein, 和 Vasishth 2016) 。

```
# 参数的置信区间
intervals(rats_lme, level = 0.95)
```

```
#> Approximate 95% confidence intervals
#>
#>  Fixed effects:
#>                  lower       est.      upper
#> (Intercept) 102.018105 106.567619 111.117133
```

©贾俊平

```
#> days          5.976633   6.185714   6.394795
#>
#>  Random Effects:
#>   Level: rats
#>                            lower       est.      upper
#> sd((Intercept))        7.5159648 10.7425835 15.3543961
#> sd(days)               0.3661761  0.5105447  0.7118322
#> cor((Intercept),days) -0.5659916 -0.1590236  0.3102623
#>
#>  Within-group standard error:
#>    lower     est.    upper
#> 5.197218 6.014661 6.960675
```

Stan 输出中，截距项 alpha、斜率项 beta 参数的标准差分别是 `tau_alpha` 和 `tau_beta` ，残差标准差参数 `tau_c` 的估计为 6.1。简单起见，没有考虑截距项和斜率项的相关性，即不考虑小鼠出生时的重量和生长率的相关性，一般来说，应该是有关系的。函数 `lme()` 的输出结果中给出了截距项和斜率项的相关性为 -0.343，随机截距和随机斜率的相关性为 -0.159。

计算与 Stan 输出中的截距项 `alpha_c` 对应的量，结合函数 `lme()` 的输出，截距、斜率加和之后，如下

```
106.56762 + 6.18571 * 22
```

```
#> [1] 242.6532
```

值得注意，Stan 代码中对时间 days 做了中心化处理，即 $x_t - \bar{x}$，目的是降低采样时参数 α_i 和 β_i 之间的相关性，而在拟合函数 `lme()` 中没有做处理，因此，结果无需转化，而且更容易解释。

```
fit_lm <- lm(weight ~ days, data = rats_data)
summary(fit_lm)
```

```
#>
#> Call:
#> lm(formula = weight ~ days, data = rats_data)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -38.253 -11.278   0.197   7.647  64.047
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 106.5676     3.2099   33.20   <2e-16 ***
#> days          6.1857     0.1331   46.49   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

©黄湘云

```
#>
#> Residual standard error: 16.13 on 148 degrees of freedom
#> Multiple R-squared:  0.9359, Adjusted R-squared:  0.9355
#> F-statistic:  2161 on 1 and 148 DF,  p-value: < 2.2e-16
```

采用简单线性模型即可获得与 **nlme** 包非常接近的估计结果，主要是小鼠重量的分布比较正态，且随时间的变化非常线性。

35.4.2 lavaan

lavaan 包 (Rosseel 2012) 主要是用来拟合结构方程模型，而生长曲线模型可以放在该框架下。所以，也可以用 **lavaan** 包来拟合，并且，它提供的函数 `growth()` 可以直接拟合生长曲线模型。

```
library(lavaan)
# 设置矩阵 y 的列名
colnames(y) <- c("t1","t2","t3","t4","t5")
rats_growt_model <- "
  # intercept and slope with fixed coefficients
  intercept =~ 1*t1 + 1*t2 + 1*t3 + 1*t4 + 1*t5
  days =~ 0*t1 + 1*t2 + 2*t3 + 3*t4 + 4*t5

  # if we fix the variances to be equal, the models are now identical.
  t1 ~~ resvar*t1
  t2 ~~ resvar*t2
  t3 ~~ resvar*t3
  t4 ~~ resvar*t4
  t5 ~~ resvar*t5
"
```

其中，算子符号 `=~` 定义潜变量，`~~` 定义残差协方差，intercept 表示截距，days 表示斜率。假定 5 次测量的测量误差（组内方差）是相同的。拟合模型的代码如下：

```
rats_growth_fit <- growth(rats_growt_model, data = y)
```

提供函数 `summary()` 获得模型输出，结果如下：

```
summary(rats_growth_fit, fit.measures = TRUE)
```

```
#> lavaan 0.6.17 ended normally after 87 iterations
#>
#>   Estimator                                         ML
#>   Optimization method                           NLMINB
#>   Number of model parameters                        10
#>   Number of equality constraints                     4
```

©黄海广

```
#>
#>   Number of observations                          30
#>
#> Model Test User Model:
#>
#>   Test statistic                             106.203
#>   Degrees of freedom                              14
#>   P-value (Chi-square)                         0.000
#>
#> Model Test Baseline Model:
#>
#>   Test statistic                             247.075
#>   Degrees of freedom                              10
#>   P-value                                      0.000
#>
#> User Model versus Baseline Model:
#>
#>   Comparative Fit Index (CFI)                  0.611
#>   Tucker-Lewis Index (TLI)                     0.722
#>
#> Loglikelihood and Information Criteria:
#>
#>   Loglikelihood user model (H0)             -548.029
#>   Loglikelihood unrestricted model (H1)     -494.927
#>
#>   Akaike (AIC)                              1108.057
#>   Bayesian (BIC)                            1116.465
#>   Sample-size adjusted Bayesian (SABIC)     1097.783
#>
#> Root Mean Square Error of Approximation:
#>
#>   RMSEA                                        0.469
#>   90 Percent confidence interval - lower       0.388
#>   90 Percent confidence interval - upper       0.554
#>   P-value H_0: RMSEA <= 0.050                  0.000
#>   P-value H_0: RMSEA >= 0.080                  1.000
#>
#> Standardized Root Mean Square Residual:
#>
#>   SRMR                                         0.151
```

©黄湘云

```
#>
#> Parameter Estimates:
#>
#>   Standard errors                             Standard
#>   Information                                 Expected
#>   Information saturated (h1) model          Structured
#>
#> Latent Variables:
#>                    Estimate  Std.Err  z-value  P(>|z|)
#>   intercept =~
#>     t1                1.000
#>     t2                1.000
#>     t3                1.000
#>     t4                1.000
#>     t5                1.000
#>   days =~
#>     t1                0.000
#>     t2                1.000
#>     t3                2.000
#>     t4                3.000
#>     t5                4.000
#>
#> Covariances:
#>                    Estimate  Std.Err  z-value  P(>|z|)
#>   intercept ~~
#>     days              8.444    8.521    0.991    0.322
#>
#> Intercepts:
#>                    Estimate  Std.Err  z-value  P(>|z|)
#>     intercept       156.053    2.123   73.516    0.000
#>     days             43.300    0.727   59.582    0.000
#>
#> Variances:
#>                    Estimate  Std.Err  z-value  P(>|z|)
#>    .t1      (rsvr)   36.176    5.393    6.708    0.000
#>    .t2      (rsvr)   36.176    5.393    6.708    0.000
#>    .t3      (rsvr)   36.176    5.393    6.708    0.000
#>    .t4      (rsvr)   36.176    5.393    6.708    0.000
#>    .t5      (rsvr)   36.176    5.393    6.708    0.000
#>     intrcpt         113.470   35.052    3.237    0.001
```

```
#>     days             12.226    4.126    2.963    0.003
```

输出结果显示 **lavaan** 包的函数 `growth()` 采用极大似然估计方法。协方差部分 `Covariances:` 随机效应中斜率和截距的协方差。截距部分 `Intercepts:` 对应于混合效应模型的固定效应部分。方差部分 `Variances:` 对应于混合效应模型的随机效应部分，包括残差方差、斜率和截距的方差。不难看出，这和前面 **nlme** 包的输出结果差别很大。原因是 **lavaan** 包将测量的次序从 0 开始计，0 代表小鼠出生后的第 8 天。也就是说，**lavaan** 采用的是次序标记，而不是实际数据。将测量发生的时间（第几天）换算成次序（第几次），并从 0 开始计，则函数 `lme()` 的输出和函数 `growth()` 就一致了。

```
# 重新组织数据
rats_data2 <- data.frame(
  weight = as.vector(y),
  rats = rep(1:30, times = 5),
  days = rep(c(0, 1, 2, 3, 4), each = 30)
)
# ML 方法估计模型参数
rats_lme2 <- lme(data = rats_data2, fixed = weight ~ days, random = ~ days | rats, method = "ML")
summary(rats_lme2)
```

```
#> Linear mixed-effects model fit by maximum likelihood
#>   Data: rats_data2
#>        AIC      BIC    logLik
#>   1108.057 1126.121 -548.0287
#>
#> Random effects:
#>  Formula: ~days | rats
#>  Structure: General positive-definite, Log-Cholesky parametrization
#>             StdDev    Corr
#> (Intercept) 10.652385 (Intr)
#> days         3.496584 0.227
#> Residual     6.014613
#>
#> Fixed effects:  weight ~ days
#>                Value Std.Error  DF  t-value p-value
#> (Intercept) 156.0533 2.1370181 119 73.02387       0
#> days         43.3000 0.7316138 119 59.18423       0
#>  Correlation:
#>      (Intr)
#> days 0.026
#>
#> Standardized Within-Group Residuals:
```

©黄湘云

```
#>        Min         Q1        Med         Q3        Max
#> -2.6317200 -0.5421564  0.1154349  0.4948020  2.6188195
#>
#> Number of Observations: 150
#> Number of Groups: 30
```

可以看到函数 `growth()` 给出的截距和斜率的协方差估计为 8.444，函数 `lme()` 给出对应截距和斜率的标准差分别是 10.652390 和 3.496588，它们的相关系数为 0.227，则函数 `lme()` 给出的协方差估计为 `10.652390*3.496588*0.227` ，即 8.455，协方差估计比较一致。同理，比较两个输出结果中的其它成分，函数 `growth()` 给出的残差方差估计为 36.176，则残差标准差估计为 6.0146，结合函数 `lme()` 给出的 `Random effects:` 中 `Residual`，结果完全一样。函数 `growth()` 给出的 `Intercepts:` 对应于函数 `lme()` 给出的固定效应部分，结果也是完全一样。

针对模型拟合对象 `rats_growth_fit` ，除了函数 `summary()` 可以汇总结果，**lavaan** 包还提供 `AIC()` 、`BIC()` 和 `logLik()` 等函数，分别可以提取 AIC、BIC 和对数似然值，`AIC()` 和 `logLik()` 结果与前面的函数 `lme()` 的输出是一样的，而 `BIC()` 不同。

35.4.3 lme4

当采用 **lme4** 包拟合数据的时候，发现输出结果与 **nlme** 包几乎相同。

```
rats_lme4 <- lme4::lmer(weight ~ days + (days | rats), data = rats_data)
summary(rats_lme4)
```

```
#> Linear mixed model fit by REML ['lmerMod']
#> Formula: weight ~ days + (days | rats)
#>    Data: rats_data
#>
#> REML criterion at convergence: 1095.4
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -2.6371 -0.5395  0.1188  0.4927  2.6091
#>
#> Random effects:
#>  Groups   Name        Variance Std.Dev. Corr
#>  rats     (Intercept) 115.4239 10.7435
#>           days          0.2607  0.5106  -0.16
#>  Residual              36.1753  6.0146
#> Number of obs: 150, groups:  rats, 30
#>
#> Fixed effects:
```

```
#>             Estimate Std. Error t value
#> (Intercept) 106.5676     2.2978   46.38
#> days          6.1857     0.1056   58.58
#>
#> Correlation of Fixed Effects:
#>      (Intr)
#> days -0.343
```

35.4.4 glmmTMB

glmmTMB 包基于 Template Model Builder (TMB)，拟合广义线性混合效应模型，公式语法与 **lme4** 包一致。

```
rats_glmmtmb <- glmmTMB::glmmTMB(weight ~ days + (days | rats), REML = TRUE, data = rats_data)
summary(rats_glmmtmb)
```

```
#>  Family: gaussian  ( identity )
#> Formula:          weight ~ days + (days | rats)
#> Data: rats_data
#>
#>      AIC      BIC   logLik deviance df.resid
#>   1107.4   1125.4   -547.7   1095.4      146
#>
#> Random effects:
#>
#> Conditional model:
#>  Groups   Name        Variance Std.Dev. Corr
#>  rats     (Intercept) 115.4195 10.7433
#>           days          0.2607  0.5106  -0.16
#>  Residual              36.1756  6.0146
#> Number of obs: 150, groups:  rats, 30
#>
#> Dispersion estimate for gaussian family (sigma^2): 36.2
#>
#> Conditional model:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept) 106.5676     2.2977   46.38   <2e-16 ***
#> days          6.1857     0.1056   58.58   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

结果与 **nlme** 包完全一样。

35.4.5 MASS

MASS 包的结果与前面完全一致。

```
rats_mass <- MASS::glmmPQL(
  fixed = weight ~ days, random = ~ days | rats,
  data = rats_data, family = gaussian(), verbose = FALSE
)
summary(rats_mass)
```

```
#> Linear mixed-effects model fit by maximum likelihood
#>   Data: rats_data
#>   AIC BIC logLik
#>    NA  NA     NA
#>
#> Random effects:
#>  Formula: ~days | rats
#>  Structure: General positive-definite, Log-Cholesky parametrization
#>             StdDev     Corr
#> (Intercept) 10.4945051 (Intr)
#> days         0.4994795 -0.15
#> Residual     6.0146664
#>
#> Variance function:
#>  Structure: fixed weights
#>  Formula: ~invwt
#> Fixed effects:  weight ~ days
#>                 Value Std.Error  DF  t-value p-value
#> (Intercept) 106.56762 2.2742917 119 46.85750       0
#> days          6.18571 0.1045112 119 59.18709       0
#>  Correlation:
#>      (Intr)
#> days -0.343
#>
#> Standardized Within-Group Residuals:
#>        Min         Q1        Med         Q3        Max
#> -2.6316763 -0.5421361  0.1154445  0.4947986  2.6187979
#>
#> Number of Observations: 150
```

```
#> Number of Groups: 30
```

35.4.6 spaMM

spaMM 包的结果与前面完全一致。

```
rats_spamm <- spaMM::fitme(weight ~ days + (days | rats), data = rats_data)
summary(rats_spamm)
```

```
#> formula: weight ~ days + (days | rats)
#> ML: Estimation of ranCoefs and phi by ML.
#>     Estimation of fixed effects by ML.
#> Estimation of phi by 'outer' ML, maximizing logL.
#> family: gaussian( link = identity )
#>  ------------ Fixed effects (beta) ------------
#>             Estimate Cond. SE t-value
#> (Intercept)  106.568   2.2591   47.17
#> days           6.186   0.1038   59.58
#>  --------------- Random effects ---------------
#> Family: gaussian( link = identity )
#>          --- Random-coefficients Cov matrices:
#>  Group        Term   Var.   Corr.
#>   rats (Intercept)  110.1
#>   rats        days 0.2495 -0.1507
#> # of obs: 150; # of groups: rats, 30
#>  -------------- Residual variance  ------------
#> phi estimate was 36.1755
#>  ------------- Likelihood values  -------------
#>                         logLik
#> logL       (p_v(h)): -548.0287
```

```
 --------------- Random effects ---------------
Family: gaussian( link = identity )
         --- Random-coefficients Cov matrices:
 Group        Term   Var.   Corr.
  rats (Intercept)  110.1
  rats        days 0.2495 -0.1507
# of obs: 150; # of groups: rats, 30
```

随机效应的截距方差 110.1，斜率方差 0.2495，则标准差分别是 10.49 和 0.499，相关性为 -0.1507。

```
 -------------- Residual variance  ------------
phi estimate was 36.1755
```

残差方差为 36.1755，则标准差为 6.0146。

35.4.7 hglm

hglm 包 (Rönnegård, Shen, 和 Alam 2010) 可以拟合分层广义线性模型，线性混合效应模型和广义线性混合效应模型，随机效应和响应变量服从的分布可以很广泛，使用语法与 **lme4** 包一样。

```
rats_hglm <- hglm::hglm2(weight ~ days + (days | rats), data = rats_data)
summary(rats_hglm)
```

```
#> Call:
#> hglm2.formula(meanmodel = weight ~ days + (days | rats), data = rats_data)
#>
#> ----------
#> MEAN MODEL
#> ----------
#>
#> Summary of the fixed effects estimates:
#>
#>             Estimate Std. Error t-value Pr(>|t|)
#> (Intercept) 106.5676     2.1787   48.91   <2e-16 ***
#> days          6.1857     0.1029   60.13   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> Note: P-values are based on 103 degrees of freedom
#>
#> Summary of the random effects estimates:
#>
#>                     Estimate Std. Error
#> (Intercept)| rats:1  -0.1705     5.3422
#> (Intercept)| rats:2  -9.8655     5.3422
#> (Intercept)| rats:3   2.7201     5.3422
#> ...
#> NOTE: to show all the random effects, use print(summary(hglm.object), print.ranef = TRUE).
#>
#> Summary of the random effects estimates:
#>
#>              Estimate Std. Error
#> days| rats:1  -0.1213      0.229
#> days| rats:2   0.7260      0.229
#> days| rats:3   0.3280      0.229
```

```
#> ...
#> NOTE: to show all the random effects, use print(summary(hglm.object), print.ranef = TRUE).
#>
#> ----------------
#> DISPERSION MODEL
#> ----------------
#>
#> NOTE: h-likelihood estimates through EQL can be biased.
#>
#> Dispersion parameter for the mean model:
#> [1] 37.09572
#>
#> Model estimates for the dispersion term:
#>
#> Link = log
#>
#> Effects:
#>   Estimate Std. Error
#>     3.6135     0.1391
#>
#> Dispersion = 1 is used in Gamma model on deviances to calculate the standard error(s).
#>
#> Dispersion parameter for the random effects:
#> [1] 103.4501   0.2407
#>
#> Dispersion model for the random effects:
#>
#> Link = log
#>
#> Effects:
#> .|Random1
#>   Estimate Std. Error
#>     4.6391     0.3069
#>
#> .|Random2
#>   Estimate Std. Error
#>    -1.4241     0.2920
#>
#> Dispersion = 1 is used in Gamma model on deviances to calculate the standard error(s).
#>
```

```
#> EQL estimation converged in 5 iterations.
```

固定效应的截距和斜率都是和 **nlme** 包的输出结果一致。值得注意，随机效应和模型残差都是以发散参数（Dispersion parameter）来表示的，模型残差方差为 37.09572，则标准差为 6.0906，随机效应的随机截距和随机斜率的方差分别为 103.4501 和 0.2407，则标准差分别为 10.1710 和 0.4906，这与 **nlme** 包的结果也是一致的。

35.4.8 mgcv

先考虑一个变截距的混合效应模型

$$y_{ij} = \beta_0 + \beta_1 * x_j + \alpha_i + \epsilon_{ij}, \quad i = 1, 2, \ldots, 30. \quad j = 1, 2, 3, 4, 5$$

假设随机效应服从独立同正态分布，等价于在似然函数中添加一个岭惩罚。广义可加模型在一定形式下和上述混合效应模型存在等价关系，在广义可加模型中，可以样条表示随机效应。**mgcv** 包拟合代码如下。

```
library(mgcv)
rats_data$rats <- as.factor(rats_data$rats)
rats_gam <- gam(weight ~ days + s(rats, bs = "re"), data = rats_data)
```

其中，参数取值 `bs = "re"` 指定样条类型，re 是 Random effects 的简写。

```
summary(rats_gam)
```

```
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> weight ~ days + s(rats, bs = "re")
#>
#> Parametric coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 106.56762    3.03797   35.08   <2e-16 ***
#> days          6.18571    0.06766   91.42   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>           edf Ref.df     F p-value
#> s(rats) 27.14     29 14.63  <2e-16 ***
#> ---
```

©王德辉

```
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.983   Deviance explained = 98.6%
#> GCV = 83.533  Scale est. = 67.303    n = 150
```

其中，残差的方差 Scale est. $= 67.303$ ，则标准差为 $\sigma_\epsilon = 8.2038$ 。随机效应的标准差如下

```
gam.vcomp(rats_gam, rescale = TRUE)
```

```
#>  s(rats)
#> 14.03351
```

rescale = TRUE 表示恢复至原数据的尺度，标准差 $\sigma_\alpha = 14.033$。可以看到，固定效应和随机效应的估计结果与 **nlme** 包等完全一致。若考虑变截距和变斜率的混合效应模型，拟合代码如下：

```
rats_gam1 <- gam(
  weight ~ days + s(rats, bs = "re") + s(rats, by = days, bs = "re"),
  data = rats_data, method = "REML"
)
summary(rats_gam1)
```

```
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> weight ~ days + s(rats, bs = "re") + s(rats, by = days, bs = "re")
#>
#> Parametric coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 106.5676     2.2365   47.65   <2e-16 ***
#> days          6.1857     0.1028   60.18   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>                edf Ref.df     F p-value
#> s(rats)      21.80     29 183.9  <2e-16 ***
#> s(rats):days 23.47     29 201.8  <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.991   Deviance explained = 99.4%
```

©黄湘云

```
#> -REML = 547.89  Scale est. = 36.834    n = 150
```

输出结果中，固定效应部分的结果和 **nlme** 包完全一样。

```
gam.vcomp(rats_gam1, rescale = TRUE)
```

```
#>
#> Standard deviations and 0.95 confidence intervals:
#>
#>                std.dev     lower      upper
#> s(rats)      10.3107538 7.2978205 14.5675882
#> s(rats):days  0.4916736 0.3571229  0.6769181
#> scale         6.0691017 5.2454835  7.0220401
#>
#> Rank: 3/3
```

输出结果中，依次是随机效应的截距、斜率和残差的标准差（标准偏差），和 **nlme** 包给出的结果非常接近。

mgcv 包还提供函数 `gamm()`，它将混合效应和固定效应分开，在拟合 LMM 模型时，它类似 **nlme** 包的函数 `lme()`。返回一个含有 lme 和 gam 两个元素的列表，前者包含随机效应的估计，后者是固定效应的估计，固定效应中可以添加样条（或样条表示的简单随机效益，比如本节前面提及的模型）。实际上，函数 `gamm()` 分别调用 **nlme** 包和 **MASS** 包来拟合 LMM 模型和 GLMM 模型。

```
rats_gamm <- gamm(weight ~ days, random = list(rats = ~days), method = "REML", data = rats_data)
# LME
summary(rats_gamm$lme)
```

```
#> Linear mixed-effects model fit by REML
#>   Data: strip.offset(mf)
#>        AIC      BIC    logLik
#>   1107.373 1125.357 -547.6867
#>
#> Random effects:
#>  Formula: ~days | rats
#>  Structure: General positive-definite, Log-Cholesky parametrization
#>             StdDev     Corr
#> (Intercept) 10.7433332 (Intr)
#> days         0.5105577 -0.159
#> Residual     6.0146119
#>
#> Fixed effects:  y ~ X - 1
#>                  Value Std.Error  DF  t-value p-value
#> X(Intercept) 106.56762 2.2977301 119 46.37952       0
```

©李贤平 江苏

```
#> Xdays          6.18571 0.1055931 119 58.58069       0
#>  Correlation:
#>       X(Int)
#> Xdays -0.343
#>
#> Standardized Within-Group Residuals:
#>        Min         Q1        Med         Q3        Max
#> -2.6371079 -0.5394997  0.1187534  0.4927191  2.6091109
#>
#> Number of Observations: 150
#> Number of Groups: 30
```

```
# GAM
summary(rats_gamm$gam)
```

```
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> weight ~ days
#>
#> Parametric coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 106.5676     2.2977   46.38   <2e-16 ***
#> days          6.1857     0.1056   58.58   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>
#> R-sq.(adj) =  0.935
#>   Scale est. = 36.176    n = 150
```

35.5 贝叶斯方法

35.5.1 rstan

初始化模型参数，设置采样算法的参数。

```
# 迭代链
chains <- 4
```

汪荣鑫©著

```
# 迭代次数
iter <- 1000
# 初始值
init <- rep(list(list(
  alpha = rep(250, 30), beta = rep(6, 30),
  alpha_c = 150, beta_c = 10,
  tausq_c = 1, tausq_alpha = 1,
  tausq_beta = 1
)), chains)
```

接下来，基于重复测量数据，建立线性生长曲线模型：

$$
\begin{aligned}
\alpha_c &\sim \mathcal{N}(0,100) \quad \beta_c \sim \mathcal{N}(0,100) \\
\tau_\alpha^2 &\sim \text{inv_gamma}(0.001, 0.001) \\
\tau_\beta^2 &\sim \text{inv_gamma}(0.001, 0.001) \\
\tau_c^2 &\sim \text{inv_gamma}(0.001, 0.001) \\
\alpha_n &\sim \mathcal{N}(\alpha_c, \tau_\alpha) \quad \beta_n \sim \mathcal{N}(\beta_c, \tau_\beta) \\
y_{nt} &\sim \mathcal{N}(\alpha_n + \beta_n * (x_t - \bar{x}), \tau_c) \\
n &= 1,2,\ldots,N \quad t = 1,2,\ldots,T
\end{aligned}
$$

其中，$\alpha_c, \beta_c, \tau_c, \tau_\alpha, \tau_\beta$ 为无信息先验，$\bar{x} = 22$ 表示第 22 天，$N = 30$ 和 $T = 5$ 分别表示实验中的小鼠数量和测量次数，下面采用 Stan 编码、编译、采样和拟合模型。

```
rats_fit <- stan(
  model_name = "rats",
  model_code = "
  data {
    int<lower=0> N;
    int<lower=0> T;
    vector[T] x;
    matrix[N,T] y;
    real xbar;
  }
  parameters {
    vector[N] alpha;
    vector[N] beta;

    real alpha_c;
    real beta_c;          // beta.c in original bugs model
```

汪诚义 ©

```
    real<lower=0> tausq_c;
    real<lower=0> tausq_alpha;
    real<lower=0> tausq_beta;
  }
  transformed parameters {
    real<lower=0> tau_c;       // sigma in original bugs model
    real<lower=0> tau_alpha;
    real<lower=0> tau_beta;

    tau_c = sqrt(tausq_c);
    tau_alpha = sqrt(tausq_alpha);
    tau_beta = sqrt(tausq_beta);
  }
  model {
    alpha_c ~ normal(0, 100);
    beta_c ~ normal(0, 100);
    tausq_c ~ inv_gamma(0.001, 0.001);
    tausq_alpha ~ inv_gamma(0.001, 0.001);
    tausq_beta ~ inv_gamma(0.001, 0.001);
    alpha ~ normal(alpha_c, tau_alpha); // vectorized
    beta ~ normal(beta_c, tau_beta);  // vectorized
    for (n in 1:N)
      for (t in 1:T)
        y[n,t] ~ normal(alpha[n] + beta[n] * (x[t] - xbar), tau_c);
  }
  generated quantities {
    real alpha0;
    alpha0 = alpha_c - xbar * beta_c;
  }
  ",
  data = list(N = N, T = T, y = y, x = x, xbar = xbar),
  chains = chains, init = init, iter = iter,
  verbose = FALSE, refresh = 0, seed = 20190425
)
```

模型输出结果如下：

```
print(rats_fit, pars = c("alpha", "beta"), include = FALSE, digits = 1)

#> Inference for Stan model: rats.
#> 4 chains, each with iter=1000; warmup=500; thin=1;
```

© 统计学习基础


```
#> post-warmup draws per chain=500, total post-warmup draws=2000.
#>
#>               mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
#> alpha_c      242.5     0.1  2.7  237.1  240.6  242.5  244.3  247.7  1728    1
#> beta_c         6.2     0.0  0.1    6.0    6.1    6.2    6.3    6.4  2205    1
#> tausq_c       37.0     0.2  5.8   27.6   32.9   36.4   40.6   50.0   947    1
#> tausq_alpha  218.0     1.6 64.1  125.3  172.0  207.8  253.0  372.8  1703    1
#> tausq_beta     0.3     0.0  0.1    0.1    0.2    0.3    0.3    0.5  1481    1
#> tau_c          6.1     0.0  0.5    5.3    5.7    6.0    6.4    7.1   938    1
#> tau_alpha     14.6     0.0  2.1   11.2   13.1   14.4   15.9   19.3  1826    1
#> tau_beta       0.5     0.0  0.1    0.4    0.5    0.5    0.6    0.7  1429    1
#> alpha0       106.3     0.1  3.6   99.5  103.9  106.4  108.8  113.2  1965    1
#> lp__        -438.0     0.3  7.0 -452.8 -442.6 -437.4 -433.2 -425.4   558    1
#>
#> Samples were drawn using NUTS(diag_e) at Tue Jan 23 09:57:08 2024.
#> For each parameter, n_eff is a crude measure of effective sample size,
#> and Rhat is the potential scale reduction factor on split chains (at
#> convergence, Rhat=1).
```

alpha_c 表示小鼠 5 次测量的平均重量，beta_c 表示小鼠体重的增长率，α_i, β_i 分别表示第 i 只小鼠在第 22 天（第 3 次测量或 $x_t = \bar{x}$ ）的重量和增长率（每日增加的重量）。

对于分量众多的参数向量，比较适合用岭线图展示后验分布，下面调用 **bayesplot** 包绘制参数向量 $\boldsymbol{\alpha}, \boldsymbol{\beta}$ 的后验分布。

```r
# plot(rats_fit, pars = "alpha", show_density = TRUE, ci_level = 0.8, outer_level = 0.95)
bayesplot::mcmc_areas_ridges(rats_fit, pars = paste0("alpha", "[", 1:30, "]")) +
  scale_y_discrete(labels = scales::parse_format())
```

©黄湘云

图 35.8: 参数 $\boldsymbol{\alpha}$ 的后验分布

参数向量 $\boldsymbol{\alpha}$ 的后验估计可以看作 $x_t = \bar{x}$ 时小鼠的重量，上图即为各个小鼠重量的后验分布。

```
# plot(rats_fit, pars = "beta", ci_level = 0.8, outer_level = 0.95)
```

© 基础统计


```
bayesplot::mcmc_areas_ridges(rats_fit, pars = paste0("beta", "[", 1:30, "]")) +
  scale_y_discrete(labels = scales::parse_format())
```

图 35.9: 参数 β 的后验分布

参数向量 β 的后验估计可以看作是小鼠的重量的增长率，上图即为各个小鼠重量的增长率的后验分布。

35.5.2 cmdstanr

从 rstan 包转 cmdstanr 包是非常容易的，只要语法兼容，模型代码可以原封不动。

```
library(cmdstanr)
mod_rats <- cmdstan_model(
  stan_file = "code/rats.stan",
  compile = TRUE, cpp_options = list(stan_threads = TRUE)
)
fit_rats <- mod_rats$sample(
  data = list(N = N, T = T, y = y, x = x, xbar = xbar), # 数据
  chains = 2,            # 总链条数
  parallel_chains = 2,   # 并行数目
  iter_warmup = 1000,    # 每条链预处理的迭代次数
  iter_sampling = 1000,  # 每条链采样的迭代次数
  threads_per_chain = 2, # 每条链设置 2 个线程
  seed = 20232023,       # 随机数种子
  show_messages = FALSE, # 不显示消息
  adapt_delta = 0.9,     # 接受率
  refresh = 0 # 不显示采样迭代的进度
)
```

模型输出

```
# 显示除了参数 alpha 和 beta 以外的结果
vars <- setdiff(fit_rats$metadata()$stan_variables, c("alpha", "beta"))
fit_rats$summary(variables = vars)
```

```
#> # A tibble: 10 x 10
#>    variable       mean   median      sd     mad       q5      q95  rhat ess_bulk
#>    <chr>         <dbl>    <dbl>   <dbl>   <dbl>    <dbl>    <dbl> <dbl>    <dbl>
#>  1 lp__       -438.    -438.     6.87    6.83   -450.    -427.    1.01      586.
#>  2 alpha_c     243.     242.     2.79    2.70    238.     247.    1.00     3528.
#>  3 beta_c        6.18     6.18   0.106   0.107     6.01     6.36  1.00     3494.
#>  4 tausq_c      37.4     36.8    5.57    5.49     28.9     47.1   1.00     1720.
#>  5 tausq_alp~  217.     208.    63.5    57.7     134.     335.    1.00     3104.
#>  6 tausq_beta    0.275    0.259  0.0975  0.0884    0.148    0.457 0.999    2070.
#>  7 tau_c         6.10     6.07   0.451   0.453     5.38     6.86  1.00     1720.
#>  8 tau_alpha    14.6     14.4    2.06    2.00     11.6     18.3   1.00     3104.
#>  9 tau_beta      0.517    0.509  0.0896  0.0882    0.385    0.676 0.999    2070.
```

©贾俊平

```
#> 10 alpha0      106.     107.     3.63    3.68    100.     112.    0.999    3565.
#> # i 1 more variable: ess_tail <dbl>
```

诊断信息

```
fit_rats$diagnostic_summary()
```

```
#> $num_divergent
#> [1] 0 0
#>
#> $num_max_treedepth
#> [1] 0 0
#>
#> $ebfmi
#> [1] 0.8995806 0.8828313
```

35.5.3 brms

brms 包是基于 **rstan** 包的，基于 Stan 语言做贝叶斯推断，提供与 lme4 包一致的公式语法，且扩展了模型种类。

```
rats_brms <- brms::brm(weight ~ days + (days | rats), data = rats_data)
summary(rats_brms)
```

```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: weight ~ days + (days | rats)
   Data: rats_data (Number of observations: 150)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Group-Level Effects:
~rats (Number of levels: 30)
                    Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)          11.27      2.23     7.36    16.08 1.00     2172     2939
sd(days)                0.54      0.09     0.37     0.74 1.00     1380     2356
cor(Intercept,days)    -0.11      0.24    -0.53     0.39 1.00      920     1541

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept   106.47      2.47   101.61   111.23 1.00     2173     2768
days          6.18      0.11     5.96     6.41 1.00     1617     2177
```

©草稿

```
Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     6.15      0.47     5.30     7.14 1.00     1832     3151

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

35.5.4 rstanarm

rstanarm 包与 **brms** 包类似，区别是前者预编译了 Stan 模型，后者根据输入数据和模型编译即时编译，此外，后者支持的模型范围更加广泛。

```
library(rstanarm)
rats_rstanarm <- stan_lmer(formula = weight ~ days + (days | rats), data = rats_data)
summary(rats_rstanarm)
```

```
Model Info:
 function:     stan_lmer
 family:       gaussian [identity]
 formula:      weight ~ days + (days | rats)
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 150
 groups:       rats (30)

Estimates:
                                      mean     sd      10%     50%     90%
(Intercept)                         106.575   2.236 103.789 106.559 109.415
days                                  6.187   0.111   6.048   6.185   6.329
sigma                                 6.219   0.497   5.626   6.183   6.862
Sigma[rats:(Intercept),(Intercept)] 103.927  42.705  57.329  98.128 159.086
Sigma[rats:days,(Intercept)]         -0.545   1.492  -2.361  -0.402   1.162
Sigma[rats:days,days]                 0.304   0.112   0.181   0.285   0.445

MCMC diagnostics
                                    mcse  Rhat  n_eff
(Intercept)                         0.043 1.000 2753
days                                0.003 1.005 1694
sigma                               0.015 1.001 1172
```

©版权所有

```
Sigma[rats:(Intercept),(Intercept)] 1.140 1.000 1403
Sigma[rats:days,(Intercept)]        0.054 1.006  772
Sigma[rats:days,days]               0.003 1.000 1456

For each parameter, mcse is Monte Carlo standard error,
n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor
on split chains (at convergence Rhat=1).
```

固定效应的部分，截距和斜率如下：

```
Estimates:
                                     mean    sd      10%     50%     90%
(Intercept)                        106.575   2.236 103.789 106.559 109.415
days                                 6.187   0.111   6.048   6.185   6.329
```

模型残差的标准差 sigma、随机效应 Sigma 的随机截距的方差 103.927 、随机斜率的方差 0.304 及其协方差 -0.545。

```
sigma                                6.219   0.497   5.626   6.183   6.862
Sigma[rats:(Intercept),(Intercept)] 103.927  42.705  57.329  98.128 159.086
Sigma[rats:days,(Intercept)]         -0.545   1.492  -2.361  -0.402   1.162
Sigma[rats:days,days]                0.304   0.112   0.181   0.285   0.445
```

rstanarm 和 **brms** 包的结果基本一致的。

35.5.5 blme

blme 包 (Chung 等 2013) 基于 **lme4** 包 (D. Bates 等 2015) 拟合贝叶斯线性混合效应模型。参考前面 **rstan** 小节中关于模型参数的先验设置，下面将残差方差的先验设置为逆伽马分布，随机效应的协方差设置为扁平分布。发现拟合结果和 **nlme** 和 **lme4** 包的几乎一样。

```
rats_blme <- blme::blmer(
  weight ~ days + (days | rats), data = rats_data,
  resid.prior = invgamma, cov.prior = NULL
)
summary(rats_blme)
```

```
#> Resid prior: invgamma(shape = 0, scale = 0, posterior.scale = var)
#> Prior dev  : 7.1328
#>
#> Linear mixed model fit by REML ['blmerMod']
#> Formula: weight ~ days + (days | rats)
#>    Data: rats_data
```

©王泽宇

```
#>
#> REML criterion at convergence: 1095.4
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -2.6697 -0.5440  0.1202  0.4968  2.6317
#>
#> Random effects:
#>  Groups   Name        Variance Std.Dev. Corr
#>  rats     (Intercept) 116.3517 10.7866
#>           days          0.2623  0.5121  -0.16
#>  Residual              35.3891  5.9489
#> Number of obs: 150, groups:  rats, 30
#>
#> Fixed effects:
#>             Estimate Std. Error t value
#> (Intercept) 106.5676     2.2977   46.38
#> days          6.1857     0.1056   58.58
#>
#> Correlation of Fixed Effects:
#>      (Intr)
#> days -0.343
```

与 **lme4** 包的函数 `lmer()` 所不同的是参数 `resid.prior` 、`fixef.prior` 和 `cov.prior` ，它们设置参数的先验分布，其它参数的含义同 `lme4` 包。`resid.prior = invgamma` 表示残差方差参数使用逆伽马分布，`cov.prior = NULL` 表示随机效应的协方差参数使用扁平先验 flat priors。

35.5.6 rjags

rjags (Plummer 2021) 是 JAGS 软件的 R 语言接口，可以拟合分层正态模型，再借助 **coda 包** (Plummer 等 2006) 可以分析 JAGS 返回的各项数据。

JAGS 代码和 Stan 代码有不少相似之处，最大的共同点在于以直观的统计模型的符号表示编码模型，仿照 Stan 代码，JAGS 编码的模型（BUGS 代码）如下：

```
model {
  alpha_c ~ dnorm(0, 1.0E-4);
  beta_c ~ dnorm(0, 1.0E-4);

  tau_c ~ dgamma(0.001, 0.001);
  tau_alpha ~ dgamma(0.001, 0.001);
```

©黄湘云

```
  tau_beta ~ dgamma(0.001, 0.001);

  sigma_c <- 1.0 / sqrt(tau_c);
  sigma_alpha <- 1.0 / sqrt(tau_alpha);
  sigma_beta <- 1.0 / sqrt(tau_beta);

  for (n in 1:N){
      alpha[n] ~ dnorm(alpha_c, tau_alpha);
      beta[n] ~ dnorm(beta_c, tau_beta);
    for (t in 1:T) {
      y[n,t] ~ dnorm(alpha[n] + beta[n] * (x[t] - xbar), tau_c);
    }
  }
}
```

转化主要集中在模型块，注意二者概率分布的名称以及参数含义对应关系，JAGS 使用 precision 而不是 standard deviation or variance，比如正态分布中的方差（标准偏差）被替换为其倒数。JAGS 可以省略类型声明（初始化模型时会补上），最后，JAGS 不支持 Stan 中的向量化操作，这种新特性是独特的。

```r
library(rjags)
# 初始值
rats_inits <- list(
  list(".RNG.name" = "base::Marsaglia-Multicarry",
       ".RNG.seed" = 20222022,
       "alpha_c" = 100, "beta_c" = 6, "tau_c" = 5, "tau_alpha" = 10, "tau_beta" = 0.5),
  list(".RNG.name" = "base::Marsaglia-Multicarry",
       ".RNG.seed" = 20232023,
       "alpha_c" = 200, "beta_c" = 10, "tau_c" = 15, "tau_alpha" = 15, "tau_beta" = 1)
)
# 模型
rats_model <- jags.model(
  file = "code/rats.bugs",
  data = list(x = x, y = y, N = 30, T = 5, xbar = 22.0),
  inits = rats_inits,
  n.chains = 2, quiet = TRUE
)
# burn-in
update(rats_model, n.iter = 2000)
# 抽样
rats_samples <- coda.samples(rats_model,
```

贝叶斯统计学 ©

```
  variable.names = c("alpha_c", "beta_c", "sigma_alpha", "sigma_beta", "sigma_c"),
  n.iter = 4000, thin = 1
)
# 参数的后验估计
summary(rats_samples)
```

```
#>
#> Iterations = 2001:6000
#> Thinning interval = 1
#> Number of chains = 2
#> Sample size per chain = 4000
#>
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#>
#>                 Mean      SD Naive SE Time-series SE
#> alpha_c     242.4752 2.72749 0.030494       0.031571
#> beta_c        6.1878 0.10798 0.001207       0.001481
#> sigma_alpha  14.6233 2.05688 0.022997       0.025070
#> sigma_beta    0.5176 0.09266 0.001036       0.001741
#> sigma_c       6.0731 0.46425 0.005191       0.007984
#>
#> 2. Quantiles for each variable:
#>
#>                 2.5%      25%      50%      75%    97.5%
#> alpha_c     237.0333 240.6832 242.5024 244.2965 247.7816
#> beta_c        5.9785   6.1150   6.1867   6.2593   6.4035
#> sigma_alpha  11.1840  13.1802  14.4152  15.8340  19.2429
#> sigma_beta    0.3571   0.4538   0.5098   0.5734   0.7187
#> sigma_c       5.2384   5.7479   6.0455   6.3803   7.0413
```

输出结果与 rstan 十分一致，且采样速度极快。类似地，`alpha0 = alpha_c - xbar * beta_c` 可得 alpha0 = 242.4752 - 22 * 6.1878 = 106.3436。

35.5.7 MCMCglmm

同前，先考虑变截距的混合效应模型，**MCMCglmm** 包 (Hadfield 2010) 给出的拟合结果与 **nlme** 包很接近。

```
## 变截距模型
prior1 <- list(
```

©南京大学出版社

```
  R = list(V = 1, nu = 0.002),
  G = list(G1 = list(V = 1, nu = 0.002))
)
set.seed(20232023)
rats_mcmc1 <- MCMCglmm::MCMCglmm(
  weight ~ days, random = ~ rats,
  data = rats_data, verbose = FALSE, prior = prior1
)
summary(rats_mcmc1)
```

```
#>
#>  Iterations = 3001:12991
#>  Thinning interval  = 10
#>  Sample size  = 1000
#>
#>  DIC: 1088.71
#>
#>  G-structure:  ~rats
#>
#>      post.mean l-95% CI u-95% CI eff.samp
#> rats       213    108.4    336.4     1000
#>
#>  R-structure:  ~units
#>
#>       post.mean l-95% CI u-95% CI eff.samp
#> units     68.58    50.63    86.58     1000
#>
#>  Location effects: weight ~ days
#>
#>             post.mean l-95% CI u-95% CI eff.samp  pMCMC
#> (Intercept)   106.568  100.464  112.897     1000 <0.001 ***
#> days            6.185    6.051    6.315     1000 <0.001 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

随机效应的方差（组间方差）为 211.4 ，则标准差为 14.539。残差方差（组内方差）为 68.77，则标准差为 8.293。

再考虑变截距和斜率的混合效应模型。

```
## 变截距、变斜率模型
prior2 <- list(
```

```
  R = list(V = 1, nu = 0.002),
  G = list(G1 = list(V = diag(2), nu = 0.002))
)
set.seed(20232023)
rats_mcmc2 <- MCMCglmm::MCMCglmm(weight ~ days,
  random = ~ us(1 + days):rats,
  data = rats_data, verbose = FALSE, prior = prior2
)
summary(rats_mcmc2)
```

```
#>
#>  Iterations = 3001:12991
#>  Thinning interval  = 10
#>  Sample size  = 1000
#>
#>  DIC: 1018.746
#>
#>  G-structure:  ~us(1 + days):rats
#>
#>                              post.mean l-95% CI u-95% CI eff.samp
#> (Intercept):(Intercept).rats  124.1327  41.5313  226.059    847.2
#> days:(Intercept).rats          -0.7457  -4.3090    2.571    896.6
#> (Intercept):days.rats          -0.7457  -4.3090    2.571    896.6
#> days:days.rats                  0.2783   0.1067    0.493    786.9
#>
#>  R-structure:  ~units
#>
#>       post.mean l-95% CI u-95% CI eff.samp
#> units     38.14    27.07    51.08     1000
#>
#>  Location effects: weight ~ days
#>
#>             post.mean l-95% CI u-95% CI eff.samp  pMCMC
#> (Intercept)    106.40   101.70   110.78    823.3 <0.001 ***
#> days             6.19     5.99     6.41    963.4 <0.001 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

G-structure 代表随机效应部分，R-structure 代表残差效应部分，Location effects 代表固定效应部分。**MCMCglmm** 包的这套模型表示术语源自商业软件 ASReml 。

©黄湘云

随机截距的方差为 124.1327，标准差为 11.1415，随机斜率的方差 0.2783，标准差为 0.5275，随机截距和随机斜率的协方差 -0.7457，相关系数为 -0.1268，这与 **nlme** 包结果很接近。

35.5.8 INLA

同前，先考虑变截距的混合效应模型。

```
library(INLA)
inla.setOption(short.summary = TRUE)
# 数值稳定性考虑
rats_data$weight <- rats_data$weight / 400
# 变截距
rats_inla1 <- inla(weight ~ days + f(rats, model = "iid", n = 30),
                   family = "gaussian", data = rats_data)
# 输出结果
summary(rats_inla1)
```

```
#> Fixed effects:
#>              mean    sd 0.025quant 0.5quant 0.975quant  mode kld
#> (Intercept) 0.266 0.008      0.252    0.266      0.281 0.266   0
#> days        0.015 0.000      0.015    0.015      0.016 0.015   0
#>
#> Model hyperparameters:
#>                                          mean     sd 0.025quant 0.5quant
#> Precision for the Gaussian observations 2414.80 311.28    1852.68  2397.51
#> Precision for rats                       888.43 244.91     495.43   858.84
#>                                         0.975quant    mode
#> Precision for the Gaussian observations    3076.02 2368.15
#> Precision for rats                         1451.76  804.50
#>
#>  is computed
```

再考虑变截距和斜率的混合效应模型。

```
# https://inla.r-inla-download.org/r-inla.org/doc/latent/iid.pdf
# 二维高斯随机效应的先验为 Wishart prior
rats_data$rats <- as.integer(rats_data$rats)
rats_data$slopeid <- 30 + rats_data$rats
# 变截距、变斜率
rats_inla2 <- inla(
  weight ~ 1 + days + f(rats, model = "iid2d", n = 2 * 30) + f(slopeid, days, copy = "rats"),
  data = rats_data, family = "gaussian"
```

```
)
# 输出结果
summary(rats_inla2)

#> Fixed effects:
#>              mean    sd 0.025quant 0.5quant 0.975quant  mode kld
#> (Intercept) 0.266 0.034       0.20    0.266      0.333 0.266   0
#> days        0.015 0.033      -0.05    0.015      0.080 0.015   0
#>
#> Model hyperparameters:
#>                                          mean      sd 0.025quant 0.5quant
#> Precision for the Gaussian observations 4522.637 666.555   3334.894 4480.315
#> Precision for rats (component 1)          32.097   7.955     18.938   31.260
#> Precision for rats (component 2)          33.234   8.227     19.603   32.377
#> Rho1:2 for rats                           -0.001   0.172     -0.335   -0.001
#>                                         0.975quant     mode
#> Precision for the Gaussian observations   5953.012 4407.810
#> Precision for rats (component 1)            50.048   29.770
#> Precision for rats (component 2)            51.773   30.859
#> Rho1:2 for rats                              0.334   -0.001
#>
#>  is computed
```

警告

对于变截距和斜率混合效应模型，还未完全弄清楚 INLA 包的输出结果。固定效应部分和残差部分都是和前面一致的，但不清楚随机效应的方差协方差矩阵的估计与 INLA 输出的对应关系。参考《Bayesian inference with INLA》第 3 章第 3 小节。

35.6 总结

基于 rats 数据建立变截距、变斜率的分层正态模型，也是线性混合效应模型的一种特殊情况，下表给出不同方法对模型各个参数的估计及置信区间。

表格 35.2: 频率派方法比较

	β_0	β_1	σ_0	σ_1	ρ_σ	σ_ϵ
nlme (REML)	106.568	6.186	10.743	0.511	-0.159	6.015
lme4 (REML)	106.568	6.186	10.744	0.511	-0.16	6.015

© 王建平

	β_0	β_1	σ_0	σ_1	ρ_σ	σ_ϵ
glmmTMB (REML)	106.568	6.186	10.743	0.511	-0.16	6.015
MASS (PQL)	106.568	6.186	10.495	0.500	-0.15	6.015
spaMM (ML)	106.568	6.186	10.49	0.499	-0.15	6.015
hglm	106.568	6.186	10.171	0.491	-	6.091
mgcv (REML)	106.568	6.186	10.311	0.492	-	6.069

表中给出截距 β_0 、斜率 β_1 、随机截距 σ_0、随机斜率 σ_1、随机截距和斜率的相关系数 ρ_σ、残差 σ_ϵ 等参数的估计及 95% 的置信区间，四舍五入保留 3 位小数。固定效应部分的结果完全相同，随机效应部分略有不同。

表格 35.3: 贝叶斯方法比较

	β_0	β_1	σ_0	σ_1	ρ_σ	σ_ϵ
rstan (NUTS)	106.4	6.2	14.6	0.5	-	6.1
cmdstanr (NUTS)	106	6.19	14.5	0.513	-	6.09
brms (NUTS)	106.47	6.18	11.27	0.54	-0.11	6.15
rstanarm (NUTS)	106.575	6.187	10.194	0.551	-0.0969	6.219
blme (REML)	106.568	6.186	10.787	0.512	-0.160	5.949
rjags (Gibbs)	106.344	6.188	14.623	0.518	-	6.073
MCMCglmm (MCMC)	106.40	6.19	11.14	0.53	-0.13	6.18

其中，**INLA** 结果的转化未完成，表格中暂缺。**rstan** 、**cmdstanr** 和 **rjags** 未考虑随机截距和随机斜率的相关性，因此，相关系数暂缺。MCMC 是一种随机优化算法，在不同的实现中，可重复性的要求不同，设置随机数种子仅是其中的一个必要条件，故而，每次运行程序结果可能略微不同，但不影响结论。Stan 相关的 R 包输出结果中，**rstan** 保留 1 位小数，**cmdstanr** 保留 3 位有效数字，**brms** 保留 2 位小数，**rstanarm** 小数点后保留 3 位有效数字，各不相同，暂未统一处理。

35.7 习题

1. 四个组的重复测量数据，如下表所示，建立贝叶斯线性混合效应模型/分层正态模型分析数据，与 nlme 包拟合的结果对比。

©黄湘云

表格 35.4: 实验数据

编号	第 1 组	第 2 组	第 3 组	第 4 组
1	62	63	68	56
2	60	67	66	62
3	63	71	71	60
4	59	64	67	61
5		65	68	63
6		66	68	64
7				63
8				59

$$
\begin{aligned}
y_{ij} \sim \mathcal{N}(\theta_i, \sigma^2) \quad \theta_i \sim \mathcal{N}(\mu, \tau^2) \\
(\mu, \log\sigma, \tau) \sim \text{uniform prior} \\
i = 1, 2, 3, 4 \quad j = 1, 2, \ldots, n_i
\end{aligned}
$$

y_{ij} 表示第 i 组的第 j 个测量值，θ_i 表示第 i 组的均值，μ 表示整体的均值，σ^2 表示组内的方差，τ^2 表示组内的方差。

```
library(nlme)
fit_lme <- lme(data = dat, fixed = y ~ 1, random = ~ 1 | group)
summary(fit_lme)
```

```
#> Linear mixed-effects model fit by REML
#>   Data: dat
#>        AIC      BIC    logLik
#>   121.7804 125.1869 -57.89019
#>
#> Random effects:
#>  Formula: ~1 | group
#>         (Intercept) Residual
#> StdDev:    3.419288 2.366309
#>
#> Fixed effects:  y ~ 1
#>                Value Std.Error DF  t-value p-value
#> (Intercept) 64.01266  1.780313 20 35.95584       0
#>
#> Standardized Within-Group Residuals:
#>         Min          Q1         Med          Q3         Max
#> -2.18490896 -0.59921167  0.09332131  0.54077636  2.17507789
```

©黄湘云

```
#>
#> Number of Observations: 24
#> Number of Groups: 4
```

随机效应（组间标准差）τ^2 3.419288 、残差效应（组内标准差）σ^2 2.366309。截距 μ 64.01266 代表整体的均值。各组的均值如下：

```
64.01266 + ranef(fit_lme)
```

```
#>   (Intercept)
#> 1    61.32214
#> 2    65.85309
#> 3    67.70525
#> 4    61.17016
```

也可以调用 **rjags** 包连接 JAGS 软件做贝叶斯推理，JAGS 代码如下：

```
model {
  ## specify the distribution for observations
  for(i in 1:n){
    y[i] ~ dnorm(theta[group[i]], 1/sigma2)
  }

  ## specify the prior for theta
  for(j in 1:J){
    theta[j] ~ dnorm(mu, 1/tau2)
  }

  ## specify the prior for hyperparameters
  mu ~ dunif(55, 75)

  log_sigma ~ dunif(-10, 3)
  sigma2 <- exp(2*log_sigma)
  sigma <- exp(log_sigma)

  tau ~ dunif(0, 8)
  tau2 <- pow(tau, 2)
}
```

完整的运行代码如下：

```
library(rjags)
# 参考值
mu_a <- min(y)
```

©贾俊平

```
mu_b <- max(y)
log_sigma_b <- 2 * log(sd(y))
tau_b <- 2 * sd(y)

J <- 4            # 4 个组
n <- length(y)    # 观察值数量
N <- 1500         # 总采样数
nthin <- 1        # 采样间隔
nchains <- 2      # 2 条链
ndiscard <- N / 2 # 预处理阶段 warm-up / burn-in

# 初始值
jags_inits <- list(
  list(".RNG.name" = "base::Marsaglia-Multicarry",
       ".RNG.seed" = 20222022,
       "theta" = rep(3, 4), "mu" = 60, "log_sigma" = 0, "tau" = 1.5),
  list(".RNG.name" = "base::Marsaglia-Multicarry",
       ".RNG.seed" = 20232023,
       "theta" = rep(2, 4), "mu" = 60, "log_sigma" = 1, "tau" = 0.375)
)
# Call JAGS from R
jags_model <- jags.model(
  file = "code/hnm.bugs",
  data = list("y" = y, "group" = group, "J" = J, "n" = n),
  inits = jags_inits, n.chains = nchains, quiet = TRUE
)
# burn-in
update(jags_model, n.iter = ndiscard)
# 抽样
jags_samples <- coda.samples(jags_model,
  variable.names = c('theta','mu','sigma','tau'), n.iter = N
)
# 参数的后验估计
summary(jags_samples)

#>
#> Iterations = 1751:3250
#> Thinning interval = 1
#> Number of chains = 2
#> Sample size per chain = 1500
```

© 统计出版社


```
#>
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#>
#>            Mean     SD Naive SE Time-series SE
#> mu       64.142 2.3470  0.04285        0.05879
#> sigma     2.473 0.4278  0.00781        0.01117
#> tau       4.372 1.5995  0.02920        0.05101
#> theta[1] 61.356 1.2301  0.02246        0.02503
#> theta[2] 65.877 1.0056  0.01836        0.01928
#> theta[3] 67.696 1.0247  0.01871        0.02119
#> theta[4] 61.186 0.8694  0.01587        0.01692
#>
#> 2. Quantiles for each variable:
#>
#>            2.5%    25%    50%    75%  97.5%
#> mu       59.145 62.700 64.167 65.510 69.027
#> sigma     1.795  2.165  2.424  2.718  3.471
#> tau       1.846  3.128  4.171  5.464  7.652
#> theta[1] 58.947 60.545 61.342 62.161 63.771
#> theta[2] 63.866 65.228 65.878 66.548 67.872
#> theta[3] 65.665 67.046 67.712 68.337 69.692
#> theta[4] 59.446 60.632 61.189 61.707 62.975
```

2. 基于 **lme4** 包中学生对老师的评价数据 `InstEval` 建立（广义）线性混合效应模型分析数据。将响应变量（学生评价）视为有序的离散型变量，比较观察两个模型拟合效果（lme4、GLMMadaptive、spaMM 都不支持有序的响应变量，brms 则支持各类有序回归，使用语法与 lme4 完全一样。但是，由于数据规模比较大，计算时间数以天计，可考虑用 Stan 直接编码）。再者，从 Stan 实现的贝叶斯模型来看，感受 Stan 建模的灵活性和扩展性。（**nlme** 包不支持此等交叉随机效应的表达。）

```r
data(InstEval, package = "lme4")
str(InstEval)
```

```
#> 'data.frame':    73421 obs. of  7 variables:
#>  $ s      : Factor w/ 2972 levels "1","2","3","4",..: 1 1 1 1 2 2 3 3 3 3 ...
#>  $ d    : Factor w/ 1128 levels "1","6","7","8",..: 525 560 832 1068 62 406 3 6 19 75 ...
#>  $ studage: Ord.factor w/ 4 levels "2"<"4"<"6"<"8": 1 1 1 1 1 1 1 1 1 1 ...
#>  $ lectage: Ord.factor w/ 6 levels "1"<"2"<"3"<"4"<..: 2 1 2 2 1 1 1 1 1 1 ...
#>  $ service: Factor w/ 2 levels "0","1": 1 2 1 2 1 1 2 1 1 1 ...
#>  $ dept   : Factor w/ 14 levels "15","5","10",..: 14 5 14 12 2 2 13 3 3 3 ...
#>  $ y      : int  5 2 5 3 2 4 4 5 5 4 ...
```

©黄湘云

- 因子型变量 s 表示 1-2972 位参与评分的学生。
- 因子型变量 d 表示 1-2160 位上课的讲师。
- 因子型变量 dept 表示课程相关的 1-15 院系。
- 因子型变量 service 表示讲师除了授课外，是否承担其它服务。
- 数值型变量 y 表示学生给课程的评分，1-5 分对应从坏到很好。

```
# 数值型的响应变量
fit_lme4 <- lme4::lmer(y ~ 1 + service + (1 | s) + (1 | d) + (1 | dept), data = InstEval)
summary(fit_lme4)
```

```
#> Linear mixed model fit by REML ['lmerMod']
#> Formula: y ~ 1 + service + (1 | s) + (1 | d) + (1 | dept)
#>    Data: InstEval
#>
#> REML criterion at convergence: 237733.8
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -3.0597 -0.7478  0.0404  0.7723  3.1988
#>
#> Random effects:
#>  Groups   Name        Variance Std.Dev.
#>  s        (Intercept) 0.105998 0.32557
#>  d        (Intercept) 0.265219 0.51499
#>  dept     (Intercept) 0.006912 0.08314
#>  Residual             1.386501 1.17750
#> Number of obs: 73421, groups:  s, 2972; d, 1128; dept, 14
#>
#> Fixed effects:
#>             Estimate Std. Error t value
#> (Intercept)  3.28259    0.02935 111.860
#> service1    -0.09264    0.01339  -6.919
#>
#> Correlation of Fixed Effects:
#>          (Intr)
#> service1 -0.152
```

lme4 包不支持响应变量为有序分类变量的情形，可用 **ordinal** 包，此等规模数据，拟合模型需要 5-10 分钟时间。

```
# 有序因子型的响应变量
InstEval$y <- factor(InstEval$y, ordered = TRUE)
```

```r
library(ordinal)
fit_ordinal <- clmm(
  y ~ 1 + service + (1 | s) + (1 | d) + (1 | dept),
  data = InstEval, link = "probit", threshold = "equidistant"
)
summary(fit_ordinal)

## MCMCglmm
library(MCMCglmm)
prior2 <- list(
  R = list(V = 1, nu = 0.002),
  G = list(
    G1 = list(V = 1, nu = 0.002),
    G2 = list(V = 1, nu = 0.002),
    G3 = list(V = 1, nu = 0.002)
  )
)
# 响应变量视为数值变量
fit_mcmc2 <- MCMCglmm(
  y ~ service, random = ~ s + d + dept, family = "gaussian",
  data = InstEval, verbose = FALSE, prior = prior2
)
# 响应变量视为有序的分类变量
fit_mcmc3 <- MCMCglmm(
  y ~ service, random = ~ s + d + dept, family = "ordinal",
  data = InstEval, verbose = FALSE, prior = prior2
)
```

当数据量较大时，**MCMCglmm** 包拟合模型需要很长时间，放弃，此时，Stan 的相对优势可以体现出来了。Stan 适合大型复杂概率统计模型。

© 基础统计

第三十六章　混合效应模型

> I think that the formula language does allow expressions with '/' to represent nested factors but I can't check right now as there is a fire in the building where my office is located. I prefer to simply write nested factors as `factor1 + factor1:factor2`.
>
> — Douglas Bates [1]

```
library(nlme)         # 线性混合效应模型
library(GLMMadaptive) # 广义线性混合效应模型
library(mgcv)         # 广义线性/可加混合效应模型
library(INLA)
library(splines)   # 样条
library(cmdstanr)  # 编译采样
library(ggplot2)   # 作图
library(bayesplot) # 后验分布
library(loo)       # LOO-CV
```

混合效应模型在心理学、生态学、计量经济学和空间统计学等领域应用十分广泛。线性混合效应模型有多个化身，比如生态学里的分层线性模型（Hierarchical linear Model，简称 HLM），心理学的多水平线性模型（Multilevel Linear Model）。模型名称的多样性正说明它应用的广泛性！混合效应模型内容非常多，非常复杂，因此，本章仅对常见的四种类型提供入门级的实战内容。从频率派和贝叶斯派两个角度介绍模型结构及说明、R 代码或 Stan 代码实现及输出结果解释。

除了 R 语言社区提供的开源扩展包，商业的软件有 Mplus 、ASReml 和 SAS 等，而开源的软件 OpenBUGS 、JAGS 和 Stan 等。混合效应模型的种类非常多，一部分可以在一些流行的 R 包能力范围内解决，其余可以放在更加灵活、扩展性强的框架 Stan 内解决。因此，本章同时介绍 Stan 框架和一些 R 包。

本章用到 4 个数据集，其中 `sleepstudy` 和 `cbpp` 均来自 **lme4** 包 (D. Bates 等 2015)，分别用于介绍线性混合效应模型和广义线性混合效应模型，`Loblolly` 来自 **datasets** 包，用于介绍非线性混合效应模型。

[1] https://stat.ethz.ch/pipermail/r-sig-mixed-models/2013q1/019945.html

© 草稿

在介绍理论的同时给出 R 语言或 S 语言实现的几本参考书籍。

- 《Mixed-Effects Models in S and S-PLUS》(Pinheiro 和 Bates 2000)
- 《Mixed Models: Theory and Applications with R》(Demidenko 2013)
- 《Linear Mixed-Effects Models Using R: A Step-by-Step Approach》(Gałecki 和 Burzykowski 2013)
- 《Linear and Generalized Linear Mixed Models and Their Applications》(Jiang 和 Nguyen 2021)

36.1 线性混合效应模型

> I think what we are seeking is the marginal variance-covariance matrix of the parameter estimators (marginal with respect to the random effects random variable, B), which would have the form of the inverse of the crossproduct of a $(q+p)$ by p matrix composed of the vertical concatenation of $-L^{-1}RZXRX^{-1}$ and RX^{-1}. (Note: You do *not* want to calculate the first term by inverting L, use `solve(L, RZX, system = "L")`
>
> - [...] don't even think about using `solve(L)`
> - don't!, don't!, don't!
> - have I made myself clear?
> - don't do that (and we all know that someone will do exactly that for a very large L and then send out messages about "R is SOOOOO SLOOOOW!!!!" :-))
>
> — Douglas Bates [2]

提示

1. 一般的模型结构和假设
2. 一般的模型表达公式
3. **nlme** 包的函数 `lme()`
4. 公式语法和示例模型表示

线性混合效应模型（Linear Mixed Models or Linear Mixed-Effects Models，简称 LME 或 LMM），介绍模型的基础理论，包括一般形式，矩阵表示，参数估计，假设检验，模型诊断，模型评估。参数方法主要是极大似然估计和限制极大似然估计。一般形式如下：

$$\boldsymbol{y} = X\boldsymbol{\beta} + Z\boldsymbol{u} + \boldsymbol{\epsilon}$$

其中，$\boldsymbol{y}$ 是一个向量，代表响应变量，X 代表固定效应对应的设计矩阵，$\boldsymbol{\beta}$ 是一个参数向量，代表固定效应对应的回归系数，Z 代表随机效应对应的设计矩阵，$\boldsymbol{u}$ 是一个参数向量，代表随机效应对应的回归系数，$\boldsymbol{\epsilon}$ 表示残差向量。

[2] https://stat.ethz.ch/pipermail/r-sig-mixed-models/2010q2/003615.html

©黄湘云

一般假定随机向量 $\boldsymbol{u}$ 服从多元正态分布，这是无条件分布，随机向量 $\boldsymbol{y}|\boldsymbol{u}$ 服从多元正态分布，这是条件分布。

$$\boldsymbol{u} \sim \mathcal{N}(0, \Sigma)$$
$$\boldsymbol{y}|\boldsymbol{u} \sim \mathcal{N}(X\boldsymbol{\beta} + Z\boldsymbol{u}, \sigma^2 W)$$

其中，方差协方差矩阵 Σ 必须是半正定的，W 是一个对角矩阵。nlme 和 lme4 等 R 包共用一套表示随机效应的公式语法。

sleepstudy 数据集来自 lme4 包，是一个睡眠研究项目的实验数据。实验对象都是有失眠情况的人，有的人有严重的失眠问题（一天只有 3 个小时的睡眠时间）。进入实验后的前 10 天的情况，记录平均反应时间、睡眠不足的天数。

```
data(sleepstudy, package = "lme4")
str(sleepstudy)
```

```
#> 'data.frame':    180 obs. of  3 variables:
#>  $ Reaction: num  250 259 251 321 357 ...
#>  $ Days    : num  0 1 2 3 4 5 6 7 8 9 ...
#>  $ Subject : Factor w/ 18 levels "308","309","310",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Reaction 表示平均反应时间（毫秒），数值型，Days 表示进入实验后的第几天，数值型，Subject 表示参与实验的个体编号，因子型。

```
xtabs(~ Days + Subject, data = sleepstudy)
```

```
#>     Subject
#> Days 308 309 310 330 331 332 333 334 335 337 349 350 351 352 369 370 371 372
#>    0   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
#>    1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
#>    2   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
#>    3   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
#>    4   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
#>    5   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
#>    6   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
#>    7   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
#>    8   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
#>    9   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
```

每个个体每天产生一条数据，下图 36.1 中每条折线代表一个个体。

```
library(ggplot2)
ggplot(data = sleepstudy, aes(x = Days, y = Reaction, group = Subject)) +
  geom_line() +
  scale_x_continuous(n.breaks = 6) +
```

© 王湘君


```
  theme_bw() +
  labs(x = "睡眠不足的天数", y = "平均反应时间")
```

图 36.1: sleepstudy 数据集

对于连续重复测量的数据（continuous repeated measurement outcomes），也叫纵向数据（longitudinal data），针对不同个体 Subject，相比于上图，下面绘制反应时间 Reaction 随睡眠时间 Days 的变化趋势更合适。图中趋势线是简单线性回归的结果，分面展示不同个体 Subject 之间对比。

```
ggplot(data = sleepstudy, aes(x = Days, y = Reaction)) +
  geom_point() +
  geom_smooth(formula = "y ~ x", method = "lm", se = FALSE) +
  scale_x_continuous(n.breaks = 6) +
  theme_bw() +
  facet_wrap(facets = ~Subject, labeller = "label_both", ncol = 6) +
  labs(x = "睡眠不足的天数", y = "平均反应时间")
```

© 黄湘云

图 36.2: 分面展示 sleepstudy 数据集

36.1.1 nlme

考虑两水平的混合效应模型，其中随机截距 β_{0j} 和随机斜率 β_{1j}，指标 j 表示分组的编号，也叫变截距和变斜率模型

©李漫 汇

$$
\begin{aligned}
\text{Reaction}_{ij} &= \beta_{0j} + \beta_{1j} \cdot \text{Days}_{ij} + \epsilon_{ij} \\
\beta_{0j} &= \gamma_{00} + U_{0j} \\
\beta_{1j} &= \gamma_{10} + U_{1j} \\
\begin{pmatrix} U_{0j} \\ U_{1j} \end{pmatrix} &\sim \mathcal{N}\left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \tau_{00}^2 & \tau_{01} \\ \tau_{01} & \tau_{10}^2 \end{pmatrix}\right] \\
\epsilon_{ij} &\sim \mathcal{N}(0, \sigma^2) \\
i = 0, 1, \cdots, 9 \quad j &= 308, 309, \cdots, 372.
\end{aligned}
$$

下面用 nlme 包 (Pinheiro 和 Bates 2000) 拟合模型。

```
library(nlme)
sleep_nlme <- lme(Reaction ~ Days, random = ~ Days | Subject, data = sleepstudy)
summary(sleep_nlme)
```

```
#> Linear mixed-effects model fit by REML
#>   Data: sleepstudy
#>        AIC      BIC    logLik
#>   1755.628 1774.719 -871.8141
#>
#> Random effects:
#>  Formula: ~Days | Subject
#>  Structure: General positive-definite, Log-Cholesky parametrization
#>             StdDev    Corr
#> (Intercept) 24.740241 (Intr)
#> Days         5.922103 0.066
#> Residual    25.591843
#>
#> Fixed effects:  Reaction ~ Days
#>                 Value Std.Error  DF  t-value p-value
#> (Intercept) 251.40510  6.824516 161 36.83853       0
#> Days         10.46729  1.545783 161  6.77151       0
#>  Correlation:
#>      (Intr)
#> Days -0.138
#>
#> Standardized Within-Group Residuals:
#>         Min          Q1         Med          Q3         Max
#> -3.95355735 -0.46339976  0.02311783  0.46339621  5.17925089
#>
#> Number of Observations: 180
```

```
#> Number of Groups: 18
```

随机效应（Random effects）部分：

```
# 前 6 个 subject
head(ranef(sleep_nlme))
```

```
#>     (Intercept)       Days
#> 308    2.258754  9.1989366
#> 309  -40.398490 -8.6197167
#> 310  -38.960098 -5.4489048
#> 330   23.690228 -4.8142826
#> 331   22.259981 -3.0698548
#> 332    9.039458 -0.2721585
```

固定效应（Fixed effects）部分：

```
fixef(sleep_nlme)
```

```
#> (Intercept)        Days
#>   251.40510    10.46729
```

ggeffects 包的函数 `ggpredict()` 和 `ggeffect()` 可以用来绘制混合效应模型的边际效应（Marginal Effects），**ggPMX** 包可以用来绘制混合效应模型的诊断图。下图 36.3 展示关于变量 Days 的边际效应图。

```
library(ggeffects)
mydf <- ggpredict(sleep_nlme, terms = "Days")
ggplot(mydf, aes(x = x, y = predicted)) +
  geom_line() +
  geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = 0.2) +
  scale_x_continuous(n.breaks = 6) +
  theme_bw() +
  labs(x = "Days", y = "Reaction")
```

©黄湘云

图 36.3: 边际效应图

36.1.2 MASS

```
sleep_mass <- MASS::glmmPQL(Reaction ~ Days,
  random = ~ Days | Subject, verbose = FALSE,
  data = sleepstudy, family = gaussian
)
summary(sleep_mass)
```

```
#> Linear mixed-effects model fit by maximum likelihood
#>   Data: sleepstudy
#>   AIC BIC logLik
#>    NA  NA     NA
#>
#> Random effects:
#>  Formula: ~Days | Subject
#>  Structure: General positive-definite, Log-Cholesky parametrization
#>             StdDev    Corr
#> (Intercept) 23.780376 (Intr)
#> Days         5.716807 0.081
#> Residual    25.591842
```

©非常统计

```
#>
#> Variance function:
#>  Structure: fixed weights
#>  Formula: ~invwt
#> Fixed effects:  Reaction ~ Days
#>                 Value Std.Error  DF  t-value p-value
#> (Intercept) 251.40510  6.669396 161 37.69533       0
#> Days         10.46729  1.510647 161  6.92901       0
#>  Correlation:
#>      (Intr)
#> Days -0.138
#>
#> Standardized Within-Group Residuals:
#>         Min          Q1         Med          Q3         Max
#> -3.94156355 -0.46559311  0.02894656  0.46361051  5.17933587
#>
#> Number of Observations: 180
#> Number of Groups: 18
```

36.1.3 lme4

```
sleep_lme4 <- lme4::lmer(Reaction ~ Days + (Days | Subject), data = sleepstudy)
summary(sleep_lme4)
```

```
#> Linear mixed model fit by REML ['lmerMod']
#> Formula: Reaction ~ Days + (Days | Subject)
#>    Data: sleepstudy
#>
#> REML criterion at convergence: 1743.6
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -3.9536 -0.4634  0.0231  0.4634  5.1793
#>
#> Random effects:
#>  Groups   Name        Variance Std.Dev. Corr
#>  Subject  (Intercept) 612.10   24.741
#>           Days         35.07    5.922   0.07
#>  Residual             654.94   25.592
#> Number of obs: 180, groups:  Subject, 18
```

©清华大学出版社

```
#>
#> Fixed effects:
#>             Estimate Std. Error t value
#> (Intercept)  251.405      6.825  36.838
#> Days          10.467      1.546   6.771
#>
#> Correlation of Fixed Effects:
#>      (Intr)
#> Days -0.138
```

36.1.4 blme

```
sleep_blme <- blme::blmer(
  Reaction ~ Days + (Days | Subject), data = sleepstudy,
  control = lme4::lmerControl(check.conv.grad = "ignore"),
  cov.prior = NULL)
summary(sleep_blme)
```

```
#> Prior dev  : 0
#>
#> Linear mixed model fit by REML ['blmerMod']
#> Formula: Reaction ~ Days + (Days | Subject)
#>    Data: sleepstudy
#> Control: lme4::lmerControl(check.conv.grad = "ignore")
#>
#> REML criterion at convergence: 1743.6
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -3.9536 -0.4634  0.0231  0.4634  5.1793
#>
#> Random effects:
#>  Groups   Name        Variance Std.Dev. Corr
#>  Subject  (Intercept) 612.10   24.741
#>           Days         35.07    5.922   0.07
#>  Residual             654.94   25.592
#> Number of obs: 180, groups:  Subject, 18
#>
#> Fixed effects:
#>             Estimate Std. Error t value
```

©南京大学出版社

```
#> (Intercept)  251.405      6.825  36.838
#> Days          10.467      1.546   6.771
#>
#> Correlation of Fixed Effects:
#>      (Intr)
#> Days -0.138
```

36.1.5 brms

```
sleep_brms <- brms::brm(Reaction ~ Days + (Days | Subject), data = sleepstudy)
summary(sleep_brms)
```

```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: Reaction ~ Days + (Days | Subject)
   Data: sleepstudy (Number of observations: 180)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Group-Level Effects:
~Subject (Number of levels: 18)
                    Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)          27.03      6.60    15.88    42.13 1.00     1728     2469
sd(Days)                6.61      1.50     4.18     9.97 1.00     1517     2010
cor(Intercept,Days)     0.08      0.29    -0.46     0.65 1.00      991     1521

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept   251.26      7.42   236.27   266.12 1.00     1982     2687
Days         10.36      1.77     6.85    13.85 1.00     1415     1982

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma    25.88      1.54    22.99    29.06 1.00     3204     2869

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
# predictions
conds <- brms::make_conditions(sleep_brms, "Subject")
```

©清华大学出版社

```
sleep_brms |>
  brms::marginal_effects(
    re_formula = NULL,
    conditions = conds
  ) |>
  plot(points = TRUE, ncol = 6)
```

36.1.6 MCMCglmm

MCMCglmm 包拟合变截距、变斜率模型，随机截距和随机斜率之间存在相关性。

```
## 变截距、变斜率模型
prior1 <- list(
  R = list(V = 1, fix = 1),
  G = list(G1 = list(V = diag(2), nu = 0.002))
)
set.seed(20232023)
sleep_mcmcglmm <- MCMCglmm::MCMCglmm(
  Reaction ~ Days, random = ~ us(1 + Days):Subject, prior = prior1,
  data = sleepstudy, family = "gaussian", verbose = FALSE
)
summary(sleep_mcmcglmm)
```

```
#>
#>  Iterations = 3001:12991
#>  Thinning interval  = 10
#>  Sample size  = 1000
#>
#>  DIC: 94714.46
#>
#>  G-structure:  ~us(1 + Days):Subject
#>
#>                                 post.mean l-95% CI u-95% CI eff.samp
#> (Intercept):(Intercept).Subject   1005.69   454.97  1840.04   1000.0
#> Days:(Intercept).Subject           -34.44  -167.15    84.09   1000.0
#> (Intercept):Days.Subject           -34.44  -167.15    84.09   1000.0
#> Days:Days.Subject                   52.36    22.74    95.60    902.3
#>
#>  R-structure:  ~units
#>
#>       post.mean l-95% CI u-95% CI eff.samp
```

©李东风

```
#> units         1        1        1        0
#>
#>  Location effects: Reaction ~ Days
#>
#>             post.mean l-95% CI u-95% CI eff.samp  pMCMC
#> (Intercept)   251.374  235.935  265.961     1000 <0.001 ***
#> Days           10.419    7.262   13.976     1000 <0.001 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

固定随机效应 R-structure 方差。固定效应 Location effects 截距 (Intercept) 为 251.374，斜率 Days 为 10.419 。

36.1.7 INLA

将数据集 sleepstudy 中的 Reaction 除以 1000，目的是数值稳定性，减小迭代序列的相关性。先考虑变截距模型

```r
library(INLA)
inla.setOption(short.summary = TRUE)
# 做尺度变换
sleepstudy$Reaction <- sleepstudy$Reaction / 1000
# 变截距
sleep_inla1 <- inla(Reaction ~ Days + f(Subject, model = "iid", n = 18),
                    family = "gaussian", data = sleepstudy)
# 输出结果
summary(sleep_inla1)
```

```
#> Fixed effects:
#>              mean    sd 0.025quant 0.5quant 0.975quant  mode kld
#> (Intercept) 0.251 0.010      0.232    0.251      0.270 0.251   0
#> Days        0.010 0.001      0.009    0.010      0.012 0.010   0
#>
#> Model hyperparameters:
#>                                          mean     sd 0.025quant 0.5quant
#> Precision for the Gaussian observations 1054.07 116.99     840.14  1048.46
#> Precision for Subject                    843.97 300.88     391.25   798.36
#>                                         0.975quant    mode
#> Precision for the Gaussian observations    1300.13 1038.99
#> Precision for Subject                      1559.54  714.37
#>
```

```
#>  is computed
```

再考虑变截距和变斜率模型

```
# https://inla.r-inla-download.org/r-inla.org/doc/latent/iid.pdf
# 二维高斯随机效应的先验为 Wishart prior
sleepstudy$Subject <- as.integer(sleepstudy$Subject)
sleepstudy$slopeid <- 18 + sleepstudy$Subject
# 变截距、变斜率
sleep_inla2 <- inla(
  Reaction ~ 1 + Days + f(Subject, model = "iid2d", n = 2 * 18) + f(slopeid, Days, copy = "Subject'
  data = sleepstudy, family = "gaussian"
)
# 输出结果
summary(sleep_inla2)
```

```
#> Fixed effects:
#>              mean    sd 0.025quant 0.5quant 0.975quant  mode kld
#> (Intercept) 0.251 0.055      0.142    0.251      0.360 0.251   0
#> Days        0.010 0.054     -0.097    0.010      0.118 0.010   0
#>
#> Model hyperparameters:
#>                                              mean      sd 0.025quant 0.5quant
#> Precision for the Gaussian observations 1549.519 181.248   1218.044 1540.850
#> Precision for Subject (component 1)       20.871   6.507     10.626   20.024
#> Precision for Subject (component 2)       21.225   6.611     10.804   20.367
#> Rho1:2 for Subject                        -0.001   0.213     -0.414   -0.002
#>                                        0.975quant     mode
#> Precision for the Gaussian observations   1930.562 1527.272
#> Precision for Subject (component 1)         35.971   18.480
#> Precision for Subject (component 2)         36.556   18.806
#> Rho1:2 for Subject                           0.413   -0.003
#>
#>  is computed
```

36.2 广义线性混合效应模型

当响应变量分布不再是高斯分布，线性混合效应模型就扩展到广义线性混合效应模型。有一些 R 包可以拟合此类模型，MASS 包的函数 `glmmPQL()` ，**mgcv** 包的函数 `gam()`，lme4 包的函数 `glmer()` ，GLMMadaptive 包的函数 `mixed_model()` ，brms 包的函数 `brm()` 等。

© 基础统计

表格 36.1: 响应变量的分布

响应变量分布	MASS	mgcv	lme4	GLMMadaptive	brms
伯努利分布	支持	支持	支持	支持	支持
二项分布	支持	支持	支持	支持	支持
泊松分布	支持	支持	支持	支持	支持
负二项分布	不支持	支持	支持	支持	支持
伽马分布	支持	支持	支持	支持	支持

函数 `glmmPQL()` 支持的分布族见函数 `glm()` 的参数 `family` ，lme4 包的函数 `glmer.nb()` 和 GLMMadaptive 包的函数 `negative.binomial()` 都可用于拟合响应变量服从负二项分布的情况。除了这些常规的分布，GLMMadaptive 和 brms 包还支持许多常见的分布，比如零膨胀的泊松分布、二项分布等，还可以自定义分布。

- 伯努利分布 `family = binomial(link = "logit")`
- 二项分布 `family = binomial(link = "logit")`
- 泊松分布 `family = poisson(link = "log")`
- 负二项分布 `lme4::glmer.nb()` 或 `GLMMadaptive::negative.binomial()`
- 伽马分布 `family = Gamma(link = "inverse")`

GLMMadaptive 包 (Rizopoulos 2023) 的主要函数 `mixed_model()` 是用来拟合广义线性混合效应模型的。下面以牛传染性胸膜肺炎（Contagious bovine pleuropneumonia，简称 CBPP）数据 cbpp 介绍函数 `mixed_model()` 的用法，该数据集来自 lme4 包。

```
data(cbpp, package = "lme4")
str(cbpp)
```

```
#> 'data.frame':    56 obs. of  4 variables:
#>  $ herd     : Factor w/ 15 levels "1","2","3","4",..: 1 1 1 1 2 2 2 3 3 3 ...
#>  $ incidence: num  2 3 4 0 3 1 1 8 2 0 ...
#>  $ size     : num  14 12 9 5 22 18 21 22 16 16 ...
#>  $ period   : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 1 2 3 ...
```

herd 牛群编号，period 时间段，incidence 感染的数量，size 牛群大小。疾病在种群内扩散

```
ggplot(data = cbpp, aes(x = herd, y = period)) +
  geom_tile(aes(fill = incidence / size)) +
  scale_fill_viridis_c(label = scales::percent_format(),
                       option = "C", name = "") +
  theme_minimal()
```

©黄湘云

图 36.4: 感染比例随变量 herd 和 period 的变化

36.2.1 MASS

```
cbpp_mass <- MASS::glmmPQL(
  cbind(incidence, size - incidence) ~ period,
  random = ~ 1 | herd, verbose = FALSE,
  data = cbpp, family = binomial("logit")
)
summary(cbpp_mass)
```

```
#> Linear mixed-effects model fit by maximum likelihood
#>   Data: cbpp
#>   AIC BIC logLik
#>    NA  NA     NA
#>
#> Random effects:
#>  Formula: ~1 | herd
#>         (Intercept) Residual
#> StdDev:   0.5563535 1.184527
#>
#> Variance function:
#>  Structure: fixed weights
#>  Formula: ~invwt
#> Fixed effects:  cbind(incidence, size - incidence) ~ period
#>                 Value Std.Error DF   t-value p-value
#> (Intercept) -1.327364 0.2390194 38 -5.553372  0.0000
```

©李东风 统计学

```
#> period2     -1.016126 0.3684079 38 -2.758156  0.0089
#> period3     -1.149984 0.3937029 38 -2.920944  0.0058
#> period4     -1.605217 0.5178388 38 -3.099839  0.0036
#>  Correlation:
#>         (Intr) perid2 perid3
#> period2 -0.399
#> period3 -0.373  0.260
#> period4 -0.282  0.196  0.182
#>
#> Standardized Within-Group Residuals:
#>        Min         Q1        Med         Q3        Max
#> -2.0591168 -0.6493095 -0.2747620  0.5170492  2.6187632
#>
#> Number of Observations: 56
#> Number of Groups: 15
```

36.2.2 GLMMadaptive

```r
library(GLMMadaptive)
cbpp_glmmadaptive <- mixed_model(
  fixed = cbind(incidence, size - incidence) ~ period,
  random = ~ 1 | herd, data = cbpp, family = binomial(link = "logit")
)
summary(cbpp_glmmadaptive)
```

```
#>
#> Call:
#> mixed_model(fixed = cbind(incidence, size - incidence) ~ period,
#>     random = ~1 | herd, data = cbpp, family = binomial(link = "logit"))
#>
#> Data Descriptives:
#> Number of Observations: 56
#> Number of Groups: 15
#>
#> Model:
#>  family: binomial
#>  link: logit
#>
#> Fit statistics:
#>    log.Lik      AIC     BIC
```

©黄湘云

```
#>  -91.98337 193.9667 197.507
#>
#> Random effects covariance matrix:
#>                StdDev
#> (Intercept) 0.6475934
#>
#> Fixed effects:
#>             Estimate Std.Err z-value    p-value
#> (Intercept)  -1.3995  0.2335 -5.9923    < 1e-04
#> period2      -0.9914  0.3068 -3.2316 0.00123091
#> period3      -1.1278  0.3268 -3.4513 0.00055793
#> period4      -1.5795  0.4276 -3.6937 0.00022101
#>
#> Integration:
#> method: adaptive Gauss-Hermite quadrature rule
#> quadrature points: 11
#>
#> Optimization:
#> method: EM
#> converged: TRUE
```

36.2.3 glmmTMB

```
cbpp_glmmtmb <- glmmTMB::glmmTMB(
  cbind(incidence, size - incidence) ~ period + (1 | herd),
  data = cbpp, family = binomial, REML = TRUE
)
summary(cbpp_glmmtmb)
```

```
#>  Family: binomial  ( logit )
#> Formula:          cbind(incidence, size - incidence) ~ period + (1 | herd)
#> Data: cbpp
#>
#>      AIC      BIC   logLik deviance df.resid
#>    196.4    206.5    -93.2    186.4       55
#>
#> Random effects:
#>
#> Conditional model:
#>  Groups Name        Variance Std.Dev.
```

©南京大学出版社

```
#>  herd   (Intercept) 0.4649   0.6819
#> Number of obs: 56, groups:  herd, 15
#>
#> Conditional model:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  -1.3670     0.2376  -5.752 8.79e-09 ***
#> period2      -0.9693     0.3055  -3.173 0.001509 **
#> period3      -1.1045     0.3255  -3.393 0.000691 ***
#> period4      -1.5519     0.4265  -3.639 0.000274 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

36.2.4 lme4

```
cbpp_lme4 <- lme4::glmer(
  cbind(incidence, size - incidence) ~ period + (1 | herd),
  family = binomial("logit"), data = cbpp
)
summary(cbpp_lme4)
```

```
#> Generalized linear mixed model fit by maximum likelihood (Laplace
#>   Approximation) [glmerMod]
#>  Family: binomial  ( logit )
#> Formula: cbind(incidence, size - incidence) ~ period + (1 | herd)
#>    Data: cbpp
#>
#>      AIC      BIC   logLik deviance df.resid
#>    194.1    204.2    -92.0    184.1       51
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -2.3816 -0.7889 -0.2026  0.5142  2.8791
#>
#> Random effects:
#>  Groups Name        Variance Std.Dev.
#>  herd   (Intercept) 0.4123   0.6421
#> Number of obs: 56, groups:  herd, 15
#>
#> Fixed effects:
#>             Estimate Std. Error z value Pr(>|z|)
```

©黄湘云

```
#> (Intercept)  -1.3983     0.2312  -6.048 1.47e-09 ***
#> period2      -0.9919     0.3032  -3.272 0.001068 **
#> period3      -1.1282     0.3228  -3.495 0.000474 ***
#> period4      -1.5797     0.4220  -3.743 0.000182 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Correlation of Fixed Effects:
#>         (Intr) perid2 perid3
#> period2 -0.363
#> period3 -0.340  0.280
#> period4 -0.260  0.213  0.198
```

36.2.5 mgcv

或使用 **mgcv** 包，可以得到近似的结果。随机效应部分可以看作可加的惩罚项

```
library(mgcv)
cbpp_mgcv <- gam(
  cbind(incidence, size - incidence) ~ period + s(herd, bs = "re"),
  data = cbpp, family = binomial(link = "logit"), method = "REML"
)
summary(cbpp_mgcv)
```

```
#>
#> Family: binomial
#> Link function: logit
#>
#> Formula:
#> cbind(incidence, size - incidence) ~ period + s(herd, bs = "re")
#>
#> Parametric coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  -1.3670     0.2358  -5.799 6.69e-09 ***
#> period2      -0.9693     0.3040  -3.189 0.001428 **
#> period3      -1.1045     0.3241  -3.407 0.000656 ***
#> period4      -1.5519     0.4251  -3.651 0.000261 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
```

©南京大学出版社

```
#>          edf Ref.df Chi.sq  p-value
#> s(herd) 9.66     14  32.03 3.21e-05 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.515   Deviance explained =   53%
#> -REML = 93.199  Scale est. = 1         n = 56
```

下面给出随机效应的标准差的估计及其上下限，和前面 **GLMMadaptive** 包和 **lme4** 包给出的结果也是接近的。

```
gam.vcomp(cbpp_mgcv)
```

```
#>
#> Standard deviations and 0.95 confidence intervals:
#>
#>           std.dev     lower    upper
#> s(herd) 0.6818673 0.3953145 1.176135
#>
#> Rank: 1/1
```

36.2.6 blme

```
cbpp_blme <- blme::bglmer(
  cbind(incidence, size - incidence) ~ period + (1 | herd),
  family = binomial("logit"), data = cbpp
)
summary(cbpp_blme)
```

```
#> Cov prior  : herd ~ wishart(df = 3.5, scale = Inf, posterior.scale = cov, common.scale = TRUE)
#> Prior dev  : 0.9901
#>
#> Generalized linear mixed model fit by maximum likelihood (Laplace
#>   Approximation) [bglmerMod]
#>  Family: binomial  ( logit )
#> Formula: cbind(incidence, size - incidence) ~ period + (1 | herd)
#>    Data: cbpp
#>
#>      AIC      BIC   logLik deviance df.resid
#>    194.2    204.3    -92.1    184.2       51
#>
#> Scaled residuals:
```

©黄湘云

```
#>     Min      1Q  Median      3Q     Max
#> -2.3670 -0.8121 -0.1704  0.4971  2.7969
#>
#> Random effects:
#>  Groups Name        Variance Std.Dev.
#>  herd   (Intercept) 0.5168   0.7189
#> Number of obs: 56, groups:  herd, 15
#>
#> Fixed effects:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  -1.4142     0.2466  -5.736  9.7e-09 ***
#> period2      -0.9803     0.3037  -3.227 0.001249 **
#> period3      -1.1171     0.3233  -3.455 0.000549 ***
#> period4      -1.5667     0.4222  -3.710 0.000207 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Correlation of Fixed Effects:
#>         (Intr) perid2 perid3
#> period2 -0.342
#> period3 -0.320  0.282
#> period4 -0.246  0.215  0.199
```

36.2.7 brms

表示二项分布，公式语法与前面的 lme4 等包不同。

```r
cbpp_brms <- brms::brm(
  incidence | trials(size) ~ period + (1 | herd),
  family = binomial("logit"), data = cbpp
)
summary(cbpp_brms)
```

```
 Family: binomial
  Links: mu = logit
Formula: incidence | trials(size) ~ period + (1 | herd)
   Data: cbpp (Number of observations: 56)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Group-Level Effects:
```

北京师范大学出版社 ©

```
~herd (Number of levels: 15)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.76      0.22     0.39     1.29 1.00     1483     1962

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept    -1.40      0.26    -1.92    -0.88 1.00     2440     2542
period2      -1.00      0.31    -1.63    -0.41 1.00     5242     2603
period3      -1.14      0.34    -1.83    -0.50 1.00     4938     3481
period4      -1.61      0.44    -2.49    -0.81 1.00     4697     2966

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

36.2.8 MCMCglmm

```r
set.seed(20232023)
cbpp_mcmcglmm <- MCMCglmm::MCMCglmm(
  cbind(incidence, size - incidence) ~ period, random = ~herd,
  data = cbpp, family = "multinomial2", verbose = FALSE
)
summary(cbpp_mcmcglmm)
```

```
#>
#>  Iterations = 3001:12991
#>  Thinning interval  = 10
#>  Sample size  = 1000
#>
#>  DIC: 538.4141
#>
#>  G-structure:  ~herd
#>
#>      post.mean  l-95% CI u-95% CI eff.samp
#> herd    0.0244 1.256e-16   0.1347    186.5
#>
#>  R-structure:  ~units
#>
#>       post.mean l-95% CI u-95% CI eff.samp
#> units     1.098   0.2471    2.158    273.9
```

©黄湘云

```
#>
#>  Location effects: cbind(incidence, size - incidence) ~ period
#>
#>             post.mean l-95% CI u-95% CI eff.samp  pMCMC
#> (Intercept)   -1.5314  -2.1484  -0.8507   1000.0 <0.001 ***
#> period2       -1.2596  -2.2129  -0.1495    854.7  0.006 **
#> period3       -1.3827  -2.3979  -0.2851    691.0  0.012 *
#> period4       -1.9612  -3.3031  -0.7745    572.6 <0.001 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

对于服从非高斯分布的响应变量，MCMCglmm 总是假定存在过度离散的情况，即存在一个与分类变量无关的随机变量，或者说存在一个残差服从正态分布的随机变量（效应），可以看作测量误差，这种假定对真实数据建模是有意义的，所以，与以上 MCMCglmm 代码等价的 lme4 包模型代码如下：

```
cbpp$id <- as.factor(1:dim(cbpp)[1])
cbpp_lme4 <- lme4::glmer(
  cbind(incidence, size - incidence) ~ period + (1 | herd) + (1 | id),
  family = binomial, data = cbpp
)
summary(cbpp_lme4)
```

```
#> Generalized linear mixed model fit by maximum likelihood (Laplace
#>   Approximation) [glmerMod]
#>  Family: binomial  ( logit )
#> Formula: cbind(incidence, size - incidence) ~ period + (1 | herd) + (1 |
#>     id)
#>    Data: cbpp
#>
#>      AIC      BIC   logLik deviance df.resid
#>    186.6    198.8    -87.3    174.6       50
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -1.2866 -0.5989 -0.1181  0.3575  1.6216
#>
#> Random effects:
#>  Groups Name        Variance Std.Dev.
#>  id     (Intercept) 0.79400  0.8911
#>  herd   (Intercept) 0.03384  0.1840
#> Number of obs: 56, groups:  id, 56; herd, 15
```

©南京大学出版社

```
#>
#> Fixed effects:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  -1.5003     0.2967  -5.056 4.27e-07 ***
#> period2      -1.2265     0.4803  -2.554  0.01066 *
#> period3      -1.3288     0.4939  -2.690  0.00713 **
#> period4      -1.8662     0.5936  -3.144  0.00167 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Correlation of Fixed Effects:
#>         (Intr) perid2 perid3
#> period2 -0.559
#> period3 -0.537  0.373
#> period4 -0.441  0.327  0.314
```

贝叶斯的结果与频率派的结果相近，但还是有明显差异。MCMCglmm 总是假定存在残差，残差的分布服从 0 均值的高斯分布，下面将残差分布的方差固定，重新拟合模型，之后再根据残差方差为 0 调整估计结果。

```
prior2 <- list(
  R = list(V = 1, fix = 1),
  G = list(G1 = list(V = 1, nu = 0.002))
)
set.seed(20232023)
cbpp_mcmcglmm <- MCMCglmm::MCMCglmm(
  cbind(incidence, size - incidence) ~ period, random = ~herd, prior = prior2,
  data = cbpp, family = "multinomial2", verbose = FALSE
)
summary(cbpp_mcmcglmm)
```

```
#>
#>  Iterations = 3001:12991
#>  Thinning interval  = 10
#>  Sample size  = 1000
#>
#>  DIC: 536.3978
#>
#>  G-structure:  ~herd
#>
#>      post.mean  l-95% CI u-95% CI eff.samp
```

©黄湘云

```
#> herd   0.09136 0.0001426   0.4399    312.5
#>
#>  R-structure:  ~units
#>
#>       post.mean l-95% CI u-95% CI eff.samp
#> units         1        1        1        0
#>
#>  Location effects: cbind(incidence, size - incidence) ~ period
#>
#>             post.mean l-95% CI u-95% CI eff.samp  pMCMC
#> (Intercept)   -1.5568  -2.1070  -0.8795    841.0 <0.001 ***
#> period2       -1.2424  -2.2081  -0.1868    843.2  0.014 *
#> period3       -1.3416  -2.4066  -0.3783    888.0  0.006 **
#> period4       -1.8745  -3.1598  -0.7545    644.5 <0.001 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

下面对结果进行调整

```
# 调整常数
c2 <- ((16 * sqrt(3)) / (15 * pi))^2
# 固定效应
cbpp_sol_adj <- cbpp_mcmcglmm$Sol / sqrt(1 + c2 * cbpp_mcmcglmm$VCV[, 2])
summary(cbpp_sol_adj)
```

```
#>
#> Iterations = 3001:12991
#> Thinning interval = 10
#> Number of chains = 1
#> Sample size per chain = 1000
#>
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#>
#>               Mean     SD Naive SE Time-series SE
#> (Intercept) -1.342 0.2831 0.008953       0.009763
#> period2     -1.071 0.4468 0.014128       0.015386
#> period3     -1.156 0.4603 0.014555       0.015445
#> period4     -1.616 0.5256 0.016621       0.020703
#>
#> 2. Quantiles for each variable:
```

©李东风 统计建模

```
#>
#>               2.5%    25%    50%     75%   97.5%
#> (Intercept) -1.847 -1.552 -1.347 -1.1431 -0.7918
#> period2     -1.921 -1.352 -1.061 -0.7827 -0.1840
#> period3     -2.066 -1.478 -1.136 -0.8395 -0.3173
#> period4     -2.702 -1.970 -1.608 -1.2584 -0.5811
```

```
# 方差成分
cbpp_vcv_adj <- cbpp_mcmcglmm$VCV / (1 + c2 * cbpp_mcmcglmm$VCV[, 2])
summary(cbpp_vcv_adj)
```

```
#>
#> Iterations = 3001:12991
#> Thinning interval = 10
#> Number of chains = 1
#> Sample size per chain = 1000
#>
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#>
#>          Mean     SD Naive SE Time-series SE
#> herd  0.06788 0.1335 0.004221        0.00755
#> units 0.74303 0.0000 0.000000        0.00000
#>
#> 2. Quantiles for each variable:
#>
#>            2.5%      25%     50%     75%  97.5%
#> herd  0.0005777 0.004345 0.01809 0.07155 0.4584
#> units 0.7430287 0.743029 0.74303 0.74303 0.7430
```

可以看到，调整后固定效应的部分和前面 lme4 等的输出非常接近，方差成分仍有差距。

36.2.9 INLA

表示二项分布，公式语法与前面的 brms 包和 lme4 等包都不同。

```
cbpp_inla <- inla(
  formula = incidence ~ period + f(herd, model = "iid", n = 15),
  Ntrials = size, family = "binomial", data = cbpp
)
summary(cbpp_inla)
```

```
#> Fixed effects:
```

©黄湘云


```
#>                mean    sd 0.025quant 0.5quant 0.975quant   mode kld
#> (Intercept) -1.382 0.224     -1.845   -1.375     -0.957 -1.375   0
#> period2     -1.032 0.304     -1.628   -1.032     -0.434 -1.032   0
#> period3     -1.174 0.324     -1.809   -1.174     -0.537 -1.174   0
#> period4     -1.662 0.425     -2.495   -1.662     -0.827 -1.662   0
#>
#> Model hyperparameters:
#>                     mean   sd 0.025quant 0.5quant 0.975quant mode
#> Precision for herd 5.10 6.39       1.02     3.29      18.55 2.17
#>
#>  is computed
```

36.3 非线性混合效应模型

Loblolly 数据集来自 R 内置的 datasets 包，记录了 14 颗火炬树种子的生长情况。

表格 36.2: Loblolly 数据集

Seed	3	5	10	15	20	25
301	4.51	10.89	28.72	41.74	52.70	60.92
303	4.55	10.92	29.07	42.83	53.88	63.39
305	4.79	11.37	30.21	44.40	55.82	64.10
307	3.91	9.48	25.66	39.07	50.78	59.07
309	4.81	11.20	28.66	41.66	53.31	63.05
311	3.88	9.40	25.99	39.55	51.46	59.64
315	4.32	10.43	27.16	40.85	51.33	60.07
319	4.57	10.57	27.90	41.13	52.43	60.69
321	3.77	9.03	25.45	38.98	49.76	60.28
323	4.33	10.79	28.97	42.44	53.17	61.62
325	4.38	10.48	27.93	40.20	50.06	58.49
327	4.12	9.92	26.54	37.82	48.43	56.81
329	3.93	9.34	26.08	37.79	48.31	56.43
331	3.46	9.05	25.85	39.15	49.12	59.49

火炬树种子基本决定了树的长势，不同种子预示最后的高度，并且在生长期也是很稳定地生长

```
ggplot(data = Loblolly, aes(x = age, y = height, color = Seed)) +
  geom_point() +
  geom_line() +
```

©黄湘云

```
  theme_bw() +
  labs(x = "age (yr)", y = "height (ft)")
```

图 36.5: 火炬松树的高度（英尺）随时间（年）的变化

36.3.1 nlme

非线性回归

```
nfm1 <- nls(height ~ SSasymp(age, Asym, R0, lrc),
            data = Loblolly, subset = Seed == 329)
summary(nfm1)
```

```
#>
#> Formula: height ~ SSasymp(age, Asym, R0, lrc)
#>
#> Parameters:
#>      Estimate Std. Error t value Pr(>|t|)
#> Asym  94.1282     8.4030  11.202 0.001525 **
```

©黄湘云

```
#> R0    -8.2508     1.2261  -6.729 0.006700 **
#> lrc   -3.2176     0.1386 -23.218 0.000175 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.7493 on 3 degrees of freedom
#>
#> Number of iterations to convergence: 0
#> Achieved convergence tolerance: 3.972e-07
```

非线性函数 `SSasymp()` 的内容如下

$$\mathrm{Asym} + (\mathrm{R0} - \mathrm{Asym}) \times \exp\big(-\exp(\mathrm{lrc}) \times \mathrm{input}\big)$$

其中，Asym 、R0 、lrc 是参数，input 是输入值。

示例来自 **nlme** 包的函数 `nlme()` 帮助文档

```
nfm2 <- nlme(height ~ SSasymp(age, Asym, R0, lrc),
  data = Loblolly,
  fixed = Asym + R0 + lrc ~ 1,
  random = Asym ~ 1,
  start = c(Asym = 103, R0 = -8.5, lrc = -3.3)
)
summary(nfm2)
```

```
#> Nonlinear mixed-effects model fit by maximum likelihood
#>   Model: height ~ SSasymp(age, Asym, R0, lrc)
#>   Data: Loblolly
#>        AIC      BIC    logLik
#>   239.4856 251.6397 -114.7428
#>
#> Random effects:
#>  Formula: Asym ~ 1 | Seed
#>             Asym  Residual
#> StdDev: 3.650642 0.7188625
#>
#> Fixed effects:  Asym + R0 + lrc ~ 1
#>          Value Std.Error DF   t-value p-value
#> Asym 101.44960 2.4616951 68  41.21128       0
#> R0    -8.62733 0.3179505 68 -27.13420       0
#> lrc   -3.23375 0.0342702 68 -94.36052       0
```

©清华大学出版社

```
#>  Correlation:
#>     Asym   R0
#> R0   0.704
#> lrc -0.908 -0.827
#>
#> Standardized Within-Group Residuals:
#>         Min          Q1         Med          Q3         Max
#> -2.23601930 -0.62380854  0.05917466  0.65727206  1.95794425
#>
#> Number of Observations: 84
#> Number of Groups: 14
```

```
# 更新模型的随机效应部分
nfm3 <- update(nfm2, random = pdDiag(Asym + lrc ~ 1))
summary(nfm3)
```

```
#> Nonlinear mixed-effects model fit by maximum likelihood
#>   Model: height ~ SSasymp(age, Asym, R0, lrc)
#>   Data: Loblolly
#>        AIC      BIC    logLik
#>   238.9662 253.5511 -113.4831
#>
#> Random effects:
#>  Formula: list(Asym ~ 1, lrc ~ 1)
#>  Level: Seed
#>  Structure: Diagonal
#>             Asym        lrc  Residual
#> StdDev: 2.806185 0.03449969 0.6920003
#>
#> Fixed effects:  Asym + R0 + lrc ~ 1
#>          Value Std.Error DF   t-value p-value
#> Asym 101.85205 2.3239828 68  43.82651       0
#> R0    -8.59039 0.3058441 68 -28.08747       0
#> lrc   -3.24011 0.0345017 68 -93.91167       0
#>  Correlation:
#>     Asym   R0
#> R0   0.727
#> lrc -0.902 -0.796
#>
#> Standardized Within-Group Residuals:
#>         Min          Q1         Med          Q3         Max
```

```
#> -2.06072906 -0.69785679  0.08721706  0.73687722  1.79015782
#>
#> Number of Observations: 84
#> Number of Groups: 14
```

36.3.2 lme4

lme4 的公式语法是与 nlme 包不同的。

```
lob_lme4 <- lme4::nlmer(
  height ~ SSasymp(age, Asym, R0, lrc) ~ (Asym + R0 + lrc) + (Asym | Seed),
  data = Loblolly,
  start = c(Asym = 103, R0 = -8.5, lrc = -3.3)
)
summary(lob_lme4)
```

```
#> Nonlinear mixed model fit by maximum likelihood  ['nlmerMod']
#> Formula: height ~ SSasymp(age, Asym, R0, lrc) ~ (Asym + R0 + lrc) + (Asym |
#>     Seed)
#>    Data: Loblolly
#>
#>      AIC      BIC   logLik deviance df.resid
#>    239.4    251.5   -114.7    229.4       79
#>
#> Scaled residuals:
#>      Min       1Q   Median       3Q      Max
#> -2.24149 -0.62546  0.08326  0.67711  1.91351
#>
#> Random effects:
#>  Groups   Name Variance Std.Dev.
#>  Seed     Asym 13.5121  3.6759
#>  Residual       0.5161  0.7184
#> Number of obs: 84, groups:  Seed, 14
#>
#> Fixed effects:
#>        Estimate Std. Error t value
#> Asym 102.119832   0.012378  8249.9
#> R0    -8.549401   0.012354  -692.0
#> lrc   -3.243973   0.008208  -395.2
#>
#> Correlation of Fixed Effects:
```

非洲渔民©

```
#>      Asym   R0
#> R0   0.000
#> lrc -0.008 -0.034
```

36.3.3 brms

根据数据的情况，设定参数的先验分布

```
lob_prior <- c(
  brms::set_prior("normal(101, 0.1)", nlpar = "Asym", lb = 100, ub = 102),
  brms::set_prior("normal(-8, 1)", nlpar = "R0", lb = -10),
  brms::set_prior("normal(-3, 3)", nlpar = "lrc", lb = -9),
  brms::set_prior("normal(3, 0.2)", class = "sigma")
)
```

根据模型表达式编码

```
lob_formula <- brms::bf(
  height ~ Asym + (R0 - Asym) * exp( - exp(lrc) * age),
  # Nonlinear variables
  # Fixed effects: Asym R0 lrc
  R0 + lrc ~ 1,
  # Nonlinear variables
  # Random effects: Seed
  Asym ~ 1 + (1 | Seed),
  # Nonlinear fit
  nl = TRUE
)
```

```
lob_brms <- brms::brm(lob_formula, data = Loblolly, prior = lob_prior)
summary(lob_brms)
```

```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: height ~ Asym + (R0 - Asym) * exp(-exp(lrc) * age)
         R0 ~ 1
         lrc ~ 1
         Asym ~ 1 + (1 | Seed)
   Data: Loblolly (Number of observations: 84)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Group-Level Effects:
```

©黄湘云

```
~Seed (Number of levels: 14)
                    Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Asym_Intercept)      3.90      1.09     2.24     6.51 1.00     1033     1647

Population-Level Effects:
                Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
R0_Intercept       -8.53      0.43    -9.37    -7.68 1.00     2236     1434
lrc_Intercept      -3.23      0.02    -3.27    -3.20 1.00      981     1546
Asym_Intercept    101.00      0.10   100.80   101.20 1.00     4443     2907

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     1.68      0.25     1.20     2.17 1.00     1910     2258

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

36.4 模拟实验比较（补充）

从广义线性混合效应模型生成模拟数据，用至少 6 个不同的 R 包估计模型参数，比较和归纳不同估计方法和实现算法的效果。举例：带漂移项的泊松型广义线性混合效应模型。y_{ij} 表示响应变量，$\boldsymbol{u}$ 表示随机效应，o_{ij} 表示漂移项。

$$
\begin{gathered}
y_{ij}|\boldsymbol{u} \sim \text{Poisson}(o_{ij}\lambda_{ij}) \\
\log(\lambda_{ij}) = \beta_{ij}x_{ij} + u_j \\
u_j \sim \mathcal{N}(0, \sigma^2) \\
i = 1, 2, \ldots, n \quad j = 1, 2, \ldots, q
\end{gathered}
$$

首先准备数据

```r
set.seed(2023)
Ngroups <- 25 # 一个随机效应分 25 个组
NperGroup <- 100 # 每个组 100 个观察值
# 样本量
N <- Ngroups * NperGroup
# 截距和两个协变量的系数
beta <- c(0.5, 0.3, 0.2)
# 两个协变量
```

©南京大学出版社

```
X <- MASS::mvrnorm(N, mu = rep(0, 2), Sigma = matrix(c(1, 0.8, 0.8, 1), 2))
# 漂移项
o <- rep(c(2, 4), each = N / 2)
# 分 25 个组 每个组 100 个观察值
g <- factor(rep(1:Ngroups, each = NperGroup))
u <- rnorm(Ngroups, sd = .5) # 随机效应的标准差 0.5
# 泊松分布的期望
lambda <- o * exp(cbind(1, X) %*% beta + u[g])
# 响应变量的值
y <- rpois(N, lambda = lambda)
# 模拟的数据集
sim_data <- data.frame(y, X, o, g)
colnames(sim_data) <- c("y", "x1", "x2", "o", "g")
```

36.4.1 lme4

```
# 模型拟合
fit_lme4 <- lme4::glmer(y ~ x1 + x2 + (1 | g),
  data = sim_data, offset = log(o), family = poisson(link = "log")
)
summary(fit_lme4)
```

```
#> Generalized linear mixed model fit by maximum likelihood (Laplace
#>   Approximation) [glmerMod]
#>  Family: poisson  ( log )
#> Formula: y ~ x1 + x2 + (1 | g)
#>    Data: sim_data
#>  Offset: log(o)
#>
#>      AIC      BIC   logLik deviance df.resid
#>  11065.6  11088.9  -5528.8  11057.6     2496
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -3.0650 -0.7177 -0.0827  0.6334  4.0103
#>
#> Random effects:
#>  Groups Name        Variance Std.Dev.
#>  g      (Intercept) 0.4074   0.6383
#> Number of obs: 2500, groups:  g, 25
```

©袁浩

```
#>
#> Fixed effects:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  0.55608    0.12805   4.343 1.41e-05 ***
#> x1           0.28442    0.01280  22.214  < 2e-16 ***
#> x2           0.20851    0.01294  16.117  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Correlation of Fixed Effects:
#>    (Intr) x1
#> x1 -0.008
#> x2 -0.008 -0.821
```

36.4.2 GLMMadaptive

对随机效应采用 adaptive Gauss-Hermite quadrature 积分

```r
library(GLMMadaptive)
fit_glmmadaptive <- mixed_model(
  fixed = y ~ x1 + x2 + offset(log(o)),
  random = ~ 1 | g, data = sim_data,
  family = poisson(link = "log")
)
summary(fit_glmmadaptive)
```

```
#>
#> Call:
#> mixed_model(fixed = y ~ x1 + x2 + offset(log(o)), random = ~1 |
#>     g, data = sim_data, family = poisson(link = "log"))
#>
#> Data Descriptives:
#> Number of Observations: 2500
#> Number of Groups: 25
#>
#> Model:
#>  family: poisson
#>  link: log
#>
#> Fit statistics:
#>   log.Lik      AIC      BIC
```

©南京大学出版社

```
#>  -5528.78 11065.56 11070.44
#>
#> Random effects covariance matrix:
#>                StdDev
#> (Intercept) 0.6417329
#>
#> Fixed effects:
#>             Estimate Std.Err z-value p-value
#> (Intercept)   0.5647  0.1288  4.3863 < 1e-04
#> x1            0.2844  0.0128 22.2059 < 1e-04
#> x2            0.2085  0.0129 16.1096 < 1e-04
#>
#> Integration:
#> method: adaptive Gauss-Hermite quadrature rule
#> quadrature points: 11
#>
#> Optimization:
#> method: hybrid EM and quasi-Newton
#> converged: TRUE
```

36.4.3 glmmTMB

```
fit_glmmtmb <- glmmTMB::glmmTMB(
  y ~ x1 + x2 + (1 | g), offset = log(o),
  data = sim_data, family = poisson, REML = TRUE
)
summary(fit_glmmtmb)
```

```
#>  Family: poisson  ( log )
#> Formula:          y ~ x1 + x2 + (1 | g)
#> Data: sim_data
#>  Offset: log(o)
#>
#>      AIC      BIC   logLik deviance df.resid
#>  11082.7  11106.0  -5537.3  11074.7     2499
#>
#> Random effects:
#>
#> Conditional model:
#>  Groups Name        Variance Std.Dev.
```

©黄文涛

```
#>  g      (Intercept) 0.4245   0.6515
#> Number of obs: 2500, groups:  g, 25
#>
#> Conditional model:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  0.55710    0.13069   4.263 2.02e-05 ***
#> x1           0.28442    0.01281  22.206  < 2e-16 ***
#> x2           0.20851    0.01294  16.111  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

36.4.4 hglm

hglm 包的名称是 Hierarchical Generalized Linear Models 的首字母缩写拼成的。

```r
# extended quasi likelihood (EQL) method
fit_hglm <- hglm::hglm(
  fixed =  y ~ x1 + x2, random = ~ 1 | g,
  family = poisson(link = "log"),
  offset = log(o), data = sim_data
)
summary(fit_hglm)
```

```
#> Call:
#> hglm.formula(family = poisson(link = "log"), fixed = y ~ x1 +
#>     x2, random = ~1 | g, data = sim_data, offset = log(o))
#>
#> ----------
#> MEAN MODEL
#> ----------
#>
#> Summary of the fixed effects estimates:
#>
#>             Estimate Std. Error t-value Pr(>|t|)
#> (Intercept)  0.55720    0.13058   4.267 2.05e-05 ***
#> x1           0.28442    0.01316  21.609  < 2e-16 ***
#> x2           0.20851    0.01330  15.678  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> Note: P-values are based on 2473 degrees of freedom
#>
```

©南京大学出版社

```
#> Summary of the random effects estimates:
#>
#>    Estimate Std. Error
#> g1   0.6643     0.1352
#> g2  -0.1421     0.1402
#> g3   0.9560     0.1344
#> ...
#> NOTE: to show all the random effects, use print(summary(hglm.object), print.ranef = TRUE).
#>
#> ----------------
#> DISPERSION MODEL
#> ----------------
#>
#> NOTE: h-likelihood estimates through EQL can be biased.
#>
#> Dispersion parameter for the mean model:
#> [1] 1.055997
#>
#> Model estimates for the dispersion term:
#>
#> Link = log
#>
#> Effects:
#>   Estimate Std. Error
#>     0.0545     0.0284
#>
#> Dispersion = 1 is used in Gamma model on deviances to calculate the standard error(s).
#>
#> Dispersion parameter for the random effects:
#> [1] 0.4236
#>
#> Dispersion model for the random effects:
#>
#> Link = log
#>
#> Effects:
#> .|Random1
#>   Estimate Std. Error
#>    -0.8589     0.2895
#>
```

©李东风

```
#> Dispersion = 1 is used in Gamma model on deviances to calculate the standard error(s).
#>
#> EQL estimation converged in 3 iterations.
#>
#> !! Observation 302 is too influential! Estimates are likely unreliable !!
```

36.4.5 glmmML

glmmML 包 Maximum Likelihood and numerical integration via Gauss-Hermite quadrature

```
fit_glmmml <- glmmML::glmmML(
  formula = y ~ x1 + x2, family = poisson,
  data = sim_data, offset = log(o), cluster = g
)
summary(fit_glmmml)
```

```
Call:  glmmML::glmmML(formula = y ~ x1 + x2, family = poisson, data = sim_data,      cluster = g, c



             coef se(coef)     z Pr(>|z|)
(Intercept) 0.556   0.1281  4.34  1.4e-05
x1          0.284   0.0128 22.21  0.0e+00
x2          0.209   0.0129 16.11  0.0e+00

Scale parameter in mixing distribution:  0.638 gaussian
Std. Error:                              0.0865

        LR p-value for H_0: sigma = 0:  0

Residual deviance: 2770 on 2496 degrees of freedom  AIC: 2780
```

36.4.6 glmm

glmm 包对随机效应的积分采用 Monte Carlo Likelihood Approximation 近似

```
# 对迭代时间没有给出预估，一旦执行，不知道什么时候会跑完
set.seed(2023)
# 设置双核并行迭代
clust <- parallel::makeCluster(2)
fit_glmm <- glmm::glmm(y ~ x1 + x2 + offset(log(o)),
  random = list(~ 1 + g), # 随机效应
```

©李东风 统计课

```
  varcomps.names = "G", # 给随机效应取个名字
  data = sim_data,
  family.glmm = glmm::poisson.glmm, # 泊松型
  m = 10^4, debug = TRUE, cluster = clust
)
parallel::stopCluster(clust)
summary(fit_glmm)
```

glmm 包的帮助文档中的示例如下，可复现结果，运行时间 1-2 分钟。

```
set.seed(1234)
clust <- makeCluster(2)
sal <- glmm(
  Mate ~ 0 + Cross, random = list(~ 0 + Female, ~ 0 + Male),
  varcomps.names = c("F", "M"), data = salamander,
  family.glmm = bernoulli.glmm, m = 10^4, debug = TRUE, cluster = clust
)
summary(sal)
stopCluster(clust)
```

```
Call:
glmm(fixed = Mate ~ 0 + Cross, random = list(~0 + Female, ~0 + Male),
    varcomps.names = c("F", "M"), data = salamander,
    family.glmm = bernoulli.glmm, m = 10^4, debug = TRUE, cluster = clust)

Link is: "logit (log odds)"

Fixed Effects:
         Estimate Std. Error z value Pr(>|z|)
CrossR/R    1.230      0.300   4.045 5.24e-05 ***
CrossR/W    0.320      0.267   1.198  0.23077
CrossW/R   -2.000      0.330  -6.042 1.52e-09 ***
CrossW/W    0.920      0.300   3.084  0.00204 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Variance Components for Random Effects (P-values are one-tailed):
  Estimate Std. Error z value Pr(>|z|)/2
F     1.46       0.31   4.695   1.33e-06 ***
M     1.64       0.33   4.918   4.36e-07 ***
---
```

©黄湘云

```
Signif. codes:  0  ‘***’  0.001  ‘**’  0.01  ‘*’  0.05  ‘.’  0.1  ‘ ’  1
```

36.4.7 gee

gee 包采用广义估计方程（Generalized Estimation Equation）方法

```
fit_gee <- gee::gee(y ~ x1 + x2 + offset(log(o)), id = g,
  data = sim_data, family = poisson(link = "log"), corstr = "exchangeable"
)
# 输出
fit_gee
```

```
 GEE:  GENERALIZED LINEAR MODELS FOR DEPENDENT DATA
 gee S-function, version 4.13 modified 98/01/27 (1998)

Model:
 Link:                      Logarithm
 Variance to Mean Relation: Poisson
 Correlation Structure:     Exchangeable

Call:
gee::gee(formula = y ~ x1 + x2 + offset(log(o)), id = g, data = sim_data,
    family = poisson(link = "log"), corstr = "exchangeable")

Number of observations :  2500
Maximum cluster size   :  100

Coefficients:
(Intercept)          x1          x2
  0.6098935   0.3003721   0.2165055

Estimated Scale Parameter:  4.979956
Number of Iterations:  3

Working Correlation[1:4,1:4]
          [,1]      [,2]      [,3]      [,4]
[1,] 1.0000000 0.7220617 0.7220617 0.7220617
[2,] 0.7220617 1.0000000 0.7220617 0.7220617
[3,] 0.7220617 0.7220617 1.0000000 0.7220617
[4,] 0.7220617 0.7220617 0.7220617 1.0000000
```

```
Returned Error Value:
[1] 0
```

输出结果中，尺度参数（Estimated Scale Parameter）的估计结果与随机效应的方差的联系？

36.4.8 geepack

geepack 包类似 **gee** 包。

```
fit_geepack <- geepack::geeglm(
  formula = y ~ x1 + x2, family = poisson(link = "log"),
  id = g, offset = log(o), data = sim_data,
  corstr = "exchangeable", scale.fix = FALSE
)
summary(fit_geepack)
```

```
Call:
geepack::geeglm(formula = y ~ x1 + x2, family = poisson(link = "log"),
    data = sim_data, offset = log(o), id = g, corstr = "exchangeable",
    scale.fix = FALSE)

 Coefficients:
            Estimate Std.err  Wald Pr(>|W|)
(Intercept)  0.60964 0.17310  12.4 0.000428 ***
x1           0.30040 0.02353 163.1  < 2e-16 ***
x2           0.21653 0.01458 220.6  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Correlation structure = exchangeable
Estimated Scale Parameters:

            Estimate Std.err
(Intercept)    4.975    1.39
  Link = identity

Estimated Correlation Parameters:
      Estimate Std.err
alpha    0.723 0.06703
Number of clusters:   25  Maximum cluster size: 100
```

©贝叶斯统计

36.4.9 blme

blme 包采用贝叶斯估计

```
fit_blme <- blme::bglmer(
  formula = y ~ x1 + x2 + (1 | g),
  data = sim_data, offset = log(o),
  family = poisson(link = "log")
)
summary(fit_blme)
```

```
#> Cov prior  : g ~ wishart(df = 3.5, scale = Inf, posterior.scale = cov, common.scale = TRUE)
#> Prior dev   : 1.2531
#>
#> Generalized linear mixed model fit by maximum likelihood (Laplace
#>   Approximation) [bglmerMod]
#>  Family: poisson  ( log )
#> Formula: y ~ x1 + x2 + (1 | g)
#>    Data: sim_data
#>  Offset: log(o)
#>
#>      AIC      BIC   logLik deviance df.resid
#>  11065.6  11088.9  -5528.8  11057.6     2496
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -3.0651 -0.7180 -0.0816  0.6335  4.0103
#>
#> Random effects:
#>  Groups Name        Variance Std.Dev.
#>  g      (Intercept) 0.4337   0.6586
#> Number of obs: 2500, groups:  g, 25
#>
#> Fixed effects:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  0.55595    0.13209   4.209 2.57e-05 ***
#> x1           0.28442    0.01280  22.214  < 2e-16 ***
#> x2           0.20851    0.01294  16.117  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
```

非常课程 ©

```
#> Correlation of Fixed Effects:
#>    (Intr) x1
#> x1 -0.008
#> x2 -0.007 -0.821
```

GLMMadaptive、glmmML、gee、geepack 和 lme4 的模型输出结果是接近的。

36.4.10 brms

```
fit_brms <- brms::brm(
  y ~ x1 + x2 + (1 | g) + offset(log(o)),
  data = sim_data, family = poisson(link = "log"),
  silent = 2, refresh = 0, seed = 20232023
)
summary(fit_brms)
```

```
 Family: poisson
  Links: mu = log
Formula: y ~ x1 + x2 + (1 | g) + offset(log(o))
   Data: sim_data (Number of observations: 2500)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Group-Level Effects:
~g (Number of levels: 25)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.68      0.11     0.51     0.94 1.01      295      491

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept     0.56      0.14     0.31     0.85 1.02      297      344
x1            0.28      0.01     0.26     0.31 1.00     1053     1625
x2            0.21      0.01     0.18     0.23 1.01     1071     1298

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

36.4.11 MCMCglmm

MCMCglmm 包采用贝叶斯估计

```r
prior1 <- list(
  R = list(V = 1, fix = 1),
  G = list(G1 = list(V = 1, nu = 0.002))
)
set.seed(20232023)
fit_mcmcglmm <- MCMCglmm::MCMCglmm(
  fixed = y ~ x1 + x2 + offset(log(o)),
  random = ~g, family = "poisson",
  data = sim_data, verbose = FALSE, prior = prior1
)
summary(fit_mcmcglmm)
```

```
#>
#>  Iterations = 3001:12991
#>  Thinning interval  = 10
#>  Sample size  = 1000
#>
#>  DIC: 12397.82
#>
#>  G-structure:  ~g
#>
#>   post.mean l-95% CI u-95% CI eff.samp
#> g    0.5443   0.2753   0.8924     1000
#>
#>  R-structure:  ~units
#>
#>       post.mean l-95% CI u-95% CI eff.samp
#> units         1        1        1        0
#>
#>  Location effects: y ~ x1 + x2 + offset(log(o))
#>
#>             post.mean l-95% CI u-95% CI eff.samp  pMCMC
#> (Intercept)    1.4315   1.1536   1.7161   1000.0 <0.001 ***
#> x1             0.3099   0.2405   0.3892    925.6 <0.001 ***
#> x2             0.2234   0.1465   0.3017    877.7 <0.001 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

随机效应的方差 G-structure 为 0.5443，则标准差为 0.738。

对于离散型响应变量，MCMCglmm 包默认添加一个可加的随机变量表示过度离散，如何将其去掉？将

残差方差设置为常数，不再作为参数去估计，`fix = 1` 表示在 R-structure 中固定方差，`V = 1` 表示残差方差为 1。

```
# 固定效应参数的后验分布
# plot(fit_mcmcglmm$Sol)
plot(fit_mcmcglmm$VCV)
```

图 36.6: 方差协方差参数的后验分布

根据响应变量的服从的分布类型，确定调整因子。固定效应乘以调整因子的平方根，随机效应的方差乘以调整因子，详见 (Peter Diggle 和 Zeger 2002) 第 136-137 页。二项分布联系函数对应的调整因子如下：

$$\frac{1+c^2\sigma_\epsilon^2}{1+c^2\sigma_{\text{units}}^2}$$

统计学习基础 ©

其中，c 是与联系函数有关的常数，二项分布联系函数对应 $c = 16\sqrt{3}/(15\pi)$。此处，假定 $\sigma_\epsilon^2 = 0$ ，代入泊松分布对应的调整因子。调整后的固定效应（回归系数）、随机效应的方差如下：

```
# 调整公式中的调整因子 c2 取决于联系函数
c2 <- ((16 * sqrt(3))/(15 * pi))^2 # 需要修改为泊松分布对应的值
# 固定效应的调整
adjusted_sol <- fit_mcmcglmm$Sol / sqrt(1 + c2 * fit_mcmcglmm$VCV[, 2])
plot(adjusted_sol)
# 随机效应的方差调整
adjusted_vcv <- fit_mcmcglmm$VCV[, 1] / (1 + c2 * fit_mcmcglmm$VCV[, 2])
plot(adjusted_vcv)
```

36.4.12 INLA

```
library(INLA)
fit_inla <- inla(
  formula = y ~ x1 + x2 + f(g, model = "iid", n = 25),
  E = o, family = "poisson", data = sim_data
)
summary(fit_inla)
```

```
#> Fixed effects:
#>              mean    sd 0.025quant 0.5quant 0.975quant  mode kld
#> (Intercept) 0.556 0.130      0.300    0.556      0.812 0.556   0
#> x1          0.284 0.013      0.259    0.284      0.310 0.284   0
#> x2          0.209 0.013      0.183    0.209      0.234 0.209   0
#>
#> Model hyperparameters:
#>                 mean    sd 0.025quant 0.5quant 0.975quant mode
#> Precision for g 2.56 0.713       1.36     2.49       4.14 2.36
#>
#>  is computed
```

随机效应的标准（偏）差为 $1/\sqrt{\text{Precision}}$ ，即 0.625。

36.5 总结

本章介绍函数 `MASS::glmmPQL()`、`nlme::lme()`、`lme4::lmer()` 和 `brms::brm()` 的用法，以及它们求解线性混合效应模型的区别和联系。在贝叶斯估计方法中，**brms** 包和 **INLA** 包都支持非常丰富的模型种类，前者是贝叶斯精确推断，后者是贝叶斯近似推断，**brms** 基于概率编程语言 Stan 框架打包了许

©黄湘云

多模型的 Stan 实现，INLA 基于求解随机偏微分方程的有限元方法和拉普拉斯近似技巧，将各类常见统计模型统一起来，计算速度快，计算结果准确。

1. 函数 `nlme::lme()` 提供极大似然估计和限制极大似然估计。
2. 函数 `MASS::glmmPQL()` 惩罚拟似然估计，MASS 是依赖 nlme 包，nlme 不支持模型中添加漂移项，所以函数 `glmmPQL()` 也不支持添加漂移项。
3. 函数 `lme4::lmer()` 拉普拉斯近似关于随机效应的高维积分。
4. 函数 `brms::brm()` 汉密尔顿蒙特卡罗抽样。HMC 方法结合自适应步长的采样器 NUTS 来抽样。
5. 函数 `INLA::inla()` 集成嵌套拉普拉斯近似。

表格 36.3: 混合效应模型及相关 R 包拟合函数

模型	nlme	MASS	lme4	GLMMadaptive	brms
线性混合效应模型	`lme()`	`glmmPQL()`	`lmer()`	不支持	`brm()`
广义线性混合效应模型	不支持	`glmmPQL()`	`glmer()`	`mixed_model()`	`brm()`
非线性混合效应模型	`nlme()`	不支持	`nlmer()`	不支持	`brm()`

通过对频率派和贝叶斯派方法的比较，发现一些有意思的结果。与 Stan 不同，INLA 包做近似贝叶斯推断，计算效率很高。

INLA 软件能处理上千个高斯随机效应，但最多只能处理 15 个超参数，因为 INLA 使用 CCD 处理超参数。如果使用 MCMC 处理超参数，就有可能处理更多的超参数，Daniel Simpson 等把 Laplace approximation 带入 Stan，这样就可以处理上千个超参数。更多理论内容见 2009 年 INLA 诞生的论文和《Advanced Spatial Modeling with Stochastic Partial Differential Equations Using R and INLA》中第一章的估计方法 CCD。

36.6 习题

1. 基于奥克兰火山地形数据集 volcano ，随机拆分成训练数据和测试数据，训练数据可以看作采样点的观测数据，建立高斯过程回归模型，比较测试数据与未采样的位置上的预测数据，在计算速度、准确度、易用性等方面总结 Stan 和 INLA 的特点。

2. 基于 `PlantGrowth` 数据集，比较将 `group` 变量视为随机变量与随机效应的异同?

```
fit_lm <- lm(weight ~ group, data = PlantGrowth)
summary(fit_lm)
fit_lme <- nlme::lme(weight ~ 1, random = ~ 1 | group, data = PlantGrowth)
summary(fit_lme)
```

```
fit_lme4 <- lme4::lmer(weight ~ 1 + (1 | group), data = PlantGrowth)
summary(fit_lme4)
```

3. **MASS** 包的数据集 epil 记录癫痫发作的次数及病人的特征，请建立混合效应模型分析癫痫病发作的风险与病人特征之间的关系。

4. 基于数据集 Puromycin 分析酶促反应的反应速率（提示：Michaelis-Menten 模型和函数 `SSmicmen()`）。

```
ggplot(data = Puromycin, aes(x = conc, y = rate, color = state)) +
  geom_point() +
  geom_line() +
  theme_minimal() +
  labs(
    x = "Substrate concentration (ppm)",
    y = "Reaction velocity (counts/min/min)"
  )
```

图 36.7: Puromycin 反应速率变化趋势

5. 基于 **MASS** 包的地形数据集 topo，建立高斯过程回归模型，比较贝叶斯预测与克里金插值预测的效果。

©黄湘云

第三十七章　广义可加模型

```
library(mgcv)      # 广义可加模型
library(splines)   # 样条
library(cmdstanr)  # 编译采样
library(ggplot2)   # 作图
library(bayesplot) # 后验分布
library(loo)       # LOO-CV
library(INLA)      # 近似贝叶斯推断
options(mc.cores = 2) # 全局设置双核
```

相比于广义线性模型，广义可加模型可以看作是一种非线性模型，模型中含有非线性的成分。

注释

- 多元适应性（自适应）回归样条 multivariate adaptive regression splines
- Friedman, Jerome H. 1991. Multivariate Adaptive Regression Splines. The Annals of Statistics. 19(1):1–67. https://doi.org/10.1214/aos/1176347963
- earth: Multivariate Adaptive Regression Splines http://www.milbo.users.sonic.net/earth
- Friedman, Jerome H. 2001. Greedy function approximation: A gradient boosting machine. The Annals of Statistics. 29(5):1189–1232. https://doi.org/10.1214/aos/1013203451
- Friedman, Jerome H., Trevor Hastie and Robert Tibshirani. Additive Logistic Regression: A Statistical View of Boosting. The Annals of Statistics. 28(2): 337–374. http://www.jstor.org/stable/2674028
- Flexible Modeling of Alzheimer's Disease Progression with I-Splines PDF 文档
- Implementation of B-Splines in Stan 网页文档

37.1　案例：模拟摩托车事故

37.1.1　mgcv

MASS 包的 mcycle 数据集

©黄湘云

```
data(mcycle, package = "MASS")
str(mcycle)
```

```
#> 'data.frame':    133 obs. of  2 variables:
#>  $ times: num  2.4 2.6 3.2 3.6 4 6.2 6.6 6.8 7.8 8.2 ...
#>  $ accel: num  0 -1.3 -2.7 0 -2.7 -2.7 -2.7 -1.3 -2.7 -2.7 ...
```

```
library(ggplot2)
ggplot(data = mcycle, aes(x = times, y = accel)) +
  geom_point() +
  theme_classic() +
  labs(x = "时间 (ms) ", y = "加速度 (g) ")
```

图 37.1: mcycle 数据集

样条回归

```
library(mgcv)
mcycle_mgcv <- gam(accel ~ s(times), data = mcycle, method = "REML")
summary(mcycle_mgcv)
```

```
#>
#> Family: gaussian
#> Link function: identity
#>
```

```
#> Formula:
#> accel ~ s(times)
#>
#> Parametric coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  -25.546      1.951  -13.09   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>            edf Ref.df    F p-value
#> s(times) 8.625  8.958 53.4  <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.783   Deviance explained = 79.7%
#> -REML = 616.14  Scale est. = 506.35    n = 133
```

方差成分

```
gam.vcomp(mcycle_mgcv, rescale = FALSE)
```

```
#>
#> Standard deviations and 0.95 confidence intervals:
#>
#>            std.dev     lower      upper
#> s(times) 807.88726 480.66162 1357.88215
#> scale     22.50229  19.85734   25.49954
#>
#> Rank: 2/2
```

```
plot(mcycle_mgcv)
```

© 黄湘云

图 37.2: mcycle 数据集

ggplot2 包的平滑图层函数 `geom_smooth()` 集成了 **mgcv** 包的函数 `gam()` 的功能。

```
library(ggplot2)
ggplot(data = mcycle, aes(x = times, y = accel)) +
  geom_point() +
  geom_smooth(method = "gam", formula = y ~ s(x, bs = "tp"), method.args = list(method = "REML"))
```

©黄湘云

图 37.3: ggplot2 平滑

37.1.2 cmdstanr

```
library(cmdstanr)
```

37.1.3 rstanarm

rstanarm 可以拟合一般的广义可加（混合）模型。

```
library(rstanarm)
mcycle_rstanarm <- stan_gamm4(accel ~ s(times),
  data = mcycle, family = gaussian(), cores = 2, seed = 20232023,
  iter = 4000, warmup = 1000, thin = 10, refresh = 0,
  adapt_delta = 0.99
)
summary(mcycle_rstanarm)
```

```
Model Info:
 function:     stan_gamm4
 family:       gaussian [identity]
 formula:      accel ~ s(times)
 algorithm:    sampling
```

©黄湘云

```
 sample:       1200 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 133

Estimates:
                      mean    sd      10%     50%     90%
(Intercept)            -25.6     2.1   -28.4   -25.5   -23.0
s(times).1             340.4   232.9    61.1   340.8   634.7
s(times).2           -1218.9   243.3 -1529.2 -1218.8  -913.5
s(times).3            -567.8   147.0  -765.2  -567.1  -385.3
s(times).4            -619.8   133.8  -791.1  -617.0  -458.9
s(times).5           -1056.2    85.8 -1162.8 -1055.7  -945.1
s(times).6             -89.2    49.8  -154.4   -89.4   -27.6
s(times).7            -232.2    33.8  -274.7  -232.2  -189.5
s(times).8              17.3   105.8  -121.0    15.5   150.1
s(times).9               4.1    33.1   -25.8     1.0    39.1
sigma                   24.7     1.6    22.6    24.6    26.8
smooth_sd[s(times)1]   399.9    59.2   327.6   395.4   479.1
smooth_sd[s(times)2]    25.2    25.4     2.9    17.5    56.6

Fit Diagnostics:
           mean   sd    10%   50%   90%
mean_PPD -25.5    3.0 -29.3 -25.5 -21.8

The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for detai

MCMC diagnostics
                     mcse Rhat n_eff
(Intercept)          0.1  1.0  1052
s(times).1           7.0  1.0  1103
s(times).2           6.7  1.0  1329
s(times).3           4.4  1.0  1101
s(times).4           3.8  1.0  1230
s(times).5           2.5  1.0  1137
s(times).6           1.5  1.0  1128
s(times).7           1.0  1.0  1062
s(times).8           3.1  1.0  1147
s(times).9           1.0  1.0  1052
sigma                0.0  1.0  1154
smooth_sd[s(times)1] 1.8  1.0  1136
```

```
smooth_sd[s(times)2] 0.7  1.0  1157
mean_PPD             0.1  1.0   997
log-posterior        0.1  1.0  1122

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample size, and
```

计算 LOO 值

```
loo(mcycle_rstanarm)
```

```
Computed from 1200 by 133 log-likelihood matrix

         Estimate   SE
elpd_loo   -611.0  8.8
p_loo         7.3  1.2
looic      1222.0 17.5
------
Monte Carlo SE of elpd_loo is 0.1.

All Pareto k estimates are good (k < 0.5).
See help('pareto-k-diagnostic') for details.
```

```
plot_nonlinear(mcycle_rstanarm)
pp_check(mcycle_rstanarm)
```

37.1.4 brms

另一个综合型的贝叶斯分析扩展包是 brms 包

```
# 拟合模型
mcycle_brms <- brms::brm(accel ~ s(times),
  data = mcycle, family = gaussian(), cores = 2, seed = 20232023,
  iter = 4000, warmup = 1000, thin = 10, refresh = 0, silent = 2,
  control = list(adapt_delta = 0.99)
)
# 模型输出
summary(mcycle_brms)
```

```
#>  Family: gaussian
#>   Links: mu = identity; sigma = identity
#> Formula: accel ~ s(times)
#>    Data: mcycle (Number of observations: 133)
#>   Draws: 4 chains, each with iter = 4000; warmup = 1000; thin = 10;
```

© 统计学习


```
#>          total post-warmup draws = 1200
#>
#> Smooth Terms:
#>               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
#> sds(stimes_1)   722.94    193.12   455.20  1174.52 1.00     1088     1246
#>
#> Population-Level Effects:
#>           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
#> Intercept   -25.43      1.95   -29.35   -21.58 1.00     1118     1254
#> stimes_1    122.17    289.80  -465.71   668.34 1.00     1210     1197
#>
#> Family Specific Parameters:
#>       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
#> sigma    22.76      1.46    20.20    25.79 1.00     1083     1173
#>
#> Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
#> and Tail_ESS are effective sample size measures, and Rhat is the potential
#> scale reduction factor on split chains (at convergence, Rhat = 1).
```

固定效应

```
brms::fixef(mcycle_brms)
```

```
#>            Estimate  Est.Error       Q2.5    Q97.5
#> Intercept -25.43013   1.946244  -29.34629 -21.5809
#> stimes_1  122.16814 289.800852 -465.70761 668.3376
```

LOO 值与 rstanarm 包计算的值很接近。

```
brms::loo(mcycle_brms)
```

```
#>
#> Computed from 1200 by 133 log-likelihood matrix
#>
#>          Estimate   SE
#> elpd_loo   -608.3 10.2
#> p_loo         8.8  1.5
#> looic      1216.7 20.4
#> ------
#> Monte Carlo SE of elpd_loo is 0.1.
#>
#> All Pareto k estimates are good (k < 0.5).
#> See help('pareto-k-diagnostic') for details.
```

©黄湘云

模型中样条平滑的效应

```
plot(brms::conditional_smooths(mcycle_brms))
brms::pp_check(mcycle_brms, ndraws = 50)
```

(a) 样条平滑效应 (b) 后验预测分布

图 37.4: 后验预测分布检查

37.1.5 GINLA

mgcv 包的简化版 INLA 算法用于贝叶斯计算

```
library(mgcv)
mcycle_mgcv <- gam(accel ~ s(times), data = mcycle, fit = FALSE)
# 简化版 INLA
mcycle_ginla <- ginla(G = mcycle_mgcv)
str(mcycle_ginla)
```

```
#> List of 2
#>  $ density: num [1:10, 1:100] 2.04e-04 2.13e-05 8.21e-06 3.47e-05 1.14e-05 ...
#>  $ beta   : num [1:10, 1:100] -32.8 -133.4 -139.4 -152.5 -153.3 ...
```

提取最大后验估计

```
idx <- apply(mcycle_ginla$density, 1, function(x) x == max(x))
mcycle_ginla$beta[t(idx)]
```

```
#>  [1]  -64.502391 -110.258456   89.506696   -9.449407 -110.635738   16.145715
#>  [7]  -25.472566   43.025985  -21.992372   35.893285
```

©黄湘云

37.1.6 INLA

```
library(INLA)
library(splines)
```

37.2 案例：朗格拉普岛核污染

从线性到可加，意味着从线性到非线性，可加模型容纳非线性的成分，比如高斯过程、样条。

37.2.1 mgcv

本节复用章节 二十八 朗格拉普岛核污染数据，相关背景不再赘述，下面首先加载数据到 R 环境。

```
# 加载数据
rongelap <- readRDS(file = "data/rongelap.rds")
rongelap_coastline <- readRDS(file = "data/rongelap_coastline.rds")
```

接着，将岛上各采样点的辐射强度展示出来，算是简单回顾一下数据概况。

在这里，从广义可加混合效应模型的角度来对核污染数据建模，空间效应仍然是用高斯过程来表示，响应变量服从带漂移项的泊松分布。采用 mgcv 包 (S. N. Wood 2004) 的函数 `gam()` 拟合模型，其中，含 49 个参数的样条近似高斯过程，高斯过程的核函数为默认的梅隆型。更多详情见 **mgcv** 包的函数 `s()` 帮助文档参数的说明，默认值是梅隆型相关函数及默认的范围参数，作者自己定义了一套符号约定。

```
library(nlme)
library(mgcv)
fit_rongelap_gam <- gam(
  counts ~ s(cX, cY, bs = "gp", k = 50), offset = log(time),
  data = rongelap, family = poisson(link = "log")
)
# 模型输出
summary(fit_rongelap_gam)
```

```
#>
#> Family: poisson
#> Link function: log
#>
#> Formula:
#> counts ~ s(cX, cY, bs = "gp", k = 50)
#>
#> Parametric coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
```

©黄湘云

图 37.5: 岛上各采样点的辐射强度

©黄湘云

```
#> (Intercept) 1.976815   0.001642    1204   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>            edf Ref.df Chi.sq p-value
#> s(cX,cY) 48.98     49  34030  <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.876   Deviance explained = 60.7%
#> UBRE = 153.78  Scale est. = 1         n = 157
```

```
# 随机效应
gam.vcomp(fit_rongelap_gam)
```

```
#> s(cX,cY)
#> 2543.376
```

值得一提的是核函数的类型和默认参数的选择，参数 m 接受一个向量，`m[1]` 取值为 1 至 5，分别代表球型 spherical, 幂指数 power exponential 和梅隆型 Matern with $\kappa = 1.5, 2.5$ or 3.5 等 5 种相关/核函数。

```
# 球型相关函数及范围参数为 0.5
fit_rongelap_gam <- gam(
  counts ~ s(cX, cY, bs = "gp", k = 50, m = c(1, .5)),
  offset = log(time), data = rongelap, family = poisson(link = "log")
)
```

接下来，基于岛屿的海岸线数据划分出网格，将格点作为新的预测位置。

```
library(sf)
```

```
#> Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

```
library(abind)
library(stars)
# 类型转化
rongelap_sf <- st_as_sf(rongelap, coords = c("cX", "cY"), dim = "XY")
rongelap_coastline_sf <- st_as_sf(rongelap_coastline, coords = c("cX", "cY"), dim = "XY")
rongelap_coastline_sfp <- st_cast(st_combine(st_geometry(rongelap_coastline_sf)), "POLYGON")
# 添加缓冲区
rongelap_coastline_buffer <- st_buffer(rongelap_coastline_sfp, dist = 50)
# 构造带边界约束的网格
rongelap_coastline_grid <- st_make_grid(rongelap_coastline_buffer, n = c(150, 75))
```

© 统计学习基础

```
# 将 sfc 类型转化为 sf 类型
rongelap_coastline_grid <- st_as_sf(rongelap_coastline_grid)
rongelap_coastline_buffer <- st_as_sf(rongelap_coastline_buffer)
rongelap_grid <- rongelap_coastline_grid[rongelap_coastline_buffer, op = st_intersects]
# 计算网格中心点坐标
rongelap_grid_centroid <- st_centroid(rongelap_grid)
# 共计 1612 个预测点
rongelap_grid_df <- as.data.frame(st_coordinates(rongelap_grid_centroid))
colnames(rongelap_grid_df) <- c("cX", "cY")
```

模型对象 `fit_rongelap_gam` 在新的格点上预测核辐射强度，接着整理预测结果数据。

```
# 预测
rongelap_grid_df$ypred <- as.vector(predict(fit_rongelap_gam, newdata = rongelap_grid_df, type = "respor
# 整理预测数据
rongelap_grid_sf <- st_as_sf(rongelap_grid_df, coords = c("cX", "cY"), dim = "XY")
rongelap_grid_stars <- st_rasterize(rongelap_grid_sf, nx = 150, ny = 75)
rongelap_stars <- st_crop(x = rongelap_grid_stars, y = rongelap_coastline_sfp)
```

最后，将岛上各个格点的核辐射强度绘制出来，给出全岛核辐射强度的空间分布。

图 37.6: 核辐射强度的预测分布

37.2.2 cmdstanr

FRK 包 (Sainsbury-Dale, Zammit-Mangion, 和 Cressie 2022)（Fixed Rank Kriging，固定秩克里金）

©黄湘云

可对有一定规模的（时空）空间区域数据和点参考数据集建模，响应变量的分布从高斯分布扩展到指数族，放在（时空）空间广义线性混合效应模型的框架下统一建模。然而，不支持带漂移项的泊松分布。

brms 包支持一大类贝叶斯统计模型，但是对高斯过程建模十分低效，当遇到有一定规模的数据，建模是不可行的，因为经过对 brms 包生成的模型代码的分析，发现它采用潜变量高斯过程（latent variable GP）模型，这也是采样效率低下的一个关键因素。

```
# 预计运行 1 个小时以上
rongelap_brm <- brms::brm(counts ~ gp(cX, cY) + offset(log(time)),
  data = rongelap, family = poisson(link = "log")
)
# 基样条近似拟合也很慢
rongelap_brm <- brms::brm(
  counts ~ gp(cX, cY, c = 5/4, k = 5) + offset(log(time)),
  data = rongelap, family = poisson(link = "log")
)
```

当设置 $k = 5$ 时，用 5 个基函数来近似高斯过程，编译完成后，采样速度很快，但是结果不可靠，采样过程中的问题很多。当将横、纵坐标值同时缩小 6000 倍，采样效率并未得到改善。当设置 $k = 15$ 时，运行时间明显增加，采样过程的诊断结果类似 $k = 5$ 的情况，还是不可靠。截止写作时间，函数 `gp()` 的参数 `cov` 只能取指数二次核函数（exponentiated-quadratic kernel）。说明 brms 包不适合处理含高斯过程的模型。

实际上，Stan 没有现成的有效算法或扩展包做有规模的高斯过程建模，详见 Bob Carpenter 在 2023 年 Stan 大会的报告，因此，必须采用一些近似方法，通过 Stan 编码实现。接下来，分别手动实现低秩和基样条两种方法近似边际高斯过程（marginal likelihood GP）(Rasmussen 和 Williams 2006)，用 Stan 编码模型。代码文件分别是 `rongelap_poisson_lr.stan` 和 `rongelap_poisson_splines.stan` 。

```
library(cmdstanr)
```

37.2.3 GINLA

mgcv 包的函数 `ginla()` 实现简化版的 Integrated Nested Laplace Approximation, INLA (Simon N. Wood 2019)。

```
rongelap_gam <- gam(
  counts ~ s(cX, cY, bs = "gp", k = 50), offset = log(time),
  data = rongelap, family = poisson(link = "log"), fit = FALSE
)
# 简化版 INLA
rongelap_ginla <- ginla(G = rongelap_gam)
str(rongelap_ginla)
```

```
#> List of 2
```

©黄湘云

```
#>  $ density: num [1:50, 1:100] 2.49e-01 9.03e-06 3.51e-06 1.97e-06 1.17e-06 ...
#>  $ beta   : num [1:50, 1:100] 1.97 -676.61 -572.67 4720.77 240.12 ...
```

其中，$k = 50$ 表示 49 个样条参数，每个参数的分布对应有 100 个采样点，另外，截距项的边际后验概率密度分布如下：

```
plot(
  rongelap_ginla$beta[1, ], rongelap_ginla$density[1, ],
  type = "l", xlab = "截距项", ylab = "概率密度"
)
```

图 37.7: 截距项的边际后验概率密度分布

不难看出，截距项在 1.976 至 1.978 之间，50 个参数的最大后验估计分别如下：

```
idx <- apply(rongelap_ginla$density, 1, function(x) x == max(x))
rongelap_ginla$beta[t(idx)]
```

```
#>  [1]  1.977019e+00 -5.124099e+02  5.461183e+03  1.515296e+03 -2.822166e+03
#>  [6] -1.598371e+04 -6.417855e+03  1.938121e+02 -4.270878e+03  3.769951e+03
#> [11] -1.002035e+04  1.914717e+03 -9.721572e+03 -3.794461e+04 -1.401549e+04
#> [16] -5.376582e+04 -1.585899e+04 -2.338235e+04  6.239053e+04 -3.574500e+02
#> [21] -4.587927e+04  1.723604e+04 -4.514781e+03  9.184026e-02  3.496526e-01
#> [26] -1.477406e+02  4.585057e+03  9.153647e+03  1.929387e+04 -1.116512e+04
```

```
#> [31] -1.166149e+04  8.079451e+02  3.627369e+03 -9.835680e+03  1.357777e+04
#> [36]  1.487742e+04  3.880562e+04 -1.708858e+03  2.775844e+04  2.527415e+04
#> [41] -3.932957e+04  3.548123e+04 -1.116341e+04  1.630910e+04 -9.789381e+02
#> [46] -2.011250e+04  2.699657e+04 -4.744393e+04  2.753347e+04  2.834356e+04
```

37.2.4 INLA

接下来，介绍完整版的近似贝叶斯推断方法 INLA — 集成嵌套拉普拉斯近似 (Integrated Nested Laplace Approximations，简称 INLA) (Rue, Martino, 和 Chopin 2009)。根据研究区域的边界构造非凸的内外边界，处理边界效应。

```
library(INLA)
library(splancs)
# 构造非凸的边界
boundary <- list(
  inla.nonconvex.hull(
    points = as.matrix(rongelap_coastline[,c("cX", "cY")]),
    convex = 100, concave = 150, resolution = 100),
  inla.nonconvex.hull(
    points = as.matrix(rongelap_coastline[,c("cX", "cY")]),
    convex = 200, concave = 200, resolution = 200)
)
```

根据研究区域的情况构造网格，边界内部三角网格最大边长为 300，边界外部最大边长为 600，边界外凸出距离为 100 米。

```
# 构造非凸的网格
mesh <- inla.mesh.2d(
  loc = as.matrix(rongelap[, c("cX", "cY")]), offset = 100,
  max.edge = c(300, 600), boundary = boundary
)
```

构建 SPDE，指定自协方差函数为指数型，则 $\nu = 1/2$ ，因是二维平面，则 $d = 2$ ，根据 $\alpha = \nu + d/2$ ，从而 `alpha = 3/2` 。

```
spde <- inla.spde2.matern(mesh = mesh, alpha = 3/2, constr = TRUE)
```

生成 SPDE 模型的指标集，也是随机效应部分。

```
indexs <- inla.spde.make.index(name = "s", n.spde = spde$n.spde)
lengths(indexs)
```

```
#>       s s.group  s.repl
#>     691     691     691
```

©黄湘云

投影矩阵，三角网格和采样点坐标之间的投影。观测数据 rongelap 和未采样待预测的位置数据 rongelap_grid_df

```
# 观测位置投影到三角网格上
A <- inla.spde.make.A(mesh = mesh, loc = as.matrix(rongelap[, c("cX", "cY")]) )
# 预测位置投影到三角网格上
coop <- as.matrix(rongelap_grid_df[, c("cX", "cY")])
Ap <- inla.spde.make.A(mesh = mesh, loc = coop)
# 1612 个预测位置
dim(Ap)
```

```
#> [1] 1612  691
```

准备观测数据和预测位置，构造一个 INLA 可以使用的数据栈 Data Stack。

```
# 在采样点的位置上估计 estimation stk.e
stk.e <- inla.stack(
  tag = "est",
  data = list(y = rongelap$counts, E = rongelap$time),
  A = list(rep(1, 157), A),
  effects = list(data.frame(b0 = 1), s = indexs)
)

# 在新生成的位置上预测 prediction stk.p
stk.p <- inla.stack(
  tag = "pred",
  data = list(y = NA, E = NA),
  A = list(rep(1, 1612), Ap),
  effects = list(data.frame(b0 = 1), s = indexs)
)

# 合并数据 stk.full has stk.e and stk.p
stk.full <- inla.stack(stk.e, stk.p)
```

指定响应变量与漂移项、联系函数、模型公式。

```
# 精简输出
inla.setOption(short.summary = TRUE)
# 模型拟合
res <- inla(formula = y ~ 0 + b0 + f(s, model = spde),
  data = inla.stack.data(stk.full),
  E = E, # E 已知漂移项
  control.family = list(link = "log"),
```

©黄湘云

```
  control.predictor = list(
    compute = TRUE,
    link = 1, # 与 control.family 联系函数相同
    A = inla.stack.A(stk.full)
  ),
  control.compute = list(
    cpo = TRUE,
    waic = TRUE, # WAIC 统计量 通用信息准则
    dic = TRUE   # DIC 统计量 偏差信息准则
  ),
  family = "poisson"
)
# 模型输出
summary(res)
```

```
#> Fixed effects:
#>     mean    sd 0.025quant 0.5quant 0.975quant  mode kld
#> b0 1.828 0.061      1.706    1.828      1.948 1.828   0
#>
#> Model hyperparameters:
#>               mean    sd 0.025quant 0.5quant 0.975quant  mode
#> Theta1 for s  2.00 0.062       1.88     2.00       2.12  2.00
#> Theta2 for s -4.85 0.130      -5.11    -4.85      -4.59 -4.85
#>
#> Deviance Information Criterion (DIC) ...............: 1834.57
#> Deviance Information Criterion (DIC, saturated) ....: 314.90
#> Effective number of parameters .....................: 156.46
#>
#> Watanabe-Akaike information criterion (WAIC) ...: 1789.32
#> Effective number of parameters .................: 80.06
#>
#>  is computed
```

- `kld` 表示 Kullback-Leibler divergence (KLD) 它的值描述标准高斯分布与 Simplified Laplace Approximation 之间的差别，值越小越表示拉普拉斯的近似效果好。
- DIC 和 WAIC 指标都是评估模型预测表现的。另外，还有两个量计算出来了，但是没有显示，分别是 CPO 和 PIT 。CPO 表示 Conditional Predictive Ordinate (CPO)，PIT 表示 Probability Integral Transforms (PIT) 。

固定效应（截距）和超参数部分

```
# 截距
```

```
res$summary.fixed
```

```
#>        mean         sd 0.025quant 0.5quant 0.975quant     mode          kld
#> b0 1.828027 0.06147355   1.706422 1.828283   1.948169 1.828279 1.782527e-08
```

```
# 超参数
res$summary.hyperpar
```

```
#>                    mean         sd 0.025quant  0.5quant 0.975quant      mode
#> Theta1 for s  2.000684 0.06235093   1.876511  2.001169   2.122007  2.003210
#> Theta2 for s -4.851258 0.12973843  -5.105068 -4.851807  -4.594242 -4.854095
```

提取预测数据，并整理数据。

```
# 预测值对应的指标集合
index <- inla.stack.index(stk.full, tag = "pred")$data
# 提取预测结果，后验均值
# pred_mean <- res$summary.fitted.values[index, "mean"]
# 95% 预测下限
# pred_ll <- res$summary.fitted.values[index, "0.025quant"]
# 95% 预测上限
# pred_ul <- res$summary.fitted.values[index, "0.975quant"]
# 整理数据
rongelap_grid_df$ypred <- res$summary.fitted.values[index, "mean"]
# 预测值数据
rongelap_grid_sf <- st_as_sf(rongelap_grid_df, coords = c("cX", "cY"), dim = "XY")
rongelap_grid_stars <- st_rasterize(rongelap_grid_sf, nx = 150, ny = 75)
rongelap_stars <- st_crop(x = rongelap_grid_stars, y = rongelap_coastline_sfp)
```

最后，类似之前 mgcv 建模的最后一步，将 INLA 的预测结果绘制出来。

```
ggplot() +
  geom_stars(data = rongelap_stars, aes(fill = ypred), na.action = na.omit) +
  geom_sf(data = rongelap_coastline_sfp, fill = NA, color = "gray50", linewidth = 0.5) +
  scale_fill_viridis_c(option = "C") +
  theme_bw() +
  labs(x = "横坐标（米）", y = "纵坐标（米）", fill = "预测值")
```

©黄湘云

图 37.8: 核辐射强度的预测分布

37.3 案例：城市土壤重金属污染

介绍多元地统计（Multivariate geostatistics）建模分析与 INLA 实现。分析某城市地表土壤重金属污染情况，找到污染最严重的地方，即寻找重金属污染的源头。

```
city_df <- readRDS(file = "data/cumcm2011A.rds")
library(sf)
city_sf <- st_as_sf(city_df, coords = c("x(m)", "y(m)"), dim = "XY")
city_sf
```

```
#> Simple feature collection with 319 features and 12 fields
#> Geometry type: POINT
#> Dimension:     XY
#> Bounding box:  xmin: 0 ymin: 0 xmax: 28654 ymax: 18449
#> CRS:           NA
#> First 10 features:
#>    编号 功能区 海拔(m) 功能区名称 As (μg/g) Cd (ng/g) Cr (μg/g) Cu (μg/g)
#> 1     1      4       5     交通区      7.84     153.8     44.31     20.56
#> 2     2      4      11     交通区      5.93     146.2     45.05     22.51
#> 3     3      4      28     交通区      4.90     439.2     29.07     64.56
#> 4     4      2       4     工业区      6.56     223.9     40.08     25.17
#> 5     5      4      12     交通区      6.35     525.2     59.35    117.53
```

©黄湘云

```
#> 6      6       2        6      工业区      14.08     1092.9      67.96     308.61
#> 7      7       4       15      交通区       8.94      269.8      95.83      44.81
#> 8      8       2        7      工业区       9.62     1066.2     285.58    2528.48
#> 9      9       4       22      交通区       7.41     1123.9      88.17     151.64
#> 10    10       4        7      交通区       8.72      267.1      65.56      29.65
#>     Hg (ng/g) Ni (μg/g) Pb (μg/g) Zn (μg/g)          geometry
#> 1         266      18.2     35.38     72.35    POINT (74 781)
#> 2          86      17.2     36.18     94.59  POINT (1373 731)
#> 3         109      10.6     74.32    218.37 POINT (1321 1791)
#> 4         950      15.4     32.28    117.35    POINT (0 1787)
#> 5         800      20.2    169.96    726.02 POINT (1049 2127)
#> 6        1040      28.2    434.80    966.73 POINT (1647 2728)
#> 7         121      17.8     62.91    166.73 POINT (2883 3617)
#> 8       13500      41.7    381.64   1417.86 POINT (2383 3692)
#> 9       16000      25.8    172.36    926.84 POINT (2708 2295)
#> 10         63      21.7     36.94    100.41 POINT (2933 1767)
```

```
ggplot(data = city_sf) +
  geom_sf(aes(color = `功能区名称`, size = `海拔(m)`)) +
  theme_classic()
```

图 37.9: 某城市的地形

类似章节 37.2.1 ，下面根据数据构造城市边界以及对城市区域划分，以便预测城市中其它地方的重金属浓度。

仅供学习


```r
# 由点构造多边形
city_sfp <- st_cast(st_combine(st_geometry(city_sf)), "POLYGON")
# 由点构造凸包
city_hull <- st_convex_hull(st_geometry(city_sfp))
# 添加缓冲区作为城市边界
city_buffer <- st_buffer(city_hull, dist = 1000)
# 构造带边界约束的网格
city_grid <- st_make_grid(city_buffer, n = c(150, 75))
# 将 sfc 类型转化为 sf 类型
city_grid <- st_as_sf(city_grid)
city_buffer <- st_as_sf(city_buffer)
city_grid <- city_grid[city_buffer, op = st_intersects]
# 计算网格中心点坐标
city_grid_centroid <- st_centroid(city_grid)
# 共计 8494 个预测点
city_grid_df <- as.data.frame(st_coordinates(city_grid_centroid))
```

城市边界线

```r
ggplot() +
  geom_sf(data = city_sf, aes(color = `功能区名称`, size = `海拔(m)`)) +
  geom_sf(data = city_hull, fill = NA) +
  geom_sf(data = city_buffer, fill = NA) +
  theme_classic()
```

©黄湘云

图 37.10: 某城市边界线

根据横、纵坐标和海拔数据，通过高斯过程回归（当然可以用其他办法，这里仅做示意）拟合获得城市其他位置的海拔，绘制等高线图，一目了然地获得城市地形信息。

```
library(mgcv)
# 提取部分数据
city_topo <- subset(city_df, select = c("x(m)", "y(m)", "海拔(m)"))
colnames(city_topo) <- c("x", "y", "z")
# 高斯过程拟合
fit_city_mgcv <- gam(z ~ s(x, y, bs = "gp", k = 50),
  data = city_topo, family = gaussian(link = "identity")
)
# 绘制等高线图
# vis.gam(fit_city_mgcv, color = "cm", plot.type = "contour", n.grid = 50)
colnames(city_grid_df) <- c("x", "y")
# 预测
city_grid_df$zpred <- as.vector(predict(fit_city_mgcv, newdata = city_grid_df, type = "response"))
# 转化数据
city_grid_sf <- st_as_sf(city_grid_df, coords = c("x", "y"), dim = "XY")
library(stars)
city_stars <- st_rasterize(city_grid_sf, nx = 150, ny = 75)

ggplot() +
```

©黄湘云

```
  geom_stars(data = city_stars, aes(fill = zpred), na.action = na.omit) +
  geom_sf(data = city_buffer, fill = NA, color = "gray50", linewidth = .5) +
  scale_fill_viridis_c(option = "C") +
  theme_bw() +
  labs(x = "横坐标（米）", y = "纵坐标（米）", fill = "海拔（米）")
```

图 37.11: 某城市地形图

```
library(ggplot2)
ggplot(data = city_sf) +
  geom_sf(aes(color = `功能区名称`, size = `As (μg/g)`)) +
  theme_classic()
ggplot(data = city_sf) +
  geom_sf(aes(color = `功能区名称`, size = `Cd (ng/g)`)) +
  theme_classic()
```

为了便于建模，对数据做标准化处理。

```
# 根据背景值将各个重金属浓度列进行转化
city_sf <- within(city_sf, {
  `As (μg/g)` <- (`As (μg/g)` - 3.6) / 0.9
  `Cd (ng/g)` <- (`Cd (ng/g)` - 130) / 30
  `Cr (μg/g)` <- (`Cr (μg/g)` - 31) / 9
  `Cu (μg/g)` <- (`Cu (μg/g)` - 13.2) / 3.6
  `Hg (ng/g)` <- (`Hg (ng/g)` - 35) / 8
```

©黄湘云

(a) 重金属砷 As

(b) 重金属镉 Cd

图 37.12: 重金属砷 As 和镉 Cd 的浓度分布

© 黄湘云


```
  `Ni (μg/g)` <- (`Ni (μg/g)` - 12.3) / 3.8
  `Pb (μg/g)` <- (`Pb (μg/g)` - 31) / 6
  `Zn (μg/g)` <- (`Zn (μg/g)` - 69) / 14
})
```

当我们逐一检查各个重金属的浓度分布时，发现重金属汞 Hg 在四个地方的浓度极高，暗示着如果数据采集没有问题，那么这几个地方很可能是污染源。

```
ggplot(data = city_sf) +
  geom_sf(aes(color = `功能区名称`, size = `Hg (ng/g)`)) +
  theme_classic()
```

图 37.13: 重金属汞 Hg 的浓度分布

37.3.1 mgcv

mgcv 包用于多元空间模型中样条参数估计和选择 (Simon N. Wood, Pya, 和 Säfken 2016)。

```
# ?mvn
```

37.3.2 INLA

INLA 包用于多元空间模型的贝叶斯推断 (Palmí-Perales 等 2022) 。

©黄湘云

第三十八章　高斯过程回归

本章主要内容分三大块，分别是多元正态分布、二维高斯过程和高斯过程回归。根据高斯过程的定义，我们知道多元正态分布和高斯过程有紧密的联系，首先介绍多元正态分布的定义、随机数模拟和分布拟合。二维高斯过程很有代表性，常用于空间统计中，而一维高斯过程常用于时间序列分析中，因而，介绍高斯过程的定义，二维高斯过程的模拟和参数拟合。在后续的高斯过程回归中，以朗格拉普岛的核辐射数据为例，建立泊松空间广义线性混合效应模型（响应变量非高斯的高斯过程回归模型），随机过程看作一组存在相关性的随机变量，这一组随机变量视为模型中的随机效应。

38.1　多元正态分布

设随机向量 $\boldsymbol{X} = (X_1, X_2, \cdots, X_p)^\top$ 服从多元正态分布 $\mathrm{MVN}(\boldsymbol{\mu}, \Sigma)$ ，其联合密度函数如下

$$p(\boldsymbol{x}) = (2\pi)^{-\frac{p}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right\},\ \boldsymbol{x} \in \mathbb{R}^p$$

其中，协方差矩阵 Σ 是正定的，其 Cholesky 分解为 $\Sigma = CC^\top$ ，这里 C 为下三角矩阵。设 $\boldsymbol{Z} = (Z_1, Z_2, \cdots, Z_p)^\top$ 服从 p 元标准正态分布 $\mathrm{MVN}(\mathbf{0}, I)$ ，则 $\boldsymbol{X} = \boldsymbol{\mu} + C\boldsymbol{Z}$ 服从多元正态分布 $\mathrm{MVN}(\boldsymbol{\mu}, \Sigma)$ 。

38.1.1　多元正态分布模拟

可以用 Stan 函数 `multi_normal_cholesky_rng` 生成随机数模拟多元正态分布。

```
data {
  int<lower=1> N;
  int<lower=1> D;
  vector[D] mu;
  matrix[D, D] Sigma;
}
```

©黄湘云

```
transformed data {
  matrix[D, D] L_K = cholesky_decompose(Sigma);
}
parameters {
}
model {
}
generated quantities {
  array[N] vector[D] yhat;
  for (n in 1:N){
    yhat[n] = multi_normal_cholesky_rng(mu, L_K);
  }
}
```

上述代码块可以同时模拟多组服从多元正态分布的随机数。其中，参数块 parameters 和模型块 model 是空白的，这是因为模拟随机数不涉及模型推断，只是采样。核心部分 generated quantities 代码块负责生成随机数。

```
# 给定二元正态分布的参数值
multi_normal_d <- list(
  N = 1, # 一组随机数
  D = 2, # 维度
  mu = c(3, 2), # 均值向量
  Sigma = rbind(c(4, 1), c(1, 1)) # 协方差矩阵
)
library(cmdstanr)
# 编译多元正态分布模型
mod_multi_normal <- cmdstan_model(
  stan_file = "code/multi_normal_simu.stan",
  compile = TRUE, cpp_options = list(stan_threads = TRUE)
)
```

抽样生成 1000 个服从二元正态分布的随机数。

```
simu_multi_normal <- mod_multi_normal$sample(
  data = multi_normal_d,
  iter_warmup = 500,     # 每条链预处理迭代次数
  iter_sampling = 1000, # 样本量
  chains = 1,            # 马尔科夫链的数目
  parallel_chains = 1,  # 指定 CPU 核心数，可以给每条链分配一个
  threads_per_chain = 1, # 每条链设置一个线程
  show_messages = FALSE, # 不显示迭代的中间过程
```

```
  refresh = 0,        # 不显示采样的进度
  fixed_param = TRUE, # 固定参数
  seed = 20232023     # 设置随机数种子，不要使用 set.seed() 函数
)
```

值得注意，这里，不需要设置参数初始值，但要设置 fixed_param = TRUE，表示根据模型生成模拟数据。

```
# 原始数据
simu_multi_normal$draws(variables = "yhat", format = "array")

# A draws_array: 1000 iterations, 1 chains, and 2 variables
, , variable = yhat[1,1]

         chain
iteration   1
        1 2.6
        2 5.3
        3 1.4
        4 3.1
        5 7.6

, , variable = yhat[1,2]

         chain
iteration    1
        1 0.65
        2 3.70
        3 4.36
        4 1.85
        5 2.32

# ... with 995 more iterations

# 数据概览
simu_multi_normal$summary(.num_args = list(sigfig = 4, notation = "dec"))

# A tibble: 2 x 10
  variable   mean median    sd    mad      q5   q95  rhat ess_bulk ess_tail
  <chr>     <dbl>  <dbl> <dbl>  <dbl>   <dbl> <dbl> <dbl>    <dbl>    <dbl>
1 yhat[1,1] 3.076  2.979 2.038 1.916  -0.3290 6.535 1.001   1140.    1023.
2 yhat[1,2] 1.990  1.964 1.018 0.9765  0.2323 3.680 1.002    966.5    982.1
```

© 王建平

以生成第一个服从二元正态分布的随机数（样本点）为例，这个随机数是通过采样获得的，采样过程中产生一个采样序列，采样序列的轨迹和分布如下：

```
library(ggplot2)
library(bayesplot)
mcmc_trace(simu_multi_normal$draws(c("yhat[1,1]", "yhat[1,2]")),
  facet_args = list(
    labeller = ggplot2::label_parsed, strip.position = "top", ncol = 1
  )
) + theme_bw(base_size = 12)

mcmc_dens(simu_multi_normal$draws(c("yhat[1,1]", "yhat[1,2]")),
  facet_args = list(
    labeller = ggplot2::label_parsed, strip.position = "top", ncol = 1
  )
) + theme_bw(base_size = 12)
```

图 38.1: 采样序列的轨迹和分布

© 黄湘云

图 38.2: 采样序列的轨迹和分布

这就是一组来自二元正态分布的随机数。

```
mcmc_scatter(simu_multi_normal$draws(c("yhat[1,1]", "yhat[1,2]"))) +
  theme_bw(base_size = 12) +
  labs(x = expression(x[1]), y = expression(x[2]))
```

© 黄湘云

图 38.3: 生成二元正态分布的随机数

提取采样数据，整理成矩阵。

```
# 抽取原始采样数据
yhat <- simu_multi_normal$draws(c("yhat[1,1]", "yhat[1,2]"))
# 合并多条链
yhat_mean <- apply(yhat, c(1, 3), mean)
# 整理成二维矩阵
x <- as.matrix(yhat_mean)
# 样本均值
colMeans(x)
```

```
yhat[1,1] yhat[1,2]
 3.076281  1.990465
```

```
# 样本方差-协方差矩阵
var(x)
```

```
          yhat[1,1] yhat[1,2]
yhat[1,1]  4.152065  1.031544
yhat[1,2]  1.031544  1.035985
```

38.1.2 多元正态分布拟合

一般地，协方差矩阵的 Cholesky 分解的矩阵表示如下：

$$
\begin{aligned}
\Sigma &= \begin{bmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 & \rho_{13}\sigma_1\sigma_3 \\ \rho_{12}\sigma_1\sigma_2 & \sigma_2^2 & \rho_{23}\sigma_2\sigma_3 \\ \rho_{13}\sigma_1\sigma_3 & \rho_{23}\sigma_2\sigma_3 & \sigma_3^2 \end{bmatrix} \\
&= \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} \underbrace{\begin{bmatrix} 1 & \rho_{12} & \rho_{13} \\ \rho_{12} & 1 & \rho_{23} \\ \rho_{13} & \rho_{23} & 1 \end{bmatrix}}_{R} \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} \\
&= \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} \underbrace{L_u L_u^\top}_{R} \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix}
\end{aligned}
$$

```
data {
  int<lower=1> N; // number of observations
  int<lower=1> K; // dimension of observations
  array[N] vector[K] y; // observations: a list of N vectors (each has K elements)
}
parameters {
  vector[K] mu;
  cholesky_factor_corr[K] Lcorr; // cholesky factor (L_u matrix for R)
  vector<lower=0>[K] sigma;
}
transformed parameters {
  corr_matrix[K] R; // correlation matrix
  cov_matrix[K] Sigma; // VCV matrix
  R = multiply_lower_tri_self_transpose(Lcorr); // R = Lcorr * Lcorr'
  Sigma = quad_form_diag(R, sigma); // quad_form_diag: diag_matrix(sig) * R * diag_matrix(sig)
}
model {
  sigma ~ cauchy(0, 5); // prior for sigma
  Lcorr ~ lkj_corr_cholesky(2.0); // prior for cholesky factor of a correlation matrix
  y ~ multi_normal(mu, Sigma);
}
```

代码中，核心部分是关于多元正态分布的协方差矩阵的参数化，先将协方差矩阵中的方差和相关矩阵剥离，然后利用 Cholesky 分解将相关矩阵分解。在 Stan 里，这是一套高效的组合。

- 类型 `cholesky_factor_corr` 表示相关矩阵的 Cholesky 分解后的矩阵 L_u

©黄湘云

- 类型 `corr_matrix` 表示相关矩阵 R 。
- 类型 `cov_matrix` 表示协方差矩阵 Σ 。
- 函数 `lkj_corr_cholesky` 为相关矩阵 Cholesky 分解后的矩阵 L_u 服从的分布，详见 Cholesky LKJ correlation distribution。函数名中的 `lkj` 是以三个人的人名的首字母命名的 Lewandowski, Kurowicka, and Joe 2009。
- 函数 `multiply_lower_tri_self_transpose` 为下三角矩阵与它的转置的乘积，详见 Correlation Matrix Distributions。
- 函数 `multi_normal` 为多元正态分布的抽样语句，详见 Multivariate normal distribution。

矩阵 L_u 是相关矩阵 R 的 Cholesky 分解的结果，在贝叶斯框架内，参数都是随机的，相关矩阵是一个随机矩阵，矩阵 L_u 是一个随机矩阵，它的分布用 Stan 代码表示为如下：

```
L ~ lkj_corr_cholesky(2.0); # implies L * L' ~ lkj_corr(2.0);
```

LKJ 分布有一个参数 η ，此处 $\eta = 2$ ，意味着变量之间的相关性较弱，LKJ 分布的概率密度函数正比于相关矩阵的行列式的 $\eta - 1$ 次幂 $(\det R)^{\eta-1}$，LKJ 分布的详细说明见Lewandowski-Kurowicka-Joe (LKJ) distribution。

有了上面的背景知识，下面先在 R 环境中模拟一组来自多元正态分布的样本。

```r
set.seed(20232023)
# 均值
mu <- c(1, 2, -5)
# 相关矩阵 (R)
R <- matrix(c(
  1, 0.7, 0.2,
  0.7, 1, -0.5,
  0.2, -0.5, 1
), 3)
# sd1 = 0.5, sd2 = 1.2, sd3 = 2.3
sigmas <- c(0.5, 1.2, 2.3)
# 方差-协方差矩阵
Sigma <- diag(sigmas) %*% R %*% diag(sigmas)
# 模拟 1000 个样本数据
dat <- MASS::mvrnorm(1000, mu = mu, Sigma = Sigma)
```

根据 1000 个样本点，估计多元正态分布的均值参数和方差协方差参数。

```r
# 来自多元正态分布的一组观测数据
multi_normal_chol_d <- list(
  N = 1000, # 样本量
  K = 3,    # 三维
  y = dat
```

©草稿 仅供内部使用

```
)
# 编译多元正态分布模型
mod_multi_normal_chol <- cmdstan_model(
  stan_file = "code/multi_normal_fitted.stan",
  compile = TRUE, cpp_options = list(stan_threads = TRUE)
)
# 拟合多元正态分布模型
fit_multi_normal <- mod_multi_normal_chol$sample(
  data = multi_normal_chol_d,
  iter_warmup = 500,    # 每条链预处理迭代次数
  iter_sampling = 1000, # 每条链采样次数
  chains = 2,           # 马尔科夫链的数目
  parallel_chains = 1,  # 指定 CPU 核心数
  threads_per_chain = 1,  # 每条链设置一个线程
  show_messages = FALSE,  # 不显示迭代的中间过程
  refresh = 0,            # 不显示采样的进度
  seed = 20232023     # 设置随机数种子
)
```

均值向量 $\boldsymbol{\mu}$ 和协方差矩阵 Σ 估计结果如下：

```
fit_multi_normal$summary(c("mu", "Sigma"), .num_args = list(sigfig = 3, notation = "dec"))
```

```
# A tibble: 12 x 10
   variable     mean median     sd    mad     q5    q95  rhat ess_bulk ess_tail
   <chr>       <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <dbl>    <dbl>    <dbl>
 1 mu[1]       0.985  0.985 0.0156 0.0151  0.958   1.01  0.999    1966.    1379.
 2 mu[2]       1.92   1.92  0.0376 0.0377  1.85    1.98  1.00     1471.     937.
 3 mu[3]      -4.89  -4.89  0.0723 0.0724 -5.00   -4.76  1.00     1940.    1019.
 4 Sigma[1,1]  0.255  0.255 0.0109 0.0107  0.238   0.275 1.00     1893.    1105.
 5 Sigma[2,1]  0.419  0.418 0.0219 0.0215  0.385   0.457 1.00     1470.    1216.
 6 Sigma[3,1]  0.250  0.249 0.0383 0.0373  0.188   0.315 1.00     1862.    1157.
 7 Sigma[1,2]  0.419  0.418 0.0219 0.0215  0.385   0.457 1.00     1470.    1216.
 8 Sigma[2,2]  1.42   1.42  0.0629 0.0643  1.32    1.52  1.00     1594.    1298.
 9 Sigma[3,2] -1.35  -1.34  0.0986 0.0980 -1.51   -1.19  1.00     1939.    1531.
10 Sigma[1,3]  0.250  0.249 0.0383 0.0373  0.188   0.315 1.00     1862.    1157.
11 Sigma[2,3] -1.35  -1.34  0.0986 0.0980 -1.51   -1.19  1.00     1939.    1531.
12 Sigma[3,3]  5.33   5.32  0.241  0.240   4.96    5.73  1.00     1858.    1496.
```

均值向量 $\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3)^\top$ 各个分量及其两两相关性，如下图所示。

```
mcmc_pairs(
  fit_multi_normal$draws(c("mu[1]", "mu[2]", "mu[3]")),
```

© 黄湘云


```
  diag_fun = "dens", off_diag_fun = "hex"
)
```

图 38.4: 三元正态分布

38.2 二维高斯过程

高斯过程定义

38.2.1 二维高斯过程模拟

二维高斯过程 $\mathcal{S}$ 的均值向量为 0 向量，自协方差函数为指数型，如下

贝叶斯统计建模 ©

$$\mathsf{Cov}\{S(x_i), S(x_j)\} = \sigma^2 \exp\left(-\frac{\|x_i - x_j\|_2}{\phi}\right)$$

其中，不妨设参数 $\sigma = 10, \phi = 1$ 。模拟高斯过程的 Stan 代码如下

```
data {
  int<lower=1> N;
  int<lower=1> D;
  array[N] vector[D] X;
  vector[N] mu;
  real<lower=0> sigma;
  real<lower=0> phi;
}
transformed data {
  real delta = 1e-9;
  matrix[N, N] L;
  matrix[N, N] K = gp_exponential_cov(X, sigma, phi) + diag_matrix(rep_vector(delta, N));
  L = cholesky_decompose(K);
}
parameters {
  vector[N] eta;
}
model {
  eta ~ std_normal();
}
generated quantities {
  vector[N] y;
  y = mu + L * eta;
}
```

在二维规则网格上采样，采样点数量为 225。

```
n <- 15
gaussian_process_d <- list(
  N = n^2,
  D = 2,
  mu = rep(0, n^2),
  sigma = 10,
  phi = 1,
  X = expand.grid(x1 = 1:n / n, x2 = 1:n / n)
)
# 编译二维高斯过程模型
```

© 统计学习

```
mod_gaussian_process_simu <- cmdstan_model(
  stan_file = "code/gaussian_process_simu.stan",
  compile = TRUE, cpp_options = list(stan_threads = TRUE)
)
```

模拟 1 个样本，因为是模拟数据，不需要设置多条链。

```
fit_multi_normal_gp <- mod_gaussian_process_simu$sample(
  data = gaussian_process_d,
  iter_warmup = 500,        # 每条链预处理迭代次数
  iter_sampling = 1000,     # 样本量
  chains = 1,               # 马尔科夫链的数目
  parallel_chains = 1,     # 指定 CPU 核心数
  threads_per_chain = 1,   # 每条链设置一个线程
  show_messages = FALSE,   # 不显示迭代的中间过程
  refresh = 0,             # 不显示采样的进度
  seed = 20232023          # 设置随机数种子
)
```

位置 1 和 2 处的随机变量的迭代轨迹，均值为 0 ，标准差 10 左右。

```
mcmc_trace(fit_multi_normal_gp$draws(c("y[1]", "y[2]")),
  facet_args = list(
    labeller = ggplot2::label_parsed,
    strip.position = "top", ncol = 1
  )
) + theme_bw(base_size = 12)
```

© 黄湘云

图 38.5: 位置 1 和 2 处的迭代轨迹

位置 1 处的随机变量及其分布

```
y1 <- fit_multi_normal_gp$draws(c("y[1]"), format = "draws_array")
# 合并链条结果
y1_mean <- apply(y1, c(1, 3), mean)
# y[1] 的方差
var(y1_mean)
```

```
         y[1]
y[1] 103.9116
```

```
# y[1] 的标准差
sd(y1_mean)
```

```
[1] 10.1937
```

100 次迭代获得 100 个样本点，每次迭代采集一个样本点，每个样本点是一个 225 维的向量。

```
# 抽取原始的采样数据
y_array <- fit_multi_normal_gp$draws(variables = "y", format = "array")
# 合并链条
y_mean <- apply(y_array, c(1, 3), mean)
```

从 100 次迭代中任意提取某一个样本点，比如预采样之后的第一次下迭代的结果，接着整理数据。

```
# 整理数据
sim_gp_data <- cbind.data.frame(gaussian_process_d$X, ysim = y_mean[1, ])
```

©黄文海

绘制二维高斯过程图形。

```
ggplot(data = sim_gp_data, aes(x = x1, y = x2)) +
  geom_point(aes(color = ysim)) +
  scale_color_distiller(palette = "Spectral") +
  theme_bw() +
  labs(x = expression(x[1]), y = expression(x[2]))
```

图 38.6: 二维高斯过程

38.2.2 二维高斯过程拟合

二维高斯过程拟合代码如下

```
data {
  int<lower=1> N;
  int<lower=1> D;
  array[N] vector[D] x;
  vector[N] y;
}
transformed data {
  real delta = 1e-9;
  vector[N] mu = rep_vector(0, N);
```

北京大学出版社 ©

```
}
parameters {
  real<lower=0> phi;
  real<lower=0> sigma;
}
model {
  matrix[N, N] L_K;
  {
    matrix[N, N] K = gp_exponential_cov(x, sigma, phi) + diag_matrix(rep_vector(delta, N));
    L_K = cholesky_decompose(K);
  }

  phi ~ std_normal();
  sigma ~ std_normal();

  y ~ multi_normal_cholesky(mu, L_K);
}

# 二维高斯过程模型
gaussian_process_d <- list(
  D = 2,
  N = nrow(sim_gp_data), # 观测记录的条数
  x = sim_gp_data[, c("x1", "x2")],
  y = sim_gp_data[, "ysim"]
)

nchains <- 2
set.seed(20232023)
# 给每条链设置不同的参数初始值
inits_gaussian_process <- lapply(1:nchains, function(i) {
  list(
    sigma = runif(1), phi = runif(1)
  )
})

# 编译模型
mod_gaussian_process <- cmdstan_model(
  stan_file = "code/gaussian_process_fitted.stan",
  compile = TRUE, cpp_options = list(stan_threads = TRUE)
)
```

© 统计出版社

```r
# 拟合二维高斯过程
fit_gaussian_process <- mod_gaussian_process$sample(
  data = gaussian_process_d,     # 观测数据
  init = inits_gaussian_process, # 迭代初值
  iter_warmup = 1000,   # 每条链预处理迭代次数
  iter_sampling = 2000, # 每条链总迭代次数
  chains = nchains,     # 马尔科夫链的数目
  parallel_chains = 2,  # 指定 CPU 核心数，可以给每条链分配一个
  threads_per_chain = 2, # 每条链设置一个线程
  show_messages = FALSE, # 不显示迭代的中间过程
  refresh = 0,           # 不显示采样的进度
  seed = 20232023        # 设置随机数种子，不要使用 set.seed() 函数
)
# 诊断
fit_gaussian_process$diagnostic_summary()
```

```
$num_divergent
[1] 0 0

$num_max_treedepth
[1] 0 0

$ebfmi
[1] 1.111623 1.059357
```

输出结果

```r
fit_gaussian_process$summary()
```

```
# A tibble: 3 x 10
  variable     mean   median     sd    mad       q5      q95  rhat ess_bulk
  <chr>       <dbl>    <dbl>  <dbl>  <dbl>    <dbl>    <dbl> <dbl>    <dbl>
1 lp__     -358.    -357.    1.03   0.744  -360.    -357.     1.00    1542.
2 phi         0.540    0.532 0.0752 0.0728    0.432    0.677  1.00    1191.
3 sigma       6.50     6.48  0.415  0.405     5.87     7.25   1.00    1046.
# i 1 more variable: ess_tail <dbl>
```

©黄湘云

38.3 高斯过程回归

38.3.1 模型介绍

朗格拉普岛是位于太平洋上的一个小岛，因美国在比基尼群岛的氢弹核试验受到严重的核辐射影响，数十年之后，科学家登岛采集核辐射强度数据以评估当地居民重返该岛的可能性。朗格拉普岛是一个十分狭长且占地面积只有几平方公里的小岛。

根据 ^{137}Cs 放出伽马射线，在 $n = 157$ 个采样点，分别以时间间隔 t_i 测量辐射量 $y(x_i)$，建立泊松型空间广义线性混合效应模型 (Diggle, Tawn, 和 Moyeed 1998)。

$$
\begin{aligned}
\log\{\lambda(x_i)\} &= \beta + S(x_i) \\
y(x_i) &\sim \text{Poisson}\big(t_i\lambda(x_i)\big)
\end{aligned}
$$

其中，β 表示截距，相当于平均水平，$\lambda(x_i)$ 表示位置 x_i 处的辐射强度，$S(x_i)$ 表示位置 x_i 处的空间效应，$S(x), x \in \mathcal{D} \subset \mathbb{R}^2$ 是二维平稳空间高斯过程 $\mathcal{S}$ 的具体实现。$\mathcal{D}$ 表示研究区域，可以理解为朗格拉普岛，它是二维实平面 $\mathbb{R}^2$ 的子集。

随机过程 $S(x)$ 的自协方差函数常用的有指数型、幂二次指数型（高斯型）和梅隆型，形式如下：

$$
\begin{aligned}
\mathsf{Cov}\{S(x_i), S(x_j)\} &= \sigma^2 \exp\big(-\frac{\|x_i - x_j\|_2}{\phi}\big) \\
\mathsf{Cov}\{S(x_i), S(x_j)\} &= \sigma^2 \exp\big(-\frac{\|x_i - x_j\|_2^2}{2\phi^2}\big) \\
\mathsf{Cov}\{S(x_i), S(x_j)\} &= \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{\|x_i - x_j\|_2}{\phi}\right)^{\nu} K_\nu\left(\sqrt{2\nu}\frac{\|x_i - x_j\|_2}{\phi}\right) \\
K_\nu(x) &= \int_0^\infty \exp(-x\cosh t)\cosh(\nu t)\mathrm{d}t
\end{aligned}
$$

待估参数：代表方差的 σ^2 和代表范围的 ϕ 。当 $\nu = 1/2$ 时，梅隆型退化为指数型。

38.3.2 观测数据

```r
# 加载数据
rongelap <- readRDS(file = "data/rongelap.rds")
rongelap_coastline <- readRDS(file = "data/rongelap_coastline.rds")
# 准备输入数据
rongelap_poisson_d <- list(
  N = nrow(rongelap), # 观测记录的条数
  D = 2, # 2 维坐标
  X = rongelap[, c("cX", "cY")] / 6000, # N x 2 矩阵
  y = rongelap$counts, # 响应变量
  offsets = rongelap$time # 漂移项
```

©贾俊平


```
)
# 准备参数初始化数据
set.seed(20232023)
nchains <- 2 # 2 条迭代链
inits_data_poisson <- lapply(1:nchains, function(i) {
  list(
    beta = rnorm(1), sigma = runif(1),
    phi = runif(1), lambda = rnorm(157)
  )
})
```

38.3.3 预测数据

预测未采样的位置的核辐射强度，根据海岸线数据网格化全岛，以格点代表未采样的位置

```
library(sf)
library(abind)
library(stars)
# 类型转化
rongelap_sf <- st_as_sf(rongelap, coords = c("cX", "cY"), dim = "XY")
rongelap_coastline_sf <- st_as_sf(rongelap_coastline, coords = c("cX", "cY"), dim = "XY")
rongelap_coastline_sfp <- st_cast(st_combine(st_geometry(rongelap_coastline_sf)), "POLYGON")
# 添加缓冲区
rongelap_coastline_buffer <- st_buffer(rongelap_coastline_sfp, dist = 50)
# 构造带边界约束的网格
rongelap_coastline_grid <- st_make_grid(rongelap_coastline_buffer, n = c(150, 75))
# 将 sfc 类型转化为 sf 类型
rongelap_coastline_grid <- st_as_sf(rongelap_coastline_grid)
rongelap_coastline_buffer <- st_as_sf(rongelap_coastline_buffer)
rongelap_grid <- rongelap_coastline_grid[rongelap_coastline_buffer, op = st_intersects]
# 计算网格中心点坐标
rongelap_grid_centroid <- st_centroid(rongelap_grid)
# 共计 1612 个预测点
rongelap_grid_df <- as.data.frame(st_coordinates(rongelap_grid_centroid))
colnames(rongelap_grid_df) <- c("cX", "cY")
```

未采样的位置 rongelap_grid_df

```
head(rongelap_grid_df)
```

```
         cX        cY
1 -5685.942 -3606.997
```

```
2 -5643.145 -3606.997
3 -5600.347 -3606.997
4 -5557.549 -3606.997
5 -5514.751 -3606.997
6 -5471.953 -3606.997
```

朗格拉普岛网格化生成格点

```
ggplot() +
  geom_point(data = rongelap_grid_df, aes(x = cX, y = cY), cex = 0.3) +
  geom_path(data = rongelap_coastline, aes(x = cX, y = cY)) +
  coord_fixed() +
  theme_bw() +
  labs(x = "横坐标（米）", y = "纵坐标（米）")
```

图 38.7: 朗格拉普岛

38.3.4 模型编码

指定各个参数 β, σ, ϕ 的先验分布

贝叶斯©

$$
\begin{aligned}
\beta &\sim \text{std_normal}(0,1)\\
\sigma &\sim \text{inv_gamma}(5,5)\\
\phi &\sim \text{half_std_normal}(0,1)\\
\boldsymbol{\lambda}|\beta,\sigma &\sim \text{multivariate_normal}(\boldsymbol{\beta},\sigma^2\Sigma)\\
\boldsymbol{y}|\boldsymbol{\lambda} &\sim \text{poisson_log}\big(\log(\text{offsets})+\boldsymbol{\lambda}\big)
\end{aligned}
$$

其中，β,σ,ϕ,Σ 的含义同前，λ 代表辐射强度，offsets 代表漂移项，这里是时间段，$\boldsymbol{y}$ 表示观测的辐射粒子数，poisson_log 表示泊松分布的对数参数化，将频率参数 rate 的对数 λ 作为参数，详见 Stan 函数手册中泊松分布的对数函数表示。

```
data {
  int<lower=1> N;
  int<lower=1> D;
  array[N] vector[D] X;
  array[N] int<lower = 0> y;
  vector[N] offsets;
}
transformed data {
  real delta = 1e-12;
  vector[N] log_offsets = log(offsets);
}
parameters {
  real beta;
  real<lower=0> sigma;
  real<lower=0> phi;
  vector[N] lambda;
}
transformed parameters {
  vector[N] mu = rep_vector(beta, N);
}
model {
  matrix[N, N] L_K;
  {
    matrix[N, N] K = gp_exponential_cov(X, sigma, phi) + diag_matrix(rep_vector(delta, N));
    L_K = cholesky_decompose(K);
  }

  beta ~ std_normal();
  sigma ~ inv_gamma(5, 5);
```

©贝叶斯统计

```
  phi ~ std_normal();

  lambda ~ multi_normal_cholesky(mu, L_K);
  y ~ poisson_log(log_offsets + lambda);
}
```

```r
# 编译模型
mod_rongelap_poisson <- cmdstan_model(
  stan_file = "code/rongelap_poisson_processes.stan",
  compile = TRUE, cpp_options = list(stan_threads = TRUE)
)
# 泊松对数模型
fit_rongelap_poisson <- mod_rongelap_poisson$sample(
  data = rongelap_poisson_d,  # 观测数据
  init = inits_data_poisson,  # 迭代初值
  iter_warmup = 500,    # 每条链预处理迭代次数
  iter_sampling = 1000, # 每条链总迭代次数
  chains = nchains,     # 马尔科夫链的数目
  parallel_chains = 2,  # 指定 CPU 核心数，可以给每条链分配一个
  threads_per_chain = 2, # 每条链设置一个线程
  show_messages = FALSE, # 不显示迭代的中间过程
  refresh = 0,           # 不显示采样的进度
  seed = 20232023
)
# 诊断
fit_rongelap_poisson$diagnostic_summary()
```

```
$num_divergent
[1] 0 0

$num_max_treedepth
[1] 0 0

$ebfmi
[1] 1.029395 1.036886
```

```r
# 泊松对数模型
fit_rongelap_poisson$summary(
  variables = c("lp__", "beta", "sigma", "phi"),
  .num_args = list(sigfig = 3, notation = "dec")
)
```

© 贝叶斯


```
# A tibble: 4 x 10
  variable          mean        median      sd       mad            q5           q95
  <chr>            <dbl>         <dbl>   <dbl>     <dbl>         <dbl>         <dbl>
1 lp__      3402193.      3402190       9.40    14.8      3402180       3402210
2 beta            1.77          1.79    0.160    0.125          1.53          1.98
3 sigma           0.640         0.615   0.118    0.0784         0.517         0.861
4 phi             0.0276        0.0239  0.0154   0.00782        0.0147        0.0549
# i 3 more variables: rhat <dbl>, ess_bulk <dbl>, ess_tail <dbl>
```

```r
# 参数的迭代轨迹
mcmc_trace(
  fit_rongelap_poisson$draws(c("sigma", "phi")),
  facet_args = list(
    labeller = ggplot2::label_parsed, strip.position = "top", ncol = 1
  )
) + theme_bw(base_size = 12)
```

图 38.8: σ 和 ϕ 的迭代轨迹

```r
# 参数的后验分布
mcmc_dens(
  fit_rongelap_poisson$draws(c("sigma", "phi")),
  facet_args = list(
    labeller = ggplot2::label_parsed, strip.position = "top", ncol = 1
  )
) + theme_bw(base_size = 12)
```

©黄湘云

图 38.9: σ 和 ϕ 的后验分布

38.3.5 预测分布

核辐射预测模型的 Stan 代码

```
functions {
  vector gp_pred_rng(array[] vector x2,
                     vector lambda,
                     array[] vector x1,
                     real beta,
                     real sigma,
                     real phi,
                     real delta) {
    int N1 = rows(lambda);
    int N2 = size(x2);
    vector[N2] f2;
    {
      matrix[N1, N1] L_K;
      vector[N1] K_div_lambda;
      matrix[N1, N2] k_x1_x2;
      matrix[N1, N2] v_pred;
      vector[N2] f2_mu;
      matrix[N2, N2] cov_f2;
```

汪漫©著

```
      matrix[N2, N2] diag_delta;
      matrix[N1, N1] K;
      K = gp_exponential_cov(x1, sigma, phi);
      L_K = cholesky_decompose(K);
      K_div_lambda = mdivide_left_tri_low(L_K, lambda - beta);
      K_div_lambda = mdivide_right_tri_low(K_div_lambda', L_K)';
      k_x1_x2 = gp_exponential_cov(x1, x2, sigma, phi);
      f2_mu = beta + (k_x1_x2' * K_div_lambda);
      v_pred = mdivide_left_tri_low(L_K, k_x1_x2);
      cov_f2 = gp_exponential_cov(x2, sigma, phi) - v_pred' * v_pred;
      diag_delta = diag_matrix(rep_vector(delta, N2));

      f2 = multi_normal_rng(f2_mu, cov_f2 + diag_delta);
    }
    return f2;
  }
}
data {
  int<lower=1> D;
  int<lower=1> N1;
  array[N1] vector[D] x1;
  array[N1] int<lower = 0> y1;
  vector[N1] offsets1;
  int<lower=1> N2;
  array[N2] vector[D] x2;
  vector[N2] offsets2;
}
transformed data {
  real delta = 1e-12;
  vector[N1] log_offsets1 = log(offsets1);
  vector[N2] log_offsets2 = log(offsets2);

  int<lower=1> N = N1 + N2;
  array[N] vector[D] x;

  for (n1 in 1:N1) {
    x[n1] = x1[n1];
  }
  for (n2 in 1:N2) {
    x[N1 + n2] = x2[n2];
```

北京大学出版社 ©

```
  }
}
parameters {
  real beta;
  real<lower=0> sigma;
  real<lower=0> phi;
  vector[N1] lambda1;
}
transformed parameters {
  vector[N1] mu = rep_vector(beta, N1);
}
model {
  matrix[N1, N1] L_K;
  {
    matrix[N1, N1] K = gp_exponential_cov(x1, sigma, phi) + diag_matrix(rep_vector(delta, N1));
    L_K = cholesky_decompose(K);
  }

  beta ~ std_normal();
  sigma ~ inv_gamma(5, 5);
  phi ~ std_normal();

  lambda1 ~ multi_normal_cholesky(mu, L_K);
  y1 ~ poisson_log(log_offsets1 + lambda1);
}
generated quantities {
  vector[N1] yhat;     // Posterior predictions for each location
  vector[N1] log_lik;  // Log likelihood for each location
  vector[N1] RR1 = log_offsets1 + lambda1;

  for(n in 1:N1) {
    log_lik[n] = poisson_log_lpmf(y1[n] | RR1[n]);
    yhat[n] = poisson_log_rng(RR1[n]);
  }

  vector[N2] ypred;
  vector[N2] lambda2 = gp_pred_rng(x2, lambda1, x1, beta, sigma, phi, delta);
  vector[N2] RR2 = log_offsets2 + lambda2;

  for(n in 1:N2) {
```

©黄湘云

```
    ypred[n] = poisson_log_rng(RR2[n]);
  }
}
```

准备数据、拟合模型

```
# 固定漂移项
rongelap_grid_df$time <- 100
# 对数高斯模型
rongelap_poisson_pred_d <- list(
  D = 2,
  N1 = nrow(rongelap), # 观测记录的条数
  x1 = rongelap[, c("cX", "cY")] / 6000,
  y1 = rongelap[, "counts"],
  offsets1 = rongelap[, "time"],
  N2 = nrow(rongelap_grid_df), # 2 维坐标
  x2 = rongelap_grid_df[, c("cX", "cY")] / 6000,
  offsets2 = rongelap_grid_df[, "time"]
)
# 迭代链数目
nchains <- 2
# 给每条链设置不同的参数初始值
inits_data_poisson_pred <- lapply(1:nchains, function(i) {
  list(
    beta = rnorm(1), sigma = runif(1),
    phi = runif(1), lambda = rnorm(157)
  )
})
# 编译模型
mod_rongelap_poisson_pred <- cmdstan_model(
  stan_file = "code/rongelap_poisson_pred.stan",
  compile = TRUE, cpp_options = list(stan_threads = TRUE)
)
# 泊松模型
fit_rongelap_poisson_pred <- mod_rongelap_poisson_pred$sample(
  data = rongelap_poisson_pred_d,   # 观测数据
  init = inits_data_poisson_pred,   # 迭代初值
  iter_warmup = 500,            # 每条链预处理迭代次数
  iter_sampling = 1000,         # 每条链总迭代次数
  chains = nchains,             # 马尔科夫链的数目
  parallel_chains = 2,      # 指定 CPU 核心数，可以给每条链分配一个
```

©贾俊平

```
  threads_per_chain = 2,     # 每条链设置一个线程
  show_messages = FALSE,     # 不显示迭代的中间过程
  refresh = 0,               # 不显示采样的进度
  seed = 20232023            # 设置随机数种子，不要使用 set.seed() 函数
)
# 诊断信息
fit_rongelap_poisson_pred$diagnostic_summary()
```

```
$num_divergent
[1] 0 0

$num_max_treedepth
[1] 0 0

$ebfmi
[1] 0.9812227 1.0499278
```

参数的后验估计

```
fit_rongelap_poisson_pred$summary(variables = c("beta", "sigma", "phi"))
```

```
# A tibble: 3 x 10
  variable   mean median    sd     mad     q5    q95  rhat ess_bulk ess_tail
  <chr>     <dbl>  <dbl> <dbl>   <dbl>  <dbl>  <dbl> <dbl>    <dbl>    <dbl>
1 beta     1.73   1.79   0.365 0.115   1.46   1.96    1.01     280.     74.7
2 sigma    0.687  0.616  0.350 0.0840  0.513  0.938   1.01     205.     55.5
3 phi      0.0415 0.0236 0.105 0.00846 0.0141 0.0639  1.01     196.     56.2
```

模型评估 LOO-CV

```
fit_rongelap_poisson_pred$loo(variables = "log_lik", cores = 2)
```

```
Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.

Computed from 2000 by 157 log-likelihood matrix

         Estimate  SE
elpd_loo   -934.0 4.0
p_loo       119.6 2.8
looic      1868.0 8.1
------
Monte Carlo SE of elpd_loo is NA.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
```

©贾俊平

```
(-Inf, 0.5]   (good)       1    0.6%   218
 (0.5, 0.7]   (ok)        23   14.6%   49
   (0.7, 1]   (bad)      105   66.9%   10
   (1, Inf)   (very bad)  28   17.8%   2
See help('pareto-k-diagnostic') for details.
```

检查辐射强度分布的拟合效果

```
# 抽取 yrep 数据
yrep <- fit_rongelap_poisson_pred$draws(variables = "yhat", format = "draws_matrix")
# Posterior predictive checks
pp_check(rongelap$counts / rongelap$time,
  yrep = sweep(yrep[1:50, ], MARGIN = 2, STATS = rongelap$time, FUN = `/`),
  fun = ppc_dens_overlay
) +
  theme_classic()
```

图 38.10: 后验预测诊断图（密度图）

后 1000 次迭代是平稳的，可取任意一个链条的任意一次迭代，获得采样点处的预测值

```
yhat_array <- fit_rongelap_poisson_pred$draws(variables = "yhat", format = "array")
lambda1_array <- fit_rongelap_poisson_pred$draws(variables = "lambda1", format = "array")
rongelap_sf$lambda <- as.vector(lambda1_array[1,1,])
rongelap_sf$yhat <- as.vector(yhat_array[1,1,])
```

数据集 rongelap_sf 的概况

汪漫 ©

```
rongelap_sf
Simple feature collection with 157 features and 4 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: -6050 ymin: -3430 xmax: -50 ymax: 0
CRS:           NA
First 10 features:
   counts time            geometry    lambda yhat
1      75  300 POINT (-6050 -3270) -1.252640   75
2     371  300 POINT (-6050 -3165)  0.265123  382
3    1931  300 POINT (-5925 -3320)  1.846460 1842
4    4357  300 POINT (-5925 -3165)  2.688390 4343
5    2114  300 POINT (-5800 -3350)  1.955840 2206
6    2318  300 POINT (-5800 -3165)  2.021350 2248
7    1975  300 POINT (-5625 -3350)  1.860100 1889
8    1912  300 POINT (-5700 -3260)  1.867950 1926
9    1902  300 POINT (-5700 -3220)  1.815670 1740
10   1882  300 POINT (-5700 -3180)  1.830060 1836
```

观测值和预测值的情况

```r
summary(rongelap_sf$counts / rongelap_sf$time)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.250   5.890   7.475   7.604   9.363  15.103 
```

```r
summary(rongelap_sf$yhat / rongelap_sf$time)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.250   5.800   7.465   7.579   9.313  14.993 
```

展示采样点处的预测值

未采样点的预测

```r
# 后验估计
ypred_tbl <- fit_rongelap_poisson_pred$summary(variables = "ypred", "mean")
rongelap_grid_df$ypred <- ypred_tbl$mean
# 查看预测结果
head(rongelap_grid_df)
```

```
         cX        cY time    ypred
1 -5685.942 -3606.997  100 736.3910
2 -5643.145 -3606.997  100 756.4195
3 -5600.347 -3606.997  100 774.6125
```

©黄湘云

图 38.11: 朗格拉普岛核辐射强度的分布

```
4 -5557.549 -3606.997  100 778.3830
5 -5514.751 -3606.997  100 777.8395
6 -5471.953 -3606.997  100 800.4850
```

```
# 预测值的分布范围
summary(rongelap_grid_df$ypred / rongelap_grid_df$time)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.4919  6.1896  7.3840  7.3044  8.5795 12.9371
```

转化数据类型，去掉缓冲区内的预测位置，准备绘制辐射强度预测值的分布

```
rongelap_grid_sf <- st_as_sf(rongelap_grid_df, coords = c("cX", "cY"), dim = "XY")
rongelap_grid_stars <- st_rasterize(rongelap_grid_sf, nx = 150, ny = 75)
rongelap_stars <- st_crop(x = rongelap_grid_stars, y = rongelap_coastline_sfp)
```

38.4 总结

从模型是否含有块金效应、不同的自相关函数和参数估计方法等方面比较。

```
library(nlme)
# 高斯分布、指数型自相关结构
fit_exp_reml <- gls(log(counts / time) ~ 1,
```

©黄湘云

图 38.12: 朗格拉普岛核辐射强度的分布

```
  correlation = corExp(value = 200, form = ~ cX + cY, nugget = FALSE),
  data = rongelap, method = "REML"
)
fit_exp_ml <- gls(log(counts / time) ~ 1,
  correlation = corExp(value = 200, form = ~ cX + cY, nugget = FALSE),
  data = rongelap, method = "ML"
)
fit_exp_reml_nugget <- gls(log(counts / time) ~ 1,
  correlation = corExp(value = c(200, 0.1), form = ~ cX + cY, nugget = TRUE),
  data = rongelap, method = "REML"
)
fit_exp_ml_nugget <- gls(log(counts / time) ~ 1,
  correlation = corExp(value = c(200, 0.1), form = ~ cX + cY, nugget = TRUE),
  data = rongelap, method = "ML"
)

# 高斯分布、高斯型自相关结构
fit_gaus_reml <- gls(log(counts / time) ~ 1,
  correlation = corGaus(value = 200, form = ~ cX + cY, nugget = FALSE),
  data = rongelap, method = "REML"
```

© 基础统计


```
)
fit_gaus_ml <- gls(log(counts / time) ~ 1,
  correlation = corGaus(value = 200, form = ~ cX + cY, nugget = FALSE),
  data = rongelap, method = "ML"
)
fit_gaus_reml_nugget <- gls(log(counts / time) ~ 1,
  correlation = corGaus(value = c(200, 0.1), form = ~ cX + cY, nugget = TRUE),
  data = rongelap, method = "REML"
)
fit_gaus_ml_nugget <- gls(log(counts / time) ~ 1,
  correlation = corGaus(value = c(200, 0.1), form = ~ cX + cY, nugget = TRUE),
  data = rongelap, method = "ML"
)
```

汇总结果见下表。

表格 38.1: 不同模型与参数估计方法的比较

响应变量分布	空间自相关结构	块金效应	估计方法	β	σ^2	ϕ	对数似然值
高斯分布	指数型	无	REML	1.826	0.3172	110.8	-89.07
高斯分布	指数型	无	ML	1.828	0.3064	105.4	-87.56
高斯分布	指数型	0.03598	REML	1.813	0.2935	169.7472	-88.22
高斯分布	指数型	0.03312	ML	1.828	0.2779	150.1324	-86.88
高斯分布	高斯型	无	REML	1.878	0.2523	41.96	-100.7
高斯分布	高斯型	无	ML	1.879	0.25	41.81	-98.62
高斯分布	高斯型	0.07055	REML	1.831	0.2532	139.1431	-84.91
高斯分布	高斯型	0.07053	ML	1.832	0.2459	137.0980	-83.32

相比于其他参数，REML 和 ML 估计方法对参数 ϕ 影响很大，ML 估计的 ϕ 和对数似然函数值更大。高斯型自相关结构中，REML 和 ML 估计方法对参数 ϕ 的估计结果差不多。函数 `gls()` 对初值要求不高，以上初值选取比较随意，只是符合要求函数定义。

对普通用户来说，想要流畅地使用 Stan 框架，需要面对很多挑战。

1. 软件安装和配置过程复杂。**rstan** 包内置的 Stan 版本常低于最新发布的 Stan 版本。
2. 编译和运行模型的参数控制选项很多。编译模型，OpenCL 和多线程支持，HMC（NUTS）、L-BFGS 和 VI 三大推理算法的参数设置
3. 模型参数先验分布设置技巧高。模型参数的先验对数据的依赖非常高，仅对线性和广义线性模型依赖较小。即使是面对模拟的简单广义线性混合效应模型，抽样过程也发散严重。
4. 面对大规模数据扩展困难。以朗格拉普岛的核污染预测任务为例，处理 157 维的积分显得吃力，对 1600 个参数的后验分布模拟和推断低效。

©黄湘云

2020 年 Stan 大会 Wade Brorsen 介绍采用 Stan 实现的贝叶斯克里金（Kriging）平滑算法估计和预测各郡县的作物产量。Stan 实现的贝叶斯空间分层正态模型，回归参数随空间区域位置变化，参数的先验分布与空间区域相关，引入大量带超参数的先验分布，运行效率不高，跑模型花费很多时间。假定所有的参数随空间位置变化，模型参数个数瞬间爆炸，跑模型花费 31 天 (Niyizibi, Brorsen, 和 Park 2018)。

Stan 总有些优势吧！

- Stan 非常灵活。Stan 同时是一门概率编程语言，只要统计模型可以被 Stan 编码，理论上就可以编译、运行、获得结果。
- Stan 功能很多。Stan 还可以解刚性的常微分方程、积分方程等。非常灵活，非常适合学术研究工作者，计算层面，可以方便地在前人的工作上扩展。
- Stan 文档很全。函数手册 提供 Stan 内建的各类函数说明。编程手册 提供 Stan 编程语法、程序块的说明，教用户如何使用 Stan 写代码。用户手册 提供 Stan 支持的各类统计模型、代数和微分方程的使用示例。

38.5 习题

1. 对核辐射污染数据，建立对数高斯过程模型，用 Stan 编码模型，预测全岛的核辐射强度分布。

$$
\begin{aligned}
\beta &\sim \text{std_normal}(0,1)\\
\sigma &\sim \text{inv_gamma}(5,5)\\
\phi &\sim \text{half_std_normal}(0,1)\\
\tau &\sim \text{half_std_normal}(0,1)\\
\boldsymbol{y} &\sim \text{multivariate_normal}(\boldsymbol{\beta}, \sigma^2\Sigma + \tau^2 I)
\end{aligned}
$$

其中，β 代表截距，先验分布为标准正态分布，σ 代表高斯过程的方差参数（信号），先验分布为逆伽马分布，ϕ 代表高斯过程的范围参数，先验分布为半标准正态分布，y 代表辐射强度的对数，给定参数和数据的条件分布为多元正态分布，Σ 代表协方差矩阵，I 代表与采样点数量相同的单位矩阵，τ^2 是块金效应。

```
functions {
  vector gp_pred_rng(array[] vector x2,
                     vector y1,
                     array[] vector x1,
                     real sigma,
                     real phi,
                     real tau,
                     real delta) {
    int N1 = rows(y1);
    int N2 = size(x2);
```

©清华大学出版社

```
    vector[N2] f2;
    {
      matrix[N1, N1] L_K;
      vector[N1] K_div_y1;
      matrix[N1, N2] k_x1_x2;
      matrix[N1, N2] v_pred;
      vector[N2] f2_mu;
      matrix[N2, N2] cov_f2;
      matrix[N2, N2] diag_delta;
      matrix[N1, N1] K;
      K = gp_exponential_cov(x1, sigma, phi);
      for (n in 1:N1) {
        K[n, n] = K[n, n] + square(tau);
      }
      L_K = cholesky_decompose(K);
      K_div_y1 = mdivide_left_tri_low(L_K, y1);
      K_div_y1 = mdivide_right_tri_low(K_div_y1', L_K)';
      k_x1_x2 = gp_exponential_cov(x1, x2, sigma, phi);
      f2_mu = (k_x1_x2' * K_div_y1);
      v_pred = mdivide_left_tri_low(L_K, k_x1_x2);
      cov_f2 = gp_exponential_cov(x2, sigma, phi) - v_pred' * v_pred;
      diag_delta = diag_matrix(rep_vector(delta, N2));

      f2 = multi_normal_rng(f2_mu, cov_f2 + diag_delta);
    }
    return f2;
  }
}
data {
  int<lower=1> D;
  int<lower=1> N1;
  array[N1] vector[D] x1;
  vector[N1] y1;
  int<lower=1> N2;
  array[N2] vector[D] x2;
}
transformed data {
  real delta = 1e-9;
}
parameters {
```

```
  real beta;
  real<lower=0> phi;
  real<lower=0> sigma;
  real<lower=0> tau;
}
transformed parameters {
  vector[N1] mu = rep_vector(beta, N1);
}
model {
  matrix[N1, N1] L_K;
  {
    matrix[N1, N1] K = gp_exponential_cov(x1, sigma, phi);
    real sq_tau = square(tau);

    // diagonal elements
    for (n1 in 1:N1) {
      K[n1, n1] = K[n1, n1] + sq_tau;
    }

    L_K = cholesky_decompose(K);
  }

  beta ~ std_normal();
  phi ~ std_normal();
  sigma ~ inv_gamma(5, 5);
  tau ~ std_normal();

  y1 ~ multi_normal_cholesky(mu, L_K);
}
generated quantities {
  vector[N2] f2;
  vector[N2] ypred;

  f2 = gp_pred_rng(x2, y1, x1, sigma, phi, tau, delta);
  for (n2 in 1:N2) {
    ypred[n2] = normal_rng(f2[n2], tau);
  }
}
```

代码中，gp_exponential_cov 表示空间相关性结构选择了指数型，详见 Stan 函数手册中的指数

型核函数表示。`cholesky_decompose` 表示对协方差矩阵做 Cholesky 分解，分解出来的下三角矩阵作为多元正态分布的参数，详见 Stan 函数手册中的 Cholesky 分解。`multi_normal_cholesky` 表示基于 Cholesky 分解的多元正态分布。详见 Stan 函数手册中的多元正态分布的 Cholesky 参数化表示。

© 贾清华

© 基础统计

第三十九章　时间序列回归

```
library(cmdstanr)
library(zoo)
library(xts) # xts 依赖 zoo
library(fGarch)
library(INLA)
library(mgcv)
library(tensorflow)
library(ggplot2)
library(bayesplot)
```

39.1　随机波动率模型

随机波动率模型主要用于股票时间序列数据建模。本节以美团股价数据为例介绍随机波动率模型，并分别以 Stan 框架和 **fGarch** 包拟合模型。

```
# 美团上市至 2023-07-15
meituan <- readRDS(file = "data/meituan.rds")
library(zoo)
library(xts)
library(ggplot2)
autoplot(meituan[, "3690.HK.Adjusted"]) +
  theme_classic() +
  labs(x = "日期", y = "股价")
```

© 黄湘云

图 39.1: 美团股价走势

对数收益率的计算公式如下：

$$\text{对数收益率} = \ln(\text{今日收盘价}/\text{昨日收盘价}) = \ln(1 + \text{普通收益率})$$

下图给出股价对数收益率变化和股价对数收益率的分布，可以看出在不同时间段，收益率波动幅度是不同的，美团股价对数收益率的分布可以看作正态分布。

```
meituan_log_return <- diff(log(meituan[, "3690.HK.Adjusted"]))[-1]
autoplot(meituan_log_return) +
  theme_classic() +
  labs(x = "日期", y = "对数收益率")
ggplot(data = meituan_log_return, aes(x = `3690.HK.Adjusted`)) +
  geom_histogram(color = "black", fill = "gray", bins = 30) +
  theme_classic() +
  labs(x = "对数收益率", y = "频数（天数）")
```

检查对数收益率序列的自相关图

```
acf(meituan_log_return, main = "")
```

©黄湘云

(a) 对数收益率的变动　　(b) 对数收益率的分布

图 39.2: 美团股价对数收益率的情况

图 39.3: 对数收益率的自相关图

发现，滞后 2、3、6、26 阶都有出界，滞后 17 阶略微出界，其它的自相关都在零水平线的界限内。

```
Box.test(meituan_log_return, lag = 12, type = "Ljung")
```

```
#>
```

©黄湘云


```
#>  Box-Ljung test
#>
#> data:  meituan_log_return
#> X-squared = 35.669, df = 12, p-value = 0.0003661
```

在 0.05 水平下拒绝了白噪声检验，说明对数收益率序列存在相关性。同理，也注意到对数收益率的绝对值和平方序列都不是独立的，存在相关性。

```
# ARCH 效应的检验
Box.test((meituan_log_return - mean(meituan_log_return))^2,
         lag = 12, type = "Ljung")
```

```
#>
#>  Box-Ljung test
#>
#> data:  (meituan_log_return - mean(meituan_log_return))^2
#> X-squared = 124.54, df = 12, p-value < 2.2e-16
```

结果高度显著，说明有 ARCH 效应。

39.1.1 Stan 框架

随机波动率模型如下

$$
\begin{aligned}
y_t &= \epsilon_t \exp(h_t/2) \\
h_{t+1} &= \mu + \phi(h_t - \mu) + \delta_t \sigma \\
h_1 &\sim \text{normal}\left(\mu, \frac{\sigma}{\sqrt{1-\phi^2}}\right) \\
\epsilon_t &\sim \text{normal}(0,1) \\
\delta_t &\sim \text{normal}(0,1)
\end{aligned}
$$

其中，y_t 表示在时间 t 时股价的回报（对数收益率），ϵ_t 表示股价回报在时间 t 时的白噪声扰/波动，δ_t 表示波动率在时间 t 时的波动。h_t 表示对数波动率，带有参数 μ （对数波动率的均值），ϕ （对数波动率的趋势）。代表波动率的序列 $\{h_t\}$ 假定是平稳 ($|\phi| < 1$) 的随机过程，h_1 来自平稳的分布（此处为正态分布），ϵ_t 和 δ_t 是服从不相关的标准正态分布。

Stan 代码如下

```
data {
  int<lower=0> T;   // # time points (equally spaced)
  vector[T] y;      // mean corrected return at time t
}
parameters {
```

©张贤达

```
  real mu;                      // mean log volatility
  real<lower=-1, upper=1> phi; // persistence of volatility
  real<lower=0> sigma;          // white noise shock scale
  vector[T] h_std;              // std log volatility time t
}
transformed parameters {
  vector[T] h = h_std * sigma;  // now h ~ normal(0, sigma)
  h[1] /= sqrt(1 - phi * phi);  // rescale h[1]
  h += mu;
  for (t in 2:T) {
    h[t] += phi * (h[t - 1] - mu);
  }
}
model {
  phi ~ uniform(-1, 1);
  sigma ~ cauchy(0, 5);
  mu ~ cauchy(0, 10);

  h_std ~ std_normal();
  y ~ normal(0, exp(h / 2));
}
```

编译和拟合模型

```
library(cmdstanr)
# 编译模型
mod_volatility_normal <- cmdstan_model(
  stan_file = "code/stochastic_volatility_models.stan",
  compile = TRUE, cpp_options = list(stan_threads = TRUE)
)
# 准备数据
mdata = list(T = 1274, y = as.vector(meituan_log_return))
# 拟合模型
fit_volatility_normal <- mod_volatility_normal$sample(
  data = mdata,
  chains = 2,
  parallel_chains = 2,
  iter_warmup = 1000,
  iter_sampling = 1000,
  threads_per_chain = 2,
  seed = 20232023,
```

© 王建平

```
  show_messages = FALSE,
  refresh = 0
)
# 输出结果
fit_volatility_normal$summary(c("mu", "phi", "sigma", "lp__"))
```

```
#> # A tibble: 4 x 10
#>   variable     mean   median      sd     mad       q5      q95  rhat ess_bulk
#>   <chr>       <dbl>    <dbl>   <dbl>   <dbl>    <dbl>    <dbl> <dbl>    <dbl>
#> 1 mu         -6.86    -6.86   0.116   0.111    -7.04    -6.67   1.00    1536.
#> 2 phi         0.912    0.917  0.0342  0.0314    0.850    0.957  1.01     287.
#> 3 sigma       0.299    0.294  0.0613  0.0575    0.209    0.407  1.00     309.
#> 4 lp__     3089.    3089.    29.1    29.8    3039.    3135.     1.00     548.
#> # i 1 more variable: ess_tail <dbl>
```

39.1.2 fGarch 包

《金融时间序列分析讲义》两个波动率建模方法

- 自回归条件异方差模型（Autoregressive Conditional Heteroskedasticity，简称 ARCH）。
- 广义自回归条件异方差模型（Generalized Autoregressive Conditional Heteroskedasticity，简称 GARCH ）

确定 ARCH 模型的阶，观察残差的平方的 ACF 和 PACF 。

```
acf((meituan_log_return - mean(meituan_log_return))^2, main = "")
pacf((meituan_log_return - mean(meituan_log_return))^2, main = "")
```

(a) 自相关图　　(b) 偏自相关图

图 39.4: 对数收益率的残差平方

©黄湘云

发现 ACF 在滞后 1、2、3 阶比较突出，PACF 在滞后 1、2、16、18、29 阶比较突出。所以下面先来考虑低阶的 ARCH(2) 模型，设 r_t 为对数收益率。

$$
\begin{aligned}
r_t &= \mu + a_t, \quad a_t = \sigma_t \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0,1) \\
\sigma_t^2 &= \alpha_0 + \alpha_1 a_{t-1}^2 + \alpha_2 a_{t-2}^2.
\end{aligned}
$$

拟合 ARCH 模型，比较模型估计结果，根据系数显著性的结果，采纳 ARCH(2) 模型。

```r
library(fGarch)
meituan_garch1 <- garchFit(
  formula = ~ 1 + garch(2, 0),
  data = meituan_log_return, trace = FALSE, cond.dist = "std"
)
summary(meituan_garch1)
```

```
#>
#> Title:
#>  GARCH Modelling
#>
#> Call:
#>  garchFit(formula = ~1 + garch(2, 0), data = meituan_log_return,
#>     cond.dist = "std", trace = FALSE)
#>
#> Mean and Variance Equation:
#>  data ~ 1 + garch(2, 0)
#> <environment: 0x7f7ec5d8e860>
#>  [data = meituan_log_return]
#>
#> Conditional Distribution:
#>  std
#>
#> Coefficient(s):
#>        mu      omega     alpha1     alpha2      shape
#> 0.0002577  0.0010729  0.1119940  0.1382923  4.9356152
#>
#> Std. Errors:
#>  based on Hessian
#>
#> Error Analysis:
#>         Estimate  Std. Error  t value Pr(>|t|)
#> mu     2.577e-04   8.970e-04    0.287  0.77390
```

© 基础统计


```
#> omega  1.073e-03   9.432e-05   11.375  < 2e-16 ***
#> alpha1 1.120e-01   4.292e-02    2.609  0.00907 **
#> alpha2 1.383e-01   4.725e-02    2.927  0.00343 **
#> shape  4.936e+00   7.008e-01    7.043 1.88e-12 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Log Likelihood:
#>  2459.345    normalized:  1.930412
#>
#> Description:
#>  Tue Jan 23 10:32:27 2024 by user:
#>
#>
#> Standardised Residuals Tests:
#>                                 Statistic      p-Value
#>  Jarque-Bera Test   R    Chi^2  260.647924 0.000000e+00
#>  Shapiro-Wilk Test  R    W        0.975911 9.515126e-14
#>  Ljung-Box Test     R    Q(10)   21.212778 1.965775e-02
#>  Ljung-Box Test     R    Q(15)   24.773595 5.306786e-02
#>  Ljung-Box Test     R    Q(20)   33.252167 3.165160e-02
#>  Ljung-Box Test     R^2  Q(10)   17.362563 6.671658e-02
#>  Ljung-Box Test     R^2  Q(15)   29.329034 1.458467e-02
#>  Ljung-Box Test     R^2  Q(20)   53.703334 6.400548e-05
#>  LM Arch Test       R    TR^2    19.254073 8.257864e-02
#>
#> Information Criterion Statistics:
#>       AIC       BIC       SIC      HQIC
#> -3.852975 -3.832764 -3.853006 -3.845384
```

函数 `garchFit()` 的参数 `cond.dist` 默认值为 `"norm"` 表示标准正态分布，`cond.dist = "std"` 表示标准 t 分布。模型均值的估计值接近 0 是符合预期的，且显著性没通过，对数收益率在 0 上下波动。将估计结果代入模型，得到

$$
\begin{aligned}
r_t &= -5.665 \times 10^{-5} + a_t, \quad a_t = \sigma_t \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0,1) \\
\sigma_t^2 &= 1.070 \times 10^{-3} + 0.1156 a_{t-1}^2 + 0.1438 a_{t-2}^2.
\end{aligned}
$$

下面考虑 GARCH(1,1) 模型

©梁清河

$$r_t = \mu + a_t, \quad a_t = \sigma_t \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0,1)$$
$$\sigma_t^2 = \alpha_0 + \alpha_1 a_{t-1}^2 + \beta_1 \sigma_{t-1}^2.$$

```
meituan_garch2 <- garchFit(
  formula = ~ 1 + garch(1, 1),
  data = meituan_log_return, trace = FALSE, cond.dist = "std"
)
summary(meituan_garch2)
```

```
#>
#> Title:
#>  GARCH Modelling
#>
#> Call:
#>  garchFit(formula = ~1 + garch(1, 1), data = meituan_log_return,
#>     cond.dist = "std", trace = FALSE)
#>
#> Mean and Variance Equation:
#>  data ~ 1 + garch(1, 1)
#> <environment: 0x7f7eab1b4790>
#>  [data = meituan_log_return]
#>
#> Conditional Distribution:
#>  std
#>
#> Coefficient(s):
#>         mu       omega      alpha1       beta1       shape
#> 2.8296e-04  3.4454e-05  5.9798e-02  9.1678e-01  5.4352e+00
#>
#> Std. Errors:
#>  based on Hessian
#>
#> Error Analysis:
#>         Estimate  Std. Error  t value Pr(>|t|)
#> mu     2.830e-04   8.702e-04    0.325  0.74505
#> omega  3.445e-05   1.937e-05    1.779  0.07525 .
#> alpha1 5.980e-02   1.855e-02    3.224  0.00127 **
#> beta1  9.168e-01   2.784e-02   32.933  < 2e-16 ***
#> shape  5.435e+00   8.137e-01    6.680 2.39e-11 ***
#> ---
```

© 王建勋


```
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Log Likelihood:
#>  2478.473    normalized:  1.945427
#>
#> Description:
#>  Tue Jan 23 10:32:27 2024 by user:
#>
#>
#> Standardised Residuals Tests:
#>                                 Statistic      p-Value
#>  Jarque-Bera Test   R    Chi^2  226.7198909 0.000000e+00
#>  Shapiro-Wilk Test  R    W        0.9781165 5.582467e-13
#>  Ljung-Box Test     R    Q(10)   16.0489249 9.824041e-02
#>  Ljung-Box Test     R    Q(15)   19.6491086 1.858104e-01
#>  Ljung-Box Test     R    Q(20)   27.2460587 1.284822e-01
#>  Ljung-Box Test     R^2  Q(10)    7.8054550 6.478332e-01
#>  Ljung-Box Test     R^2  Q(15)    9.8336579 8.300707e-01
#>  Ljung-Box Test     R^2  Q(20)   24.7640454 2.106076e-01
#>  LM Arch Test       R    TR^2     9.5999700 6.510091e-01
#>
#> Information Criterion Statistics:
#>       AIC       BIC       SIC      HQIC
#> -3.883004 -3.862792 -3.883034 -3.875413
```

波动率的贡献主要来自 σ_{t-1}^2 ，其系数 β_1 为 0.918。通过对数似然的比较，可以发现 GARCH(1,1) 模型比 ARCH(2) 模型更好。

39.2 贝叶斯可加模型

大规模时间序列回归，观察值是比较多的，可达数十万、数百万，乃至更多。粗粒度时时间跨度往往很长，比如数十年的天粒度数据，细粒度时时间跨度可短可长，比如数年的半小时级数据，总之，需要包含多个季节的数据，各种季节性重复出现。通过时序图可以观察到明显的季节性，而且往往是多种周期不同的季节性混合在一起，有时还包含一定的趋势性。举例来说，比如 2018-2023 年美国旧金山犯罪事件报告数据，事件数量的变化趋势，除了上述季节性因素，特殊事件疫情肯定会影响，数据规模约 200 M 。再比如 2018-2023 年美国境内和跨境旅游业中的航班数据，原始数据非常大，R 包 nycflights13 提供纽约机场的部分航班数据。

为简单起见，下面以 R 内置的数据集 AirPassengers 为例，介绍 Stan 框架和 INLA 框架建模的过程。数据集 AirPassengers 包含周期性（季节性）和趋势性。作为对比的基础，下面建立非线性回归模型，趋

© 基础统计

势项和周期项是可加的形式：

$$y = at + b + c\sin(\frac{t}{12} \times 2\pi) + d\cos(\frac{t}{12} \times 2\pi) + \epsilon$$

根据数据变化的周期规律，设置周期为 12，还可以在模型中添加周期为 3 或 4 的小周期。其中，y 代表观察值，a, b, c, d 为待定的参数，ϵ 代表服从标准正态分布的随机误差。

```
air_passengers_df <- data.frame(y = as.vector(AirPassengers), t = 1:144)
fit_lm1 <- lm(y ~ t + sin(t / 12 * 2 * pi) + cos(t / 12 * 2 * pi), data = air_passengers_df)
fit_lm2 <- update(fit_lm1, . ~ . +
  sin(t / 12 * 2 * 2 * pi) + cos(t / 12 * 2 * 2 * pi), data = air_passengers_df
)
fit_lm3 <- update(fit_lm2, . ~ . +
  sin(t / 12 * 3 * 2 * pi) + cos(t / 12 * 3 * 2 * pi), data = air_passengers_df
)
plot(y ~ t, air_passengers_df, type = "l")
lines(x = air_passengers_df$t, y = fit_lm1$fitted.values, col = "red")
lines(x = air_passengers_df$t, y = fit_lm2$fitted.values, col = "green")
lines(x = air_passengers_df$t, y = fit_lm3$fitted.values, col = "orange")
```

图 39.5: 非线性回归

模型 1 已经很好地捕捉到趋势和周期信息，当添加小周期后，略有改善，继续添加更多的小周期，不再

©黄湘云

有明显改善。实际上，小周期对应的回归系数也将不再显著。所以，这类模型的优化空间见顶了，需要进一步观察和利用残差的规律，使用更加复杂的模型。

39.2.1 Stan 框架

非线性趋势、多季节性（多个周期混合）、特殊节假日、突发热点事件、残差成分（平稳），能同时应对这五种情况的建模方法是贝叶斯可加模型和神经网络模型，比如基于 Stan 实现的 prophet 包和 tensorflow 框架。

> 提示
>
> prophet 包是如何同时处理这些情况，是否可以在 cmdstanr 包中实现，是否可以在 mgcv 和 INLA 中实现？

```
library(cmdstanr)
```

39.2.2 INLA 框架

阿卜杜拉国王科技大学（King Abdullah University of Science and Technology 简称 KAUST）的 Håvard Rue 等开发了 INLA 框架 (Rue, Martino, 和 Chopin 2009)。《贝叶斯推断与 INLA 》的第 3 章混合效应模型中随机游走部分 (Gómez-Rubio 2020)，一个随机过程（如随机游走、AR(p) 过程）作为随机效应。AirPassengers 的方差在变大，取对数尺度后，方差基本保持不变，一阶差分后基本保持平稳。

```
library(ggfortify)
autoplot(log(AirPassengers)) +
  theme_classic() +
  labs(x = "年月", y = "对数值")
autoplot(diff(log(AirPassengers))) +
  theme_classic() +
  labs(x = "年月", y = "差分对数值")
```

因此，下面基于对数尺度建模。首先考虑 RW1 随机游走模型，而后考虑季节性。RW1 模型意味着取对数、一阶差分后序列平稳高斯过程，序列值服从高斯分布。下面设置似然函数的高斯先验 $\mathcal{N}(1, 0.2)$ ，目的是防止过拟合。

```
library(INLA)
inla.setOption(short.summary = TRUE)
air_passengers_df <- data.frame(
  y = as.vector(AirPassengers),
  year = as.factor(rep(1949:1960, each = 12)),
  month = as.factor(rep(1:12, times = 12)),
  ID = 1:length(AirPassengers)
```

©黄湘云

(a) 对数尺度 (b) 一阶差分

图 39.6: AirPassengers 的时序图

```
)
mod_inla_rw1 <- inla(
  formula = log(y) ~ year + f(ID, model = "rw1"),
  family = "gaussian", data = air_passengers_df,
  control.family = list(hyper = list(prec = list(param = c(1, 0.2)))),
  control.predictor = list(compute = TRUE)
)
summary(mod_inla_rw1)
```

```
#> Fixed effects:
#>              mean    sd 0.025quant 0.5quant 0.975quant  mode kld
#> (Intercept) 5.159 0.252      4.666    5.158      5.657 5.158   0
#> year1950    0.050 0.134     -0.215    0.050      0.313 0.050   0
#> year1951    0.174 0.190     -0.201    0.174      0.546 0.174   0
#> year1952    0.262 0.233     -0.198    0.262      0.717 0.262   0
#> year1953    0.330 0.269     -0.201    0.331      0.857 0.331   0
#> year1954    0.357 0.301     -0.236    0.358      0.945 0.358   0
#> year1955    0.442 0.329     -0.208    0.443      1.086 0.443   0
#> year1956    0.510 0.356     -0.193    0.511      1.206 0.511   0
#> year1957    0.567 0.381     -0.184    0.568      1.311 0.568   0
#> year1958    0.576 0.403     -0.220    0.577      1.365 0.577   0
#> year1959    0.647 0.425     -0.191    0.648      1.478 0.648   0
#> year1960    0.683 0.445     -0.195    0.684      1.555 0.684   0
#>
#> Model hyperparameters:
```

©贾俊平

```
#>                                          mean    sd 0.025quant 0.5quant
#> Precision for the Gaussian observations 101.38 18.14      69.78    99.99
#> Precision for ID                        155.14 40.69      92.69   149.32
#>                                         0.975quant   mode
#> Precision for the Gaussian observations     140.94  97.59
#> Precision for ID                            251.66 137.50
#>
#>  is computed
```

这里，将年份作为因子型变量，从输出结果可以看出，以 1949 年作为参照，回归系数的后验均值在逐年变大，这符合 AirPassengers 时序图呈现的趋势。

存在周期性的波动规律，考虑季节性

```
mod_inla_sea <- inla(
  formula = log(y) ~ year + f(ID, model = "seasonal", season.length = 12),
  family = "gaussian", data = air_passengers_df,
  control.family = list(hyper = list(prec = list(param = c(1, 0.2)))),
  control.predictor = list(compute = TRUE)
)
summary(mod_inla_sea)
```

```
#> Fixed effects:
#>              mean    sd 0.025quant 0.5quant 0.975quant  mode kld
#> (Intercept) 4.836 0.020      4.797    4.836      4.875 4.836   0
#> year1950    0.095 0.028      0.039    0.095      0.150 0.095   0
#> year1951    0.295 0.028      0.240    0.295      0.351 0.295   0
#> year1952    0.441 0.028      0.386    0.441      0.496 0.441   0
#> year1953    0.573 0.028      0.517    0.573      0.628 0.573   0
#> year1954    0.630 0.028      0.575    0.630      0.686 0.630   0
#> year1955    0.803 0.028      0.748    0.803      0.859 0.803   0
#> year1956    0.948 0.028      0.893    0.948      1.004 0.948   0
#> year1957    1.062 0.028      1.007    1.062      1.118 1.062   0
#> year1958    1.094 0.028      1.039    1.094      1.150 1.094   0
#> year1959    1.212 0.028      1.157    1.212      1.268 1.212   0
#> year1960    1.318 0.028      1.263    1.318      1.373 1.318   0
#>
#> Model hyperparameters:
#>                                          mean       sd 0.025quant 0.5quant
#> Precision for the Gaussian observations   213.03    27.48     163.45   211.49
#> Precision for ID                        42062.72 27484.88   10112.29 35356.76
#>                                         0.975quant     mode
```

```
#> Precision for the Gaussian observations     271.43   208.84
#> Precision for ID                          113332.18 24334.75
#>
#>  is computed
```

最后，将两个模型的拟合结果展示出来，见下图，黑线表示原对数值，红线表示拟合值，灰色区域表示在置信水平 95% 下的区间。区间更短说明季节性模型更好。

```
mod_inla_rw1_fitted <- data.frame(
  ID = 1:length(AirPassengers),
  y = as.vector(log(AirPassengers)),
  mean = mod_inla_rw1$summary.fitted.values$mean,
  `0.025quant` = mod_inla_rw1$summary.fitted.values$`0.025quant`,
  `0.975quant` = mod_inla_rw1$summary.fitted.values$`0.975quant`,
  check.names = FALSE
)
mod_inla_sea_fitted <- data.frame(
  ID = 1:length(AirPassengers),
  y = as.vector(log(AirPassengers)),
  mean = mod_inla_sea$summary.fitted.values$mean,
  `0.025quant` = mod_inla_sea$summary.fitted.values$`0.025quant`,
  `0.975quant` = mod_inla_sea$summary.fitted.values$`0.975quant`,
  check.names = FALSE
)
ggplot(data = mod_inla_rw1_fitted, aes(ID)) +
  geom_ribbon(aes(ymin = `0.025quant`, ymax = `0.975quant`), fill = "gray") +
  geom_line(aes(y = y)) +
  geom_line(aes(y = mean), color = "red") +
  theme_classic() +
  labs(x = "序号", y = "对数值")
ggplot(data = mod_inla_sea_fitted, aes(ID)) +
  geom_ribbon(aes(ymin = `0.025quant`, ymax = `0.975quant`), fill = "gray") +
  geom_line(aes(y = y)) +
  geom_line(aes(y = mean), color = "red") +
  theme_classic() +
  labs(x = "序号", y = "对数值")
```

©黄湘云

(a) 随机游走模型 (b) 季节效应模型

图 39.7: AirPassengers 的拟合图

39.3 一些非参数模型

39.3.1 mgcv 包

mgcv 包 (S. N. Wood 2017) 是 R 软件内置的推荐组件，由 Simon Wood 开发和维护，历经多年，成熟稳定。函数 `bam()` 相比于函数 `gam()` 的优势是可以处理大规模的时间序列数据。对于时间序列数据预测，数万和百万级观测值都可以 (Simon N. Wood, Goude, 和 Shaw 2015)。

```
air_passengers_tbl <- data.frame(
  y = as.vector(AirPassengers),
  year = rep(1949:1960, each = 12),
  month = rep(1:12, times = 12)
)
mod1 <- gam(y ~ s(year) + s(month, bs = "cr", k = 12),
  data = air_passengers_tbl, family = gaussian
)
summary(mod1)
```

```
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> y ~ s(year) + s(month, bs = "cr", k = 12)
#>
#> Parametric coefficients:
```

统计建模 ©

```
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  280.299      1.957   143.2   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>             edf Ref.df      F p-value
#> s(year)  6.102  7.265 441.39  <2e-16 ***
#> s(month) 8.796 10.097  38.25  <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.962   Deviance explained = 96.6%
#> GCV =  619.9  Scale est. = 551.47    n = 144
```

观察年和月的趋势变化，逐年增长趋势基本是线性的，略有波动，逐月变化趋势比较复杂，不过，可以明显看出在 7-9 月是高峰期，11 月和 1-3 月是低谷期。

```
layout(matrix(1:2, nrow = 1))
plot(mod1, shade = TRUE)
```

图 39.8: 年和月的趋势变化

将拟合效果绘制出来，见下图，整体上，捕捉到了趋势和周期，不过，存在欠拟合，年周期内波动幅度

©黄湘云

随时间有变化趋势，趋势和周期存在交互作用。

```
air_passengers_ts <- ts(mod1$fitted.values, start = c(1949, 1), frequency = 12)
plot(AirPassengers)
lines(air_passengers_ts, col = "red")
```

图 39.9: 趋势拟合效果

整体上，乘客数逐年呈线性增长，每年不同月份呈现波动，淡季和旺季出行的流量有很大差异，近年来，这种差异的波动在扩大。为了刻画这种情况，考虑年度趋势和月度波动的交互作用。

```
mod2 <- gam(y ~ s(year, month), data = air_passengers_tbl, family = gaussian)
summary(mod2)
```

```
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> y ~ s(year, month)
#>
#> Parametric coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  280.299      1.059   264.7   <2e-16 ***
```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>                edf Ref.df     F p-value
#> s(year,month) 28.21  28.96 435.5  <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.989   Deviance explained = 99.1%
#> GCV = 202.62  Scale est. = 161.52    n = 144
```

可以看到，调整的 R^2 明显增加，拟合效果更好，各年各月份的乘客数变化，见下图。

```
op <- par(mar = c(4, 4, 2, 0))
plot(mod2)
on.exit(par(op), add = TRUE)
```

图 39.10: 交互作用

上图是轮廓图，下面用透视图展示趋势拟合的效果。

```
op <- par(mar = c(0, 1.5, 0, 0))
vis.gam(mod2, theta = -35, phi = 20, ticktype = "detailed", expand = .65, zlab = "")
on.exit(par(op), add = TRUE)
```

© 黄湘云

图 39.11: 趋势拟合效果

最后，在原始数据的基础上，添加拟合数据，得到如下拟合趋势图，与前面的拟合图比较，可以看出效果提升很明显。

```
air_passengers_ts <- ts(mod2$fitted.values, start = c(1949, 1), frequency = 12)
plot(AirPassengers)
lines(air_passengers_ts, col = "red")
```

©黄湘云

图 39.12: 趋势拟合效果

39.3.2 tensorflow 框架

前面介绍的模型都具有非常强的可解释性，比如各个参数对模型的作用。对于复杂的时间序列数据，比较适合用复杂的模型来拟合，看重模型的泛化能力，而不那么关注模型的机理。

多层感知机是一种全连接层的前馈神经网络。**nnet** 包的函数 `nnet()` 实现了单隐藏层的简单前馈神经网络，可用于时间序列预测，也可用于分类数据的预测。作为对比的基础，下面先用 nnet 包训练和预测数据。

```
# 准备数据
air_passengers <- as.matrix(embed(AirPassengers, 4))
colnames(air_passengers) <- c("y", "x3", "x2", "x1")
data_size <- nrow(air_passengers)
# 拆分数据集
train_size <- floor(data_size * 0.67)
train_data <- air_passengers[1:train_size, ]
test_data <- air_passengers[train_size:data_size, ]

# 随机数种子对结果的影响非常大 试试 set.seed(20232023)
set.seed(20222022)
# 单隐藏层 8 个神经元
```

© 统计学基础

```
mod_nnet <- nnet::nnet(
  y ~ x1 + x2 + x3,
  data = air_passengers, # 数据集
  subset = 1:train_size, # 训练数据的指标向量
  linout = TRUE, size = 4, rang = 0.1,
  decay = 5e-4, maxit = 400, trace = FALSE
)
# 预测
train_pred <- predict(mod_nnet, newdata = air_passengers[1:train_size,], type = "raw")
# 训练集 RMSE
sqrt(mean((air_passengers[1:train_size, "y"] - train_pred )^2))
```

```
#> [1] 21.59392
```

```
# 预测
test_pred <- predict(mod_nnet, newdata = air_passengers[-(1:train_size),], type = "raw")
# 测试集 RMSE
sqrt(mean((air_passengers[-(1:train_size), "y"] - test_pred)^2))
```

```
#> [1] 53.79107
```

下面将原观测序列，训练集和测试集上的预测序列放在一张图上展示。图中，红色曲线表示训练集上的预测结果，绿色曲线为测试集上预测结果。

```
train_pred_ts <- ts(data = train_pred, start = c(1949, 3), frequency = 12)
test_pred_ts <- ts(data = test_pred, start = c(1957, 1), frequency = 12)
plot(AirPassengers)
lines(train_pred_ts, col = "red")
lines(test_pred_ts, col = "green")
```

©黄湘云

图 39.13: 单层感知机预测

由图可知，在测试集上，随着时间拉长，预测越来越不准。

下面使用 tensorflow 包构造**多层**感知机训练数据和预测。

```
library(tensorflow)
library(keras)
set_random_seed(20222022)
# 模型结构
mod_mlp <- keras_model_sequential() |>
  layer_dense(units = 12, activation = "relu", input_shape = c(3)) |>
  layer_dense(units = 8, activation = "relu") |>
  layer_dense(units = 1)
# 训练目标
compile(mod_mlp,
  loss = "mse", # 损失函数
  optimizer = "adam", # 优化器
  metrics = "mae" # 监控度量
)
# 模型概览
summary(mod_mlp)
```

©黄湘云


```
#> Model: "sequential"
#> _________________________________________________________________________________
#>  Layer (type)                    Output Shape                  Param #
#> =================================================================================
#>  dense_2 (Dense)                 (None, 12)                    48
#>  dense_1 (Dense)                 (None, 8)                     104
#>  dense (Dense)                   (None, 1)                     9
#> =================================================================================
#> Total params: 161 (644.00 Byte)
#> Trainable params: 161 (644.00 Byte)
#> Non-trainable params: 0 (0.00 Byte)
#> _________________________________________________________________________________
```

输入层为 3 个节点，中间两个隐藏层，第一层为 12 个节点，第二层为 8 个节点，全连接网络，最后输出为一层单节点，意味着单个输出。每一层都有节点和权重，参数总数为 161。

```
# 拟合模型
fit(mod_mlp,
  x = train_data[, c("x1", "x2", "x3")],
  y = train_data[, "y"],
  epochs = 200,
  batch_size = 10, # 每次更新梯度所用的样本量
  validation_split = 0.2, # 从训练数据中拆分一部分用作验证集
  verbose = 0 # 不显示训练进度
)
# 将测试数据代入模型，计算损失函数和监控度量
evaluate(mod_mlp, test_data[, c("x1", "x2", "x3")], test_data[, "y"])
```

```
#> 2/2 - 0s - loss: 2517.1836 - mae: 40.8876 - 17ms/epoch - 9ms/step
```

```
#>       loss        mae
#> 2517.18359   40.88759
```

```
# 测试集上的预测
mlp_test_pred <- predict(mod_mlp, test_data[, c("x1", "x2", "x3")])
```

```
#> 2/2 - 0s - 64ms/epoch - 32ms/step
```

```
mlp_train_pred <- predict(mod_mlp, train_data[, c("x1", "x2", "x3")])
```

```
#> 3/3 - 0s - 15ms/epoch - 5ms/step
```

```
sqrt(mean((test_data[, "y"] - mlp_test_pred)^2)) # 计算均方根误差
```

```
#> [1] 50.17154
```

从 RMSE 来看，MLP（多层感知机）预测效果比单层感知机稍好些，可网络复杂度是增加很多的。

统计学习基础 ©

```
mlp_train_pred_ts <- ts(data = mlp_train_pred, start = c(1949, 3), frequency = 12)
mlp_test_pred_ts <- ts(data = mlp_test_pred, start = c(1957, 1), frequency = 12)
plot(AirPassengers)
lines(mlp_train_pred_ts, col = "red")
lines(mlp_test_pred_ts, col = "green")
```

图 39.14: 多层感知机预测

下面用 LSTM （长短期记忆）神经网络来训练时间序列数据，预测未来一周的趋势。输出不再是一天（单点输出），而是 7 天的预测值（多点输出）。参考 **tensorflow** 包的官网中 RNN 递归神经网络的介绍。

39.4 习题

1. 基于 R 软件内置的数据集 `sunspots` 和 `sunspot.month` 比较 INLA 和 **mgcv** 框架的预测效果。

   图中黑线和红线分别表示 1749-1983 年、1984-2014 年每月太阳黑子数量。

© 黄湘云

图 39.15: 预测月粒度太阳黑子数量

© 黄湘云

第八部分

机器学习

© 黄湘云

第四十章　分类问题

```
library(nnet)     # 多项回归/神经网络 multinom / nnet
library(MASS)     # 线性/二次判别分析 lda / qda
library(glmnet)   # 惩罚多项回归 glmnet
library(e1071)    # 朴素贝叶斯 naiveBayes 和支持向量机 svm
library(kernlab)  # 支持向量机分类 ksvm
library(class)    # K 最近邻 knn
library(rpart)    # 决策树分类 rpart
library(randomForest) # 随机森林 randomForest
# library(gbm)        # 梯度提升机
library(xgboost)      # 集成学习
library(lattice)
```

以 iris 数据集为例，简单，方便介绍模型和算法，定位入门。分类间隔最大化，也是一个优化问题，找一条分界线，一个分割面，一个超平面划分不同的种类。本章篇幅：每个算法 4 页，共计 40 页。10 个算法的介绍按照分类思路，模型，代码和参数说明，分类性能评估。应用案例是手写数字识别。要点不是数据如何复杂，而是怎样把理论写得通俗、准确，看了之后能够应用到复杂的真实数据分析场景中去。理论解释、绘图说明、经验总结。

1. 线性分类器
    2. 多项回归模型
    3. 线性判别分析
2. 非线性分类器
    1. 二次判别分析
    2. 朴素贝叶斯
    3. 支持向量机
    4. K 最近邻
    5. 神经网络
    6. 决策树
    7. 随机森林
    8. 集成学习

iris 数据集也来自 Base R 自带的 **datasets** 包，由 Anderson Edgar 收集，最早见于 1935 年的文章，后

©黄湘云

被 Ronald Fisher 在研究分类问题时引用 (Fisher 1936)。到如今，在机器学习的社区里，提及 iris 数据集，一般只知 Fisher 不知 Anderson。

> 提示
>
> 1. 鸢尾花数据集，逻辑回归拟合，绘制分类边界图，实现 R 版本。
> 2. 参考文献《机器学习的概率视角导论》(Murphy 2022) 书中图 2.13 的 Python 代码
> 3. 将回归模型用 SQL 表达出来，放在数据库上高性能地执行分类预测。

40.1 多项回归模型

```
library(nnet) # 多项逻辑回归
iris_multinom <- multinom(Species ~ ., data = iris, trace = FALSE)
summary(iris_multinom)
```

```
Call:
multinom(formula = Species ~ ., data = iris, trace = FALSE)

Coefficients:
           (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
versicolor    18.69037    -5.458424   -8.707401     14.24477   -3.097684
virginica    -23.83628    -7.923634  -15.370769     23.65978   15.135301

Std. Errors:
           (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
versicolor    34.97116     89.89215    157.0415     60.19170    45.48852
virginica     35.76649     89.91153    157.1196     60.46753    45.93406

Residual Deviance: 11.89973
AIC: 31.89973
```

```
table(predict(iris_multinom, iris[, -5], type = "class"), iris[, 5])
```

```
             setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         49         1
  virginica       0          1        49
```

在有的数据中，观测变量之间存在共线性，采用变量选择方法，比如 Lasso 方法压缩掉一部分变量。

```
library(glmnet) # 多项回归
iris_glmnet <- glmnet(x = iris[, -5], y = iris[, 5], family = "multinomial")
```

©李东风

```
plot(iris_glmnet)
plot(iris_glmnet$lambda,
  ylab = expression(lambda), xlab = "迭代次数", main = "惩罚系数的迭代路径"
)
```

选择一个迭代趋于稳定时的 lambda，比如 `iris_glmnet$lambda[80]` 。

```
coef(iris_glmnet, s = 0.0002796185)
```

```
$setosa
5 x 1 sparse Matrix of class "dgCMatrix"
                     1
(Intercept)  17.015429
Sepal.Length  .
Sepal.Width   4.486992
Petal.Length -3.250342
Petal.Width  -3.315393

$versicolor
5 x 1 sparse Matrix of class "dgCMatrix"
                    1
(Intercept)  8.132656
Sepal.Length 2.123980
Sepal.Width  .
Petal.Length .
Petal.Width  .

$virginica
5 x 1 sparse Matrix of class "dgCMatrix"
                      1
(Intercept)  -25.148085
Sepal.Length   .
Sepal.Width   -5.176029
Petal.Length   7.536940
Petal.Width   14.481524
```

```
iris_pred_glmnet <- predict(
  object = iris_glmnet, newx = as.matrix(iris[, -5]),
  s = 0.0002796185, type = "class"
)
```

```
table(iris_pred_glmnet, iris[, 5])
```

©黄湘云

(a) 回归系数 setosa 的迭代路径

(b) 回归系数 versicolor 的迭代路径

(c) 回归系数 virginica 的迭代路径

(d) 惩罚系数的迭代路径

图 40.1: 迭代路径

©黄湘云

```
iris_pred_glmnet setosa versicolor virginica
      setosa         50          0         0
      versicolor      0         49         1
      virginica       0          1        49
```

40.2 线性判别分析

```r
library(MASS)
# lda
iris_lda <- lda(Species ~ ., data=iris)
iris_lda
```

```
Call:
lda(Species ~ ., data = iris)

Prior probabilities of groups:
    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333

Group means:
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa            5.006       3.428        1.462       0.246
versicolor        5.936       2.770        4.260       1.326
virginica         6.588       2.974        5.552       2.026

Coefficients of linear discriminants:
                    LD1         LD2
Sepal.Length  0.8293776 -0.02410215
Sepal.Width   1.5344731 -2.16452123
Petal.Length -2.2012117  0.93192121
Petal.Width  -2.8104603 -2.83918785

Proportion of trace:
   LD1    LD2
0.9912 0.0088
```

```r
# 预测
iris_lda_pred <- predict(iris_lda, iris[, -5])$class
# 预测结果
table(iris_lda_pred, iris[, 5])
```

©黄湘云

```
iris_lda_pred setosa versicolor virginica
   setosa         50          0         0
   versicolor      0         48         1
   virginica       0          2        49
```

40.3 二次判别分析

```r
# Quadratic Discriminant Analysis 二次判别分析
iris_qda <- qda(Species ~ ., data=iris)
iris_qda
```

```
Call:
qda(Species ~ ., data = iris)

Prior probabilities of groups:
    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333

Group means:
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa            5.006       3.428        1.462       0.246
versicolor        5.936       2.770        4.260       1.326
virginica         6.588       2.974        5.552       2.026
```

```r
# 预测
iris_qda_pred <- predict(iris_qda, iris[, -5])$class
# 预测结果
table(iris_qda_pred, iris[, 5])
```

```
iris_qda_pred setosa versicolor virginica
   setosa         50          0         0
   versicolor      0         48         1
   virginica       0          2        49
```

40.4 朴素贝叶斯

```r
library(e1071) # 朴素贝叶斯
iris_nb <- naiveBayes(Species ~ ., data = iris)
iris_nb
```

©黄文彬

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333

Conditional probabilities:
            Sepal.Length
Y             [,1]      [,2]
  setosa     5.006 0.3524897
  versicolor 5.936 0.5161711
  virginica  6.588 0.6358796

            Sepal.Width
Y             [,1]      [,2]
  setosa     3.428 0.3790644
  versicolor 2.770 0.3137983
  virginica  2.974 0.3224966

            Petal.Length
Y             [,1]      [,2]
  setosa     1.462 0.1736640
  versicolor 4.260 0.4699110
  virginica  5.552 0.5518947

            Petal.Width
Y             [,1]      [,2]
  setosa     0.246 0.1053856
  versicolor 1.326 0.1977527
  virginica  2.026 0.2746501
```

```r
# 预测
iris_nb_pred <- predict(iris_nb, newdata = iris, type = "class")
# 预测结果
table(iris_nb_pred, iris[, 5])
```

```
iris_nb_pred setosa versicolor virginica
```

©谢邦昌

```
  setosa         50          0         0
  versicolor      0         47         3
  virginica       0          3        47
```

40.5 支持向量机

e1071 包也提供支持向量机

```
# e1071
iris_svm <- svm(Species ~ ., data = iris)
iris_svm
```

```
Call:
svm(formula = Species ~ ., data = iris)



Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  51
```

```
# 预测
iris_svm_pred <- predict(iris_svm, newdata = iris, probability = FALSE)
# 预测结果
table(iris_svm_pred, iris[, 5])
```

```
iris_svm_pred setosa versicolor virginica
   setosa         50          0         0
   versicolor      0         48         2
   virginica       0          2        48
```

kernlab 包提供核支持向量机。

```
library(kernlab)
iris_ksvm <- ksvm(Species ~ ., data = iris)
iris_ksvm
```

```
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1
```

©谢湘

```
Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.643354744345197

Number of Support Vectors : 59

Objective Function Value : -4.0949 -4.4463 -20.8573
Training error : 0.026667
```

kernlab 包 (Karatzoglou 等 2004) 的绘图函数 `plot()` 仅支持二分类模型。

```r
iris_pred_svm <- predict(iris_ksvm, iris[, -5], type = "response")
table(iris_pred_svm, iris[, 5])
```

```
iris_pred_svm setosa versicolor virginica
   setosa         50          0         0
   versicolor      0         48         2
   virginica       0          2        48
```

40.6 K 最近邻

```r
# 将 iris3 数据集拆分为训练集和测试集
iris_train <- rbind(iris3[1:25, , 1], iris3[1:25, , 2], iris3[1:25, , 3])
iris_test <- rbind(iris3[26:50, , 1], iris3[26:50, , 2], iris3[26:50, , 3])
iris_species <- factor(rep(c("setosa", "versicolor", "virginica"), each = 25))
```

```r
library(class)
# 分 3 类
iris_knn <- knn(
  train = iris_train, test = iris_test,
  cl = iris_species, k = 3, prob = TRUE
)
# 分类结果汇总
table(iris_knn, iris_species)
```

```
            iris_species
iris_knn     setosa versicolor virginica
  setosa         25          0         0
  versicolor      0         23         4
  virginica       0          2        21
```

统计学习理论 ©

40.7 神经网络

```
library(nnet)
iris_nnet <- nnet(Species ~ ., data = iris, size = 4, trace = FALSE)
summary(iris_nnet)
```

```
a 4-4-3 network with 35 weights
options were - softmax modelling
   b->h1   i1->h1   i2->h1   i3->h1   i4->h1
   11.94   199.26    -9.28   403.26   173.80
   b->h2   i1->h2   i2->h2   i3->h2   i4->h2
   63.22   317.15   213.49   101.40    20.01
   b->h3   i1->h3   i2->h3   i3->h3   i4->h3
   -1.56   -48.91  -233.73   196.57   221.32
   b->h4   i1->h4   i2->h4   i3->h4   i4->h4
-1058.27  -217.60  -233.79   490.81   402.56
   b->o1   h1->o1   h2->o1   h3->o1   h4->o1
  334.96   511.80    44.70 -1399.54  -411.12
   b->o2   h1->o2   h2->o2   h3->o2   h4->o2
  338.12    73.66   325.60   290.07  -569.38
   b->o3   h1->o3   h2->o3   h3->o3   h4->o3
 -672.89  -585.25  -368.82  1109.59   979.35
```

size 隐藏层中的神经元数量

```
iris_pred_nnet <- predict(iris_nnet, newdata = iris[,-5], type = "class")
table(iris_pred_nnet, iris[, 5])
```

```
iris_pred_nnet setosa versicolor virginica
    setosa         50          0         0
    versicolor      0         49         0
    virginica       0          1        50
```

40.8 决策树

```
library(rpart)
iris_rpart <- rpart(Species ~ ., data = iris)
iris_rpart
```

```
n= 150

node), split, n, loss, yval, (yprob)
```

©黄湘云

```
      * denotes terminal node

1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
  2) Petal.Length< 2.45 50   0 setosa (1.00000000 0.00000000 0.00000000) *
  3) Petal.Length>=2.45 100  50 versicolor (0.00000000 0.50000000 0.50000000)
    6) Petal.Width< 1.75 54   5 versicolor (0.00000000 0.90740741 0.09259259) *
    7) Petal.Width>=1.75 46   1 virginica (0.00000000 0.02173913 0.97826087) *
```

```r
library(rpart.plot)
rpart.plot(iris_rpart)
```

图 40.2: 分类回归树

预测结果，训练误差

```r
# 预测
iris_pred_rpart <- predict(iris_rpart, iris[, -5], type = "class")
# 预测结果
table(iris_pred_rpart, iris[, 5])
```

```
iris_pred_rpart setosa versicolor virginica
     setosa         50          0         0
     versicolor      0         49         5
     virginica       0          1        45
```

party 包和 **partykit** 包也提供类似的功能，前者是基于 C 语言实现，后者基于 R 语言实现。

40.9 随机森林

```r
library(randomForest) # 随机森林
iris_rf <- randomForest(
  Species ~ ., data = iris,
  importance = TRUE, proximity = TRUE
)
# 分类结果
print(iris_rf)
```

```
Call:
 randomForest(formula = Species ~ ., data = iris, importance = TRUE,     proximity = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 4%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         50          0         0        0.00
versicolor      0         47         3        0.06
virginica       0          3        47        0.06
```

图 40.3: 随机森林

©黄湘云


```
varImpPlot(iris_rf, main = "变量重要性")
```

图 40.4: 变量重要性

```
iris_pred_rf <- predict(iris_rf, iris[, -5], type = "response")
table(iris_pred_rf, iris[, 5])
```

```
iris_pred_rf setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         50         0
  virginica       0          0        50
```

40.10 集成学习

在训练模型之前，需要先对数据集做预处理，包括分组采样、类别编码、数据拆分、类型转换等。

制作一个函数对数据集添加新列 `mark` 作为训练集 train 和测试集 test 的采样标记，返回数据。

```
# 输入数据 x 和采样比例 prop
add_mark <- function(x = iris, prop = 0.7) {
  idx <- sample(x = nrow(x), size = floor(nrow(x) * prop))
  rbind(
    cbind(x[idx, ], mark = "train"),
    cbind(x[-idx, ], mark = "test")
```

©黄湘云

```
  )
}
```

为了使采样结果可重复，设置随机数种子，然后对 iris 数据集按列 Species 分组添加采样标记，分组随机抽取 70% 的样本作为训练数据，余下的作为测试数据。就 iris 数据集来说，训练集有 35*3 = 105 条记录，测试集有 15*3 = 45 条记录。

```
set.seed(20232023)
iris_df <- do.call(rbind, lapply(split(iris, iris$Species), add_mark, prop = 0.7))
```

为了使用函数 fcase() 对分类变量 Species 做重编码操作，加载 **data.table** 包，将数据集 iris_df 转为 data.table 类型。值得注意，**xgboost** 包要求分类变量的类别序号必须从 0 开始。

```
# 数据准备
library(data.table)
iris_dt <- as.data.table(iris_df)
iris_dt <- iris_dt[, Species := fcase(
  Species == "setosa", 0,
  Species == "versicolor", 1,
  Species == "virginica", 2
)]
```

将数据 iris_dt 拆分成训练集和测试集，并以列表结构存储数据，样本数据及标签以矩阵类型存储。

```
# 训练数据
iris_train <- list(
  data = as.matrix(iris_dt[iris_dt$mark == "train", -c("mark", "Species")]),
  label = as.matrix(iris_dt[iris_dt$mark == "train", "Species"])
)
# 测试数据
iris_test <- list(
  data = as.matrix(iris_dt[iris_dt$mark == "test", -c("mark", "Species")]),
  label = as.matrix(iris_dt[iris_dt$mark == "test", "Species"])
)
```

数据准备好后，加载 **xgboost** 包，设置训练参数，开始训练分类模型。此分类任务中类别超过 2，是多分类任务，学习任务是分类，目标函数可以是 objective = "multi:softprob" 或者 objective = "multi:softmax"，相应的评估指标可以是 eval_metric = "mlogloss" 或者 eval_metric = "merror"。iris 数据集的分类变量 Species 共有 3 类，所以 num_class = 3 。

```
library(xgboost)
iris_xgb <- xgboost(
  data = iris_train$data,
  label = iris_train$label,
  objective = "multi:softmax",  # 学习任务
```

统计学习基础 ©


```
  eval_metric = "mlogloss",     # 评估指标
  nrounds = 2,   # 提升迭代的最大次数
  num_class = 3  # 分类数
)
```

```
[1] train-mlogloss:0.747373
[2] train-mlogloss:0.540389
```

将训练好的模型放在测试集数据上进行预测。

```
# ?predict.xgb.Booster
iris_pred <- predict(object = iris_xgb, newdata = iris_test$data)
```

将预测结果与测试集中的样本标签对比，检查分类效果。

```
table(iris_test$label, iris_pred)
```

```
   iris_pred
     0  1  2
  0 15  0  0
  1  0 14  1
  2  0  2 13
```

40.11 总结

不同的分类算法分布在不同的 R 包中，在使用方式上既有相通之处，又有不同之处。下表对多个 R 包的使用做了归纳。R 包之间的不一致性，计算预测分类的概率的语法。

函数	R 包	代码
lda()	**MASS**	predict(obj)
glm()	**stats**	predict(obj, type = "response")
gbm()	**gbm**	predict(obj, type = "response", n.trees)
naiveBayes()	**e1071**	predict(obj, type = "class")
svm()	**e1071**	predict(obj, probability = FALSE)
ksvm()	**kernlab**	predict(obj, type = "response")
mda()	**mda**	predict(obj, type = "posterior")
rpart()	**rpart**	predict(obj, type = "prob")
Weka()	**RWeka**	predict(obj, type = "probability")
ctree()	**partykit**	predict(obj, type = "response")
bagging()	**ipred**	predict(obj, type = "class")

© 王湘君

40.12 习题

1. **titanic** 包整理了来自 kaggle 的 Titanic 数据集，详细记录了 891 位乘客的信息，它比 Base R 内置的 Titanic 数据集更加原始，细节更多，信息更加丰富。原数据集拆分为训练集 `titanic_train` 和测试集 `titanic_test`。因为有每个乘客的原始信息，我们可以在个体水平上建模，采用更加复杂的模型分析泰坦尼克号乘客存活率及其影响因素。

© 黄湘云

第四十一章　聚类问题

41.1　层次聚类

来自 stats 包的函数 `hclust()` 和来自 cluster 包的函数 `agnes()` 和函数 `diana()` (Kaufman 和 Rousseeuw 1990)

41.2　快速聚类

来自 stats 包的函数 `kmeans()` 和来自 cluster 包的函数 `pam()` ，**kernlab** 包 (Karatzoglou 等 2004) 的 K-means 聚类函数 `kkmeans()` 和谱聚类函数 `specc()` 。

41.3　模糊聚类

来自 e1071 包 (Meyer 等 2023) 的 fuzzy clustering 模糊聚类和 bagged clustering 装袋聚类两种聚类方法

41.4　基于模型的聚类

来自 mclust 包 (Scrucca 等 2016) 有限混合模型

41.5　基于密度的聚类

来自 dbscan 包 (Hahsler, Piekenbrock, 和 Doran 2019)

©黄湘云

第四十二章　回归问题

```
library(MASS)
library(pls)     # PC / PLS
library(glmnet)  # 惩罚回归
library(ncvreg)  # MCP / SCAD
library(lars)    # LAR
library(abess)   # Best subset
library(kernlab) # 基于核的支持向量机 ksvm
library(nnet)    # 神经网络 nnet
library(rpart)   # 决策树
library(randomForest)  # 随机森林
library(xgboost)       # 梯度提升
library(lattice)
# Root Mean Squared Error 均方根误差
rmse <- function(y, y_pred) {
  sqrt(mean((y - y_pred)^2))
}
```

本章基于波士顿郊区房价数据集 Boston 介绍处理回归问题的 10 种方法。数据集 Boston 来自 R 软件内置的 **MASS** 包，一共 506 条记录 14 个变量，由 Boston Standard Metropolitan Statistical Area (SMSA) 在 1970 年收集。

```
data("Boston", package = "MASS")
str(Boston)
```

```
#> 'data.frame':    506 obs. of  14 variables:
#>  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
#>  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
#>  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
#>  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
#>  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
```

© 基础统计


```
#>  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
#>  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
#>  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
#>  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
#>  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
#>  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
#>  $ black  : num  397 397 393 395 397 ...
#>  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
#>  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

14 个变量的含义如下：

- crim: 城镇人均犯罪率 per capita crime rate by town
- zn: 占地面积超过 25,000 平方尺的住宅用地比例 proportion of residential land zoned for lots over 25,000 sq.ft.
- indus: 每个城镇非零售业务的比例 proportion of non-retail business acres per town.
- chas: 查尔斯河 Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
- nox: 氮氧化物浓度 nitrogen oxides concentration (parts per 10 million).
- rm: 每栋住宅的平均房间数量 average number of rooms per dwelling. 容积率
- age: 1940 年以前建造的自住单位比例 proportion of owner-occupied units built prior to 1940. 房龄
- dis: 到波士顿五个就业中心的加权平均值 weighted mean of distances to five Boston employment centres. 商圈
- rad: 径向高速公路可达性指数 index of accessibility to radial highways. 交通
- tax: 每 10,000 美元的全额物业税率 full-value property-tax rate per $10,000. 物业
- ptratio: 城镇的师生比例 pupil-teacher ratio by town. 教育
- black: 城镇黑人比例 $1000(Bk-0.63)^2$ where Bk is the proportion of blacks by town. 安全
- lstat: 较低的人口状况（百分比）lower status of the population (percent).
- medv: 自住房屋的中位数为 1000 美元 median value of owner-occupied homes in $1000s. 房价，这是响应变量。

42.1 线性回归

对于线性回归问题，为了处理变量之间的相关关系，衍生出许多处理办法。有的办法是线性的，有的办法是非线性的。

42.1.1 最小二乘回归

$$\mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2$$

```
fit_lm <- lm(medv ~ ., data = Boston)
summary(fit_lm)
```

```
#>
#> Call:
#> lm(formula = medv ~ ., data = Boston)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -15.595  -2.730  -0.518   1.777  26.199
#>
#> Coefficients:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
#> crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
#> zn           4.642e-02  1.373e-02   3.382 0.000778 ***
#> indus        2.056e-02  6.150e-02   0.334 0.738288
#> chas         2.687e+00  8.616e-01   3.118 0.001925 **
#> nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
#> rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
#> age          6.922e-04  1.321e-02   0.052 0.958229
#> dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
#> rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
#> tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
#> ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
#> black        9.312e-03  2.686e-03   3.467 0.000573 ***
#> lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 4.745 on 492 degrees of freedom
#> Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
#> F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

42.1.2 逐步回归

逐步回归是筛选变量，有向前、向后和两个方向同时进行三个方法。

- `direction = "both"` 双向
- `direction = "backward"` 向后
- `direction = "forward"` 向前

```
fit_step <- step(fit_lm, direction = "both", trace = 0)
summary(fit_step)
```

```
#>
#> Call:
#> lm(formula = medv ~ crim + zn + chas + nox + rm + dis + rad +
#>     tax + ptratio + black + lstat, data = Boston)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -15.5984  -2.7386  -0.5046   1.7273  26.2373
#>
#> Coefficients:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  36.341145   5.067492   7.171 2.73e-12 ***
#> crim         -0.108413   0.032779  -3.307 0.001010 **
#> zn            0.045845   0.013523   3.390 0.000754 ***
#> chas          2.718716   0.854240   3.183 0.001551 **
#> nox         -17.376023   3.535243  -4.915 1.21e-06 ***
#> rm            3.801579   0.406316   9.356  < 2e-16 ***
#> dis          -1.492711   0.185731  -8.037 6.84e-15 ***
#> rad           0.299608   0.063402   4.726 3.00e-06 ***
#> tax          -0.011778   0.003372  -3.493 0.000521 ***
#> ptratio      -0.946525   0.129066  -7.334 9.24e-13 ***
#> black         0.009291   0.002674   3.475 0.000557 ***
#> lstat        -0.522553   0.047424 -11.019  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 4.736 on 494 degrees of freedom
#> Multiple R-squared:  0.7406, Adjusted R-squared:  0.7348
#> F-statistic: 128.2 on 11 and 494 DF,  p-value: < 2.2e-16
```

42.1.3 偏最小二乘回归

偏最小二乘回归适用于存在多重共线性问题或变量个数远大于样本量的情况。

10 折交叉验证，`ncomp = 6` 表示 6 个主成分，拟合方法 `kernelpls` 表示核算法，`validation = "CV"` 表示采用交叉验证的方式调整参数。

```
fit_pls <- pls::plsr(medv ~ ., ncomp = 6, data = Boston, validation = "CV")
summary(fit_pls)
```

```
#> Data:    X dimension: 506 13
#>  Y dimension: 506 1
#> Fit method: kernelpls
#> Number of components considered: 6
#>
#> VALIDATION: RMSEP
#> Cross-validated using 10 random segments.
#>        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
#> CV           9.206    8.073    7.934    7.680    6.636    5.991    5.854
#> adjCV        9.206    8.069    7.936    7.675    6.625    5.981    5.844
#>
#> TRAINING: % variance explained
#>       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
#> X       80.51    94.45    98.97    99.34    99.80    99.91
#> medv    24.23    26.94    32.05    51.05    60.08    62.49
```

交叉验证的方法还可选留一交叉验证 `validation = "LOO"` 。预测的均方根误差 RMSEP 来评估交叉验证的结果。

```
pls::validationplot(fit_pls, val.type = "RMSEP")
```

©黄湘云

图 42.1: RMSE 随成分数量的变化

42.1.4 主成分回归

主成分回归采用降维的方法处理高维和多重共线性问题。

10 折交叉验证，6 个主成分，拟合方法 svdpc 表示奇异值分解算法。

```
fit_pcr <- pls::pcr(medv ~ ., ncomp = 6, data = Boston, validation = "CV")
summary(fit_pcr)
```

```
#> Data:    X dimension: 506 13
#>  Y dimension: 506 1
#> Fit method: svdpc
#> Number of components considered: 6
#>
#> VALIDATION: RMSEP
#> Cross-validated using 10 random segments.
#>        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
#> CV           9.206    8.060    8.037    7.807    7.804    7.645    6.122
#> adjCV        9.206    8.058    8.035    7.804    7.800    7.652    6.113
```

©李漫 文稿

```
#>
#> TRAINING: % variance explained
#>       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
#> X       80.58    96.89    99.02    99.72    99.85    99.92
#> medv    23.71    24.28    28.77    29.33    32.71    57.77
```

42.2 惩罚回归

本节主要介绍 4 个 R 包的使用，分别是 **glmnet** 包 (Friedman, Tibshirani, 和 Hastie 2010)、**ncvreg** 包 (Breheny 和 Huang 2011) 、**lars** 包 (Bradley Efron 和 Tibshirani 2004) 和 **abess** 包 (Zhu 等 2022)。

表格 42.1: 惩罚回归的 R 包实现

R 包	惩罚方法	函数实现
glmnet	岭回归	`glmnet(...,alpha = 0)`
glmnet	Lasso 回归	`glmnet(...,alpha = 1)`
glmnet	弹性网络回归	`glmnet(...,alpha)`
glmnet	自适应 Lasso 回归	`glmnet(...,penalty.factor)`
glmnet	松驰 Lasso 回归	`glmnet(...,relax = TRUE)`
ncvreg	MCP	`ncvreg(...,penalty = "MCP")`
ncvreg	SCAD	`ncvreg(...,penalty = "SCAD")`
lars	最小角回归	`lars(...,type = "lar")`
abess	最优子集回归	`abess()`

函数 `glmnet()` 的参数 `penalty.factor` 表示惩罚因子，默认值为全 1 向量，自适应 Lasso 回归中需要指定。弹性网络回归要求参数 `alpha` 介于 0-1 之间。

42.2.1 岭回归

岭回归

$$\mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2 + \lambda\|\boldsymbol{\beta}\|_2^2$$

```r
library(glmnet)
fit_ridge <- glmnet(x = Boston[, -14], y = Boston[, "medv"], family = "gaussian", alpha = 0)

plot(fit_ridge)
plot(fit_ridge$lambda,
  ylab = expression(lambda), xlab = "迭代次数", main = "惩罚系数的迭代路径"
)
```

© 黄湘云

(a) 回归系数的迭代路径

(b) 惩罚系数的迭代路径

图 42.2: 岭回归

```
fit_ridge$lambda[60]

#> [1] 28.00535

coef(fit_ridge, s = 28.00535)

#> 14 x 1 sparse Matrix of class "dgCMatrix"
#>                       s1
#> (Intercept) 23.047750109
#> crim        -0.045815821
#> zn           0.014330186
#> indus       -0.063634086
#> chas         1.358311700
#> nox         -3.075514644
#> rm           1.653490217
#> age         -0.009926222
#> dis         -0.025465898
#> rad         -0.026390778
#> tax         -0.002435665
#> ptratio     -0.331740062
#> black        0.004145613
#> lstat       -0.151396406
```

©黄湘云

42.2.2 Lasso 回归

Lasso 回归

$$\mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2 + \lambda\|\boldsymbol{\beta}\|_1$$

```
fit_lasso <- glmnet(x = Boston[, -14], y = Boston[, "medv"], family = "gaussian", alpha = 1)

plot(fit_lasso)
plot(fit_lasso$lambda,
  ylab = expression(lambda), xlab = "迭代次数",
  main = "惩罚系数的迭代路径"
)
```

(a) 回归系数的迭代路径

(b) 惩罚系数的迭代路径

图 42.3: Lasso 回归

```
fit_lasso$lambda[60]

#> [1] 0.02800535

coef(fit_lasso, s = 0.02800535)

#> 14 x 1 sparse Matrix of class "dgCMatrix"
#>                        s1
#> (Intercept)  34.426424733
#> crim         -0.098346337
#> zn            0.041441612
```

©黄湘云

```
#> indus         .
#> chas          2.685187735
#> nox         -16.306645191
#> rm            3.866938879
#> age           .
#> dis          -1.396021610
#> rad           0.252686499
#> tax          -0.009826799
#> ptratio      -0.929988657
#> black         0.009025875
#> lstat        -0.522499839
```

42.2.3 弹性网络

弹性网络 (Zou 和 Hastie 2005)

$$\mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2 + \lambda(\frac{1-\alpha}{2}\|\boldsymbol{\beta}\|_2^2 + \alpha\|\boldsymbol{\beta}\|_1)$$

```
fit_elasticnet <- glmnet(x = Boston[, -14], y = Boston[, "medv"], family = "gaussian")
```

```
plot(fit_elasticnet)
plot(fit_elasticnet$lambda,
  ylab = expression(lambda), xlab = "迭代次数",
  main = "惩罚系数的迭代路径"
)
```

```
fit_elasticnet$lambda[60]
```

```
#> [1] 0.02800535
```

```
coef(fit_elasticnet, s = 0.02800535)
```

```
#> 14 x 1 sparse Matrix of class "dgCMatrix"
#>                        s1
#> (Intercept)  34.426424733
#> crim         -0.098346337
#> zn            0.041441612
#> indus         .
#> chas          2.685187735
#> nox         -16.306645191
#> rm            3.866938879
#> age           .
#> dis          -1.396021610
```

©黄湘云

(a) 回归系数的迭代路径

(b) 惩罚系数的迭代路径

图 42.4: 弹性网络

```
#> rad           0.252686499
#> tax          -0.009826799
#> ptratio      -0.929988657
#> black         0.009025875
#> lstat        -0.522499839
```

42.2.4 自适应 Lasso

自适应 Lasso (Zou 2006)

$$\mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2 + \lambda_n \sum_{j=1}^{p} \frac{1}{w_j}|\beta_j|$$

普通最小二乘估计或岭回归估计的结果作为适应性 Lasso 回归的权重。其中 $w_j = (|\hat{\beta}_{ols_j}|)^\gamma$ 或 $w_j = (|\hat{\beta}_{ridge_j}|)^\gamma$ ，γ 是一个用于调整自适应权重向量的正常数，一般建议的正常数是 0.5，1 或 2。

```
# 岭权重 gamma = 1
g <- 1
set.seed(20232023)
## 岭回归
ridge_model <- cv.glmnet(
  x = as.matrix(Boston[, -14]),
```

©黄湘云

```
  y = Boston[, 14], alpha = 0
)
ridge_coef <- as.matrix(coef(ridge_model, s = ridge_model$lambda.min))
ridge_weight <- 1 / (abs(ridge_coef[-1, ]))^g

## Adaptive Lasso
set.seed(20232023)
fit_adaptive_lasso <- cv.glmnet(
  x = as.matrix(Boston[, -14]),
  y = Boston[, 14], alpha = 1,
  penalty.factor = ridge_weight # 惩罚权重
)
```

岭回归和自适应 Lasso 回归模型的超参数

```
plot(ridge_model)
plot(fit_adaptive_lasso)
```

(a) 岭回归 (b) 自适应 Lasso 回归

图 42.5: 自适应 Lasso 回归模型的超参数选择

λ 超参数

```
fit_adaptive_lasso$lambda.min
```

```
#> [1] 0.2273152
```

自适应 Lasso 回归参数

```
coef(fit_adaptive_lasso, s = fit_adaptive_lasso$lambda.min)
```

```
#> 14 x 1 sparse Matrix of class "dgCMatrix"
#>                        s1
#> (Intercept)  38.291419779
#> crim         -0.098901950
#> zn            0.023328430
#> indus        -0.016769750
#> chas          3.119585761
#> nox         -20.461629406
#> rm            3.946726706
#> age           .
#> dis          -1.354180874
#> rad           0.100046239
#> tax           .
#> ptratio      -1.019940695
#> black         0.002119703
#> lstat        -0.545149921
```

预测

```
pred_medv_adaptive_lasso <- predict(
  fit_adaptive_lasso, newx = as.matrix(Boston[, -14]),
  s = fit_adaptive_lasso$lambda.min, type = "response"
)
```

预测的均方根误差

```
rmse(Boston[, 14], pred_medv_adaptive_lasso)
```

```
#> [1] 4.77706
```

42.2.5 松弛 Lasso

Lasso 回归倾向于将回归系数压缩到 0，松弛 Lasso

$$\hat{\beta}_{relax}(\lambda, \gamma) = \gamma \hat{\beta}_{lasso}(\lambda) + (1 - \gamma) \hat{\beta}_{ols}(\lambda)$$

其中，$\gamma \in [0, 1]$ 是一个超参数。

```
fit_relax_lasso <- cv.glmnet(
  x = as.matrix(Boston[, -14]),
  y = Boston[, "medv"], relax = TRUE
)
```

```
plot(fit_relax_lasso)
```

©黄湘云

图 42.6: 回归系数的迭代路径

CV 交叉验证筛选出来的超参数 λ 和 γ ，$\gamma = 0$ 意味着松弛 Lasso 退化为 OLS 估计

```
fit_relax_lasso$relaxed$lambda.min
```

```
#> [1] 0.1240811
```

```
fit_relax_lasso$relaxed$gamma.min
```

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

松弛 Lasso 回归系数与 OLS 估计的结果一样

```
coef(fit_relax_lasso, s = "lambda.min", gamma = "gamma.min")
```

```
#> 14 x 1 sparse Matrix of class "dgCMatrix"
#>                        s1
#> (Intercept)  36.386415340
#> crim         -0.107642467
#> zn            0.046225884
#> indus         0.019914638
```

```
#> chas          2.692467531
#> nox         -17.703655696
#> rm            3.817657573
#> age           .
#> dis          -1.478133649
#> rad           0.303685310
#> tax          -0.012233266
#> ptratio      -0.951640287
#> black         0.009315797
#> lstat        -0.523702685
```

松弛 Lasso 预测

```
pred_medv_relax_lasso <- predict(
  fit_relax_lasso,
  newx = as.matrix(Boston[, -14]),
  s = "lambda.min", gamma = "gamma.min"
)
```

```
rmse(Boston[, 14], pred_medv_relax_lasso)
```

```
#> [1] 4.679209
```

42.2.6 MCP

ncvreg 包 (Breheny 和 Huang 2011) 提供额外的两种非凸/凹惩罚类型，分别是 MCP（minimax concave penalty）和 SCAD（smoothly clipped absolute deviation）。

```
library(ncvreg)
fit_mcp <- ncvreg(X = Boston[, -14], y = Boston[, "medv"], penalty = "MCP")
```

```
plot(fit_mcp)
```

©黄湘云

图 42.7: 回归系数的迭代路径

回归系数

```
coef(fit_mcp, lambda = 0.85)
```

```
#> (Intercept)        crim          zn       indus        chas         nox
#>  14.9613035   0.0000000   0.0000000   0.0000000   0.2355167   0.0000000
#>          rm         age         dis         rad         tax     ptratio
#>   4.6134961   0.0000000   0.0000000   0.0000000   0.0000000  -0.7607830
#>       black       lstat
#>   0.0000000  -0.5847017
```

```
summary(fit_mcp, lambda = 0.85)
```

```
#> Using a basic kernel estimate for local fdr; consider installing the ashr package for more accurate e
local_mfdr
```

```
#> MCP-penalized linear regression with n=506, p=13
#> At lambda=0.8500:
#> ---------------------------------------------------
#>   Nonzero coefficients         :   4
#>   Expected nonzero coefficients:   0.01
#>   Average mfdr (4 features)    :   0.001
#>
```

```
#>         Estimate       z      mfdr Selected
#> lstat    -0.5847 -17.956   < 1e-04        *
#> rm        4.6135  13.940   < 1e-04        *
#> ptratio  -0.7608  -8.381   < 1e-04        *
#> chas      0.2355   3.831 0.0051025        *
```

10 折交叉验证，选择超参数 λ

```
fit_mcp_cv <- cv.ncvreg(
  X = Boston[, -14], y = Boston[, "medv"],
  penalty = "MCP", seed = 20232023
)
summary(fit_mcp_cv)
```

```
#> MCP-penalized linear regression with n=506, p=13
#> At minimum cross-validation error (lambda=0.1800):
#> -------------------------------------------------
#>   Nonzero coefficients: 11
#>   Cross-validation error (deviance): 23.45
#>   R-squared: 0.72
#>   Signal-to-noise ratio: 2.60
#>   Scale estimate (sigma): 4.843
#> MCP-penalized linear regression with n=506, p=13
#> At lambda=0.1800:
#> --------------------------------------------------
#>   Nonzero coefficients         :  11
#>   Expected nonzero coefficients:   0.08
#>   Average mfdr (11 features)   :   0.007
#>
#>          Estimate       z      mfdr Selected
#> lstat    -0.52253 -17.314   < 1e-04        *
#> dis      -1.49319 -14.590   < 1e-04        *
#> rm        3.80092  12.392   < 1e-04        *
#> rad       0.29997  12.118   < 1e-04        *
#> ptratio  -0.94664  -9.510   < 1e-04        *
#> nox     -17.38650  -9.347   < 1e-04        *
#> tax      -0.01179  -9.220   < 1e-04        *
#> zn        0.04587   4.963   < 1e-04        *
#> crim     -0.10852  -4.330 0.0010795        *
#> black     0.00929   3.936 0.0053435        *
#> chas      2.71850   3.204 0.0713275        *
```

© 王兆军

在 $\lambda = 0.1362$ 时，交叉验证的误差最小，非 0 回归系数 11 个。

```
plot(fit_mcp_cv)
```

图 42.8: 惩罚系数的迭代路径

42.2.7 SCAD

```
fit_scad <- ncvreg(X = Boston[, -14], y = Boston[, "medv"], penalty = "SCAD")
```

```
plot(fit_scad)
```

©黄湘云

图 42.9: 回归系数的迭代路径

```
coef(fit_scad, lambda = 0.85)
```

```
#>   (Intercept)          crim            zn         indus          chas
#>  9.3713059437  0.0000000000  0.0000000000  0.0000000000  0.3518918853
#>           nox            rm           age           dis           rad
#>  0.0000000000  4.7729149463  0.0000000000  0.0000000000  0.0000000000
#>           tax       ptratio         black         lstat
#>  0.0000000000 -0.5040003090  0.0002038813 -0.6030152355
```

```
summary(fit_scad, lambda = 0.85)
```

```
#> SCAD-penalized linear regression with n=506, p=13
#> At lambda=0.8500:
#> -----------------------------------------------
#>   Nonzero coefficients         :   5
#>   Expected nonzero coefficients:   0.01
#>   Average mfdr (5 features)    :   0.002
#>
#>           Estimate       z      mfdr Selected
#> lstat   -0.6030152 -18.329   < 1e-04        *
#> rm       4.7729149  14.274   < 1e-04        *
#> ptratio -0.5040003  -7.888   < 1e-04        *
```

```
#> chas     0.3518919   4.002 0.0027534        *
#> black    0.0002039   3.673 0.0093789        *
```

10 折交叉验证，选择超参数 λ

```
fit_scad_cv <- cv.ncvreg(
  X = Boston[, -14], y = Boston[, "medv"],
  penalty = "SCAD", seed = 20232023
)
summary(fit_scad_cv)
```

```
#> SCAD-penalized linear regression with n=506, p=13
#> At minimum cross-validation error (lambda=0.1362):
#> ---------------------------------------------------
#>   Nonzero coefficients: 11
#>   Cross-validation error (deviance): 23.45
#>   R-squared: 0.72
#>   Signal-to-noise ratio: 2.60
#>   Scale estimate (sigma): 4.843
#> SCAD-penalized linear regression with n=506, p=13
#> At lambda=0.1362:
#> ---------------------------------------------------
#>   Nonzero coefficients         :  11
#>   Expected nonzero coefficients:   0.08
#>   Average mfdr (11 features)   :   0.007
#>
#>           Estimate       z      mfdr Selected
#> lstat    -0.522521 -17.314   < 1e-04        *
#> dis      -1.492829 -14.586   < 1e-04        *
#> rm        3.801459  12.393   < 1e-04        *
#> rad       0.299790  12.111   < 1e-04        *
#> ptratio  -0.946635  -9.509   < 1e-04        *
#> nox     -17.381556  -9.345   < 1e-04        *
#> tax      -0.011784  -9.215   < 1e-04        *
#> zn        0.045846   4.961   < 1e-04        *
#> crim     -0.108459  -4.328 0.0010887        *
#> black     0.009291   3.936 0.0053408        *
#> chas      2.718640   3.204 0.0712933        *
```

在 $\lambda = 0.1362$ 时，交叉验证的误差最小，非 0 回归系数 11 个。

```
plot(fit_scad_cv)
```

© 黄湘云

图 42.10: 惩罚系数的迭代路径

42.2.8 最小角回归

lars 包提供 Lasso 回归和最小角（Least Angle）回归 (Bradley Efron 和 Tibshirani 2004)。

```
library(lars)
# Lasso 回归
fit_lars_lasso <- lars(
  x = as.matrix(Boston[, -14]), y = as.matrix(Boston[, "medv"]),
  type = "lasso", trace = FALSE, normalize = TRUE, intercept = TRUE
)
# LAR 回归
fit_lars_lar <- lars(
  x = as.matrix(Boston[, -14]), y = as.matrix(Boston[, "medv"]),
  type = "lar", trace = FALSE, normalize = TRUE, intercept = TRUE
)
```

参数 `type = "lasso"` 表示采用 Lasso 回归，参数 `trace = FALSE` 表示不显示迭代过程，参数 `normalize = TRUE` 表示每个变量都标准化，使得它们的 L2 范数为 1，参数 `intercept = TRUE` 表示模型中包含截

© 王贤勇

距项，且不参与惩罚。

Lasso 和最小角回归系数的迭代路径见下图。

```
plot(fit_lars_lasso)
plot(fit_lars_lar)
```

(a) Lasso 回归　　(b) 最小角回归

图 42.11: Lasso 和最小角回归系数的迭代路径

采用 10 折交叉验证筛选变量

```
set.seed(20232023)
cv.lars(
  x = as.matrix(Boston[, -14]), y = as.matrix(Boston[, "medv"]),
  type = "lasso", trace = FALSE, plot.it = TRUE, K = 10
)
set.seed(20232023)
cv.lars(
  x = as.matrix(Boston[, -14]), y = as.matrix(Boston[, "medv"]),
  type = "lar", trace = FALSE, plot.it = TRUE, K = 10
)
```

42.2.9 最优子集回归

$$\mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2 + \lambda\|\boldsymbol{\beta}\|_0$$

©黄湘云

(a) Lasso 回归 (b) 最小角回归

图 42.12: 交叉验证均方误差的变化

最优子集回归，添加 L0 惩罚，abess 包 (Zhu 等 2022) 支持线性回归、泊松回归、逻辑回归、多项回归等模型，可以非常高效地做最优子集筛选变量。

```
library(abess)
fit_abess <- abess(medv ~ ., data = Boston, family = "gaussian",
                   tune.type = "cv", nfolds = 10, seed = 20232023)
```

参数 `tune.type = "cv"` 表示交叉验证的方式确定超参数来筛选变量，参数 `nfolds = 10` 表示将数据划分为 10 份，采用 10 折交叉验证，参数 `seed` 用来设置随机数，以便可重复交叉验证 CV 的结果。惩罚系数的迭代路径见下左图。使用交叉验证筛选变量个数，不同的 support size 表示进入模型中的变量数目。

```
plot(fit_abess, label = TRUE, main = "惩罚系数的迭代路径")
plot(fit_abess, type = "tune", main = "交叉验证筛选变量个数")
```

从上右图可以看出，选择 6 个变量是比较合适的，作为最终的模型。

```
best_model <- extract(fit_abess, support.size = 6)
# 模型的结果，惩罚参数值、各个变量的系数
str(best_model)
```

```
#> List of 7
#>  $ beta        :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
#>   .. ..@ i       : int [1:6] 3 4 5 7 10 12
#>   .. ..@ p       : int [1:2] 0 6
#>   .. ..@ Dim     : int [1:2] 13 1
#>   .. ..@ Dimnames:List of 2
```

©黄湘云

(a) 惩罚系数的迭代路径 (b) 交叉验证筛选变量个数

图 42.13: 最优子集回归

```
#>   .. .. ..$ : chr [1:13] "crim" "zn" "indus" "chas" ...
#>   .. .. ..$ : chr "6"
#>   .. ..@ x       : num [1:6] 3.24 -18.74 4.11 -1.14 -1 ...
#>   .. ..@ factors : list()
#>  $ intercept   : num 36.9
#>  $ support.size: num 6
#>  $ support.vars: chr [1:6] "chas" "nox" "rm" "dis" ...
#>  $ support.beta: num [1:6] 3.24 -18.74 4.11 -1.14 -1 ...
#>  $ dev         : num 12
#>  $ tune.value  : num 12.9
```

42.3 支持向量机

```
library(kernlab)
fit_ksvm <- ksvm(medv ~ ., data = Boston)
fit_ksvm
```

```
#> Support Vector Machine object of class "ksvm"
#>
#> SV type: eps-svr  (regression)
#>  parameter : epsilon = 0.1  cost C = 1
#>
```

©黄文强

```
#> Gaussian Radial Basis kernel function.
#>  Hyperparameter : sigma =  0.128519353805178
#>
#> Number of Support Vectors : 333
#>
#> Objective Function Value : -78.7942
#> Training error : 0.091636
```

```r
# 预测
pred_medv_svm <- predict(fit_ksvm, newdata = Boston)
# RMSE
rmse(Boston$medv, pred_medv_svm)
```

```
#> [1] 2.78409
```

42.4 神经网络

单隐藏层的神经网络

```r
library(nnet)
fit_nnet <- nnet(medv ~ .,
  data = Boston, trace = FALSE,
  size = 12, # 隐藏层单元数量
  maxit = 500, # 最大迭代次数
  linout = TRUE, # 线性输出单元
  decay = 0.01 # 权重下降的参数
)
pred_medv_nnet <- predict(fit_nnet, newdata = Boston[, -14], type = "raw")
rmse(Boston$medv, pred_medv_nnet)
```

```
#> [1] 3.030015
```

42.5 决策树

```r
library(rpart)
fit_rpart <- rpart(medv ~ .,
  data = Boston, control = rpart.control(minsplit = 5)
)

pred_medv_rpart <- predict(fit_rpart, newdata = Boston[, -14])
```

©黄湘云

```
rmse(Boston$medv, pred_medv_rpart)

#> [1] 3.565888

library(rpart.plot)
rpart.plot(fit_rpart)
```

图 42.14: 分类回归树

42.6 随机森林

```
library(randomForest)
fit_rf <- randomForest(medv ~ ., data = Boston)
print(fit_rf)

#>
#> Call:
#>  randomForest(formula = medv ~ ., data = Boston)
#>                Type of random forest: regression
#>                      Number of trees: 500
#> No. of variables tried at each split: 4
#>
#>           Mean of squared residuals: 10.17194
#>                     % Var explained: 87.95
```

©黄湘云

```
pred_medv_rf <- predict(fit_rf, newdata = Boston[, -14])
rmse(Boston$medv, pred_medv_rf)

#> [1] 1.404252
```

42.7 集成学习

```
# 输入数据 x 和采样比例 prop
add_mark <- function(x = Boston, prop = 0.7) {
  idx <- sample(x = nrow(x), size = floor(nrow(x) * prop))
  rbind(
    cbind(x[idx, ], mark = "train"),
    cbind(x[-idx, ], mark = "test")
  )
}

set.seed(20232023)
Boston_df <- add_mark(Boston, prop = 0.7)

library(data.table)
Boston_dt <- as.data.table(Boston_df)

# 训练数据
Boston_train <- list(
  data = as.matrix(Boston_dt[Boston_dt$mark == "train", -c("mark", "medv")]),
  label = as.matrix(Boston_dt[Boston_dt$mark == "train", "medv"])
)
# 测试数据
Boston_test <- list(
  data = as.matrix(Boston_dt[Boston_dt$mark == "test", -c("mark", "medv")]),
  label = as.matrix(Boston_dt[Boston_dt$mark == "test", "medv"])
)

library(xgboost)
Boston_xgb <- xgboost(
  data = Boston_train$data,
  label = Boston_train$label,
  objective = "reg:squarederror",  # 学习任务
  eval_metric = "rmse",    # 评估指标
  nrounds = 6
```

统计学习基础 ©

```
)
#> [1]  train-rmse:17.424982
#> [2]  train-rmse:12.641765
#> [3]  train-rmse:9.241521
#> [4]  train-rmse:6.833056
#> [5]  train-rmse:5.139463
#> [6]  train-rmse:3.949495
```

```
# ?predict.xgb.Booster
Boston_pred <- predict(object = Boston_xgb, newdata = Boston_test$data)
# RMSE
rmse(Boston_test$label, Boston_pred)
```

```
#> [1] 4.509274
```

© 草稿

参考文献

Abril-Pla O, Carroll C, Andreani V. 2023. 《PyMC: a modern, and comprehensive probabilistic programming framework in Python》. *PeerJ Computer Science* 9 (e1516). https://doi.org/10.7717/peerj-cs.1516.

Agresti, Alan. 2007. *An Introduction to Categorical Data Analysis*. 2nd 本. Hoboken, New Jersey: John Wiley & Sons, Inc.

Allaire, JJ, Romain Francois, Kevin Ushey, Gregory Vandenbrouck, Marcus Geelnard, 和 Intel. 2023. *RcppParallel: Parallel Programming Tools for Rcpp*. https://CRAN.R-project.org/package=RcppParallel.

Ansari, A. R., 和 R. A. Bradley. 1960. 《Rank-Sum Tests for Dispersions》. *The Annals of Mathematical Statistics* 31 (4): 1174–89. https://doi.org/10.1214/aoms/1177705688.

Anscombe, F. J. 1973. 《Graphs in Statistical Analysis》. *The American Statistician* 27 (1): 17. https://doi.org/10.2307/2682899.

Arnold, Jeffrey B. 2021. *ggthemes: Extra Themes, Scales and Geoms for ggplot2*. https://CRAN.R-project.org/package=ggthemes.

Attali, Dean, 和 Christopher Baker. 2022. *ggExtra: Add Marginal Histograms to ggplot2, and More ggplot2 Enhancements*. https://CRAN.R-project.org/package=ggExtra.

Bai, H., L. Wang, W. Pan, 和 M. Frey. 2009. 《Measuring mathematics anxiety: Psychometric analysis of a bidimensional affective scale》. *Journal of Instructional Psychology* 36 (3): 185–93.

Bates, Douglas M., 和 Donald G. Watts. 1988. *Nonlinear Regression Analysis and Its Applications*. New York, NY: John Wiley & Sons. https://doi.org/10.1002/9780470316757.app2.

Bates, Douglas, 和 Dirk Eddelbuettel. 2013. 《Fast and Elegant Numerical Linear Algebra Using the RcppEigen Package》. *Journal of Statistical Software* 52 (5): 1–24. https://doi.org/10.18637/jss.v052.i05.

Bates, Douglas, Martin Mächler, Ben Bolker, 和 Steve Walker. 2015. 《Fitting Linear Mixed-Effects Models Using lme4》. *Journal of Statistical Software* 67 (1): 1–48. https://doi.org/10.18637/jss.v067.i01.

Berkelaar, Michel 等. 2023. *lpSolve: Interface to Lp_solve v. 5.5 to Solve Linear/Integer Programs*. https://CRAN.R-project.org/package=lpSolve.

Bhadra, Anindya, Jyotishka Datta, Nicholas G. Polson, 和 Brandon Willard. 2019. 《Lasso Meets Horseshoe: A Survey》. *Statistical Science* 34 (3): 405–27. https://doi.org/10.1214/19-STS700.

内部使用 ©

Bickel, P. J., E. A. Hammel, 和 J. W. O'Connell. 1975. 《Sex Bias in Graduate Admissions: Data from Berkeley》. *Science* 187 (4175): 398–404. https://doi.org/10.1126/science.187.4175.398.

Biecek, Przemyslaw. 2023. *DrWhy: Explain, Explore and Debug Predictive Machine Learning Models*. https://github.com/ModelOriented/DrWhy.

Bion, Ricardo. 2018. *ggtech: ggplot2 tech themes and scales*.

Bivand, Roger. 2001. 《More on Spatial Data Analysis》. *R News* 1 (3): 13–17. https://www.r-project.org/doc/Rnews/Rnews_2001-3.pdf.

Blangiardo, Marta, Michela Cameletti, Gianluca Baio, 和 Håvard Rue. 2013. 《Spatial and spatio-temporal models with R-INLA》. *Spatial and Spatio-temporal Epidemiology* 7 (十二月): 39–55. https://doi.org/10.1016/j.sste.2013.07.003.

Blyth, Colin R., 和 David W. Hutchinson. 1960. 《Table of Neyman-Shortest Unbiased Confidence Intervals for the Binomial Parameter》. *Biometrika* 47 (3/4): 381–91. https://www.jstor.org/stable/2333308.

Box, G. E. P., 和 D. R. Cox. 1964. 《An analysis of transformations (with discussion)》. *Journal of the Royal Statistical Society, Series B* 26 (2): 211–52. https://www.jstor.org/stable/2984418.

Bradley Efron, Iain Johnstone, Trevor Hastie, 和 Robert Tibshirani. 2004. 《Least Angle Regression》. *Annals of Statistics* 32 (2): 407–99. https://hastie.su.domains/Papers/LARS/LeastAngle_2002.pdf.

Brandao, Filipe. 2023. *rAMPL: AMPL API for R*. https://github.com/ampl/rAMPL.

Breheny, Patrick, 和 Jian Huang. 2011. 《Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection》. *Annals of Applied Statistics* 5 (1): 232–53. https://doi.org/10.1214/10-AOAS388.

Brown, Lawrence D., T. Tony Cai, 和 Anirban DasGupta. 2001. 《Interval Estimation for a Binomial Proportion》. *Statistical Science*, 期 2: 101–33. https://projecteuclid.org/euclid.ss/1009213286.

Brunson, Jason Cory. 2020. 《ggalluvial: Layered Grammar for Alluvial Plots》. *Journal of Open Source Software* 5 (49): 2017. https://doi.org/10.21105/joss.02017.

Bürkner, Paul-Christian. 2017. 《brms: An R Package for Bayesian Multilevel Models Using Stan》. *Journal of Statistical Software* 80 (1): 1–28. https://doi.org/10.18637/jss.v080.i01.

Cabral, Rafael, David Bolin, 和 Håvard Rue. 2022. 《Controlling the Flexibility of Non-Gaussian Processes Through Shrinkage Priors》. *Bayesian Analysis* -1 (-1): 1–24. https://doi.org/10.1214/22-BA1342.

Carpenter, Bob, Andrew Gelman, Matthew Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, 和 Allen Riddell. 2017. 《Stan: A Probabilistic Programming Language》. *Journal of Statistical Software* 76 (1): 1–32. https://doi.org/10.18637/jss.v076.i01.

Carpenter, Bob, Matthew D. Hoffman, Marcus Brubaker, Daniel Lee, Peter Li, 和 Michael Betancourt. 2015. 《The Stan Math Library: Reverse-Mode Automatic Differentiation in C++》. https://arxiv.org/abs/1509.07164.

Christensen, O. F., 和 P. J. Ribeiro Jr. 2002. 《geoRglm: A package for generalised linear spatial models》. *R News* 2 (2): 26–28.

Chung, Yeojin, Sophia Rabe-Hesketh, Vincent Dorie, Andrew Gelman, 和 Jingchen Liu. 2013. 《A nondegenerate penalized likelihood estimator for variance parameters in multilevel models》.

Psychometrika 78 (4): 685–709. https://doi.org/10.1007/s11336-013-9328-2.

Clopper, C. J., 和 E. S. Pearson. 1934. 《The Use of Confidence or Fiducial Limits Illustrated In The Case of The Binomial》. *Biometrika* 26 (4): 404–13. https://doi.org/10.1093/biomet/26.4.404.

Cohen, Jacob. 1994. 《The Earth Is Round ($p < .05$)》. *American Psychologist* 49 (12): 997–1003. https://doi.org/10.1037/0003-066x.49.12.997.

Constantin, Ahlmann-Eltze, 和 Indrajeet Patil. 2021. 《ggsignif: R Package for Displaying Significance Brackets for ggplot2》. *PsyArxiv*. https://doi.org/10.31234/osf.io/7awm6.

Davies, Rhian, Steph Locke, 和 Lucy D'Agostino McGowan. 2022. *datasauRus: Datasets from the Datasaurus Dozen*. https://CRAN.R-project.org/package=datasauRus.

Demidenko, Eugene. 2013. *Mixed Models: Theory and Applications with R*. 2nd 本. Hoboken, New Jersey: John Wiley & Sons. https://doi.org/10.1002/9781118651537.

Diggle, P. J., J. A. Tawn, 和 R. A. Moyeed. 1998. 《Model-based geostatistics》. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 47 (3): 299–350. https://doi.org/10.1111/1467-9876.00113.

Dobson, Annette J. 1983. *An Introduction to Statistical Modelling*. 1st 本. London: Chapman; Hall/CRC. https://doi.org/10.1007/978-1-4899-3174-0.

Donegan, Connor. 2022.《geostan: An R package for Bayesian spatialanalysis》. *Journal of Open Source Software* 7 (79): 4716. https://doi.org/10.21105/joss.04716.

Dunning, Iain, Joey Huchette, 和 Miles Lubin. 2017. 《JuMP: A Modeling Language for Mathematical Optimization》. *SIAM Review* 59 (2): 295–320. https://doi.org/10.1137/15M1020575.

Eddelbuettel, Dirk, 和 James Joseph Balamuta. 2018.《Extending R with C++: A Brief Introduction to Rcpp》. *The American Statistician* 72 (1): 28–36. https://doi.org/10.1080/00031305.2017.1375990.

Eddelbuettel, Dirk, John W. Emerson, 和 Michael J. Kane. 2023. *BH: Boost C++ Header Files*. https://CRAN.R-project.org/package=BH.

Eddelbuettel, Dirk, 和 Romain François. 2011. 《Rcpp: Seamless R and C++ Integration》. *Journal of Statistical Software* 40 (8): 1–18. https://doi.org/10.18637/jss.v040.i08.

Efron, Bradley, Trevor Hastie, Iain Johnstone, 和 Robert Tibshirani. 2004. 《Least angle regression》. *The Annals of Statistics* 32 (2): 407–99. https://doi.org/10.1214/009053604000000067.

Epps, T. W., 和 Lawrence B. Pulley. 1983.《A Test for Normality Based on the Empirical Characteristic Function》. *Biometrika* 70 (3): 723–26. https://doi.org/10.2307/2336512.

Feinerer, Ingo, Kurt Hornik, 和 David Meyer. 2008. 《Text Mining Infrastructure in R》. *Journal of Statistical Software* 25 (5): 1–54. https://doi.org/10.18637/jss.v025.i05.

Feng, Dai, 和 Luke Tierney. 2008.《Computing and Displaying Isosurfaces in R》. *Journal of Statistical Software* 28 (1). https://doi.org/10.18637/jss.v028.i01.

Fisher, R. A. 1936.《The Use Of Multiple Measurements In Taxonomic Problems》. *Annals of Eugenics* 7 (2): 179–88. https://doi.org/10.1111/j.1469-1809.1936.tb02137.x.

Fligner, Michael A., 和 Timothy J. Killeen. 1976. 《Distribution-Free Two-Sample Tests for Scale》. *Journal of the American Statistical Association* 71 (353): 210–13. https://doi.org/10.1080/01621459.1976.10481517.

Friedman, Jerome, Robert Tibshirani, 和 Trevor Hastie. 2010. 《Regularization Paths for Generalized

Linear Models via Coordinate Descent》. *Journal of Statistical Software* 33 (1): 1–22. https://doi.org/10.18637/jss.v033.i01.

Friendly, Michael. 2021. *HistData: Data Sets from the History of Statistics and Data Visualization*. https://CRAN.R-project.org/package=HistData.

Friendly, Michael, 和 David Meyer. 2016. *Discrete Data Analysis with R: Visualization and Modeling Techniques for Categorical and Count Data*. 1st 本. Boca Raton, Florida: Chapman; Hall/CRC.

Fu, Anqi, 和 Balasubramanian Narasimhan. 2023. *ECOSolveR: Embedded Conic Solver in R*. https://CRAN.R-project.org/package=ECOSolveR.

Gabry, Jonah, Rok Češnovar, 和 Andrew Johnson. 2023. *cmdstanr: R Interface to CmdStan*. https://mc-stan.org/cmdstanr/.

Gabry, Jonah, Daniel Simpson, Aki Vehtari, Michael Betancourt, 和 Andrew Gelman. 2019. 《Visualization in Bayesian workflow》. *Journal of the Royal Statistical Society Series A: Statistics in Society* 182: 389–402. https://doi.org/10.1111/rssa.12378.

Gałecki, Andrzej, 和 Tomasz Burzykowski. 2013. *Linear Mixed-Effects Models Using R: A Step-by-Step Approach*. 1st 本. New York, NY: Springer New York. https://doi.org/10.1007/978-1-4614-3900-4.

Galton, F. 1886. 《Regression Towards Mediocrity in Hereditary Stature》. *Journal of the Anthropological Institute* 15: 246–63.

Garnier, Simon, Ross, Noam, Rudis, Robert, Camargo, 等. 2021. *viridis: Colorblind-Friendly Color Maps for R*. https://doi.org/10.5281/zenodo.4679424.

Gelfand, Alan E., Susan E. Hills, Amy Racine-Poon, 和 Adrian F. M. Smith. 1990. 《Illustration of Bayesian Inference in Normal Data Models Using Gibbs Sampling》. *Journal of the American Statistical Association* 85 (412): 972–85. https://doi.org/10.2307/2289594.

Gelman, Andrew, Daniel Lee, 和 Jiqiang Guo. 2015. 《Stan: A Probabilistic Programming Language for Bayesian Inference and Optimization》. *Journal of Educational and Behavioral Statistics* 40 (5): 530–43. https://doi.org/10.3102/1076998615606113.

Geyer, Charles J., 和 Glen D. Meeden. 2005. 《Fuzzy and Randomized Confidence Intervals and P-Values》. *Statistical Science* 20 (4): 358–66. https://www.jstor.org/stable/20061193.

Gómez-Rubio, Virgilio. 2020. *Bayesian inference with INLA*. Boca Raton, Florida: Chapman; Hall/CRC. https://becarioprecario.bitbucket.io/inla-gitbook/.

Goodrich, Ben, Jonah Gabry, Imad Ali, 和 Sam Brilleman. 2023. 《rstanarm: Bayesian applied regression modeling via Stan.》 https://mc-stan.org/rstanarm/.

Gross, Calli, 和 Philipp Ottolinger. 2016. *ggThemeAssist: Add-in to Customize ggplot2 Themes*. https://CRAN.R-project.org/package=ggThemeAssist.

Grün, Bettina, 和 Kurt Hornik. 2011. 《topicmodels: An R Package for Fitting Topic Models》. *Journal of Statistical Software* 40 (13): 1–30. https://doi.org/10.18637/jss.v040.i13.

Hadfield, Jarrod D. 2010. 《MCMC Methods for Multi-Response Generalized Linear Mixed Models: The MCMCglmm R Package》. *Journal of Statistical Software* 33 (2): 1–22. https://www.jstatsoft.org/v33/i02/.

Hahsler, Michael, 和 Kurt Hornik. 2007. 《TSP: Infrastructure for the traveling salesperson problem》. *Journal of Statistical Software* 23 (2): 1–21. https://doi.org/10.18637/jss.v023.i02.

Hahsler, Michael, Matthew Piekenbrock, 和 Derek Doran. 2019.《dbscan: Fast Density-Based Clustering with R》. *Journal of Statistical Software* 91 (1): 1–30. https://doi.org/10.18637/jss.v091.i01.

Hanley, James A. 2004.《'Transmuting' women into men: Galton's family data on human stature》. *The American Statistician* 58 (3): 237–43.

Hart, William E, Jean-Paul Watson, 和 David L Woodruff. 2011.《Pyomo: modeling and solving mathematical programs in Python》. *Mathematical Programming Computation* 3 (3): 219–60.

Hasselblad, Victor. 1969.《Estimation of Finite Mixtures of Distributions from the Exponential Family》. *Journal of the American Statistical Association* 64 (328): 1459–71. https://doi.org/10.1080/01621459.1969.10501071.

Hawkins, Oliver. 2022. *pilot: A minimal ggplot2 theme with an accessible discrete color palette*. https://github.com/olihawkins/pilot.

Heyde, C. C., E. Seneta, P. Crépel, S. E. Fienberg, 和 J. Gani. 2001. *Statisticians of the Centuries*. New York, NY: Springer-Verlag. https://doi.org/10.1007/978-1-4613-0179-0.

Hoaglin, David C., 和 Roy E. Welsch. 1978.《The Hat Matrix in Regression and ANOVA》. *The American Statistician* 32 (1): 17–22. https://www.jstor.org/stable/2683469.

Holt, Charles C. 2004.《Forecasting seasonals and trends by exponentially weighted moving averages》. *International Journal of Forecasting* 20 (1): 5–10. https://doi.org/10.1016/j.ijforecast.2003.09.015.

HSU, P. L. 1938.《Contribution to the theory of "Student's" T-test as applied to the problem of two samples》. *Statistical Research Memoirs* 2: 1–24.

———. 1983. *Collected Papers*. New York, NY: Springer-Verlag.

Hvitfeldt, Emil. 2021. *paletteer: Comprehensive Collection of Color Palettes*. https://github.com/EmilHvitfeldt/paletteer.

Hvitfeldt, Emil, 和 Julia Silge. 2021. *Supervised Machine Learning for Text Analysis in R*. New York: Chapman; Hall/CRC. https://smltar.com/.

Jiang, Jiming, 和 Thuan Nguyen. 2021. *Linear and Generalized Linear Mixed Models and Their Applications*. 2nd 本. New York, NY: Springer New York. https://doi.org/10.1007/978-1-0716-1282-8.

Johnson, Steven G. 2023. *The NLopt nonlinear optimization package*. https://CRAN.R-project.org/package=nloptr.

José. Chacón, Tarn Duong. 2018. *Multivariate Kernel Smoothing and Its Applications*. Boca Raton, Florida: Chapman; Hall/CRC. https://www.mvstat.net/mvksa/.

Joshua V. Dillon, Dustin Tran, Ian Langmore. 2017.《TensorFlow Distributions》. https://arxiv.org/abs/1711.10604.

Kabacoff, Robert I. 2022. *R in Action: Data Analysis and graphics with R and Tidyverse*. 3rd 本. Shelter Island, NY: Manning Publications Co.

Karambelkar, Bhaskar. 2016. *colormap: Color Palettes using Colormaps Node Module*. https://CRAN.R-project.org/package=colormap.

Karatzoglou, Alexandros, Alex Smola, Kurt Hornik, 和 Achim Zeileis. 2004.《kernlab: An S4 Package for Kernel Methods in R》. *Journal of Statistical Software* 11 (9): 1–20. https://doi.org/10.18637/jss.v011.i09.

内部使用

Kassambara, Alboukadel. 2022. *ggpubr: ggplot2 Based Publication Ready Plots*. https://CRAN.R-project.org/package=ggpubr.

Kaufman, Leonard, 和 Peter J. Rousseeuw. 1990. *Finding Groups in Data: An Introduction to Cluster Analysis*. 1 本. Wiley Series in Probability and Statistics. Wiley. https://doi.org/10.1002/9780470316801.

Kay, Matthew. 2022. *ggdist: Visualizations of Distributions and Uncertainty*. https://doi.org/10.5281/zenodo.3879620.

Kim, Seock-Ho, 和 Allan S. Cohen. 1998. 《On the Behrens-Fisher Problem: A Review》. *Journal of Educational and Behavioral Statistics* 23 (4): 356–77. https://doi.org/10.2307/1165281.

Kim, Yongdai, Hosik Choi, 和 Hee-Seok Oh. 2008. 《Smoothly Clipped Absolute Deviation on High Dimensions》. *Journal of the American Statistical Association* 103 (484): 1665–73. https://doi.org/10.1198/016214508000001066.

Kolaczyk, Eric D., 和 Gábor Csárdi. 2020. *Statistical Analysis of Network Data with R*. 2nd 本. Springer, New York, NY. https://doi.org/10.1007/978-3-030-44129-6.

Kothari, Aditya. 2022. *ggTimeSeries: Time Series Visualisations Using the Grammar of Graphics*. https://CRAN.R-project.org/package=ggTimeSeries.

Kuhn, Max, 和 Hadley Wickham. 2020. *Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles*. https://www.tidymodels.org.

Lang, Michel, 和 Patrick Schratz. 2023. *mlr3verse: Easily Install and Load the mlr3 Package Family*. https://CRAN.R-project.org/package=mlr3verse.

Lemon, Jim. 2006. 《plotrix: a package in the red light district of R》. *R-News* 6 (4): 8–12.

Ligges, Uwe, 和 Martin Mächler. 2003. 《scatterplot3d: An R Package for Visualizing Multivariate Data》. *Journal of Statistical Software* 8 (11): 1–20. https://doi.org/10.18637/jss.v008.i11.

Likert, Rensis. 1932. 《A Technique for the Measurement of Attitudes》. *Archives of Psychology* 142 (1): 1–55.

Lüdecke, Daniel. 2019. *strengejacke: Load Packages Associated with Strenge Jacke!* https://github.com/strengejacke/strengejacke.

Lüdecke, Daniel, Mattan S. Ben-Shachar, Indrajeet Patil, Brenton M. Wiernik, Etienne Bacher, Rémi Thériault, 和 Dominique Makowski. 2022. 《easystats: Framework for Easy Statistical Modeling, Visualization, and Reporting》. *CRAN*. https://easystats.github.io/easystats/.

Marron, J. S., 和 Ian L. Dryden. 2022. *Object Oriented Data Analysis*. 1st 本. Boca Raton, Florida: Chapman; Hall/CRC.

McGill, Tukey, R., 和 W. A. Larsen. 1978. 《Variations of box plots》. *The American Statistician* 32 (1): 12–16. https://www.jstor.org/stable/2683468.

Meyer, David, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, 和 Friedrich Leisch. 2023. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. https://CRAN.R-project.org/package=e1071.

Meyer, David, Achim Zeileis, 和 Kurt Hornik. 2006. 《The Strucplot Framework: Visualizing Multi-Way Contingency Tables with vcd》. *Journal of Statistical Software* 17 (3): 1–48. https://doi.org/10.18637/jss.v017.i03.

Mood, A. M. 1954. 《On the Asymptotic Efficiency of Certain Nonparametric Two-Sample Tests》. *The Annals of Mathematical Statistics* 25 (3): 514–22. https://doi.org/10.1214/aoms/1177728719.

Moraga, Paula. 2020. *Geospatial health data: modeling and visualization with R-INLA and Shiny*. Boca Raton, Florida: Chapman; Hall/CRC. https://www.paulamoraga.com/book-geospatial/.

Morris, Mitzi, Katherine Wheeler-Martin, Dan Simpson, Stephen J. Mooney, Andrew Gelman, 和 Charles DiMaggio. 2019. 《Bayesian hierarchical spatial models: Implementing the Besag York Mollié model in stan》. *Spatial and Spatio-temporal Epidemiology* 31 (十一月): 100301. https://doi.org/10.1016/j.sste.2019.100301.

Murphy, Kevin P. 2022. *Probabilistic Machine Learning: An introduction*. Cambridge, Massachusetts: MIT Press. https://probml.github.io/pml-book/book1.html.

Neitmann, Thomas. 2020. *ggcharts: Shorten the Distance from Data Visualization Idea to Actual Plot*. https://CRAN.R-project.org/package=ggcharts.

Neuwirth, Erich. 2022. *RColorBrewer: ColorBrewer Palettes*. https://CRAN.R-project.org/package=RColorBrewer.

Newcombe, Robert G. 1998. 《Interval estimation for the difference between independent proportions: comparison of eleven methods》. *Statistics in Medicine* 17 (8): 873–90. https://doi.org/10.1002/(SICI)1097-0258(19980430)17:8%3C873::AID-SIM779%3E3.0.CO;2-I.

Niyizibi, Bart, Wade Brorsen, 和 Eunchun Park. 2018. 《Using Bayesian Kriging for Spatial Smoothing of Trends in the Means and Variances of Crop Yield Densities》. *Economic Geography*. https://doi.org/10.22004/ag.econ.274403.

O'Donoghue, Brendan, Eric Chu, Parikh Neal, 和 Stephen Boyd. 2016. 《Operator Splitting for Conic Optimization via Homogeneous Self-Dual Embedding》. *Journal of Optimization Theory and Applications* 169 (3): 1042–68. https://doi.org/10.1007/s10957-016-0892-3.

Otto, James, 和 David Kahle. 2023. 《ggdensity: Improved Bivariate Density Visualization in R》. *The R Journal* 15: 220–36. https://doi.org/10.32614/RJ-2023-048.

Palmí-Perales, Francisco, Virgilio Gómez-Rubio, Roger S. Bivand, Michela Cameletti, 和 Håvard Rue. 2022. 《Bayesian Inference for Multivariate Spatial Models with R-INLA》. *The R Journal*. https://doi.org/10.48550/arXiv.2212.10976.

Patil, Indrajeet. 2021. 《Visualizations with statistical details: The ggstatsplot approach》. *Journal of Open Source Software* 6 (61): 3167. https://doi.org/10.21105/joss.03167.

Pebesma, Edzer. 2018. 《Simple Features for R: Standardized Support for Spatial Vector Data》. *The R Journal* 10 (1): 439–46. https://doi.org/10.32614/RJ-2018-009.

———. 2022. *stars: Spatiotemporal Arrays, Raster and Vector Data Cubes*. https://CRAN.R-project.org/package=stars.

Pedersen, Thomas Lin, 和 Fabio Crameri. 2022. *scico: Colour Palettes Based on the Scientific ColourMaps*. https://CRAN.R-project.org/package=scico.

Peter Diggle, Kung-Yee Liang, Patrick Heagerty, 和 Scott Zeger. 2002. *Analysis of longitudinal data*. 2nd 本. Oxford: Oxford University Press.

Piironen, Juho, Markus Paasiniemi, 和 Aki Vehtari. 2020. 《Projective Inference in High-Dimensional Problems: Prediction and Feature Selection》. *Electronic Journal of Statistics* 14 (1): 2155–97. https:

统计出版社 ©

//doi.org/10.1214/20-EJS1711.

Piironen, Juho, 和 Aki Vehtari. 2017a. 《Comparison of Bayesian Predictive Methods for Model Selection》. *Statistics and Computing* 27 (3): 711–35. https://doi.org/10.1007/s11222-016-9649-y.

———. 2017b. 《Sparsity information and regularization in the horseshoe and other shrinkage priors》. *Electronic Journal of Statistics* 11 (2): 5018–51. https://doi.org/10.1214/17-EJS1337SI.

Pinheiro, JoséC., 和 Douglas M. Bates. 2000. *Mixed-Effects Models in S and S-PLUS*. New York, NY: Springer-Verlag.

Plummer, Martyn. 2021. *rjags: Bayesian Graphical Models using MCMC*. https://CRAN.R-project.org/package=rjags.

Plummer, Martyn, Nicky Best, Kate Cowles, 和 Karen Vines. 2006. 《coda: Convergence Diagnosis and Output Analysis for MCMC》. *R News* 6 (1): 7–11. https://journal.r-project.org/archive/.

Pu, Xiaoying, 和 Matthew Kay. 2020. 《A Probabilistic Grammar of Graphics》. 收入 *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–13. ACM. https://doi.org/10.1145/3313831.3376466.

Rasmussen, Carl Edward, 和 Christopher K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. Cambridge, Massachusetts: MIT Press. https://gaussianprocess.org/gpml/.

Rigby, R. A., 和 D. M. Stasinopoulos. 2005. 《Generalized additive models for location, scale and shape (with discussion)》. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 54 (3): 507–54. https://doi.org/10.1111/j.1467-9876.2005.00510.x.

Rizopoulos, Dimitris. 2023. *GLMMadaptive: Generalized Linear Mixed Models using Adaptive Gaussian Quadrature*. https://CRAN.R-project.org/package=GLMMadaptive.

Rönnegård, Lars, Xia Shen, 和 Moudud Alam. 2010. 《hglm: A Package for Fitting Hierarchical Generalized Linear Models》. *The R Journal* 2 (2): 20–28. https://doi.org/10.32614/RJ-2010-009.

Rosseel, Yves. 2012. 《lavaan: An R Package for Structural Equation Modeling》. *Journal of Statistical Software* 48 (2): 1–36. https://doi.org/10.18637/jss.v048.i02.

Rubin, Donald B. 1981. 《Estimation in Parallel Randomized Experiments》. *Journal of Educational Statistics* 6 (4): 377–401. https://doi.org/10.3102/10769986006004377.

Rue, Håvard, Sara Martino, 和 Nicholas Chopin. 2009. 《Approximate Bayesian Inference for Latent Gaussian Models Using Integrated Nested Laplace Approximations (with discussion)》. *Journal of the Royal Statistical Society, Series B* 71 (2): 319–92.

Ryan, Jeffrey A., 和 Joshua M. Ulrich. 2022. *quantmod: Quantitative Financial Modelling Framework*. https://CRAN.R-project.org/package=quantmod.

S original by Berwin A. Turlach, Fortran contributions from Cleve Moler dpodi/LINPACK), R port by Andreas Weingessel. 2019. *quadprog: Functions to Solve Quadratic Programming Problems*. https://CRAN.R-project.org/package=quadprog.

Sainsbury-Dale, Matthew, Andrew Zammit-Mangion, 和 Noel Cressie. 2022. 《Modelling Big, Heterogeneous, Non-Gaussian Spatial and Spatio-Temporal Data using FRK》. *Journal of Statistical Software*. https://doi.org/10.48550/arXiv.2110.02507.

Sarkar, Deepayan. 2008. *lattice: Multivariate Data Visualization with R*. New York: Springer. http://lmdvr.r-forge.r-project.org.

Schilling, Walter. 1947. 《A Frequency Distribution Represented as the Sum of Two Poisson Distributions》. *Journal of the American Statistical Association* 42 (239): 407–24.

Schwendinger, Florian, 和 Hans W. Borchers. 2023. *CRAN Task View: Optimization and Mathematical Programming*. https://CRAN.R-project.org/view=Optimization.

Schwendinger, Florian, 和 Dirk Schumacher. 2023. *highs: HiGHS Optimization Solver*. https://CRAN.R-project.org/package=highs.

Scrucca, Luca. 2013. 《GA: A Package for Genetic Algorithms in R》. *Journal of Statistical Software* 53 (4): 1–37. https://doi.org/10.18637/jss.v053.i04.

Scrucca, Luca, Michael Fop, T. Brendan Murphy, 和 Adrian E. Raftery. 2016. 《mclust 5: clustering, classification and density estimation using Gaussian finite mixture models》. *The R Journal* 8 (1): 289–317. https://doi.org/10.32614/RJ-2016-021.

Shapiro, S. S., 和 M. B. Wilk. 1965. 《An analysis of variance test for normality (complete samples)》. *Biometrika* 52 (3-4): 591–611. https://doi.org/10.1093/biomet/52.3-4.591.

Sidiropoulos, Nikos, Sina Hadi Sohi, Thomas Lin Pedersen, Bo Torben Porse, Ole Winther, Nicolas Rapin, 和 Frederik Otzen Bagger. 2018. 《SinaPlot: An Enhanced Chart for Simple and Truthful Representation of Single Observations Over Multiple Classes》. *Journal of Computational and Graphical Statistics* 27 (3): 673–76. https://doi.org/10.1080/10618600.2017.1366914.

Silge, Julia, 和 David Robinson. 2017. *Text Mining with R*. New York: O'Reilly Media, Inc. https://www.tidytextmining.com/.

Simon, Noah, Jerome H. Friedman, Trevor Hastie, 和 Rob Tibshirani. 2011. 《Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent》. *Journal of Statistical Software* 39 (5): 1–13. https://doi.org/10.18637/jss.v039.i05.

Slowikowski, Kamil. 2021. *ggrepel: Automatically Position Non-Overlapping Text Labels with ggplot2*. https://CRAN.R-project.org/package=ggrepel.

Soetaert, Karline. 2021. *plot3D: Plotting Multi-Dimensional Data*. https://CRAN.R-project.org/package=plot3D.

Sorensen, Tanner, Sven Hohenstein, 和 Shravan Vasishth. 2016. 《Bayesian linear mixed models using Stan: A tutorial for psychologists, linguists, and cognitive scientists》. *The Quantitative Methods for Psychology* 12 (3): 175–200. https://doi.org/10.20982/tqmp.12.3.p175.

Stan Development Team. 2023a. 《RStan: the R interface to Stan》. https://mc-stan.org/.

———. 2023b. 《StanHeaders: Headers for the R interface to Stan》. https://mc-stan.org/.

"Student". 1908. 《The probable error of a mean》. *Biometrika* 6: 1–25.

Stylianou, Nassos, Will Dahlgreen, Robert Cuffe, Tom Calver, 和 Ransome Mpini. 2022. *bbplot: making ggplot2 graphics in BBC NEWS style*.

Tang, Yuan, Masaaki Horikoshi, 和 Wenxuan Li. 2016. 《ggfortify: Unified Interface to Visualize Statistical Result of Popular R Packages》. *The R Journal* 8 (2): 474–85. https://doi.org/10.32614/RJ-2016-060.

Terry M. Therneau, 和 Patricia M. Grambsch. 2000. *Modeling Survival Data: Extending the Cox Model*. New York: Springer.

Theussl, Stefan, 和 Kurt Hornik. 2023. *Rglpk: R/GNU Linear Programming Kit Interface*. https://

© 中国统计出版社

CRAN.R-project.org/package=Rglpk.

Theußl, Stefan, Florian Schwendinger, 和 Kurt Hornik. 2020. 《ROI: An Extensible R Optimization Infrastructure》. *Journal of Statistical Software* 94 (15): 1–64. https://doi.org/10.18637/jss.v094.i15.

Tibshirani, Robert. 1996. 《Regression Shrinkage and Selection via the Lasso》. *Journal of the Royal Statistical Society. Series B (Methodological)* 58 (1): 267–88. http://www.jstor.org/stable/2346178.

Tobin, Ciaran. 2020. *ggthemr: Themes for ggplot2*.

Tobin, J. 1958. 《Estimation of relationships for limited dependent variables》. *Econometrica* 26 (1): 24–36. https://doi.org/10.2307/1907382.

Tobler, Waldo. 1970. 《A computer movie simulating urban growth in the Detroit region》. *Economic Geography* 46 (Supplement): 234–40. https://doi.org/10.2307/143141.

Tyner, Sam, François Briatte, 和 Heike Hofmann. 2017. 《Network Visualization with ggplot2》. *The R Journal* 9 (1): 27–59. https://doi.org/10.32614/RJ-2017-023.

Varadhan, Ravi, 和 Paul Gilbert. 2009. 《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. https://www.jstatsoft.org/v32/i04/.

Vehtari, Aki, Andrew Gelman, 和 Jonah Gabry. 2017. 《Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC》. *Statistics and Computing* 27: 1413–32. https://doi.org/10.1007/s11222-016-9696-4.

Vehtari, Aki, Andrew Gelman, Daniel Simpson, Bob Carpenter, 和 Paul-Christian Bürkner. 2021. 《Rank-Normalization, Folding, and Localization: An Improved $\widehat{R}$ for Assessing Convergence of MCMC (with Discussion)》. *Bayesian Analysis* 16 (2): 667–718. https://doi.org/10.1214/20-BA1221.

Wand, M. P., 和 M. C. Jones. 1995. *Kernel Smoothing*. 1st 本. Boca Raton, Florida: Chapman; Hall/CRC. http://matt-wand.utsacademics.info/webWJbook/.

Warnes, J. J., 和 B. D. Ripley. 1987. 《Problems with likelihood estimation of covariance functions of spatial gaussian processes》. *Biometrika* 74 (3): 640–42.

Wickham, Charlotte. 2018. *munsell: Utilities for Using Munsell Colours*. https://CRAN.R-project.org/package=munsell.

Wickham, Hadley. 2016. *ggplot2: Elegant Graphics for Data Analysis*. 2nd 本. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, 等. 2019. 《Welcome to the tidyverse》. *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Wickham, Hadley, Mine Çetinkaya-Rundel, 和 Garrett Grolemund. 2023. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. 2nd 本. Sebastopol, California: O'Reilly Media, Inc. https://r4ds.hadley.nz/.

Wickham, Hadley, Danielle Navarro, 和 Thomas Lin Pedersen. 2024. *ggplot2: Elegant Graphics for Data Analysis*. 3rd 本. Springer-Verlag New York. https://ggplot2-book.org/.

Wickham, Hadley, 和 Dana Seidel. 2022. *scales: Scale Functions for Visualization*. https://CRAN.R-project.org/package=scales.

Wilke, Claus O. 2020. *ggtext: Improved Text Rendering Support for ggplot2*. https://CRAN.R-project.

org/package=ggtext.

Wilson, Edwin B. 1927. 《Probable inference, the law of succession, and statistical inference》. *Journal of the American Statistical Association* 22 (158): 209–12. https://doi.org/10.1080/01621459.1927.10502953.

Winters, Peter R. 1960. 《Forecasting sales by exponentially weighted moving averages》. *Management Science* 6 (3): 324–42. https://doi.org/10.1287/mnsc.6.3.324.

Wood, S. N. 2004. 《Stable and efficient multiple smoothing parameter estimation for generalized additive models》. *Journal of the American Statistical Association* 99 (467): 673–86.

———. 2017. *Generalized Additive Models: An Introduction with R*. 2nd 本. Chapman; Hall/CRC. https://www.maths.ed.ac.uk/~swood34/igam/.

Wood, Simon N. 2019. 《Simplified integrated nested Laplace approximation》. *Biometrika* 107 (1): 223–30. https://doi.org/10.1093/biomet/asz044.

Wood, Simon N., Yannig Goude, 和 Simon Shaw. 2015. 《Generalized Additive Models for Large Data Sets》. *Journal of the Royal Statistical Society Series C: Applied Statistics* 64 (1): 139–55. https://doi.org/10.1111/rssc.12068.

Wood, Simon N., Natalya Pya, 和 Benjamin Säfken. 2016. 《Smoothing Parameter and Model Selection for General Smooth Models》. *Journal of the American Statistical Association* 111 (516): 1548–63. https://doi.org/10.1080/01621459.2016.1180986.

Xiao, Nan. 2018. *ggsci: Scientific Journal and Sci-Fi Themed Color Palettes for ggplot2*. https://CRAN.R-project.org/package=ggsci.

Xie, Yihui. 2015. *Dynamic Documents with R and knitr*. 2nd 本. Boca Raton, Florida: Chapman; Hall/CRC. https://yihui.org/knitr/.

Yu, Guangchuang. 2022. *ggimage: Use Image in ggplot2*. https://CRAN.R-project.org/package=ggimage.

Yutani, Hiroaki. 2022. *string2path: Rendering Font into data.frame*. https://CRAN.R-project.org/package=string2path.

Zeileis, Achim, Jason C. Fisher, Kurt Hornik, Ross Ihaka, Claire D. McWhite, Paul Murrell, Reto Stauffer, 和 Claus O. Wilke. 2020. 《colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes》. *Journal of Statistical Software* 96 (1): 1–49. https://doi.org/10.18637/jss.v096.i01.

Zeileis, Achim, David Meyer, 和 Kurt Hornik. 2007. 《Residual-based Shadings for Visualizing (Conditional) Independence》. *Journal of Computational and Graphical Statistics* 16 (3): 507–25. https://doi.org/10.1198/106186007X237856.

Zhang, Cun-Hui. 2010. 《Nearly unbiased variable selection under minimax concave penalty》. *The Annals of Statistics* 38 (2): 894–942. https://doi.org/10.1214/09-AOS729.

Zhu, Jin, Xueqin Wang, Liyuan Hu, Junhao Huang, Kangkang Jiang, Yanhang Zhang, Shiyun Lin, 和 Junxian Zhu. 2022. 《abess: A Fast Best Subset Selection Library in Python and R》. *Journal of Machine Learning Research* 23 (202): 1–7. https://www.jmlr.org/papers/v23/21-1060.html.

Zou, Hui. 2006. 《The Adaptive Lasso and Its Oracle Properties》. *Journal of the American Statistical Association* 101 (476): 407–99. https://doi.org/10.1198/016214506000000735.

Zou, Hui, 和 Trevor Hastie. 2005. 《Regularization and Variable Selection Via the Elastic Net》. *Journal*

© 黄湘云

of the Royal Statistical Society Series B: Statistical Methodology 67 (2): 301–20. https://doi.org/10.1111/j.1467-9868.2005.00503.x.

宋泽熙. 2011.《两个二项总体成功概率的比较》. 中国校外教育（理论）z1: 81. https://doi.org/10.3969/j.issn.1004-8502-B.2011.z1.0919.

赵鹏, 谢益辉, 和黄湘云. 2021. 现代统计图形. 北京: 人民邮电出版社. https://bookdown.org/xiangyun/msg.

韦博成. 2009.《《红楼梦》前 80 回与后 40 回某些文风差异的统计分析（两个独立二项总体等价性检验的一个应用）》. 应用概率统计 25 (4): 441–48. https://doi.org/10.3969/j.issn.1001-4268.2009.04.012.

©基础与实践

© 黄湘云

附录 A　数学符号

表格 A.1: 数学符号表

符号	含义
$\mathbb{R}^n$	n 维实数
$\mathbb{R}^{n\times p}$	$n\times p$ 维实矩阵
$\mathbb{Z}$	整数
$\mathcal{N}$	正态分布
$\mathcal{D}$	研究区域
$\mathcal{S}$	随机过程
$\mathcal{G}$	图
$\mathcal{L}$	似然
MVN	多元正态分布
Σ	协方差矩阵
x	标量
$\boldsymbol{x}$	向量
X	矩阵
$X^\top$	矩阵转置
X^{-1}	矩阵求逆
I	单位矩阵
J	全 1 矩阵
$\mathbf{1}$	全 1 向量
$\mathbf{0}$	全 0 向量
β	截距
$\boldsymbol{\beta}$	系数向量
ℓ	对数似然
E	期望
Var	方差

© 黄湘云

符号	含义
Cov	协方差
Bernoulli	伯努利分布
Binomial	二项分布
Poisson	泊松分布
Gamma	伽马分布
Beta	贝塔分布
Γ	伽马函数
$\|\boldsymbol{x}\|_0$	向量的 0 范数
$\|\boldsymbol{x}\|_1$	向量的 1 范数
$\|\boldsymbol{x}\|_2$	向量的 2 范数
$\|\boldsymbol{x}\|_p$	向量的 p 范数

全书英文字母表示数据，希腊字母表示参数，加粗表示向量，大写表示矩阵，花体字母各有含义。所有的向量都是列向量，如上表中的 $\boldsymbol{x}$ ，而 $\boldsymbol{x}^\top$ 则表示行向量。

下表给出本书用到的一些统计术语的英文缩写。

表格 A.2: 统计术语的英文缩写

统计术语	英文缩写
最小二乘估计	LSE
极大似然估计	MLE
最佳线性无偏估计	BLUE
最小方差无偏估计	MVUE
一致最小方差无偏估计	UMVUE
最小范数二次无偏估计	MINQUE
普通最小二乘估计	OLS
偏最小二乘估计	PLS
广义最小二乘估计	GLS
带权最小二乘估计	WLS
Lasso 估计	LASSO
均方误差	MSE
均方根误差	RMSE
平均绝对误差	MAE
惩罚拟似然	PQL
剖面极大似然	PML
限制极大似然	REML
线性模型	LM

© 黄湘云

统计术语	英文缩写
广义线性模型	GLM
广义可加模型	GAM
线性混合效应模型	LMM
广义线性混合效应模型	GLMM
广义可加混合效应模型	GAMM

©黄湘云

附录 B　矩阵运算

> There's probably some examples, but there are some examples of people using `solve(t(X) %*% W %*% X) %*% W %*% Y` to compute regression coefficients, too.
>
> — Thomas Lumley [1]

本文主要介绍 Base R 提供的矩阵运算，包括加、减、乘等基础矩阵运算和常用的矩阵分解方法，总结 Base R 、**Matrix** 包和 Eigen 库对应的矩阵运算函数，分别对应基础、进阶和高阶的读者。最后，介绍矩阵运算在线性回归中的应用。

```
library(Matrix)
```

B.1　基础运算

约定符号

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

B.1.1　加、减、乘

矩阵 A

```
A <- matrix(c(1, 1.2, 1.2, 3), nrow = 2)
A
```

```
     [,1] [,2]
[1,]  1.0  1.2
[2,]  1.2  3.0
```

[1] https://stat.ethz.ch/pipermail/r-help/2006-March/101596.html

© 统计学基础

```
B <- matrix(c(1, 2, 3, 4), nrow =2)
B
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

```
A + A # 对应元素相加
```

```
     [,1] [,2]
[1,]  2.0  2.4
[2,]  2.4  6.0
```

```
A - A # 对应元素相减
```

```
     [,1] [,2]
[1,]    0    0
[2,]    0    0
```

```
A %*% A # 矩阵乘法
```

```
     [,1]  [,2]
[1,] 2.44  4.80
[2,] 4.80 10.44
```

B.1.2 对数、指数与幂

矩阵 A 的对数 $\log A$ ，就是找一个矩阵 L 使得 $A = \mathrm{e}^L$

```
expm::logm(A)
```

```
           [,1]      [,2]
[1,] -0.4485660 0.8050907
[2,]  0.8050907 0.8932519
```

矩阵 A 的指数 e^A 的定义

$$\mathrm{e}^A = \sum_{k=1}^{\infty} \frac{A^k}{k!}$$

expm 包可以计算矩阵的指数、开方、对数等。

```
expm::expm(A)
```

```
         [,1]     [,2]
[1,]  7.60987 12.93908
[2,] 12.93908 29.17501
```

©肖清文

或者使用奇异值分解 $A = UDV^\top$ ，则 $\mathrm{e}^A = U\mathrm{e}^D V^\top$ ，其中，D 是对角矩阵。

```
(res <- svd(A))
```

```
$d
[1] 3.5620499 0.4379501

$u
           [,1]       [,2]
[1,] -0.4241554 -0.9055894
[2,] -0.9055894  0.4241554

$v
           [,1]       [,2]
[1,] -0.4241554 -0.9055894
[2,] -0.9055894  0.4241554
```

```
res$u %*% diag(exp(res$d)) %*% res$v
```

```
         [,1]     [,2]
[1,]  7.60987 12.93908
[2,] 12.93908 29.17501
```

矩阵 A 的 n 次幂 A^n ，利用奇异值分解 $A = UDV^\top$

$$
\begin{aligned}
A^n &= A \times A \times \cdots \times A \\
&= UDV^\top UDV^\top \cdots UDV^\top
\end{aligned}
$$

计算 A^3

```
res$u %*% (diag(res$d)^3) %*% res$v
```

```
       [,1]   [,2]
[1,]  8.200 17.328
[2,] 17.328 37.080
```

B.1.3 迹、秩、条件数

矩阵 A 的迹 $\mathrm{tr}(A) = \sum_{i=1}^n a_{ii}$

```
sum(diag(A))
```

```
[1] 4
```

```
qr(A)$rank
```

```
[1] 2
```

```
kappa(A)
```

```
[1] 10.41469
```

B.1.4 求逆与广义逆

Moore-Penrose Generalized Inverse 摩尔广义逆 A^-。

$$A^- = (A^\top A)^{-1} A$$

如果 A 可逆，则广义逆就是逆。

```
solve(A) # 逆
```

```
           [,1]       [,2]
[1,]  1.9230769 -0.7692308
[2,] -0.7692308  0.6410256
```

```
MASS::ginv(A) # 广义逆
```

```
           [,1]       [,2]
[1,]  1.9230769 -0.7692308
[2,] -0.7692308  0.6410256
```

B.1.5 行列式与伴随

矩阵必须是方阵

伴随矩阵 $A * A^\star = A^\star * A = |A| * I, A^\star = |A| * A^{-1}$

- $|A^\star| = |A|^{n-1}, A \in \mathbb{R}^{n\times n}, n \geq 2$
- $(A^\star)^\star = |A|^{n-2} A, A \in \mathbb{R}^{n\times n}, n \geq 2$
- $(A^\star)^\star$ A 的 n 次伴随是?

```
det(A)
```

```
[1] 1.56
```

```
det(A) * solve(A)
```

```
     [,1] [,2]
[1,]  3.0 -1.2
[2,] -1.2  1.0
```

©草稿

B.1.6 外积、直积与交叉积

通常的矩阵乘法也叫矩阵内积

```
A %*% B
```

```
     [,1] [,2]
[1,]  3.4  7.8
[2,]  7.2 15.6
```

外积

```
A %o% B # outer(A, B, FUN = "*")
```

```
, , 1, 1

     [,1] [,2]
[1,]  1.0  1.2
[2,]  1.2  3.0

, , 2, 1

     [,1] [,2]
[1,]  2.0  2.4
[2,]  2.4  6.0

, , 1, 2

     [,1] [,2]
[1,]  3.0  3.6
[2,]  3.6  9.0

, , 2, 2

     [,1] [,2]
[1,]  4.0  4.8
[2,]  4.8 12.0
```

直积/克罗内克积

```
A %x% B # kronecker(A, B, FUN = "*")
```

```
     [,1] [,2] [,3] [,4]
[1,]  1.0  3.0  1.2  3.6
[2,]  2.0  4.0  2.4  4.8
```

©黄湘云


```
[3,]  1.2  3.6  3.0  9.0
[4,]  2.4  4.8  6.0 12.0
```

交叉积 $A^\top A$

```
crossprod(A, A)  #  t(x) %*% y
```

```
     [,1]  [,2]
[1,] 2.44  4.80
[2,] 4.80 10.44
```

```
tcrossprod(A, A) #  x %*% t(y)
```

```
     [,1]  [,2]
[1,] 2.44  4.80
[2,] 4.80 10.44
```

B.1.7 Hadamard 积

Hadamard 积（法国数学家 Jacques Hadamard）也叫 Schur 积（德国数学家 Issai Schur ）或 entrywise 积是两个维数相同的矩阵对应元素相乘，特别地，A^2 表示将矩阵 A 的每个元素平方

$$(A \circ B)_{ij} = (A)_{ij}(B)_{ij}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \circ \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & a_{13}b_{13} \\ a_{21}b_{21} & a_{22}b_{22} & a_{23}b_{23} \\ a_{31}b_{31} & a_{32}b_{32} & a_{33}b_{33} \end{bmatrix}$$

```
fastmatrix::hadamard(A, B)
```

```
     [,1] [,2]
[1,]  1.0  3.6
[2,]  2.4 12.0
```

```
A^2     # 每个元素平方 a_ij ^ 2
```

```
     [,1] [,2]
[1,] 1.00 1.44
[2,] 1.44 9.00
```

```
A ** A  # 每个元素的幂 a_ij ^ a_ij
```

```
         [,1]      [,2]
[1,] 1.000000  1.244565
[2,] 1.244565 27.000000
```

```
2^A     # 每个元素的指数 2 ^ a_ij
```

©黄文海

```
         [,1]     [,2]
[1,] 2.000000 2.297397
[2,] 2.297397 8.000000
```

```
exp(A)  # 每个元素的指数 exp(a_ij)
```

```
         [,1]      [,2]
[1,] 2.718282  3.320117
[2,] 3.320117 20.085537
```

B.1.8 矩阵范数

矩阵的范数，包括 1，2，无穷范数

1-范数 列和绝对值最大的

2 - 范数 又称谱范数，矩阵最大的奇异值，如果是方阵，就是最大的特征值

∞ - 范数 行和绝对值最大的

Frobenius - 范数 Euclidean 范数

M - 范数 矩阵里模最大的元素，矩阵里面的元素可能含有复数，所以取模最大

```
norm(A, type = "1") # max(abs(colSums(A)))
```

```
[1] 4.2
```

```
norm(A, type = "I") # max(abs(rowSums(A)))
```

```
[1] 4.2
```

```
norm(A, type = "F")
```

```
[1] 3.588872
```

```
norm(A, type = "M") #
```

```
[1] 3
```

```
norm(A, type = "2") # max(svd(A)$d)
```

```
[1] 3.56205
```

B.1.9 转置与旋转

矩阵 A

```
t(A) # 转置
```

© 基础


```
     [,1] [,2]
[1,]  1.0  1.2
[2,]  1.2  3.0
```

B.1.10 正交与投影

矩阵 A 的投影

$$I - A(A^\top A)^{-1}A^\top$$

```
diag(rep(1, 2)) - A %*% solve(t(A) %*% A) %*% t(A)
```

```
              [,1]          [,2]
[1,] -4.440892e-16  4.440892e-16
[2,]  3.330669e-16 -2.220446e-16
```

B.1.11 Givens 变换 (*)

- Givens 旋转
- 帽子矩阵在统计中的应用，回归与方差分析 (Hoaglin 和 Welsch 1978)

B.1.12 Householder 变换 (*)

Householder 变换是平面反射的一般情况：要计算 $N \times P$ 维矩阵 X 的 QR 分解，我们采用 Householder 变换

$$\mathbf{H}_u = \mathbf{I} - 2\mathbf{u}\mathbf{u}^\top$$

其中 I 是 $N \times N$ 维的单位矩阵，u 是 N 维单位向量，即 $\|\mathbf{u}\| = \sqrt{\mathbf{u}\mathbf{u}^\top} = 1$。则 H_u 是对称正交的，因为

$$\mathbf{H}_u^\top = \mathbf{I}^\top - 2\mathbf{u}\mathbf{u}^\top = \mathbf{H}_u$$

并且

$$\mathbf{H}_u^\top \mathbf{H}_u = \mathbf{I} - 4\mathbf{u}\mathbf{u}^\top + 4\mathbf{u}\mathbf{u}^\top\mathbf{u}\mathbf{u}^\top = \mathbf{I}$$

让 $\mathbf{H}_u$ 乘以向量 $\mathbf{y}$，即

$$\mathbf{H}_u\mathbf{y} = \mathbf{y} - 2\mathbf{u}\mathbf{u}^\top\mathbf{y}$$

©黄湘云

它是 y 关于垂直于过原点的 u 的直线的反射，只要

$$\mathbf{u} = \frac{\mathbf{y} - \|\mathbf{y}\|\mathbf{e}_1}{\|\mathbf{y} - \|\mathbf{y}\|\mathbf{e}_1\|} \tag{B.1}$$

或者

$$\mathbf{u} = \frac{\mathbf{y} + \|\mathbf{y}\|\mathbf{e}_1}{\|\mathbf{y} + \|\mathbf{y}\|\mathbf{e}_1\|} \tag{B.2}$$

其中 $\mathbf{e}_1 = (1, 0, \ldots, 0)^\top$，Householder 变换使得向量 y 成为 x 轴，在新的坐标系统中，向量 $H_u y$ 的坐标为 $(\pm\|y\|, 0, \ldots, 0)^\top$

举个例子

借助 Householder 变换做 QR 分解的优势：

1. 更快、数值更稳定比直接构造 Q，特别当 N 大于 P 的时候
2. 相比于存储矩阵 Q 的 N^2 个元素，Householder 变换只存储 P 个向量 $u_1, \ldots, u_P$
3. QR 分解的真实实现，比如在 LINPACK 中，定义 u 的时候，方程式 B.1 或方程式 B.2 的选择基于 y 的第一个坐标的符号。如果坐标是负的，使用方程式 B.1 ，如果是正的，使用方程式 B.2 ，这个做法可以使得数值计算更加稳定。

用 Householder 变换做 QR 分解 (D. M. Bates 和 Watts 1988) 及其 R 语言、Eigen 实现。

B.1.13 单位矩阵

矩阵对角线上全是 1，其余位置都是 0

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
diag(rep(3))
```

```
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

而全 1 矩阵是所有元素都是 1 的矩阵，可以借助外积运算构造，如 3 阶全 1 矩阵

```
rep(1,3) %o% rep(1,3)
```

```
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
```

©黄湘云


```
[3,]    1    1    1
```

B.1.14 对角矩阵

```
diag(A)       # 矩阵的对角
```

```
[1] 1 3
```

```
diag(x = c(1, 2, 3)) # 构造对角矩阵
```

```
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    2    0
[3,]    0    0    3
```

B.1.15 稀疏矩阵

稀疏矩阵的典型构造方式是通过三元组。

```
i <- c(1, 3:8) # 行指标
j <- c(2, 9, 6:10) # 列指标
x <- 7 * (1:7) # 数据
Matrix::sparseMatrix(i, j, x = x)
```

```
8 x 10 sparse Matrix of class "dgCMatrix"

[1,] . 7 . . .  .  .  .  .  .
[2,] . . . . .  .  .  .  .  .
[3,] . . . . .  .  .  . 14  .
[4,] . . . . . 21  .  .  .  .
[5,] . . . . .  . 28  .  .  .
[6,] . . . . .  .  . 35  .  .
[7,] . . . . .  .  .  . 42  .
[8,] . . . . .  .  .  .  . 49
```

B.1.16 上、下三角矩阵

```
m <- A
m
```

```
     [,1] [,2]
[1,]  1.0  1.2
[2,]  1.2  3.0
```

```
upper.tri(m) # 矩阵上三角
```

```
      [,1]  [,2]
[1,] FALSE  TRUE
[2,] FALSE FALSE
```

```
m[upper.tri(m)]
```

```
[1] 1.2
```

```
m[lower.tri(m)] <- 0 # 获得上三角矩阵
m
```

```
     [,1] [,2]
[1,]    1  1.2
[2,]    0  3.0
```

矩阵 A 的下三角矩阵

```
m <- matrix(c(1, 2, 2, 3), nrow = 2)
m[row(m) < col(m)] <- 0
m
```

```
     [,1] [,2]
[1,]    1    0
[2,]    2    3
```

B.2 矩阵分解

B.2.1 LU 分解

矩阵 A 的 LU 分解 $A = LU$ ，L 是下三角矩阵，U 是上三角矩阵

```
Matrix::lu(A)
```

```
LU factorization of Formal class 'denseLU' [package "Matrix"] with 4 slots
  ..@ x       : num [1:4] 1.2 0.833 3 -1.3
  ..@ perm    : int [1:2] 2 2
  ..@ Dim     : int [1:2] 2 2
  ..@ Dimnames:List of 2
  .. ..$ : NULL
  .. ..$ : NULL
```

©黄湘云

B.2.2 Schur 分解

矩阵 A 的 Schur 分解 $A = QTQ^\top$

```
(res <- Matrix::Schur(A))
```

```
$Q
           [,1]       [,2]
[1,] -0.9055894 -0.4241554
[2,]  0.4241554 -0.9055894

$T
          [,1]    [,2]
[1,] 0.4379501 0.00000
[2,] 0.0000000 3.56205

$EValues
[1] 0.4379501 3.5620499
```

其中 Q 是一个正交矩阵 $QQ = I$ ，T 是一个分块上三角矩阵

```
res$Q %*% t(res$Q)
```

```
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

```
res$Q %*% res$T %*% t(res$Q)
```

```
     [,1] [,2]
[1,]  1.0  1.2
[2,]  1.2  3.0
```

B.2.3 QR 分解

矩阵 A 的 QR 分解 $A = QR$

```
(res <- qr(A))
```

```
$qr
           [,1]       [,2]
[1,] -1.5620499 -3.0728851
[2,]  0.7682213  0.9986877

$rank
```

©北京大学出版社

```
[1] 2

$qraux
[1] 1.6401844 0.9986877

$pivot
[1] 1 2

attr(,"class")
[1] "qr"
```

QR 分解结果中的 Q

```
qr.Q(res)
```

```
           [,1]       [,2]
[1,] -0.6401844 -0.7682213
[2,] -0.7682213  0.6401844
```

QR 分解结果中的 R

```
qr.R(res)
```

```
         [,1]       [,2]
[1,] -1.56205 -3.0728851
[2,]  0.00000  0.9986877
```

恢复矩阵 A

```
qr.Q(res) %*% qr.R(res)
```

```
     [,1] [,2]
[1,]  1.0  1.2
[2,]  1.2  3.0
```

B.2.4 Cholesky 分解

矩阵 A 的 Cholesky 分解 $A = L^\top L$ ，其中 L 是上三角矩阵

```
(res <- chol(A))
```

```
     [,1]  [,2]
[1,]    1 1.200
[2,]    0 1.249
```

```
t(res) %*% res
```

©黄湘云


```
     [,1] [,2]
[1,]  1.0  1.2
[2,]  1.2  3.0
```

B.2.5 特征值分解

特征值分解（Eigenvalues Decomposition）也叫谱分解（Spectral Decomposition）

矩阵 A 的特征值分解 $A = V\Lambda V^{-1}$

```
(res <- eigen(A))
```

```
eigen() decomposition
$values
[1] 3.5620499 0.4379501

$vectors
          [,1]       [,2]
[1,] 0.4241554 -0.9055894
[2,] 0.9055894  0.4241554
```

返回值列表中的元素 vectors 就是 V

```
res$vectors %*% diag(res$values) %*% solve(res$vectors)
```

```
     [,1] [,2]
[1,]  1.0  1.2
[2,]  1.2  3.0
```

计算特征值，即求解如下一元 n 次方程

$|A - \lambda I| = 0$

```
rootSolve::uniroot.all(
  f = function(x) (x - 1) * (x - 3) - 1.2^2,
  lower = -10, upper = 10
)
```

```
[1] 0.4379747 3.5620253
```

B.2.6 SVD 分解

矩阵 A 的 SVD 分解 $A = UDV^\top$ ，矩阵 U 和 V 是正交的，矩阵 D 是对角的，矩阵 D 的对角元素是按降序排列的奇异值。

当矩阵是对称矩阵时，SVD 分解和特征值分解结果是一样的。

```
(res <- svd(A))

$d
[1] 3.5620499 0.4379501

$u
           [,1]       [,2]
[1,] -0.4241554 -0.9055894
[2,] -0.9055894  0.4241554

$v
           [,1]       [,2]
[1,] -0.4241554 -0.9055894
[2,] -0.9055894  0.4241554

# A = U D V'
res$u %*% diag(res$d) %*% t(res$v)

     [,1] [,2]
[1,]  1.0  1.2
[2,]  1.2  3.0

# D = U'AV
t(res$u) %*% A %*% res$v

              [,1]          [,2]
[1,]  3.562050e+00 -2.220446e-16
[2,] -5.551115e-17  4.379501e-01

# I = VV'
res$v %*% t(res$v)

     [,1] [,2]
[1,]    1    0
[2,]    0    1

# I = UU'
res$u %*% t(res$u)

              [,1]          [,2]
[1,]  1.000000e+00 -5.551115e-17
[2,] -5.551115e-17  1.000000e+00
```

仅供内部使用

B.3 Eigen 库

Eigen 是一个高性能的线性代数计算库，基于 C++ 编写，有 R 语言接口 **RcppEigen** 包。示例来自 **RcppEigen** 包，本文增加了特征向量，下面介绍如何借助 **RcppEigen** 包调用 Eigen 库做 SVD 矩阵分解。

```
#include <RcppEigen.h>

// [[Rcpp::depends(RcppEigen)]]

using Eigen::Map;                       // 'maps' rather than copies
using Eigen::MatrixXd;                  // variable size matrix, double precision
using Eigen::VectorXd;                  // variable size vector, double precision
using Eigen::SelfAdjointEigenSolver;    // one of the eigenvalue solvers

// [[Rcpp::export]]
VectorXd getEigenValues(Map<MatrixXd> M) {
  SelfAdjointEigenSolver<MatrixXd> es(M);
  return es.eigenvalues();
}
// [[Rcpp::export]]
MatrixXd getEigenVectors(Map<MatrixXd> M) {
  SelfAdjointEigenSolver<MatrixXd> es(M);
  return es.eigenvectors();
}
```

对上面的代码做几点说明：

1. `// [[Rcpp::depends(RcppEigen)]]` 可以看作一种标记，表示依赖 **RcppEigen** 包提供的 C++ 头文件，并导入到 C++ 命名空间中。`// [[Rcpp::export]]` 也可以看作一种标记，表示下面的函数需要导出到 R 语言环境中，这样 C++ 中定义的函数可以在 R 语言环境中使用。
2. `MatrixXd` 和 `VectorXd` 分别是 Eigen 库中定义的可变大小的双精度矩阵、向量类型。
3. `SelfAdjointEigenSolver` 是 Eigen 库中关于特征值分解方法中的一个求解器，特征值分解的结果有两个部分：一个是由特征值构成的向量，一个是特征向量构成的矩阵。求解器 `SelfAdjointEigenSolver` 名称中 `SelfAdjoint` 是伴随的意思，它是做矩阵 A 的伴随矩阵 $A^\star$ 的特征值分解。
4. `getEigenValues` 和 `getEigenVectors` 是用户自定义的两个函数名称，分别计算特征值和特征向量。

伴随矩阵的特征值分解和原矩阵的特征值分解有何关系？为什么不直接求原矩阵的特征值分解呢？

1. 伴随矩阵的特征值与原矩阵是一样的。
2. 伴随矩阵的特征向量有一个符号差异。

RcppEigen 包封装了 Eigen 库，它在 **RcppEigen** 包的源码路径为

`RcppEigen/inst/include/Eigen/src/Eigenvalues/SelfAdjointEigenSolver.h`

在 Eigen 库的源码路径如下：

`Eigen/src/Eigenvalues/SelfAdjointEigenSolver.h` 。

如何使用 **RcppEigen** 包加速计算？还是要看 Eigen 库的文档和源码，通过阅读源码，可以知道有哪些求解器，比如名称 `SelfAdjointEigenSolver` ，以及求解器包含的方法，比如 `eigenvalues()` 和 `eigenvectors()`，还有参数和返回值类型等。以特征值分解器 `SelfAdjointEigenSolver` 为例，编译上面的 C++ 代码，获得在 R 语言环境中可直接使用的函数 `getEigenValues()` 。

```
# 编译代码
Rcpp::sourceCpp(file = "code/rcpp_eigen.cpp")
```

然后，函数 `getEigenValues()` 计算特征值，返回一个向量。

```
# 计算特征值
getEigenValues(A)
```

```
[1] 0.4379501 3.5620499
```

返回一个矩阵，列是特征向量。

```
# 计算特征向量
getEigenVectors(A)
```

```
           [,1]       [,2]
[1,] -0.9055894 -0.4241554
[2,]  0.4241554 -0.9055894
```

根据上述分解结果计算矩阵 A 的伴随矩阵 $A^\star$ 。

```
t(getEigenVectors(A)) %*% diag(getEigenValues(A)) %*% getEigenVectors(A)
```

```
     [,1] [,2]
[1,]  1.0 -1.2
[2,] -1.2  3.0
```

B.4 应用

以线性模型为例讲述一些初步的计算性能提升办法。回顾一下线性回归的矩阵表示。

$$
\begin{aligned}
\boldsymbol{y} &= X\boldsymbol{\beta} + \boldsymbol{\epsilon} \\
\boldsymbol{\epsilon} &\sim \mathrm{MVN}(\mathbf{0}, \sigma^2 I)
\end{aligned}
$$

模型中 $\boldsymbol{\beta}, \sigma^2$ 是待估的参数，它们的最小二乘估计分别记为 $\hat{\boldsymbol{\beta}}, \hat{\sigma^2}$ 。

©黄湘云

$$\hat{\boldsymbol{\beta}} = (X^\top X)^{-1} X^\top \boldsymbol{y}$$
$$\hat{\sigma^2} = \frac{\boldsymbol{y}^\top (I - X(X^\top X)^{-1} X^\top) \boldsymbol{y}}{n - \operatorname{rank}(X)}$$

在获得参数的估计后，响应变量 $\boldsymbol{y}$ 的预测 $\hat{\boldsymbol{y}}$ 及其预测方差 $\mathsf{Var}(\hat{\boldsymbol{y}})$ 如下。

$$\hat{\boldsymbol{y}} = X(X^\top X)^{-1} X^\top \boldsymbol{y}$$
$$\mathsf{Var}(\hat{\boldsymbol{y}}) = \sigma^2 X(X^\top X)^{-1} X^\top$$

```
set.seed(2023)
n <- 200
p <- 50
x <- matrix(rnorm(n * p), n)
y <- rnorm(n)
fit_lm <- lm(y ~ x + 0)
```

下面不同的方法来计算预测值 $\hat{\boldsymbol{y}}$ ，从慢到快地优化。教科书版就是从左至右依次计算。

```
fit_base = function(x, y) {
  x %*% solve(t(x) %*% x) %*% t(x) %*% y
}
```

矩阵乘向量比矩阵乘矩阵快。虽然矩阵乘法没有交换律，但是有结合律。先向量计算，然后矩阵计算。

$$\hat{\boldsymbol{y}} = X(X^\top X)^{-1} X^\top \boldsymbol{y}$$

```
fit_vector <- function(x, y) {
  x %*% (solve(t(x) %*% x) %*% (t(x) %*% y))
}
```

解线性方程组比求逆快。$X^\top X$ 是对称的，通过解线性方程组来避免求逆。

$$\hat{\boldsymbol{y}} = X(X^\top X)^{-1} X^\top \boldsymbol{y}$$

```
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。

$$\begin{aligned}
\hat{\boldsymbol{y}} &= X(X^\top X)^{-1} X^\top \boldsymbol{y} \\
&= QR\big((QR)^\top QR\big)^{-1} (QR)^\top \boldsymbol{y} \\
&= QR(R^\top R)^{-1} R^\top Q^\top \boldsymbol{y} \\
&= QQ^\top \boldsymbol{y}
\end{aligned}$$

```
fit_qr <- function(x, y) {
  decomp <- qr(x)
  qr.qy(decomp, qr.qty(decomp, y))
}
fit_qr2 <- lm.fit(x, y)
```

其中，函数 `qr.qy(decomp, y)` 表示 `Q %*% y` ，函数 `qr.qty(decomp, y)` 表示 `t(Q) %*% y` 。实际上，Base R 提供的线性回归拟合函数 `lm()` 就采用 QR 分解。

Cholesky 分解。记 $A = X^\top X$ ，若 A 是正定矩阵，则 A 可做 Cholesky 分解。不妨设 $A = L^\top L$，其中 L 是上三角矩阵。

$$
\begin{aligned}
\hat{\boldsymbol{y}} &= X(X^\top X)^{-1} X^\top \boldsymbol{y} \\
&= X\left(L^\top L\right)^{-1} X^\top \boldsymbol{y} \\
&= X L^{-1} (L^\top)^{-1} X^\top \boldsymbol{y}
\end{aligned}
$$

```
fit_chol <- function(x, y) {
  decomp <- chol(crossprod(x))
  lxy <- backsolve(decomp, crossprod(x, y), transpose = TRUE)
  b <- backsolve(decomp, lxy)
  x %*% b
}
```

函数 `backsolve()` 求解上三角线性方程组。

© 基础课

附录 C　Git 和 Github

Git 是一个代码、数据和文档的版本管理工具，Github 提供一个用户界面。

C.1　安装配置

C.1.1　创建账户

登陆 Github 官网 (https://github.com/)，点击左上角注册按钮，开始注册 Github 账户。

图 C.1: 点击注册

接着，输入注册用的邮箱地址，比如 Outlook 和 Gmail 等。

除了邮箱外，继续输入密码、用户名等，密码可以选用浏览器自动生成的复杂字符串，只要没有被别人占用，用户名可以按着自己的喜好填写。

接着，系统要验证来注册 Github 账户的人是否是真人。

正确回答界面上出现的问题后，进入下一步，系统会给你之前提供的邮箱发送一个验证码。

图 C.2: 输入邮箱

图 C.3: 输入用户名

© 电子工业出版社

图 C.4: 回答问题

图 C.5: 输入验证码

将收到的验证码输入进去，完成账户验证。

图 C.6: 验证账户

创建账户后，将自动进入如下界面，接下来，可以创建代码仓库了。

C.1.2 安装 Git

在 MacOS 系统上，系统自带 Git 工具，无需安装。在 Ubuntu 系统上，安装最新稳定版的命令如下：

```
sudo add-apt-repository -y ppa:git-core/ppa
sudo apt update && sudo apt install git
```

在 Windows 系统上，安装最新稳定版的命令如下：

```
winget install --id Git.Git -e --source winget
```

C.1.3 配置密钥

在配置 GitHub 账户和安装完 Git 客户端后，接着配置密钥，以便将本地的代码推送到远程 Github 账户下的代码仓库。

```
git config --global user.name "用户名"
git config --global user.email "邮箱地址"
```

图 C.7: 创建代码仓库

C.1.4 (*) 账户共存

在公司往往会有自己的一套代码管理系统，比如 Gitlab 或者某种类似 Gitlab 的工具。本节介绍如何使 Gitlab / Github 账户共存在一台机器上。

如何生成 SSH 密钥见 Github 文档 — 使用 SSH 连接到 GitHub。有了密钥之后只需在目录 `~/.ssh` 下创建一个配置文件 config。

Github 对应个人的私有邮箱，Gitlab 对应公司分配的个人邮箱。

生成 SSH Key

```
ssh-keygen -t rsa -f ~/.ssh/id_rsa_github -C "个人邮箱地址"
ssh-keygen -t rsa -f ~/.ssh/id_rsa_gitlab -C "公司邮箱地址"
```

将 GitHub/GitLab 公钥分别上传至服务器，然后创建配置文件

```
touch ~/.ssh/config
```

配置文件内容如下

```
#
# Github
#
Host github.com // Github 代码仓库的服务器地址
HostName github.com
User XiangyunHuang
IdentityFile ~/.ssh/id_rsa_github
```

```
#
# company
#
Host xx.xx.xx.xx // 公司代码仓库的服务器地址
IdentityFile ~/.ssh/id_rsa_gitlab
```

配置成功，你会看到

```
ssh -T git@xx.xx.xx.xx
```

```
Welcome to GitLab, xiangyunhuang!
```

和

```
ssh -T git@github.com
```

```
Hi XiangyunHuang! You've successfully authenticated, but GitHub does not provide shell access.
```

C.2 基本操作

C.2.1 初始化仓库

git init

C.2.2 添加文件

git add

追踪当前目录下的内容

```
git add .
```

追踪被修改 (modified) 文件，不包括新添加的文件和被删除 (deleted) 的文件，`-u` 是 `--update` 的缩写

```
git add -u
```

添加所有文件，`-A` 是 `--all` 的缩写

```
git add -A
```

C.2.3 记录修改

git commit

```
git commit -m "添加提交说明"
```

©黄云清

C.2.4 推送修改

git push

```
git push -u origin master
```

C.2.5 克隆项目

克隆项目 `git clone`

```
git clone git@github.com:XiangyunHuang/data-analysis-in-action.git
```

有的项目包含子模块，添加选项 `--recursive` 可以将子模块也克隆下来。

```
git clone --recursive git@github.com:cosname/cosx.org.git
```

C.3 分支操作

对每一个新的问题，创建新的分支，提交新的 PR。

与人协作开发代码项目，往往涉及 Git 分支操作。通常有两个场景，其一是独立地在分支上进行开发，包含创建分支、修改分支、提交分支、合并分支和删除分支。其二是与人合作互相评审代码修改分支，除了之前的基础操作，还包含在分支上解决代码冲突，同步分支内容。

图 C.8: Git 分支操作

C.3.1 创建分支

```
git checkout -b 分支名称
```

C.3.2 分支切换

```
git checkout 分支名称
```

C.3.3 修改 PR

```
# 拉取合作者的 PR
git fetch origin refs/pull/771/head:patch-2
```

©黄湘云

```
# 771 是 PR 对应的编号
git checkout patch-2

# 你的修改

git add -u # 追踪修改的内容
git commit -m "描述修改内容"

git remote add LalZzy https://github.com/LalZzy/cosx.org.git
git push --set-upstream LalZzy patch-2
```

C.3.4 (*) 创建 gh-pages 分支

基于 GitHub Pages 创建站点用于存放图片和数据。

1. 在 Github 上创建一个空的仓库，命名为 uploads。
2. 在本地创建目录 uploads。
3. 切换到 uploads 目录下，执行如下命令。

```
git init
git checkout -b gh-pages
git remote add origin https://github.com/XiangyunHuang/uploads.git
```

添加图片或者数据，并推送到 gh-pages 分支。

```
git add README.md
git commit "消息"
git push --set-upstream origin gh-pages
```

这样仓库 uploads 只包含 gh-pages 分支，`README.md` 文件地址为

https://xiangyunhuang.github.io/uploads/README.md

C.4 R 与 Git 交互

usethis 包将 Git 操作封装了，特别是一些复杂的操作，比如修改他人的 PR

C.4.1 从 R 操作 Git

拉取编号为 1019 的 PR

```
usethis::pr_fetch(1019)
```

1019 是 PR 的编号，修改完，清理

© 统计出版社


```
usethis::pr_finish()
```

C.4.2 分析 Git 记录

给我的仓库点赞的人有哪些，如果有很多，仅显示第一页。

```
library(gh)
my_repos <- gh("GET /repos/:owner/:repo/stargazers",
               owner = "XiangyunHuang", page = 1,
               repo = "data-analysis-in-action")
vapply(my_repos, "[[", "", "login")
```

Jeroen Ooms 开发的 **gert** 包，提供了 `git_rm()`、`git_status()`、`git_add()` 和 `git_commit()` 等函数，其中包含 `git_reset()` 、`git_branch_*()` 等高级 Git 操作。查看最近的 5 条提交记录。

```
library(gert)
git_log(max = 5)
```

更多内容，读者请看 Gert: A minimal git client for R。

git2r 包对 Git 仓库进行概要。

```
summary(git2r::repository())
```

gitdown 包将 Git 提交日志转化为 GitBook

截止 2023 年 6 月 1 日，统计之都的主站仓库，提交量最大的 10 个人。

```
git shortlog -sn | head -n 10
```

```
 153    Dawei Lang
 127    Yihui Xie
 101    Ryan Feng Lin
  93    Beilei Bian
  65    Xiangyun Huang
  46    王佳
  42    雷博文
  39    Miao YU
  35    xiangyun
  32    fanchao
```

C.5 (*) 辅助工具

Git 扩展 git-delta 和 tig 是两款辅助工具。tig 用于查看提交的历史日志。

C.5.1 语法高亮

git-delta

```
brew install git-delta
```

对 `git diff` 的输出提供语法高亮

C.5.2 文本接口

在 MacOS 上，推荐用 Homebrew 安装

```
brew install tig
```

C.5.3 大文件存储

Git Large File Storage (LFS) Git LFS

```
# MacOS
brew install git-lfs
# Ubuntu
sudo apt install git-lfs
```

配置 Git LFS

```
git lfs install
```

项目中的大型数据文件

```
git lfs track "*.csv"
git add .gitattributes
git commit -m "Git LFS 追踪数据文件"
git push origin master
```

© 黄湘云

索引

Quarto, 1

统计检验, 308