R 语言数据分析实战

黄湘云

2023 年 8 月 27 日

目录

插图目录

列表目录

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，最新一次编译时间是 2023-08-27 12:10:00。

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

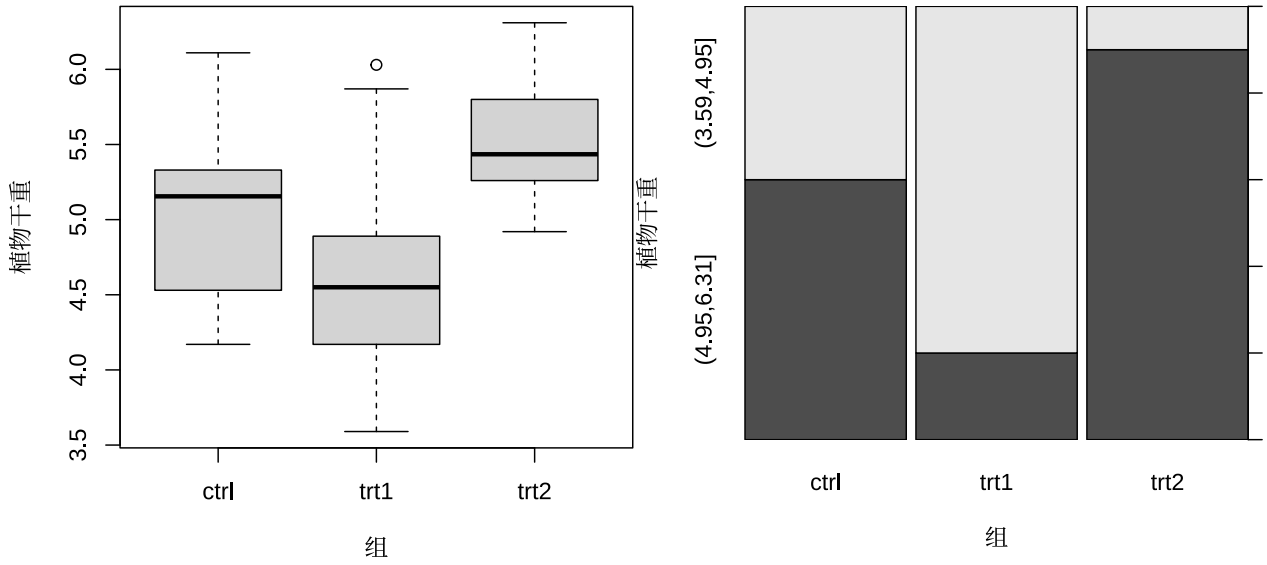
```
  ), dependencies = FALSE)
```

```
#> R version 4.3.1 (2023-06-16)
#> Platform: x86_64-redhat-linux-gnu (64-bit)
#> Running under: Fedora Linux 38 (Container Image)
#>
#>
#> Locale:
#>   LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
#>   LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
#>   LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
#>   LC_PAPER=en_US.UTF-8       LC_NAME=C
#>   LC_ADDRESS=C               LC_TELEPHONE=C
#>   LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
#>
#> time zone: UTC
#> tzcode source: system (glibc)
#>
#> Package version:
#>   data.table_1.14.8 dplyr_1.1.2       DT_0.28           gganimate_1.0.8
#>   ggbeeswarm_0.7.2  ggdensity_1.0.0   ggeffects_1.3.0   ggforce_0.4.1
#>   ggnewscale_0.4.9  ggplot2_3.4.3     ggraph_2.1.0      ggrepel_0.9.3
#>   ggridges_0.5.4    ggsignif_0.6.4    gifski_1.12.0.2   glmnet_4.1.8
#>   gt_0.9.0          knitr_1.43        lattice_0.21.8    lme4_1.1.34
#>   magick_2.7.5      patchwork_1.1.3   plotly_4.10.2     purrr_1.0.2
#>   rmarkdown_2.24    rsconnect_1.0.2   sf_1.0.14         showtext_0.9.6
#>   stars_0.6.3       tidyr_1.3.0       tinytex_0.46      xgboost_1.7.5.1
#>
#> Pandoc version: 3.1.1
#>
#> LaTeX version used:
#>   TeX Live 2022 with tlmgr 2022-04-18
```

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")
```

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
- 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。

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。

第一章　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。

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 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/。

1.1 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 1.1 。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 1.1 的截距和斜率，R^2 代表模型拟合数据的程度。

$$y = \beta_0 + \beta_1 x + \epsilon \tag{1.1}$$

表格 1.1: 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
v_lines	54.270	16.770	47.837	26.938	53.891	-0.112	0.005

子数据集	$\bar{x}$	σ_x	$\bar{y}$	σ_y	β_0	β_1	R^2
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 1.1 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。

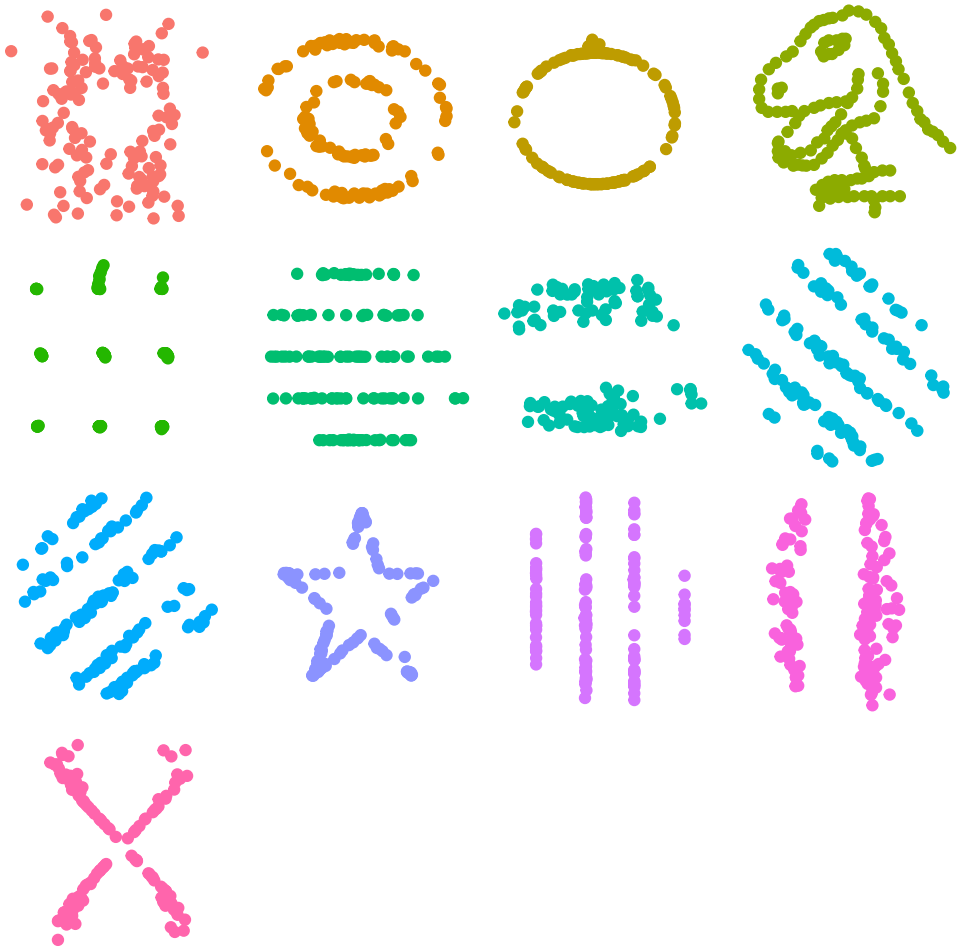


图 1.1: 数据可视化为何如此重要

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 1.1 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 1.2 所示。

表格 1.2: 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

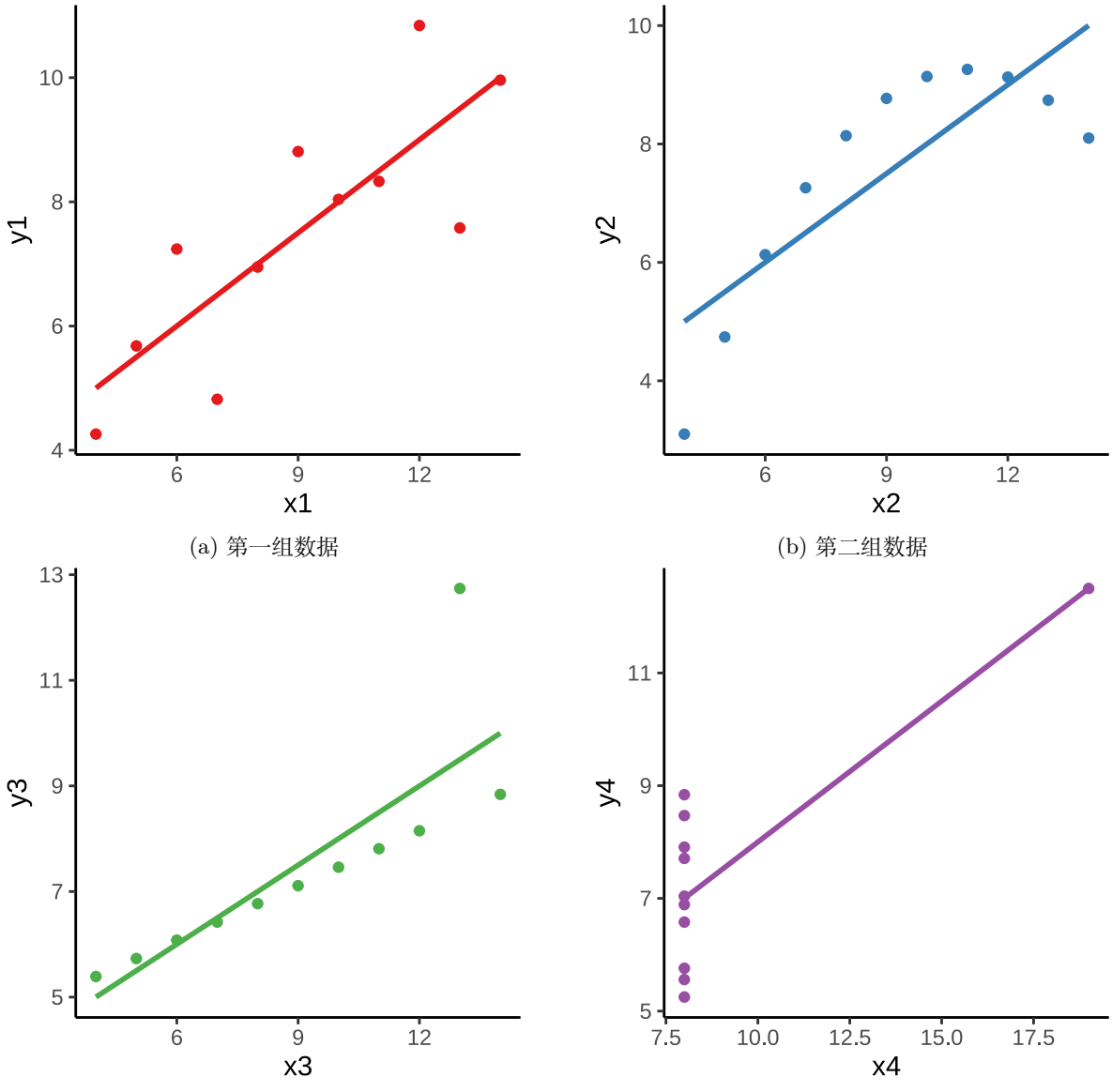
用统计的方法发现四组数据的样本均值、方差、相关系数和回归系数几乎是相同的，实际上，借助散点图 11.15 分别描述各组数据的关系时，却发现四组数据之间有极大的差异，且只有第一组数据看起来符合线性模型的条件 (Anscombe 1973)。

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。

1.2 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。



(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 1.2: 数据可视化为何如此重要

第章首先介绍 **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。

第章首先介绍 **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。

第章首先介绍 **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。

第章首先回顾 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。

第一部分

数据准备

第二章　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。

2.1 数据类型

2.1.1 整型

2.1.2 逻辑型

2.1.3 字符型

2.1.4 日期型

2.1.5 数值型

2.2 数据结构

2.2.1 向量

2.2.2 矩阵

2.2.3 数组

2.2.4 列表

2.2.5 因子

2.2.6 数据框

2.2.7 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,
  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 AirPassengers 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(AirPassengers)
```

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

```
# 存储类型
mode(AirPassengers)
```

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

```
# 数据结构
str(AirPassengers)
```

```
 Time-Series [1:144] from 1949 to 1961: 112 118 132 129 121 135 148 148 136 119 ...
```

函数 `start()` 和 `end()` 查看开始和结束的时间点。

```
c(start(AirPassengers), end(AirPassengers))
```

```
[1] 1949    1 1960   12
```

函数 `time()` 可以查看在以上时间区间的划分。

```
time(AirPassengers)
```

```
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
1949 1949.000 1949.083 1949.167 1949.250 1949.333 1949.417 1949.500 1949.583
1950 1950.000 1950.083 1950.167 1950.250 1950.333 1950.417 1950.500 1950.583
1951 1951.000 1951.083 1951.167 1951.250 1951.333 1951.417 1951.500 1951.583
1952 1952.000 1952.083 1952.167 1952.250 1952.333 1952.417 1952.500 1952.583
1953 1953.000 1953.083 1953.167 1953.250 1953.333 1953.417 1953.500 1953.583
1954 1954.000 1954.083 1954.167 1954.250 1954.333 1954.417 1954.500 1954.583
1955 1955.000 1955.083 1955.167 1955.250 1955.333 1955.417 1955.500 1955.583
1956 1956.000 1956.083 1956.167 1956.250 1956.333 1956.417 1956.500 1956.583
1957 1957.000 1957.083 1957.167 1957.250 1957.333 1957.417 1957.500 1957.583
1958 1958.000 1958.083 1958.167 1958.250 1958.333 1958.417 1958.500 1958.583
1959 1959.000 1959.083 1959.167 1959.250 1959.333 1959.417 1959.500 1959.583
```

```
1960 1960.000 1960.083 1960.167 1960.250 1960.333 1960.417 1960.500 1960.583
          Sep       Oct       Nov       Dec
1949 1949.667 1949.750 1949.833 1949.917
1950 1950.667 1950.750 1950.833 1950.917
1951 1951.667 1951.750 1951.833 1951.917
1952 1952.667 1952.750 1952.833 1952.917
1953 1953.667 1953.750 1953.833 1953.917
1954 1954.667 1954.750 1954.833 1954.917
1955 1955.667 1955.750 1955.833 1955.917
1956 1956.667 1956.750 1956.833 1956.917
1957 1957.667 1957.750 1957.833 1957.917
1958 1958.667 1958.750 1958.833 1958.917
1959 1959.667 1959.750 1959.833 1959.917
1960 1960.667 1960.750 1960.833 1960.917
```

函数 tsp() 可以查看其期初、期末和周期。

```
tsp(AirPassengers)
```

```
[1] 1949.000 1960.917   12.000
```

第三章　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。

3.1　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

3.1.1　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。

3.1.2　xlsx 文件

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

openxlsx 读/写 xlsx 文件

3.1.3 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。

3.2 从数据库中导入

从各类数据库导入数据，比如 RSQLite 等

3.2.1 RSQLite

3.2.2 odbc

3.2.3 RJDBC

很多数据库都有 Java 接口驱动

3.3 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。

3.3.1 豆瓣排行榜

3.3.2 链家二手房

3.4 从数据接口中获取

3.4.1 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。

3.4.2 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。

3.4.3 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。

3.4.4 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

第四章　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` 。

4.1　正则表达式

4.1.1　量词

4.1.2　级联

4.1.3　断言

正向查找 / 反向查找

4.1.4　反向引用

4.1.5　命名捕捉

4.2　字符串操作

4.2.1　查找

`grep()` / `grepl()` 返回是否匹配的结果

4.2.2　替换

`sub()` / `gsub()` 替换一次和多次

4.2.3 提取

`regexpr()` / `gregexpr()`

`regexec()` / `gregexec()`

第五章　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。

5.1　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

5.1.1　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
```

5.1.2　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
```

5.1.3 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
```

5.1.4 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。

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

```
# Source:   SQL [3 x 2]
# Database: sqlite 3.41.2 [/usr/local/lib/R/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。

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

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

5.2 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

5.2.1 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.Wid
```

```
   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
```

5.2.2 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 ...
 $ Species_N   : int  1 1 1 1 1 1 1 1 1 1 ...
```

5.2.3 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
```

5.2.4 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
```

5.2.5 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
```

```
# 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
```

```
# 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
```

5.2.6 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。

```
# 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
```

```
  # 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
```

5.3 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。
3. **data.table** 特有的高级数据操作 `on`、`.SD` 、`.I` 、`.J` 等。

5.3.1 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
```

5.3.2 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>
```

5.3.3 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
```

5.3.4 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
```

5.3.5 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"))
```

```r
# 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
```

```r
# 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
```

```r
# 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
```

```r
# 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
```

5.3.6 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。

```r
# 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
```

第六章　数据处理

6.1　缺失值处理

缺失是一种非常常见的数据问题。

6.1.1　查找

缺失值在数据框中的位置

6.1.2　汇总

缺失值的占比、分布情况，可视化获得缺失的结构 VIM

6.1.3　替换

替换数据框中的缺失值

6.1.4　插补

mice Multivariate Imputation by Chained Equations 缺失值插补

6.2　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。

6.2.1 检测

6.2.2 识别

6.2.3 处理

6.3 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。

6.3.1 检测

6.3.2 识别

6.3.3 处理

第二部分

数据探索

第七章　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 7.1 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 7.1: 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。

7.1 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 7.1a 仅提供数据，只渲染出来一个绘图区域。图 7.1b 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 7.1c 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 7.2: **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
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。

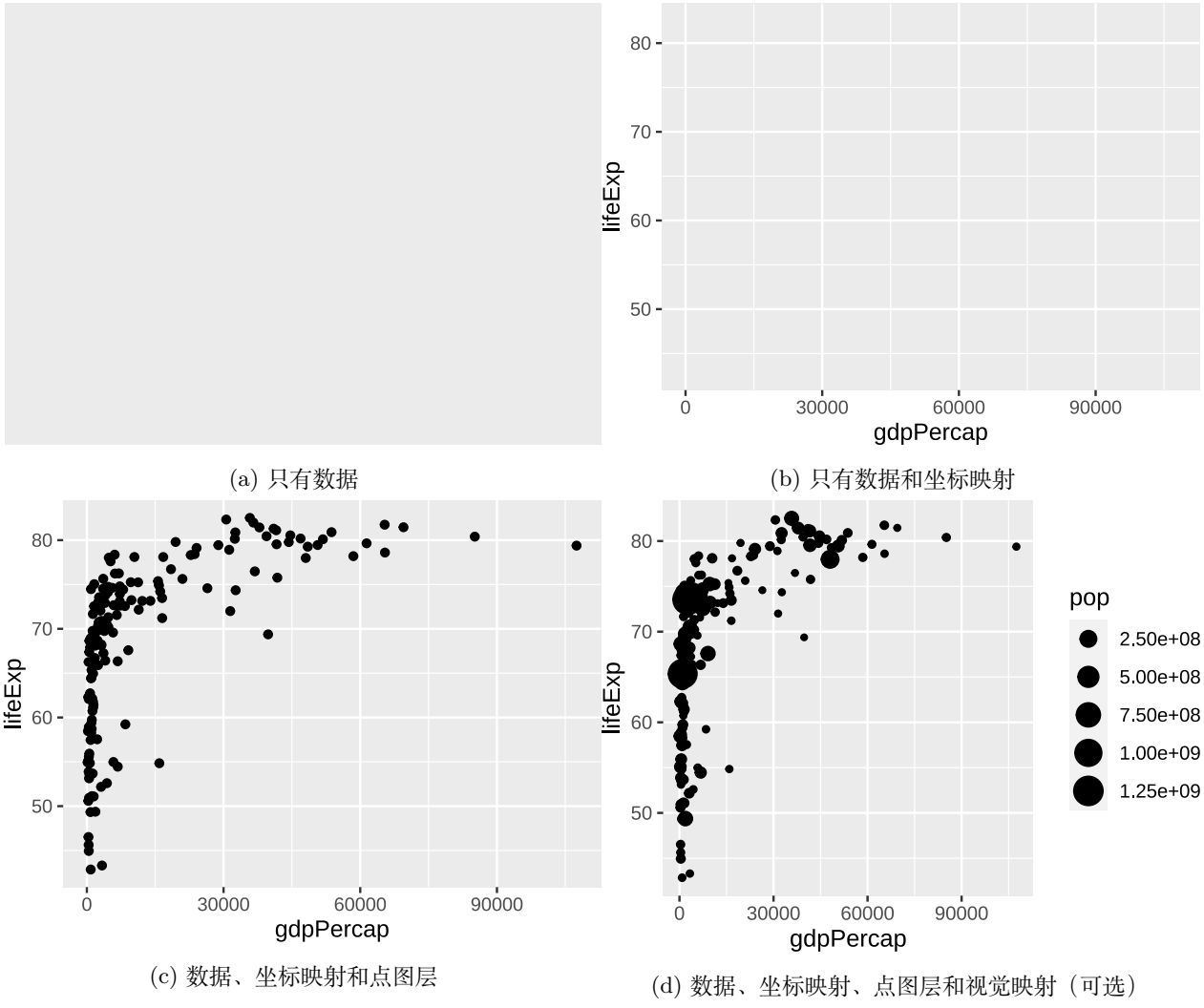
7.2 标签

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 7.2a 是默认设置下显示的标签内容，而图 7.2b 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")
```

7.3 刻度

有时候图 7.1c 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 7.3 。



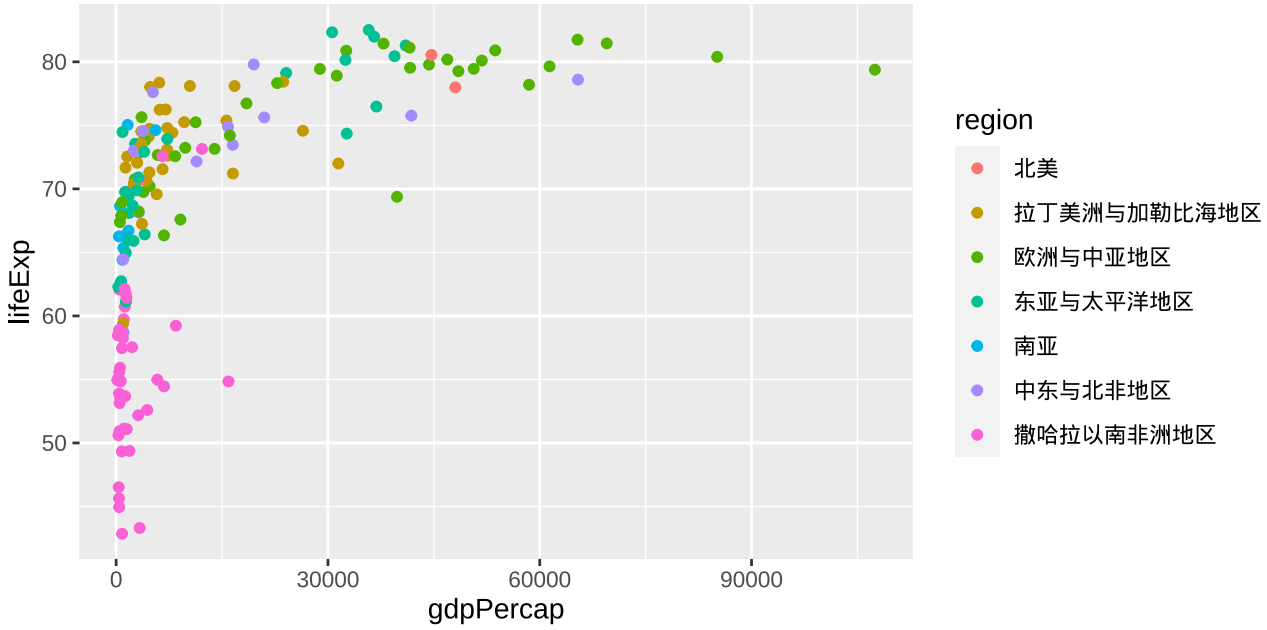
(a) 只有数据

(b) 只有数据和坐标映射

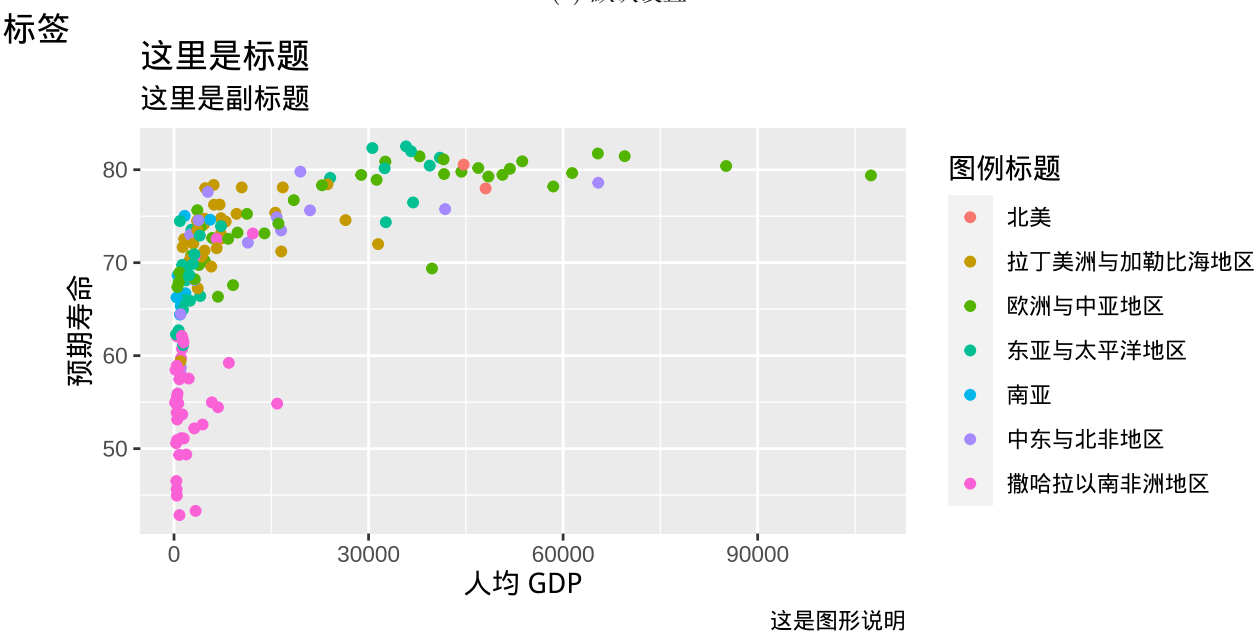
(c) 数据、坐标映射和点图层

(d) 数据、坐标映射、点图层和视觉映射（可选）

图 7.1: ggplot2 绘图三要素



(a) 默认设置



(b) 自定义标签

图 7.2: 添加标签

```
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")
```

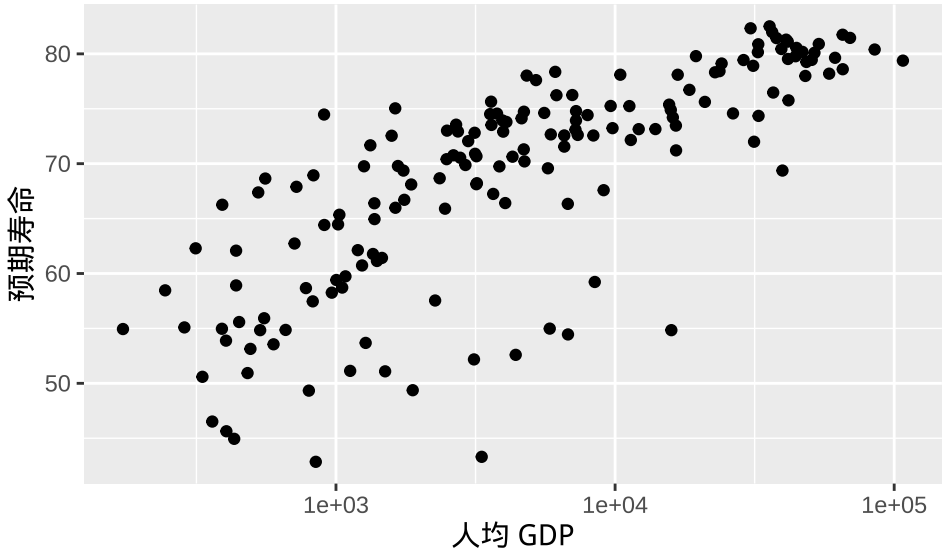


图 7.3: 人均 GDP 做对数变换

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 7.4 。

```
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")
```

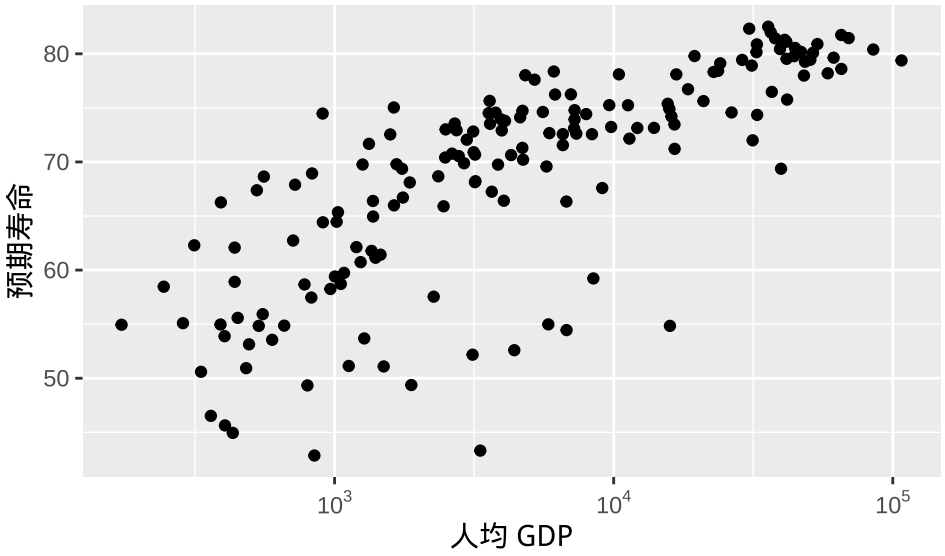


图 7.4: 刻度标签随数据变换调整

这其实还不够，有的刻度标签含义不够显然，且看图 7.4 的横轴第一个刻度标签 $10^{2.48}$ 是用来替换图 7.3 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 7.5 。

```
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")
```

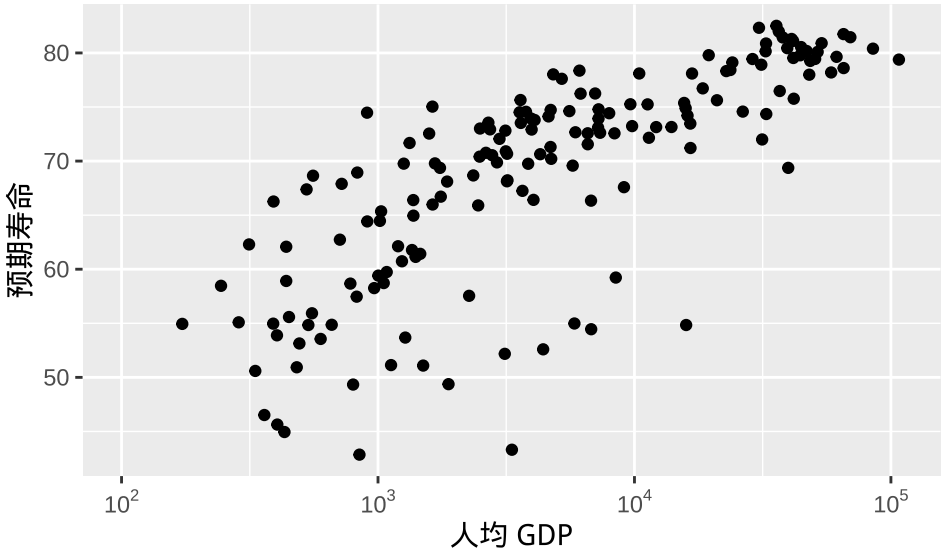


图 7.5: 设置数据展示范围

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 7.6 。

```
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")
```

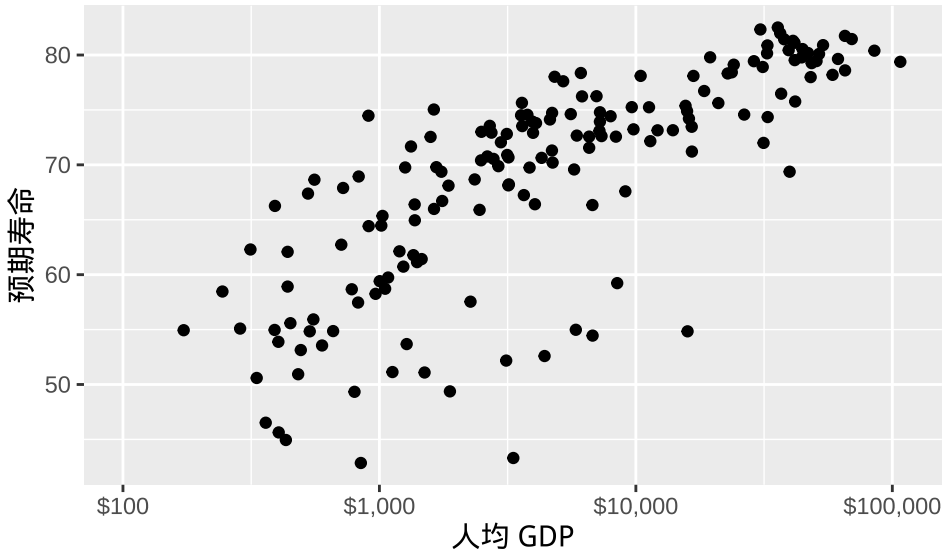


图 7.6: 设置数据展示范围

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 7.7 。

```
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")
```

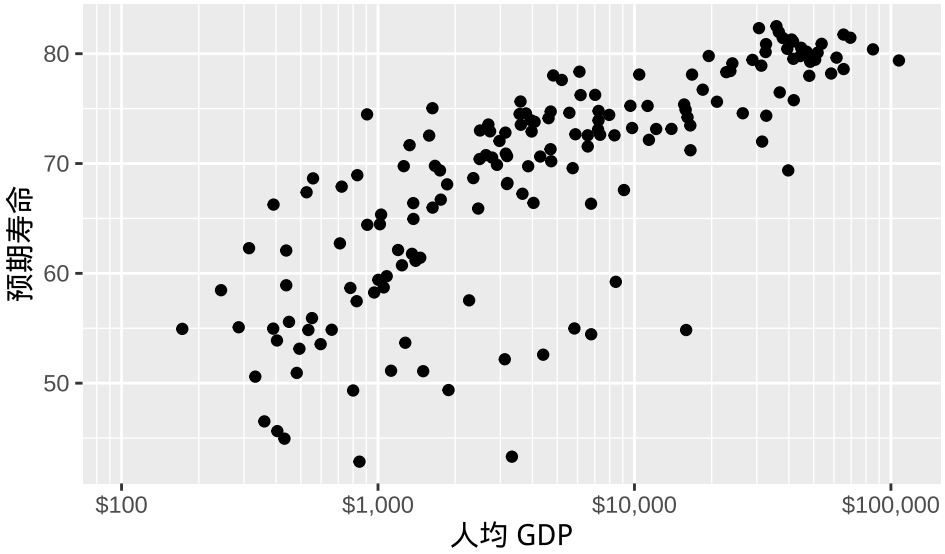


图 7.7: 添加次刻度线，提供更多参考

7.4 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 7.7 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 7.8 。

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

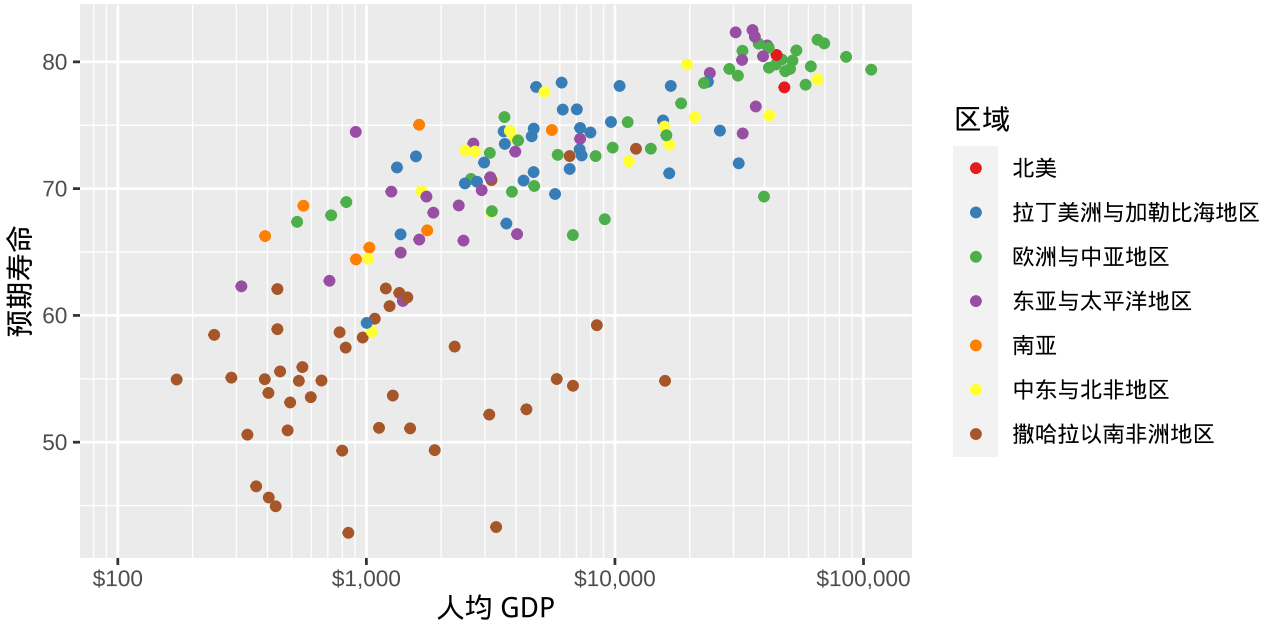


图 7.8: 使用 RColorBrewer 包提供的 Set1 调色板

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 7.9 。

```
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")
```

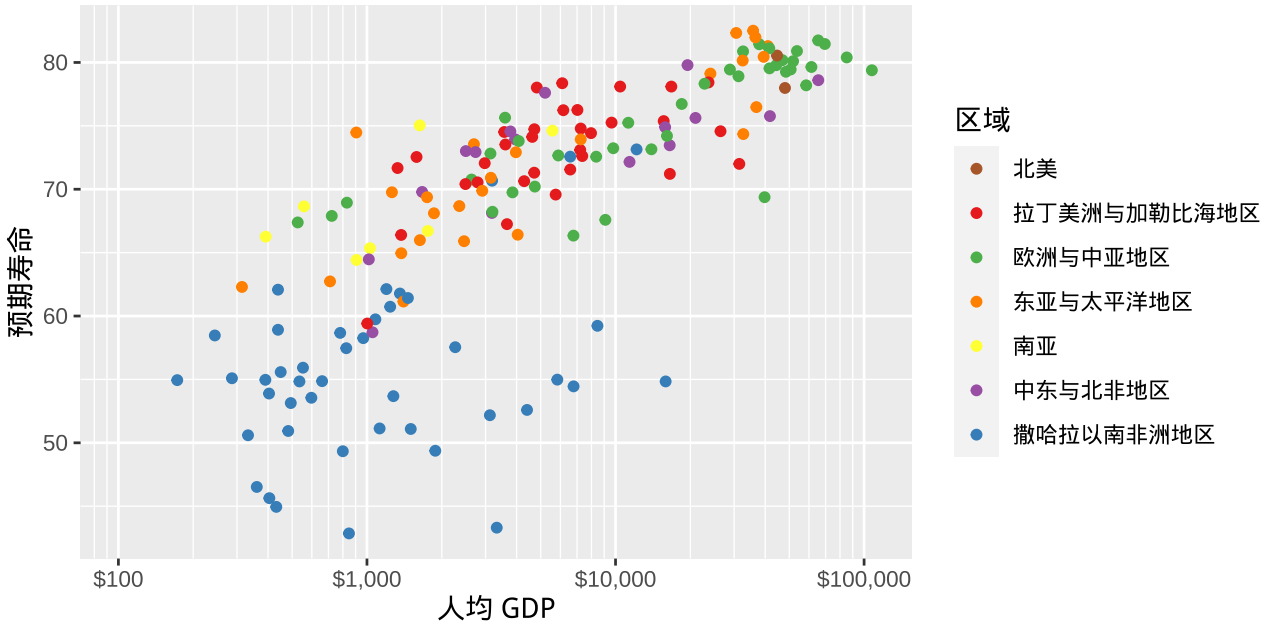


图 7.9: 手动挨个指定分类变量的颜色

7.5 图例

在图 7.8 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 7.10 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")
```

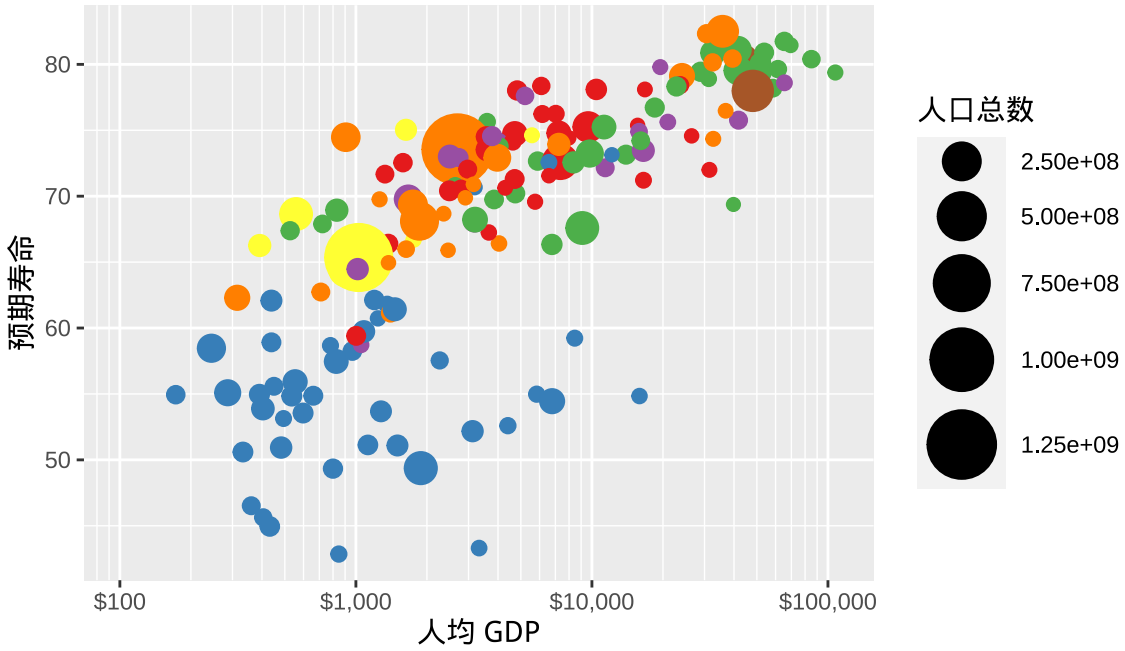


图 7.10: 在两个图例中保留一个

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 7.11 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")
```

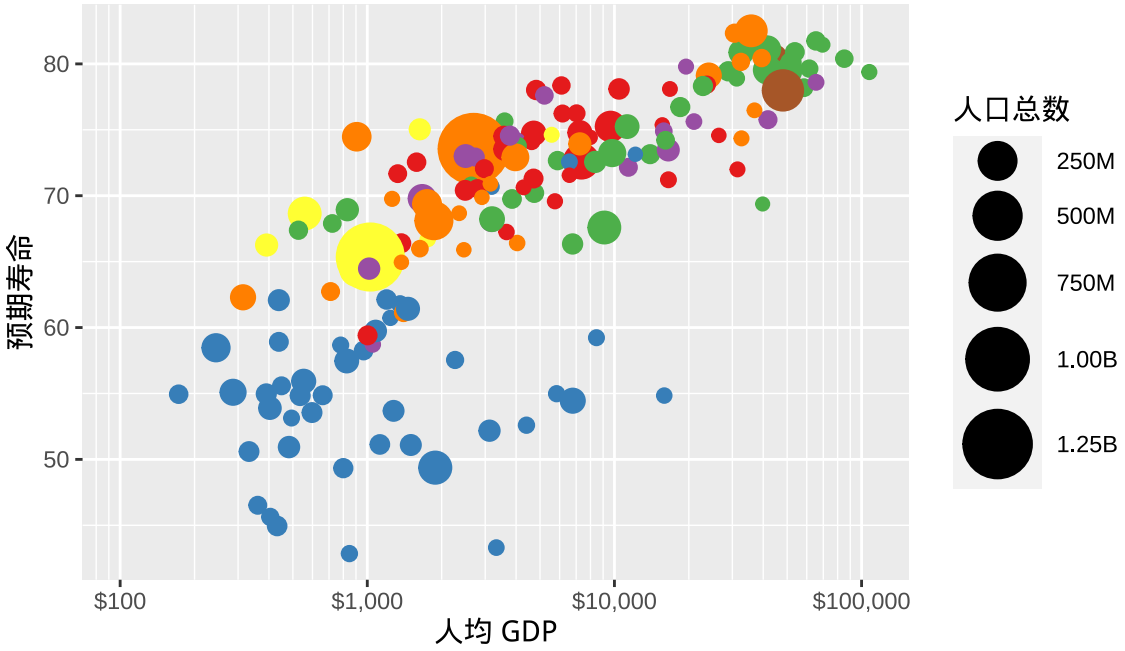


图 7.11: 修改图例刻度标签

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

在图 7.11 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 7.12，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),
    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)) +
  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")
```

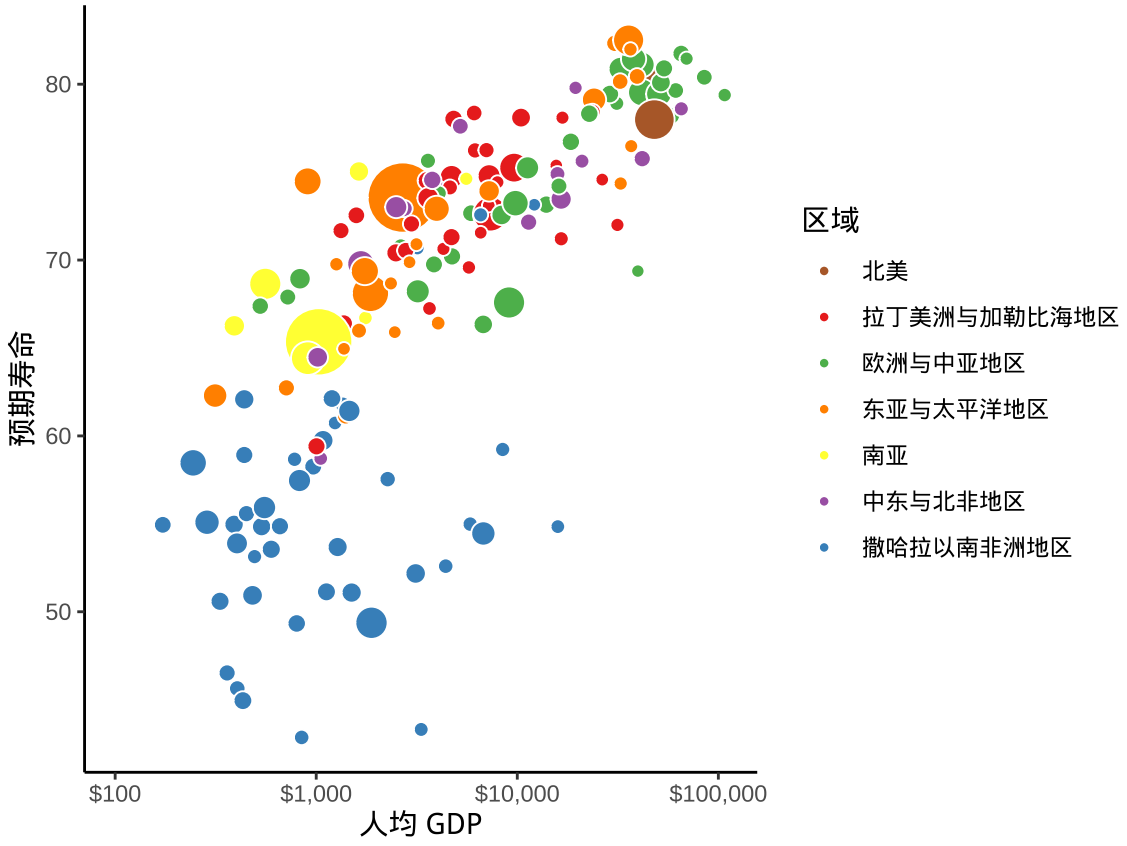


图 7.12: ggplot2 内置的经典主题风格

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 7.13 。

```
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)) +
  scale_x_log10(
    labels = label_dollar(), minor_breaks = mb, limits = c(100, 110000)
```

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

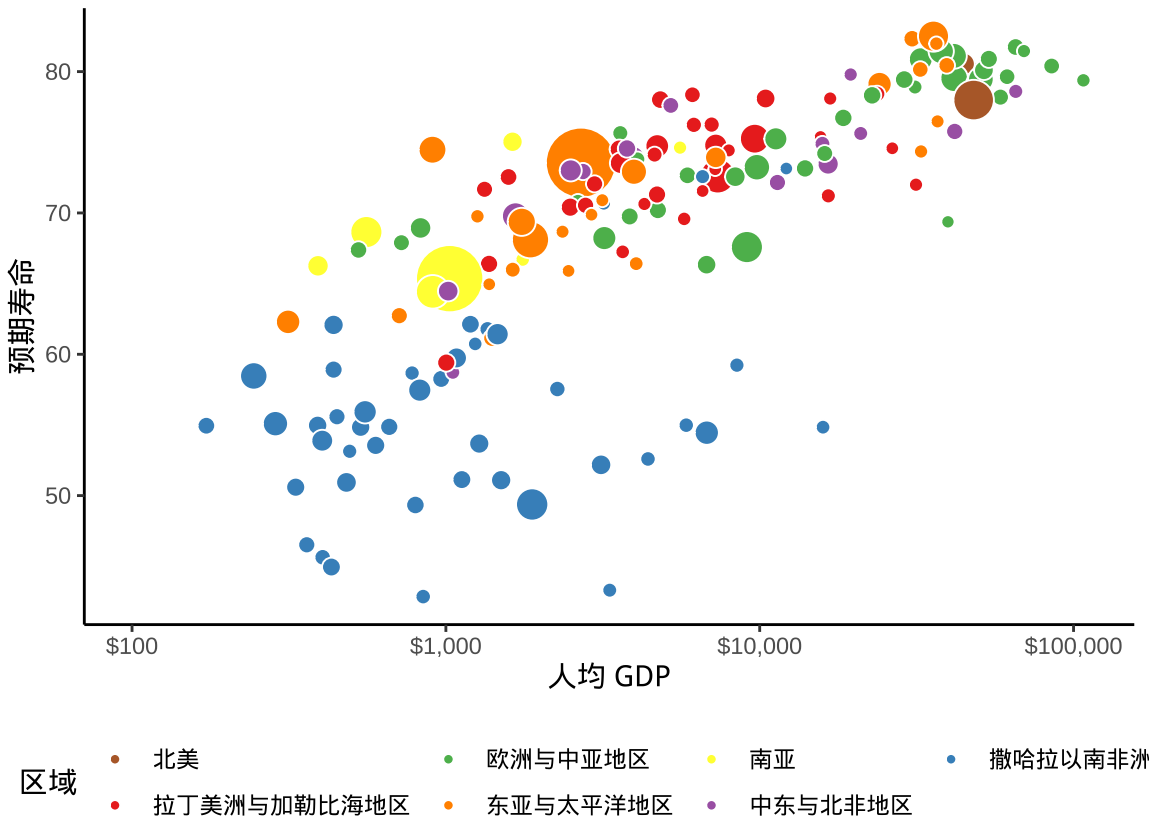


图 7.13: 图例置于图形下方

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),
    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)) +
scale_x_log10(
  labels = label_dollar(), minor_breaks = mb, limits = c(100, 110000)
) +
theme_classic() +
theme(legend.position =  c(0.875, 0.3)) +
labs(x = "人均 GDP", y = "预期寿命", fill = "区域")
```

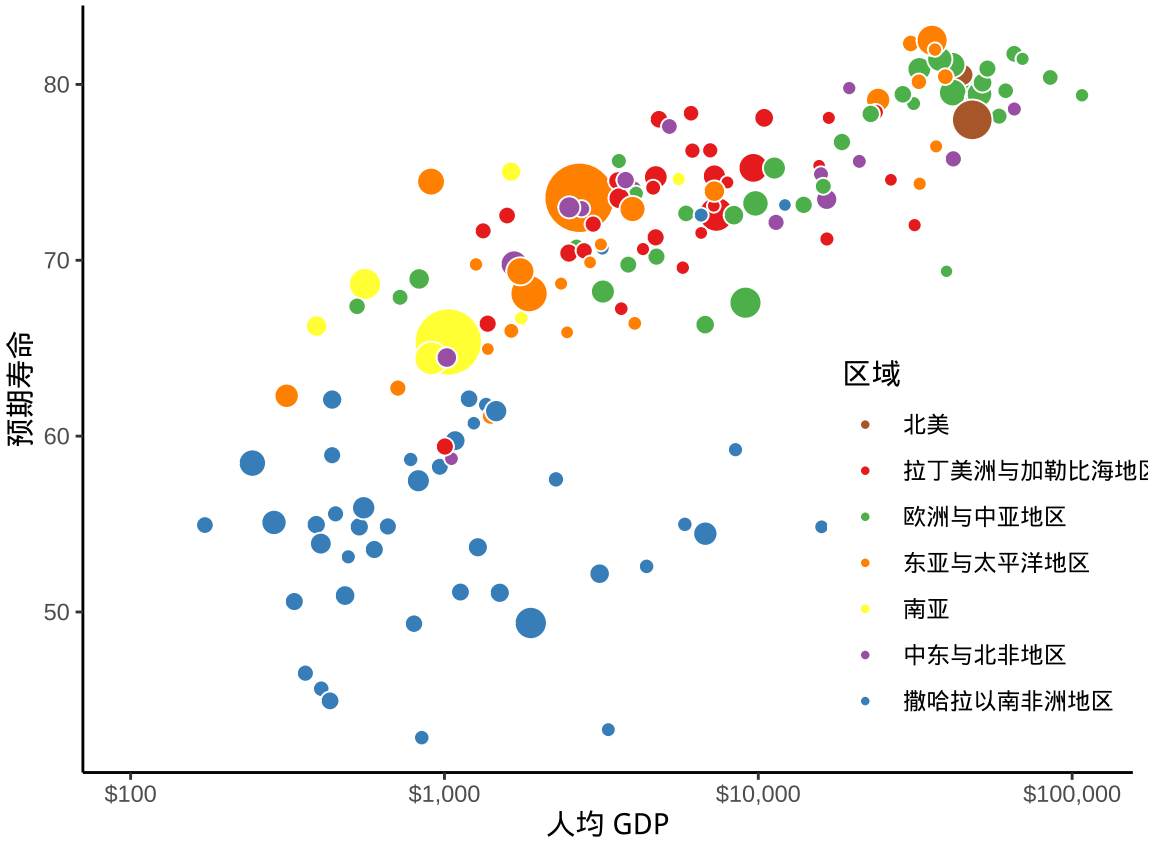


图 7.14: 微调图例位置

或者更换其它主题，比如 **ggplot2** 包内置极简主题 `theme_minimal()`，它还可以保留主、次刻度线，更加适合当下的数据。

```
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)) +
scale_x_log10(
  labels = label_dollar(), minor_breaks = mb, limits = c(100, 110000)
) +
theme_minimal() +
theme(legend.position =  c(0.875, 0.3)) +
labs(x = "人均 GDP", y = "预期寿命", fill = "区域")
```

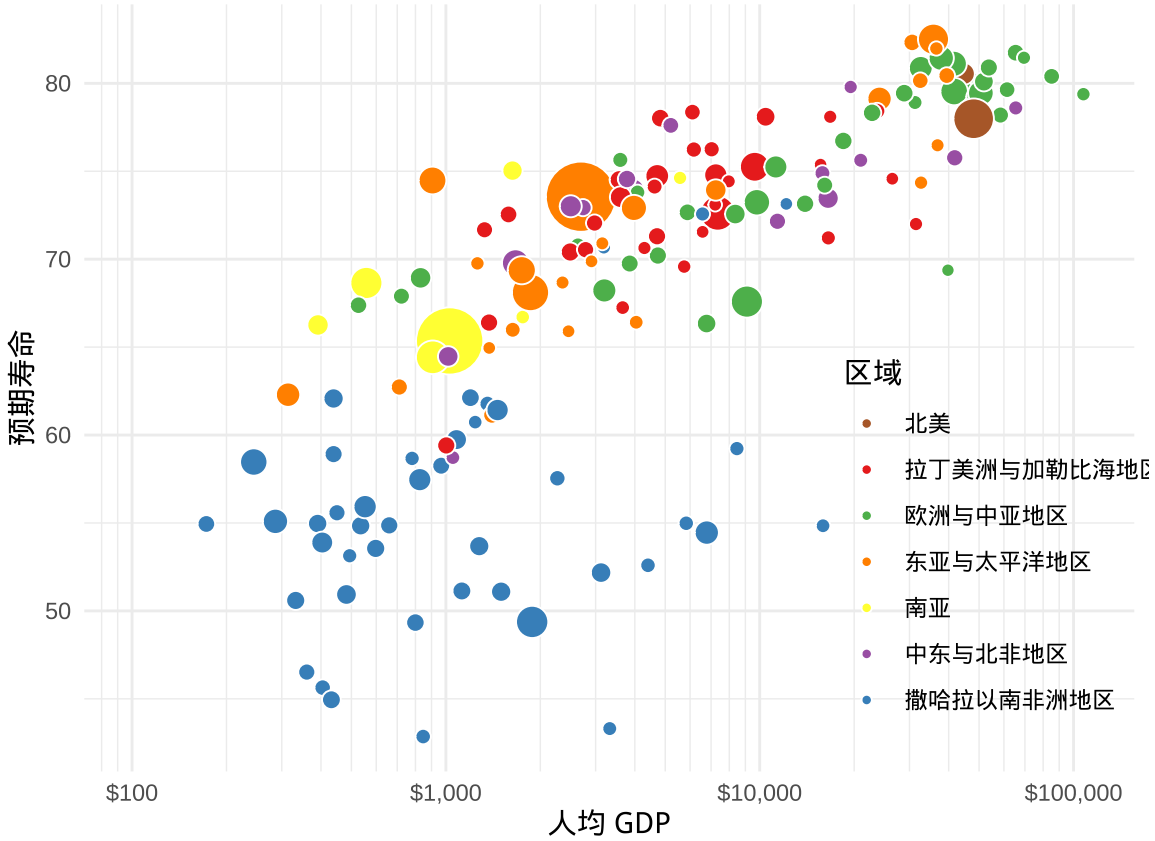


图 7.15: ggplot2 内置的极简主题风格

7.7 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 7.15 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 7.16 。

```
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_minimal() +
```

```
  theme(legend.position = c(0.9, 0.35)) +
  labs(x = "人均 GDP", y = "预期寿命", fill = "区域")
```

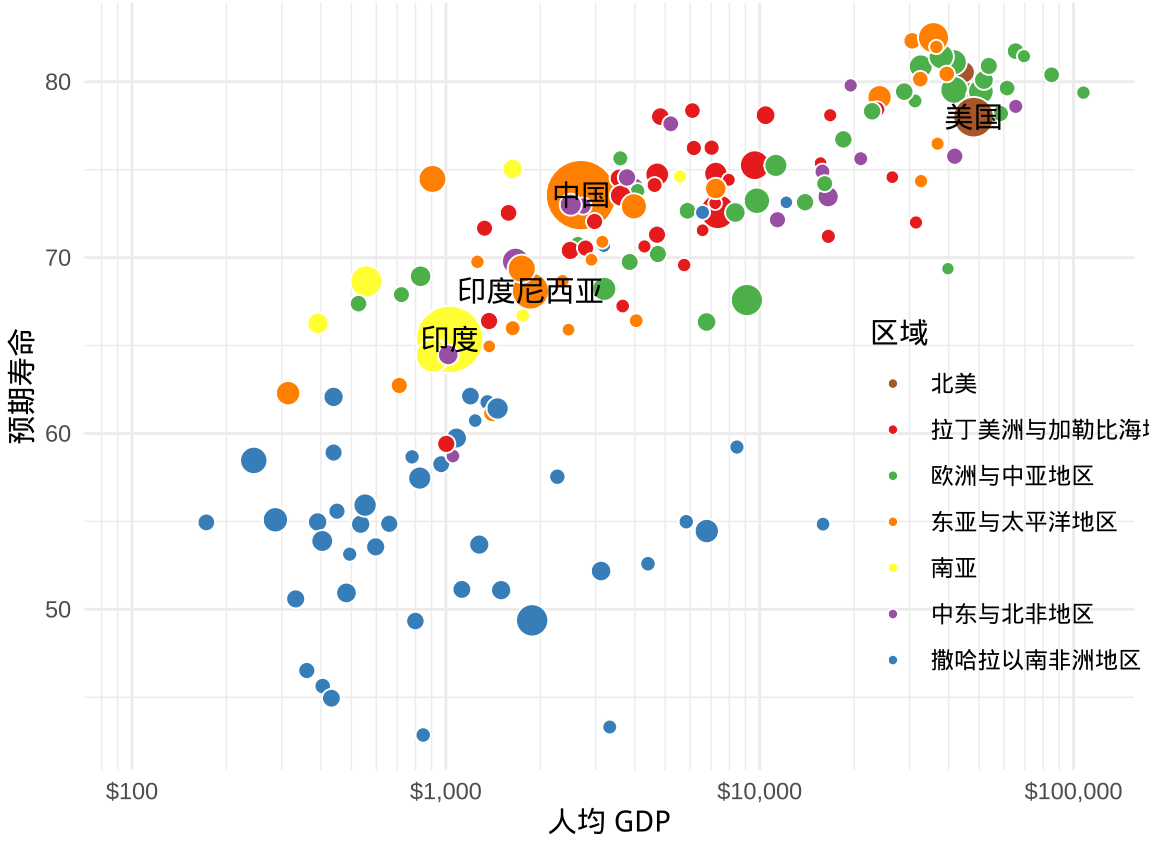


图 7.16: 添加文本注释

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(
    `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_minimal() +
theme(legend.position = c(0.9, 0.35)) +
labs(x = "人均 GDP", y = "预期寿命", fill = "区域")
```

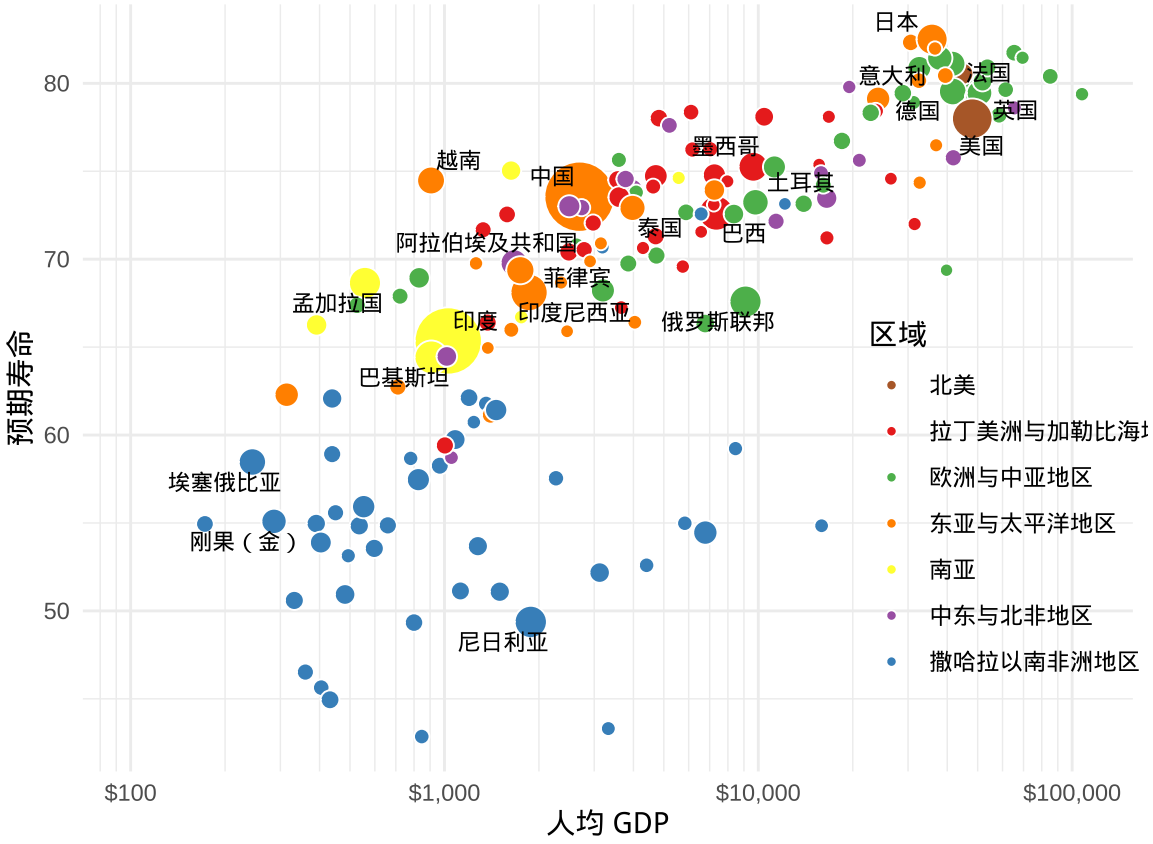


图 7.17: 缓解文本注释相互覆盖的问题

7.8 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 7.15 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 7.18 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)) +
  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")
```

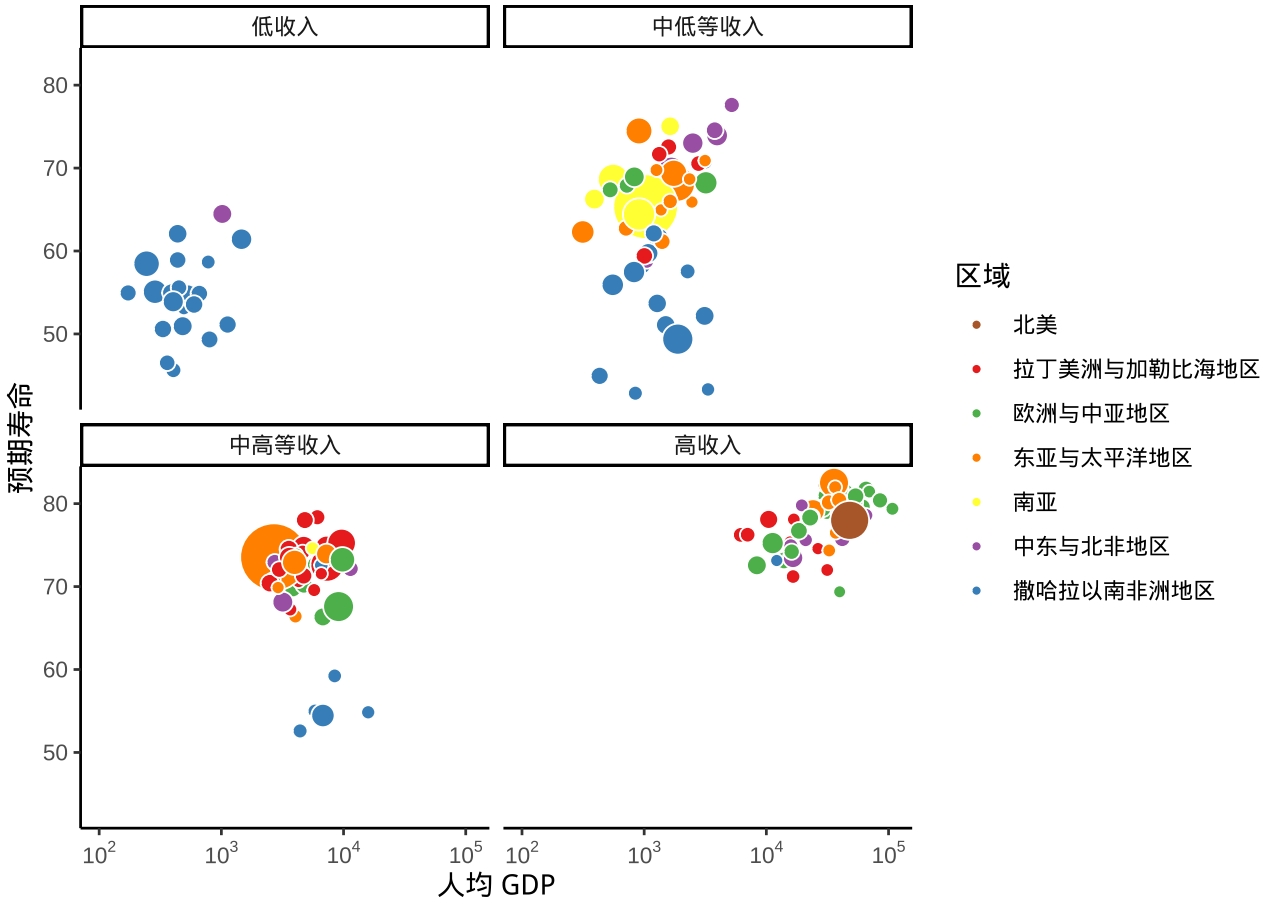


图 7.18: 按收入水平变量分面

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 7.19 。

```
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",
    `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 = c(0.9, 0.2)) +
  labs(x = "人均 GDP", y = "预期寿命", fill = "区域")
```

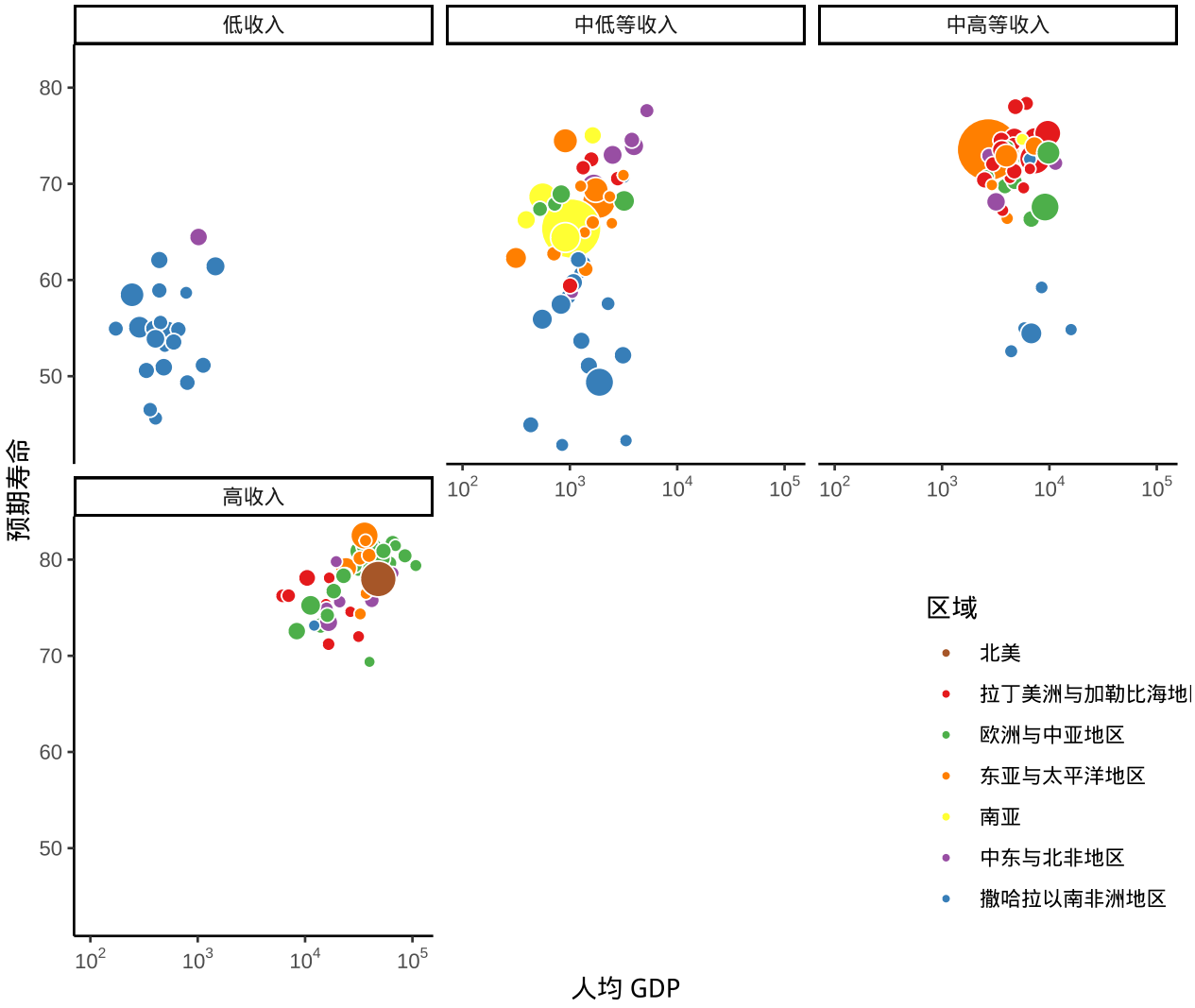


图 7.19: 按区域变量分面

7.9 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 **?@fig-animate-facets** ，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(color = region, size = pop),
    show.legend = c(color = TRUE, size = TRUE), alpha = 0.65
  ) +
  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_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", color = "区域"
  ) +
  transition_time(time = year)
```

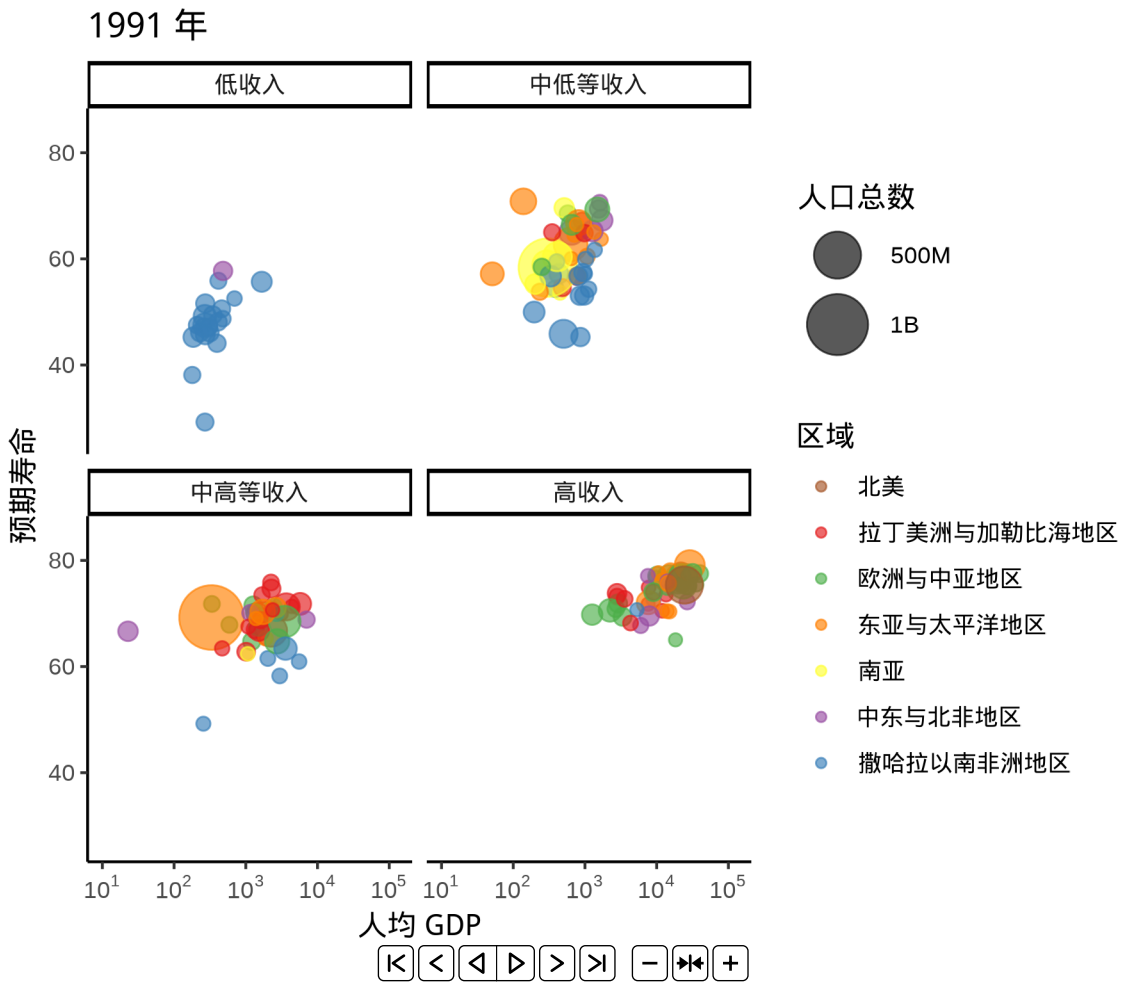


图 7.20: 生成 GIF 动画

第八章　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 8.1 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 8.2 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 8.3 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。

8.1　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。

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！

8.1.1 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")
})
```

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 8.1 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")
```

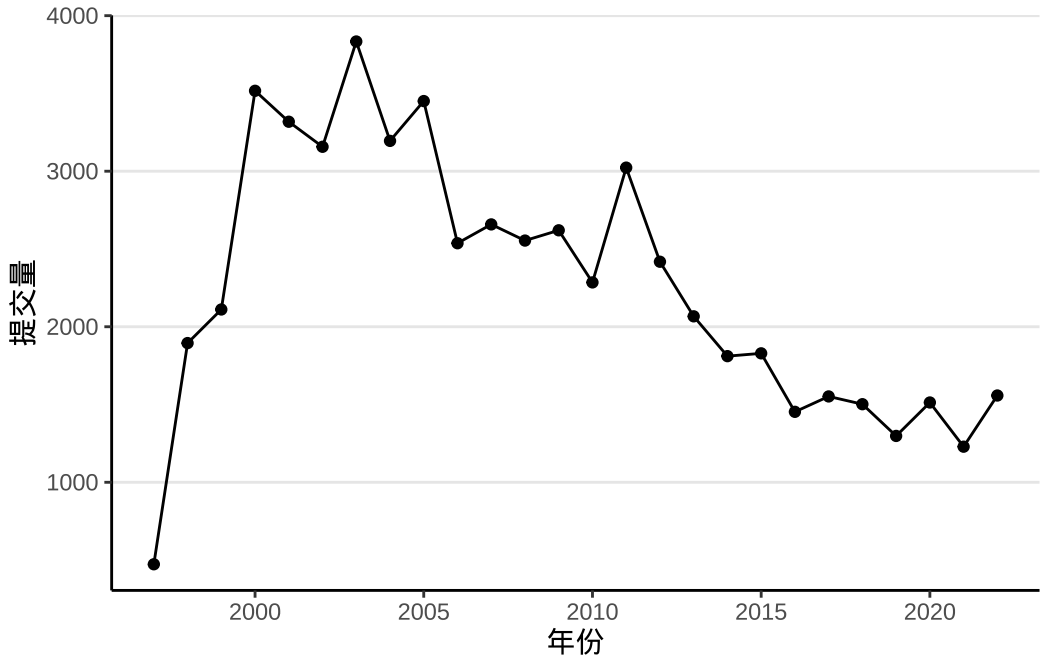


图 8.1: 过去 25 年代码提交次数的变化情况

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。

小时趋势

上午高峰

```
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")
```

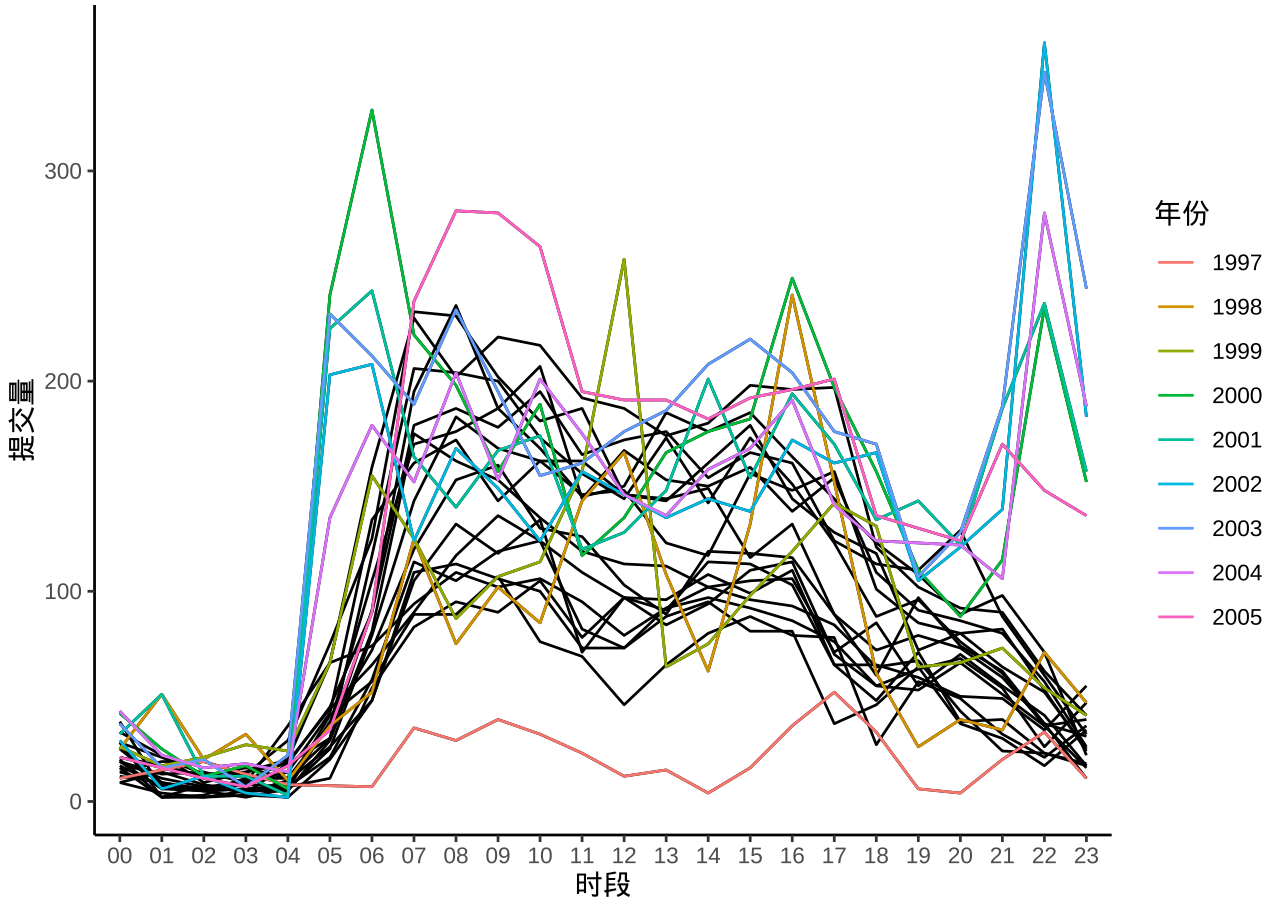


图 8.2: 提交代码的时段分布

月趋势

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。是否有新人加入？

```
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")
```

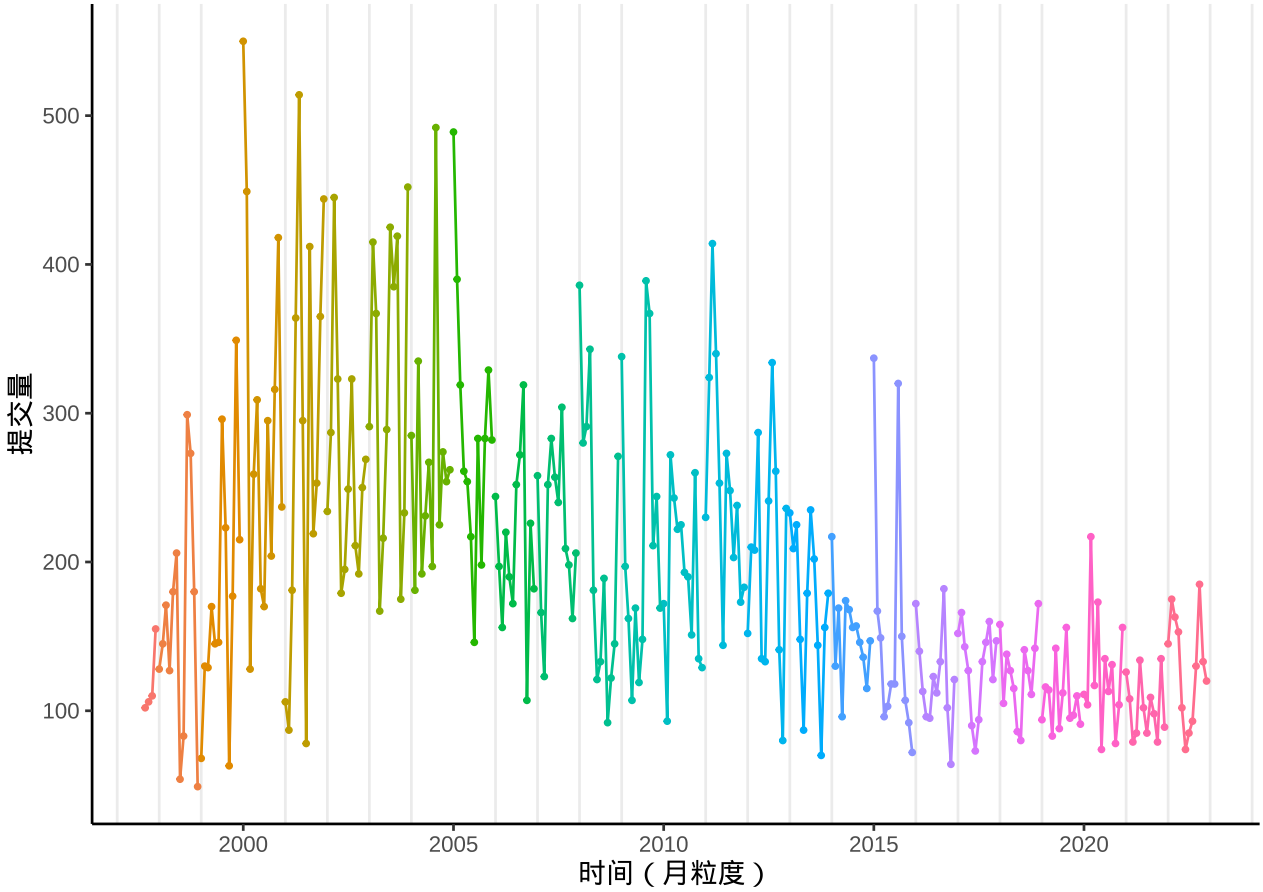


图 8.3: 提交代码的月份分布

8.1.2 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 8.4 所示，每年的提交量就好像瀑布上的水，图中当期相对于上一期的增减量

```
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() +
  geom_rect(
    data = trunk_year_tmp,
    aes(
      xmin = xmin, ymin = ymin,
      xmax = xmax, ymax = ymax, fill = fill
    ), show.legend = FALSE
  ) +
  theme_classic() +
  theme(panel.grid.major.y = element_line(colour = "gray90")) +
  labs(x = "年份", y = "提交量")
```

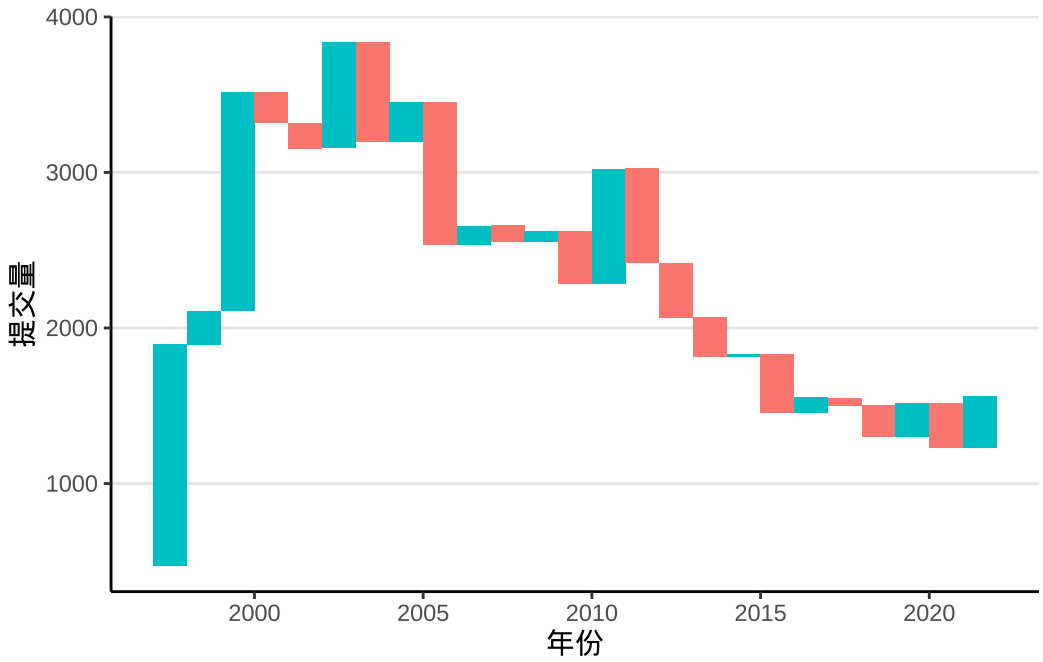


图 8.4: 25 年代码逐年提交量的变化趋势

瀑布图是以数据点作为对角点构造矩形，用对撞型的颜色表示增减，下图 8.5 在图 8.4 基础上添加了数据点，用以直观说明矩形图层 `geom_rect()` 构造瀑布图的方法。

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

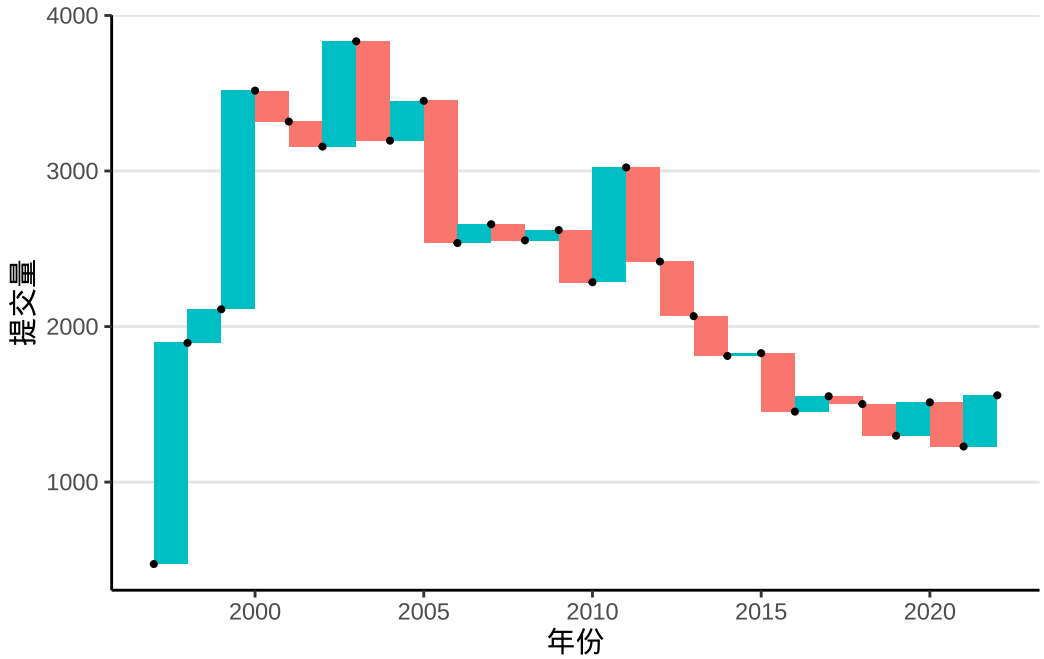


图 8.5: 矩形图层构造瀑布图

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 8.6 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")) +
  labs(x = "年份", y = "提交量")
```

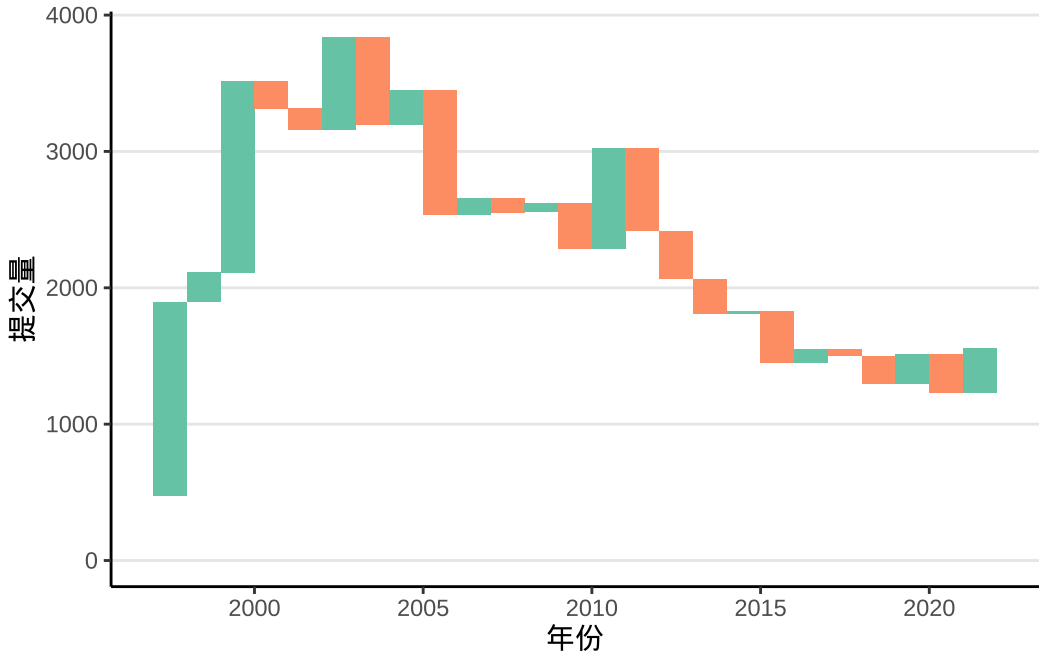


图 8.6: 矩形图层构造瀑布图

8.1.3 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 8.7 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'
```

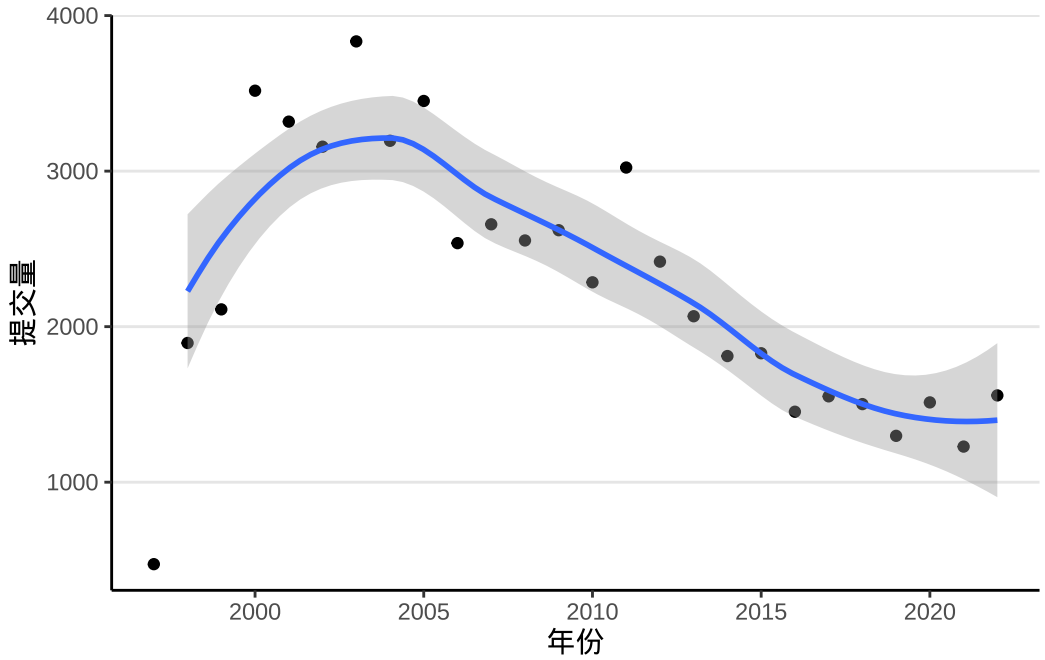


图 8.7: 过去 25 年代码提交次数的变化情况

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 8.8

```
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")
```

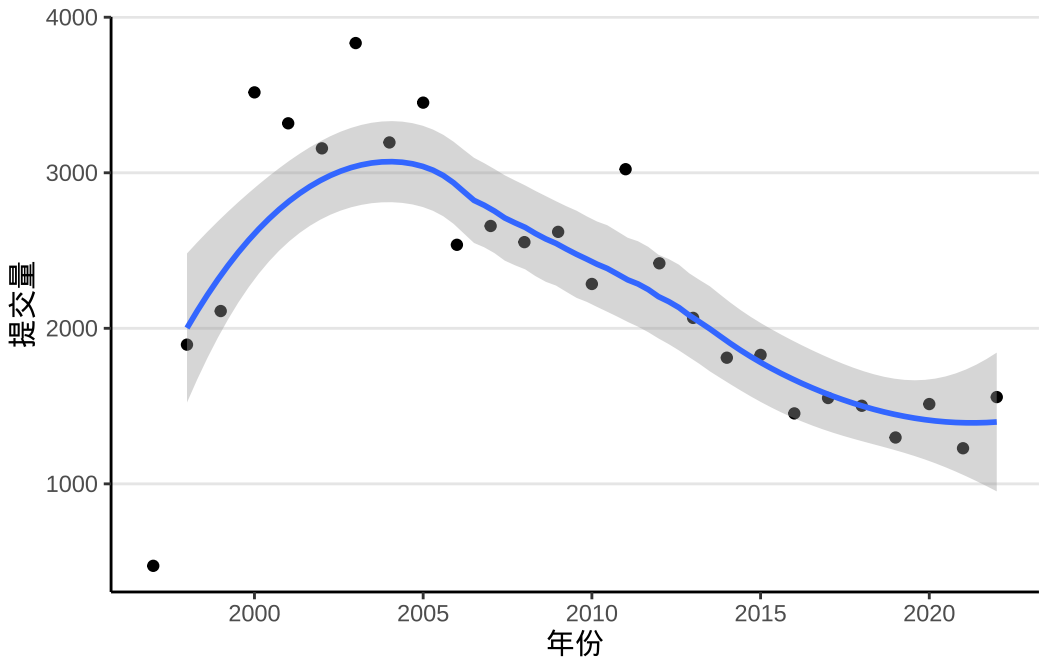


图 8.8: 过去 25 年代码提交次数的变化情况

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 8.9 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),
  ) +
```

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

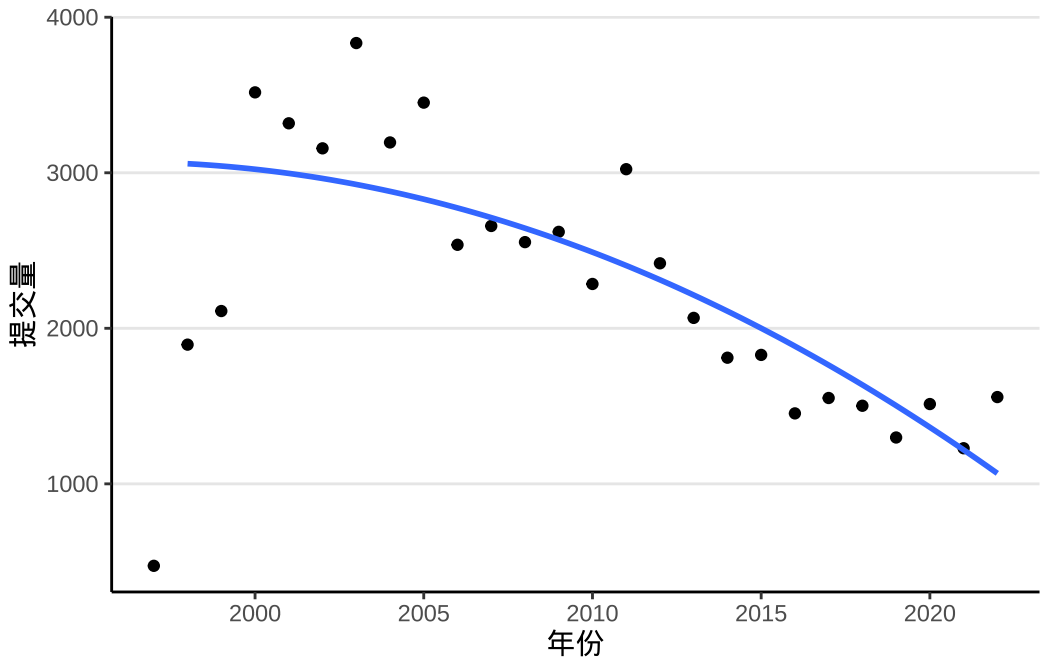


图 8.9: 过去 25 年代码提交次数的变化情况

> **警告**
>
> 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 8.8 和图 8.9 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:
#>   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。

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

```
  class(res) <- "xxspline"
  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")
```

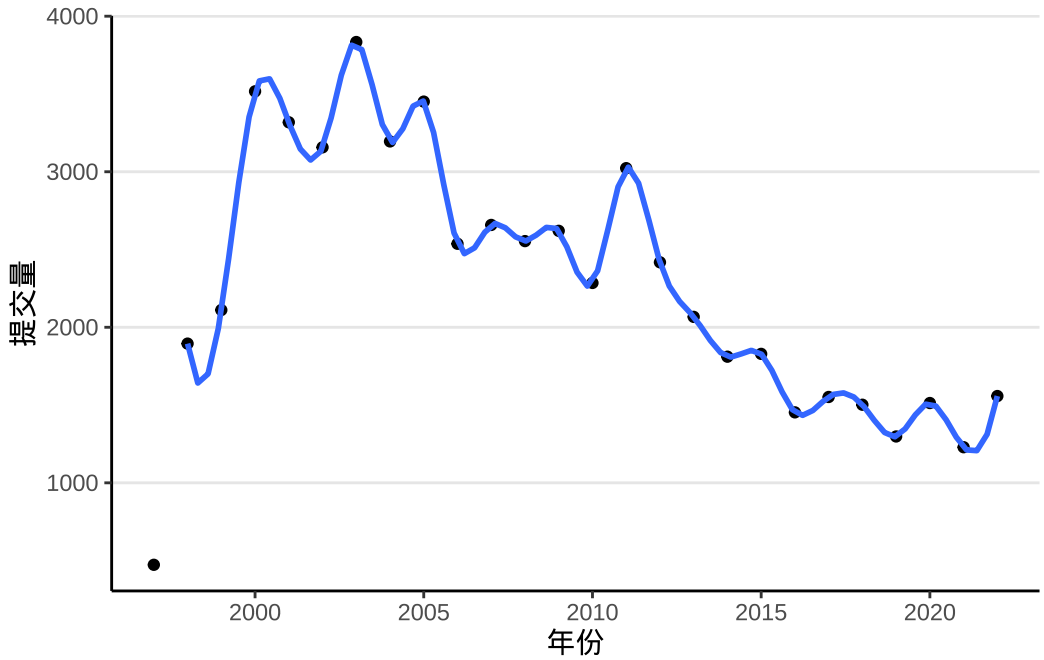


图 8.10: 过去 25 年代码提交次数的变化情况

借助 spline 插值获得平滑曲线，ggalt 包的函数 `geom_xspline` 也提供类似的功能。

ggplot2 当前支持的平滑方法，还有很多

```
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")
```

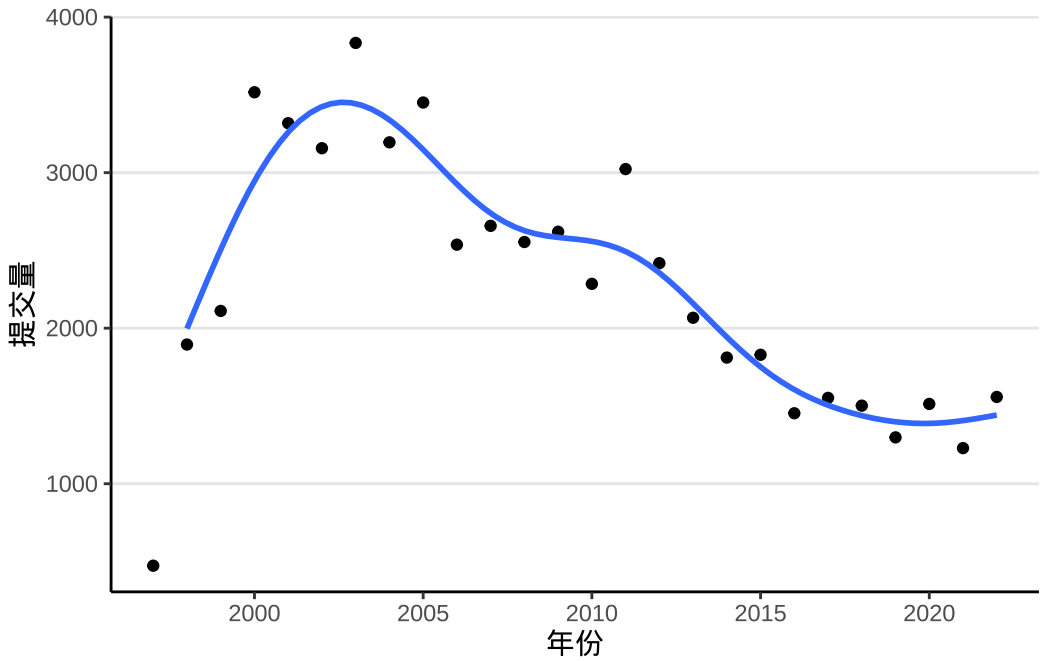


图 8.11: 过去 25 年代码提交次数的变化情况

自由度为 3 的正交多项式拟合

```
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")
```

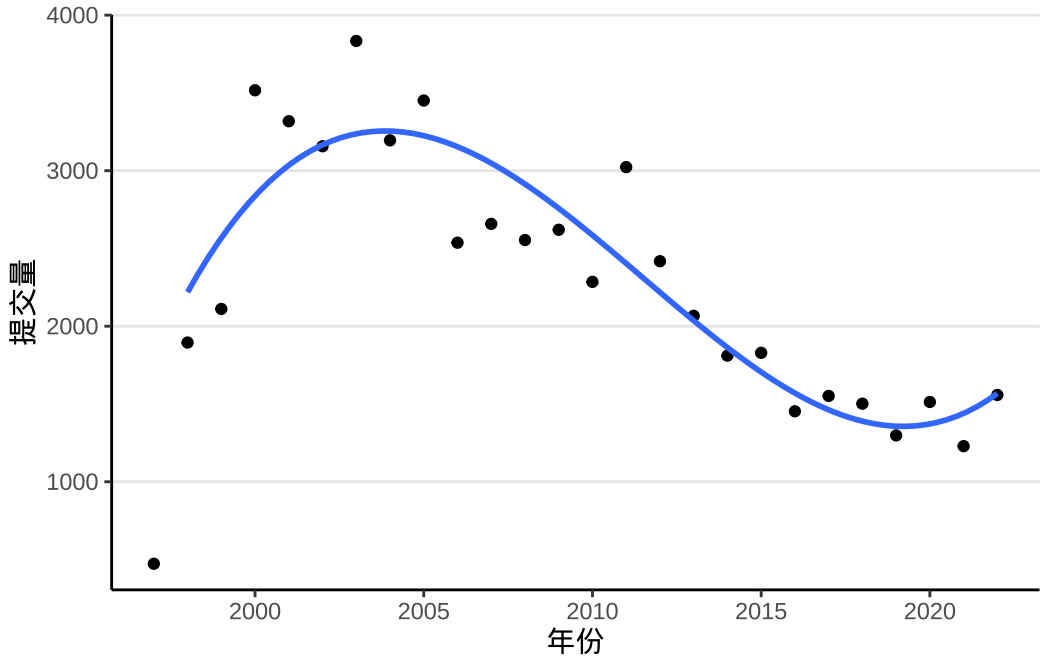


图 8.12: 过去 25 年代码提交次数的变化情况

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 8.1 ，更多详情见帮助文档 `?formula`。

表格 8.1: 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`
$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$	`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)`

自由度为 1 的正交多项式 `poly(x, 1)` 回归模型，函数 `glm()` 的参数 `offset` 的含义，`weights` 的含义广义可加模型 `gam()` 中回归样条的方法 `s()`

8.1.4 曲面图

ggplot2 暂不支持三维曲面图，而 **lattice** 包支持，也是非常有限的支持，**lattice** 和 **ggplot2** 都是基于图层的概念，层层叠加会出现覆盖。用来绘制函数图像还是可以的。

真三维图形可以用 rayrender 和 rayshader 包绘制。交互式三维图形可以用 rgl 或 plotly 包绘制。

图 8.13 是用 **lattice** 包的 `wireframe()` 函数绘制三维曲面透视图，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。

每天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 8.14 所示。

8.1.5 热力图

图 8.15 提交量变化趋势

```
ggplot(data = trunk_year_day, aes(x = as.integer(nday) , 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() +
  labs(x = "第几天", y = "年份", fill = "提交量")
```

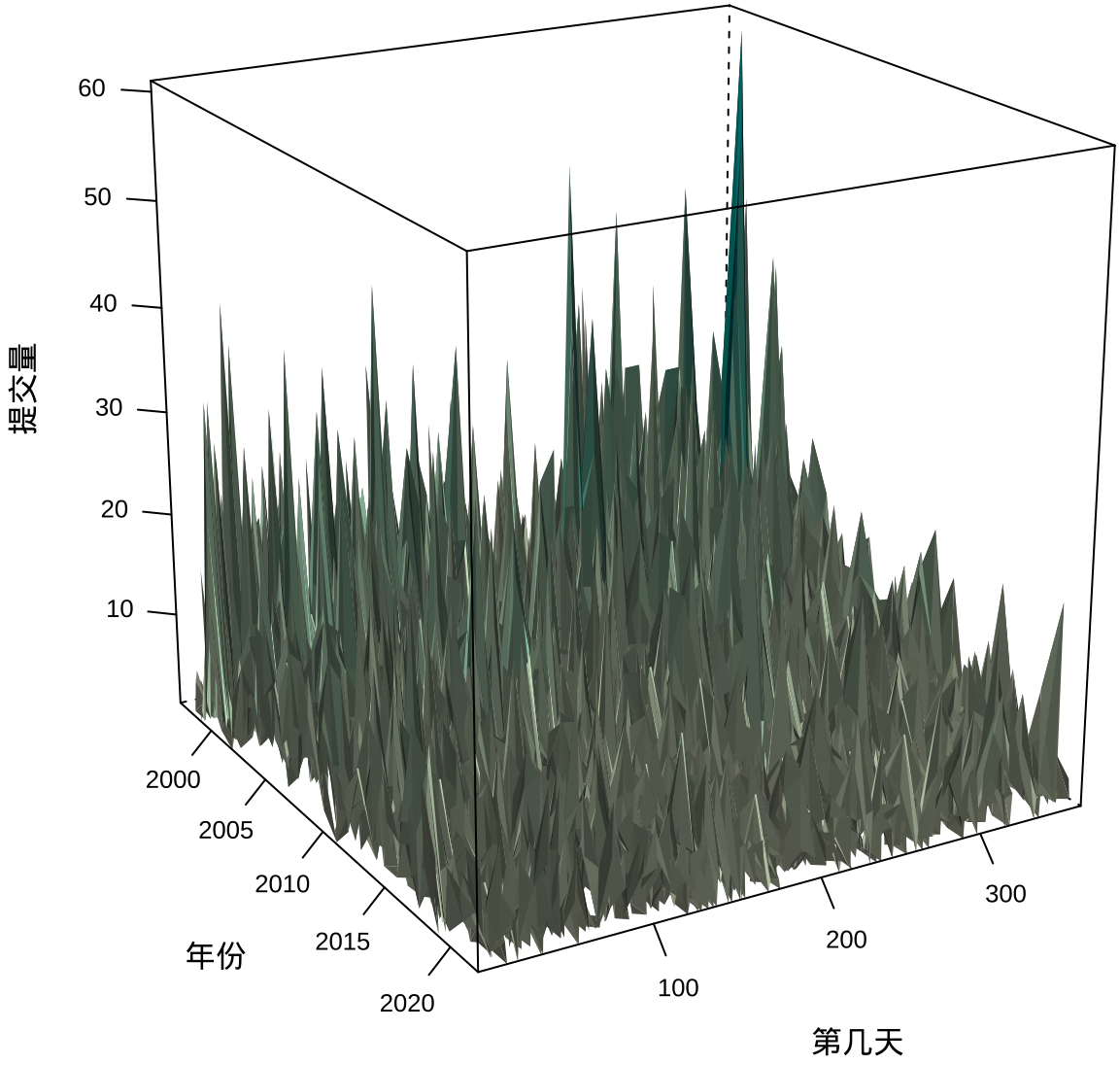


图 8.13: 25 年代码提交量变化趋势图

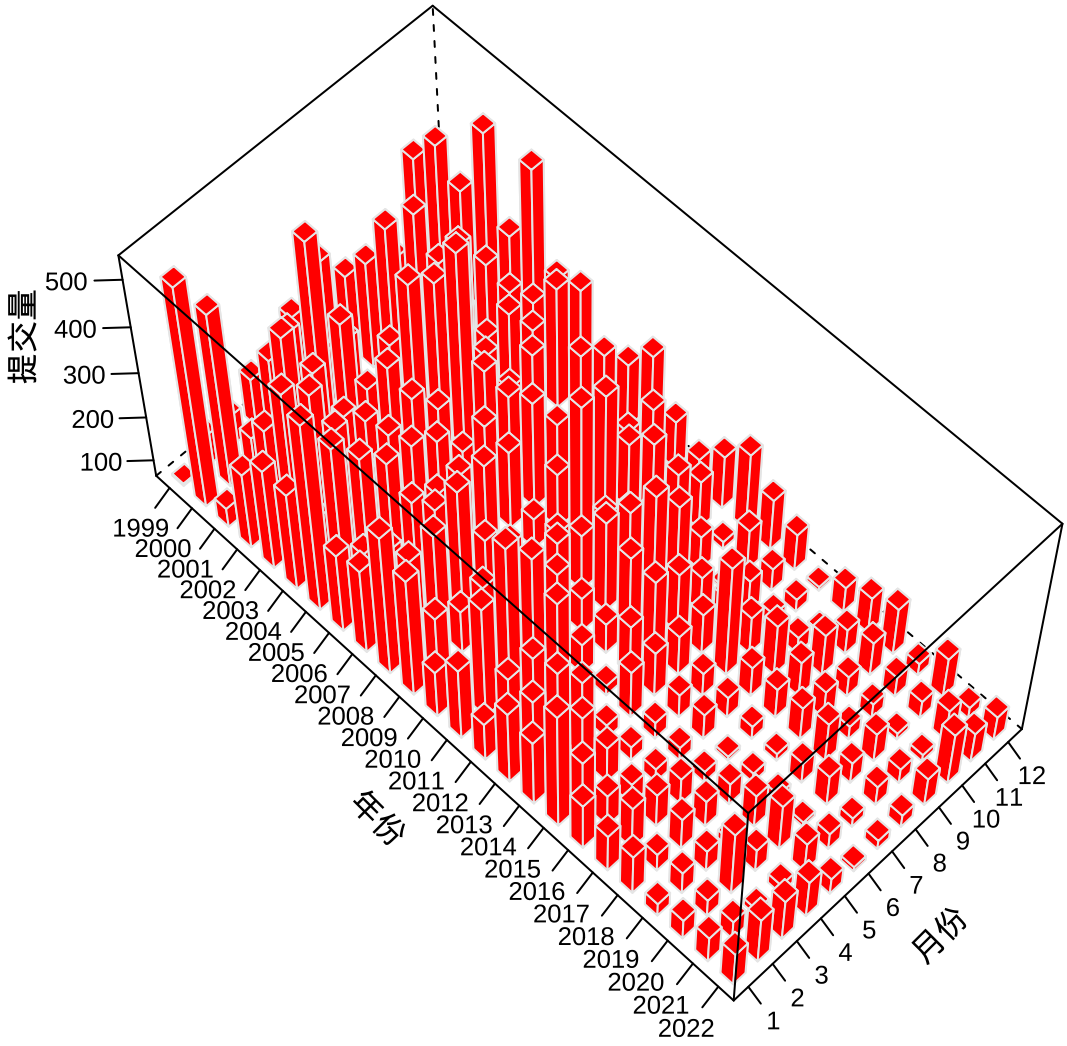


图 8.14: 25 年代码提交量变化趋势图

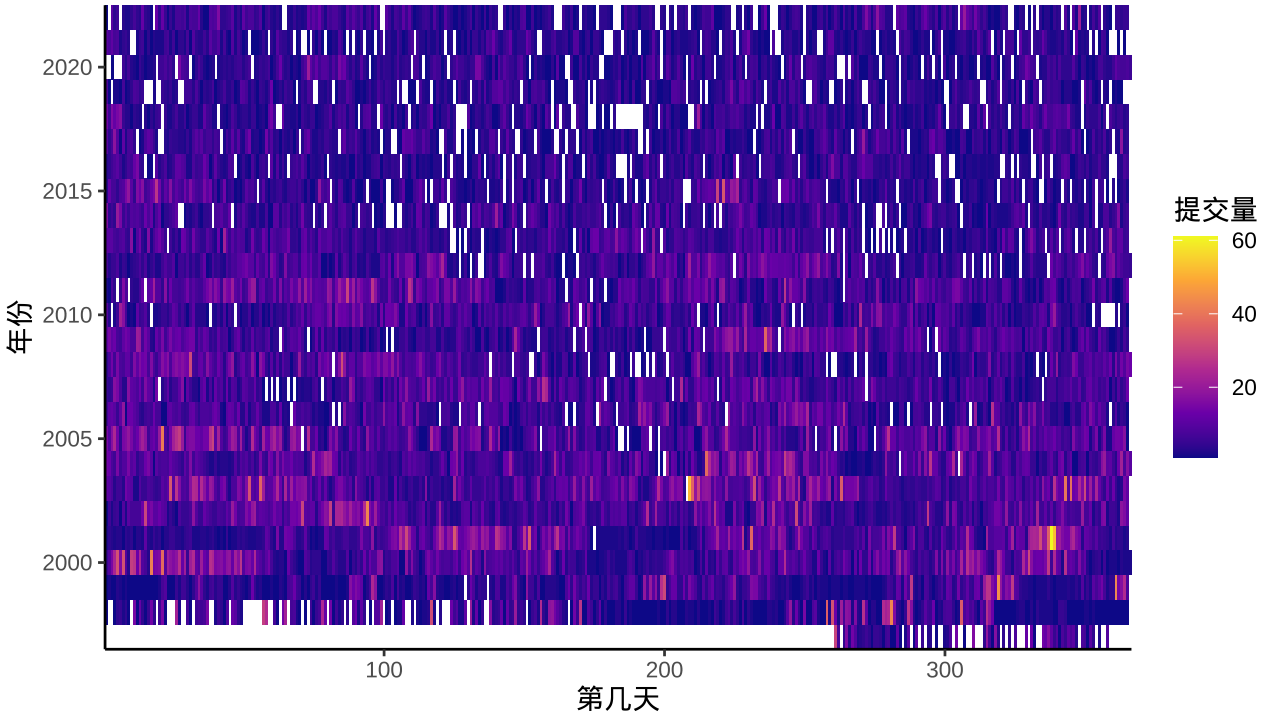


图 8.15: 25 年代码提交量变化热力图

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")
```

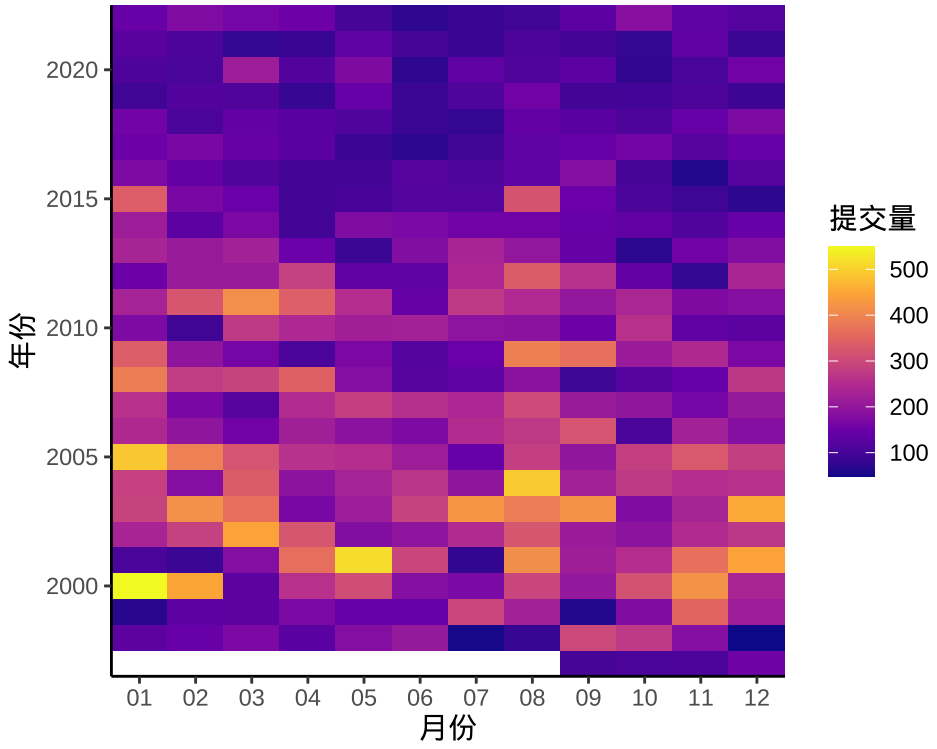


图 8.16: 25 年代码提交量变化热力图

8.1.6 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)
   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")
```

图 8.17: 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

8.1.7 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 8.18

```
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")
```

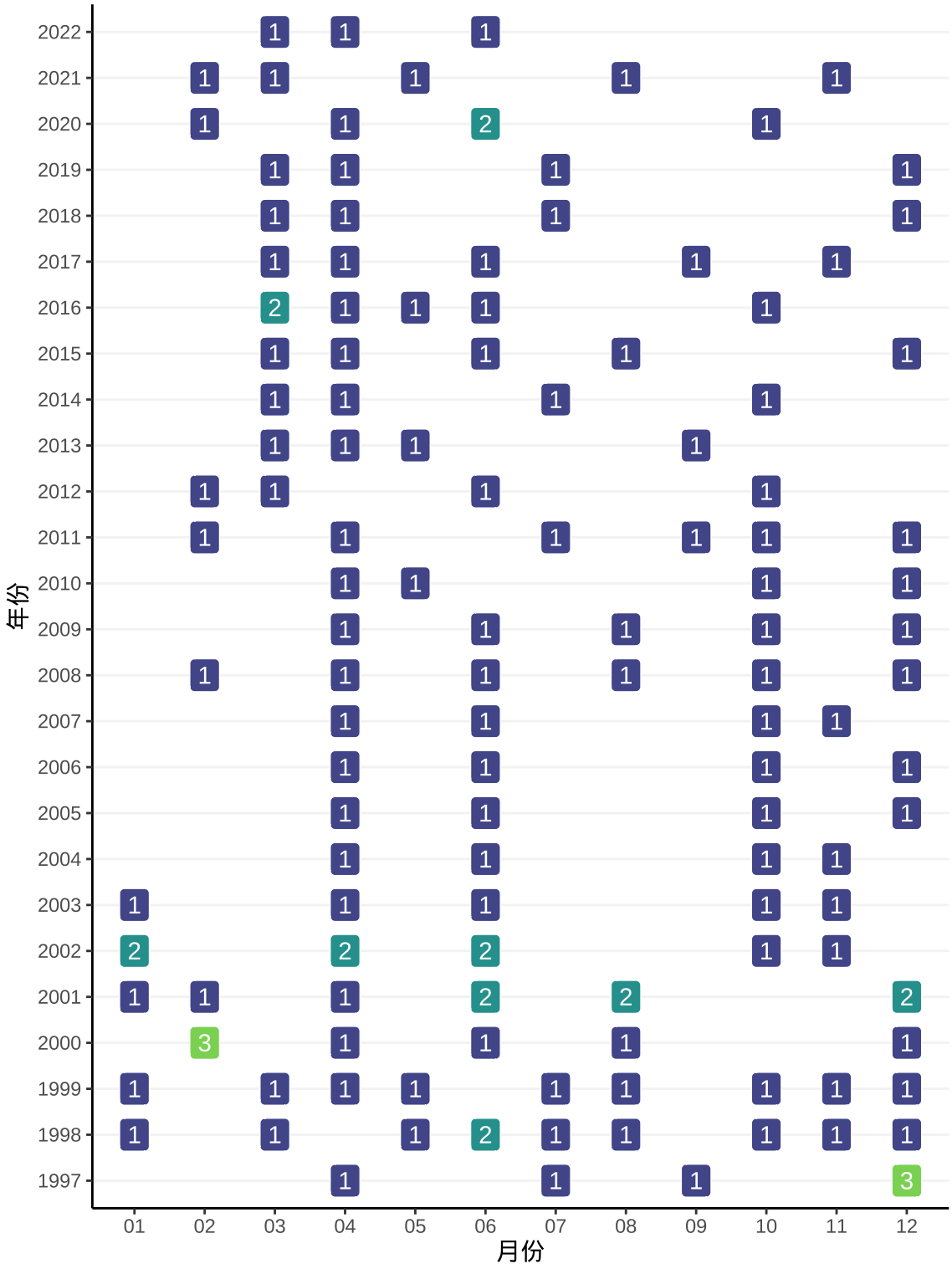


图 8.18: 25 年 R 软件发版情况

8.1.8 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 8.2

表格 8.2: 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

```
# 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 8.19 展示 2020-2022 年 R 软件发版情况

```
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()
```

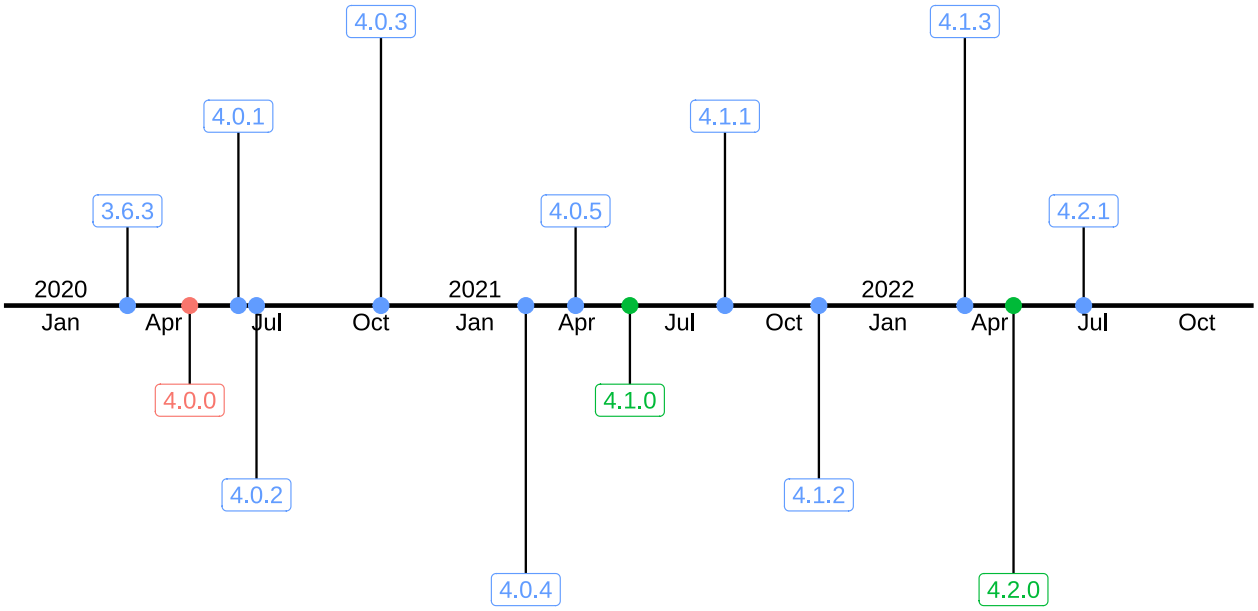


图 8.19: 2020-2022 年 R 软件发版情况

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()
```

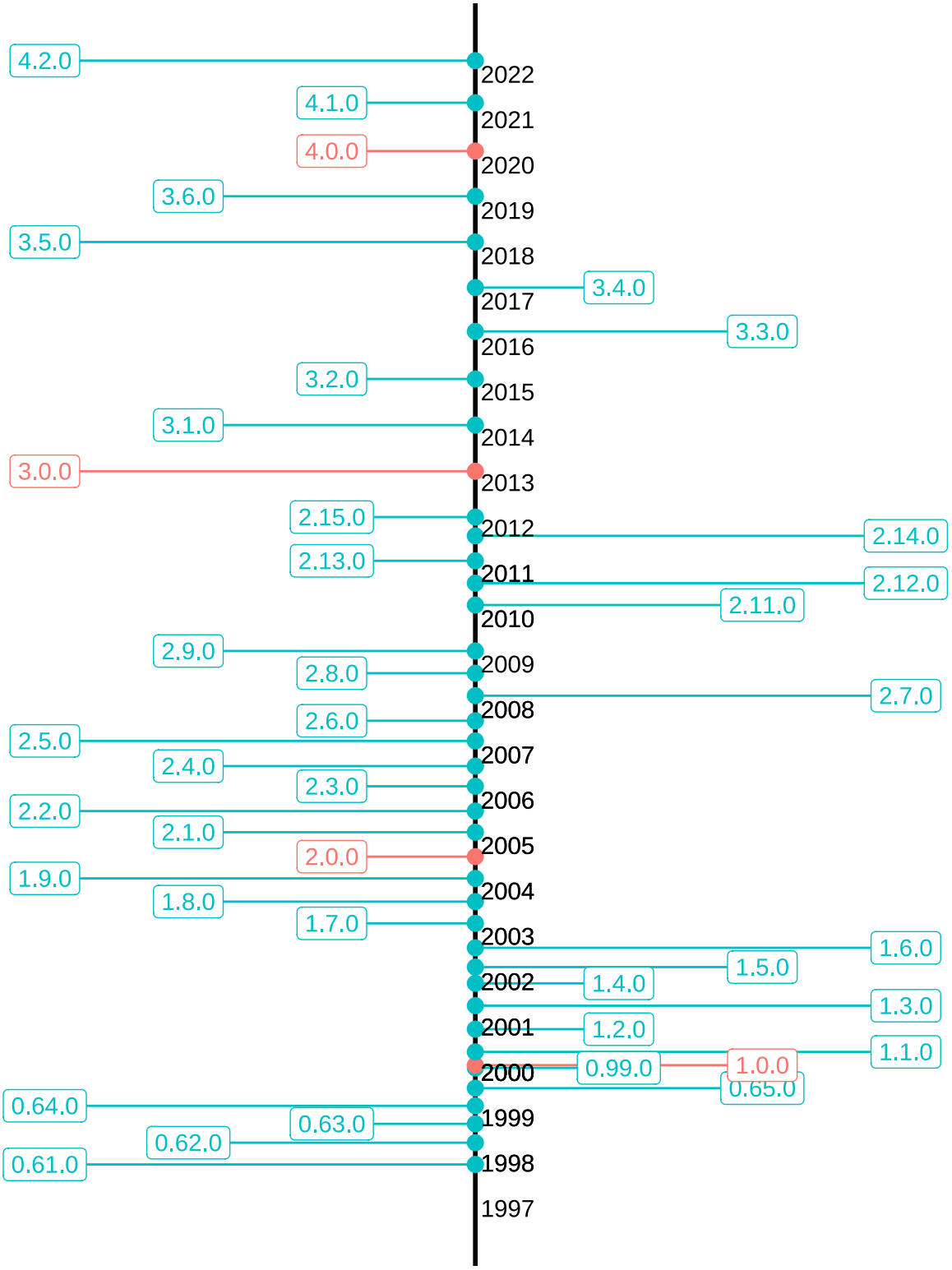


图 8.20: 25 年里 R 软件重大及主要版本发布情况

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。

8.2 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 8.3: 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？

8.2.1 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()
```

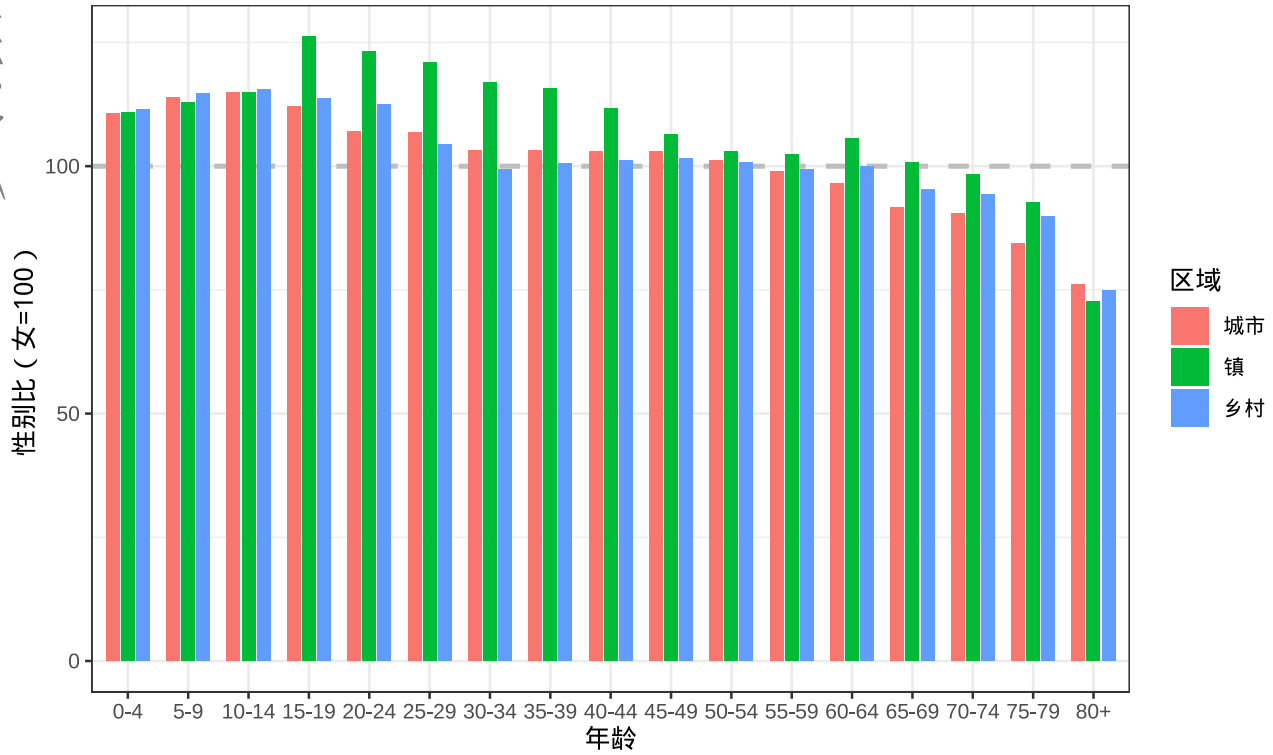


图 8.21: 分年龄段比较城市、镇和乡村的性别比数据

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()
```

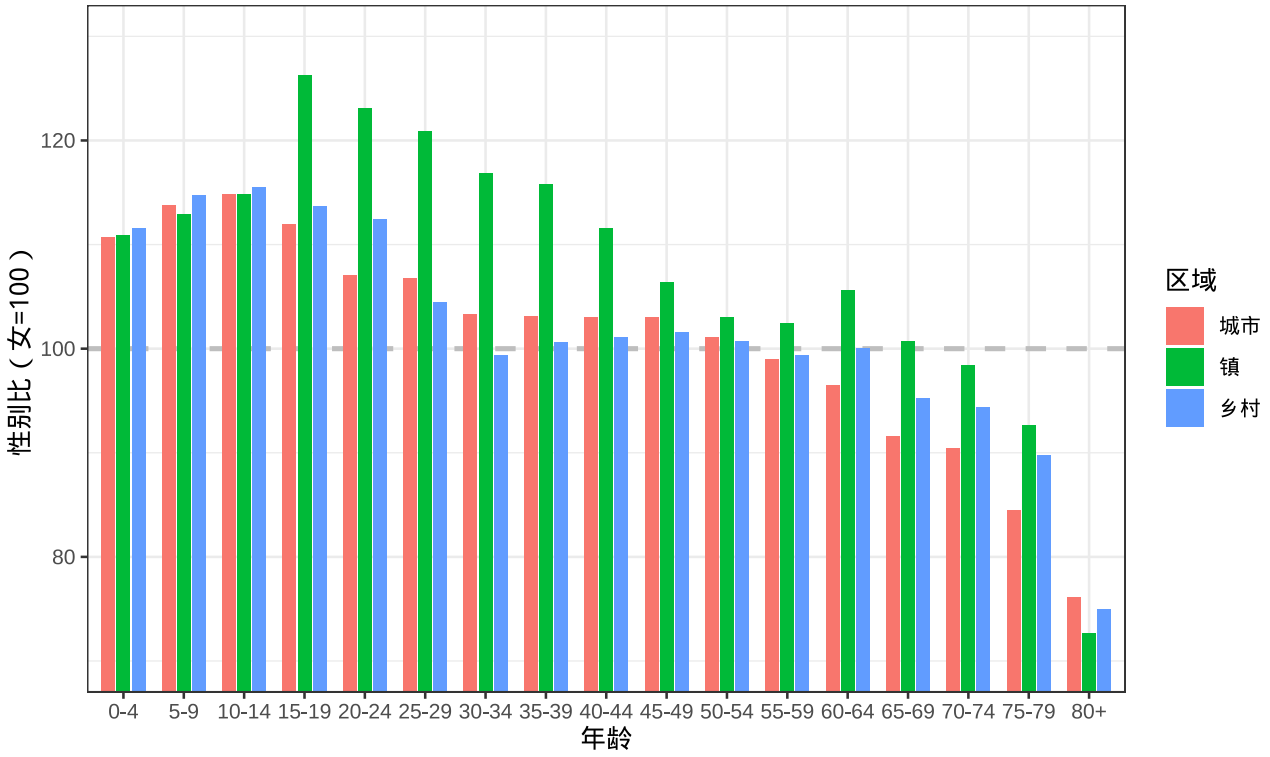


图 8.22: 分年龄段比较城市、镇和乡村的性别比数据

8.2.2 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()
```

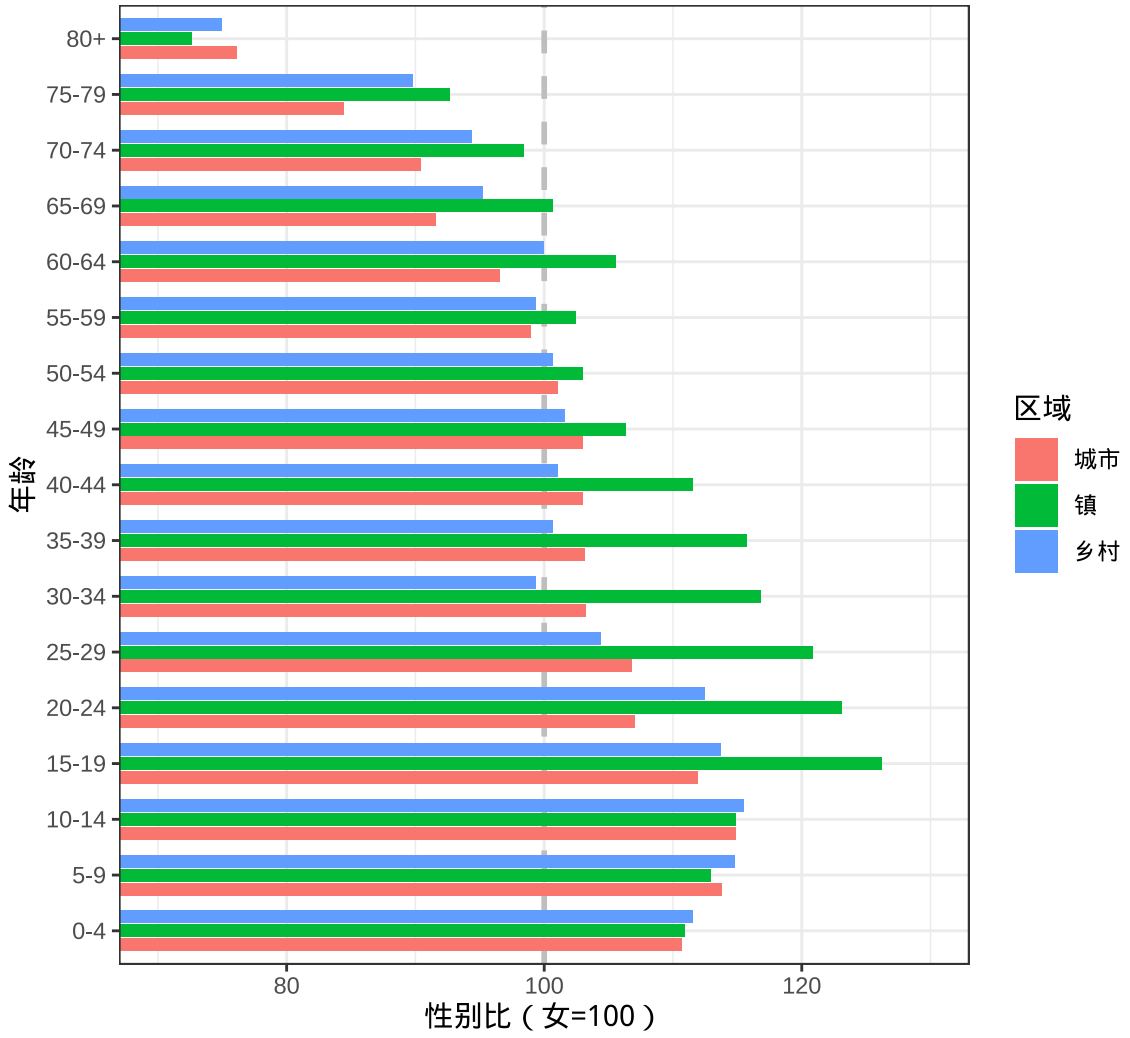


图 8.23: 分年龄段比较城市、镇和乡村的性别比数据

8.2.3 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()
```

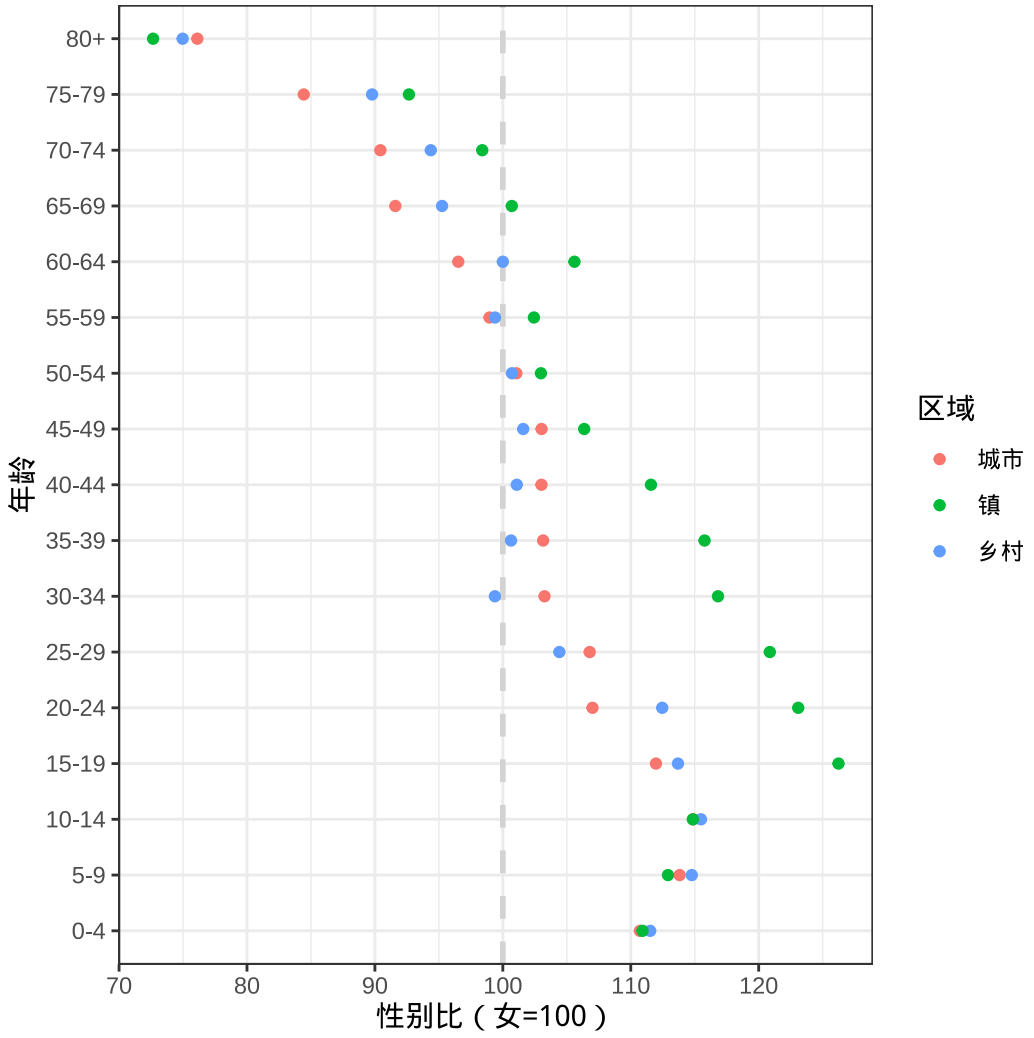


图 8.24: 分年龄段比较城市、镇和乡村的性别比数据

8.2.4 词云图

词云图帮助我们从众多的因素中展现出影响力比较大的主题。目前，R 语言社区官方发布的 R 包超过 18000 个，这些 R 包都是干什么的呢？热门的方向是什么呢？根据 R 包的标题内容分词，统计词频就可以帮助我们初步了解一些信息。根据各位开发者提交的代码量制作。

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 = 1, max_grid_size = 24) +
```

```
  scale_size_area(max_size = 20) +
  theme_minimal()
```

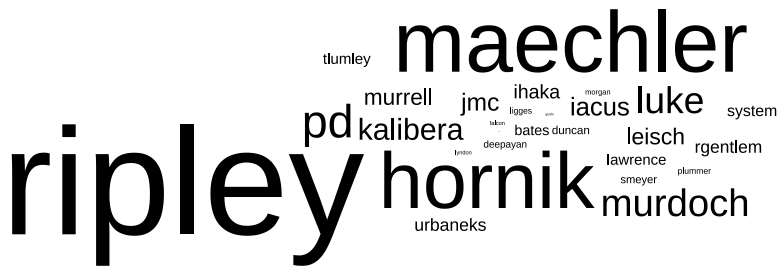


图 8.25: 词云图

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")
```

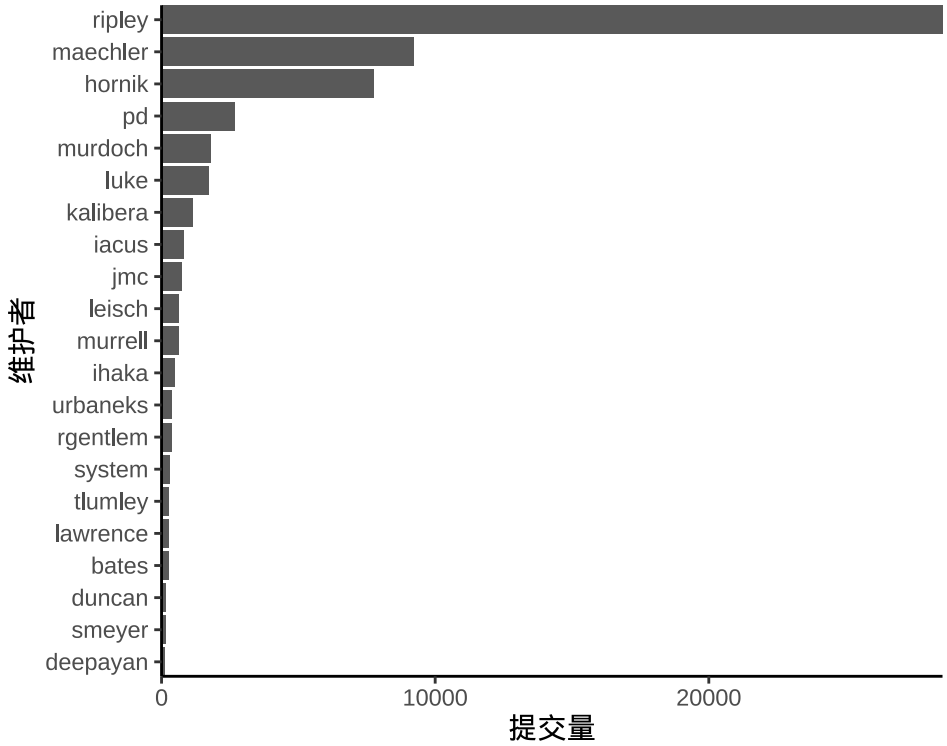


图 8.26: 开发者提交量排行榜

8.3 描述占比

8.3.1 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 8.27 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"
```

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

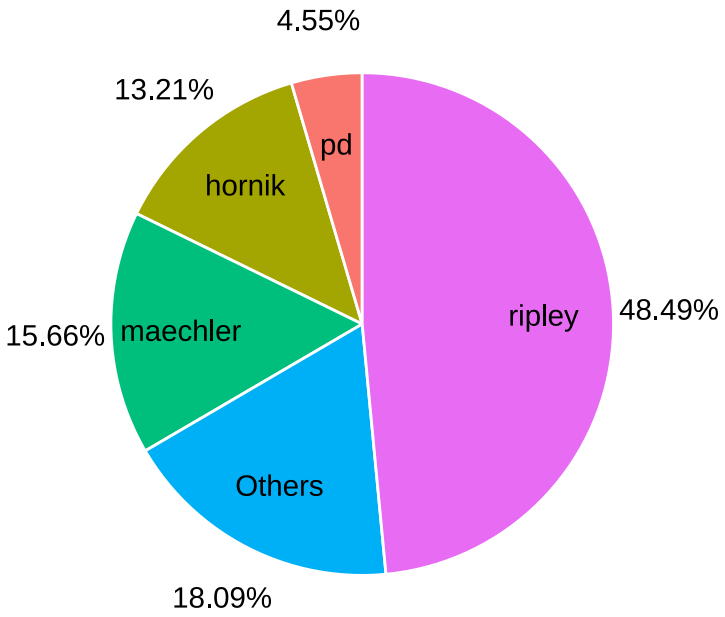


图 8.27: 维护者提交量占比

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)
```

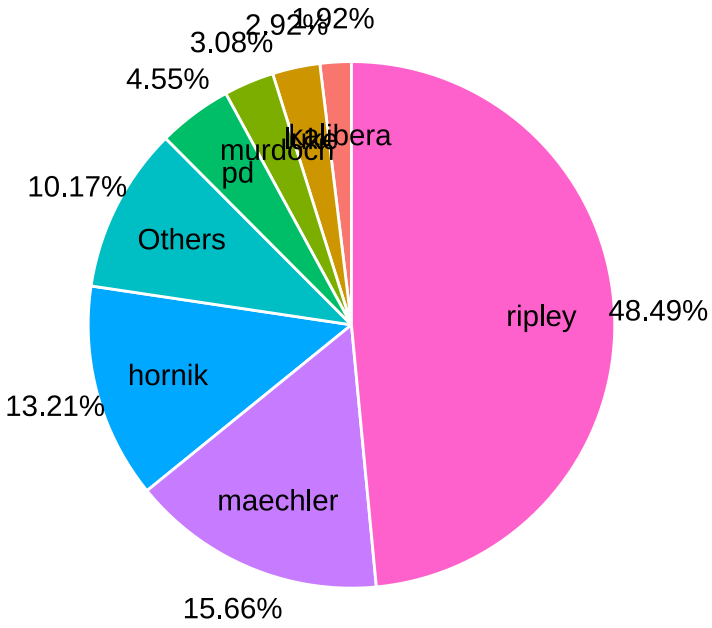


图 8.28: 维护者提交量占比

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)
```

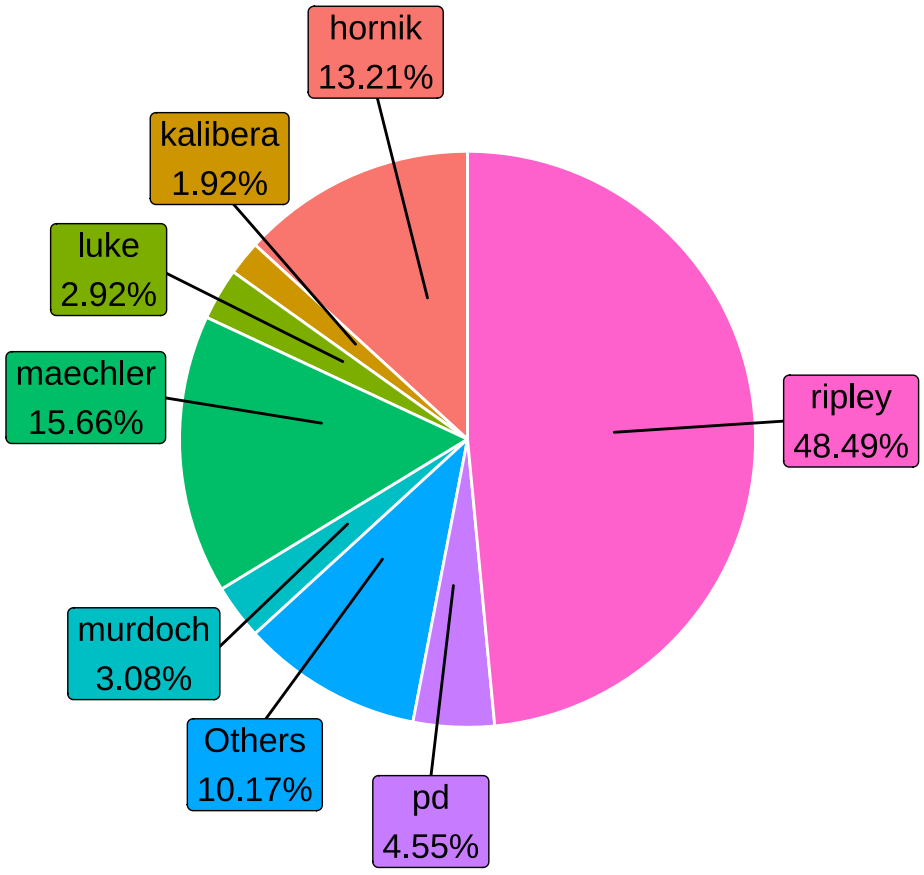


图 8.29: 维护者提交量占比

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 8.4: 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

SVN 花名	真实名字	主要贡献
bates	Douglas Bates	nlme / lme4 包核心开发者
pd	Peter Dalgaard	《统计导论与 R 语言》作者
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

8.3.2 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))
```

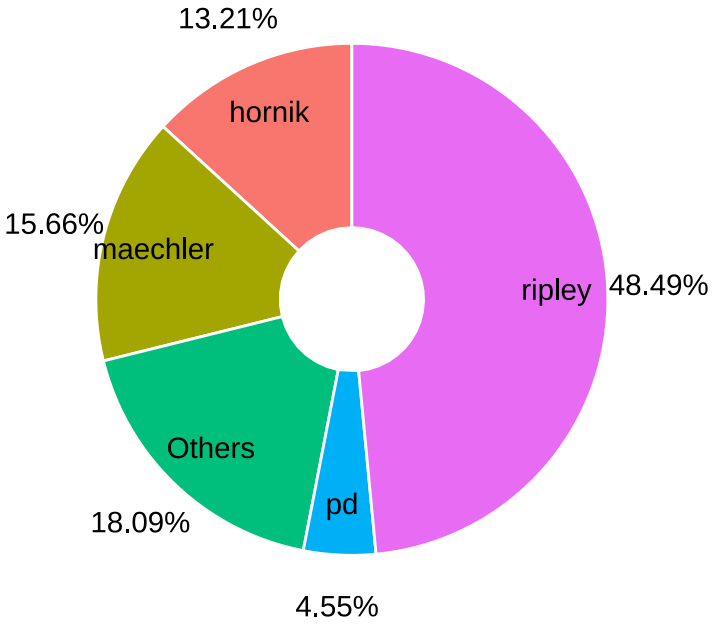


图 8.30: 维护者提交量占比

8.3.3 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)
```

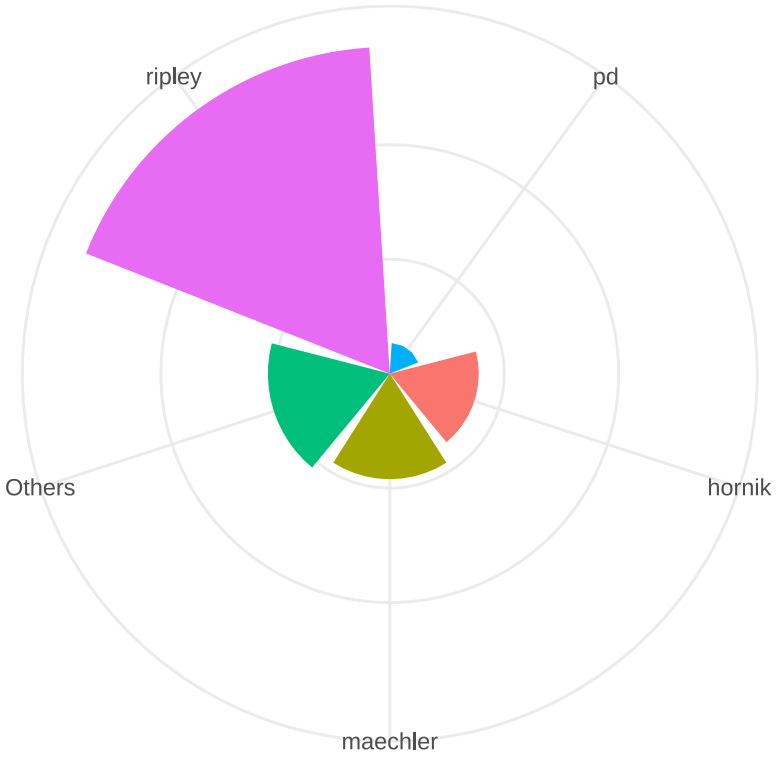


图 8.31: 维护者提交量分布

8.3.4 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 8.32 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) +
  labs(x = "年份", y = "提交量", fill = "开发者")
```

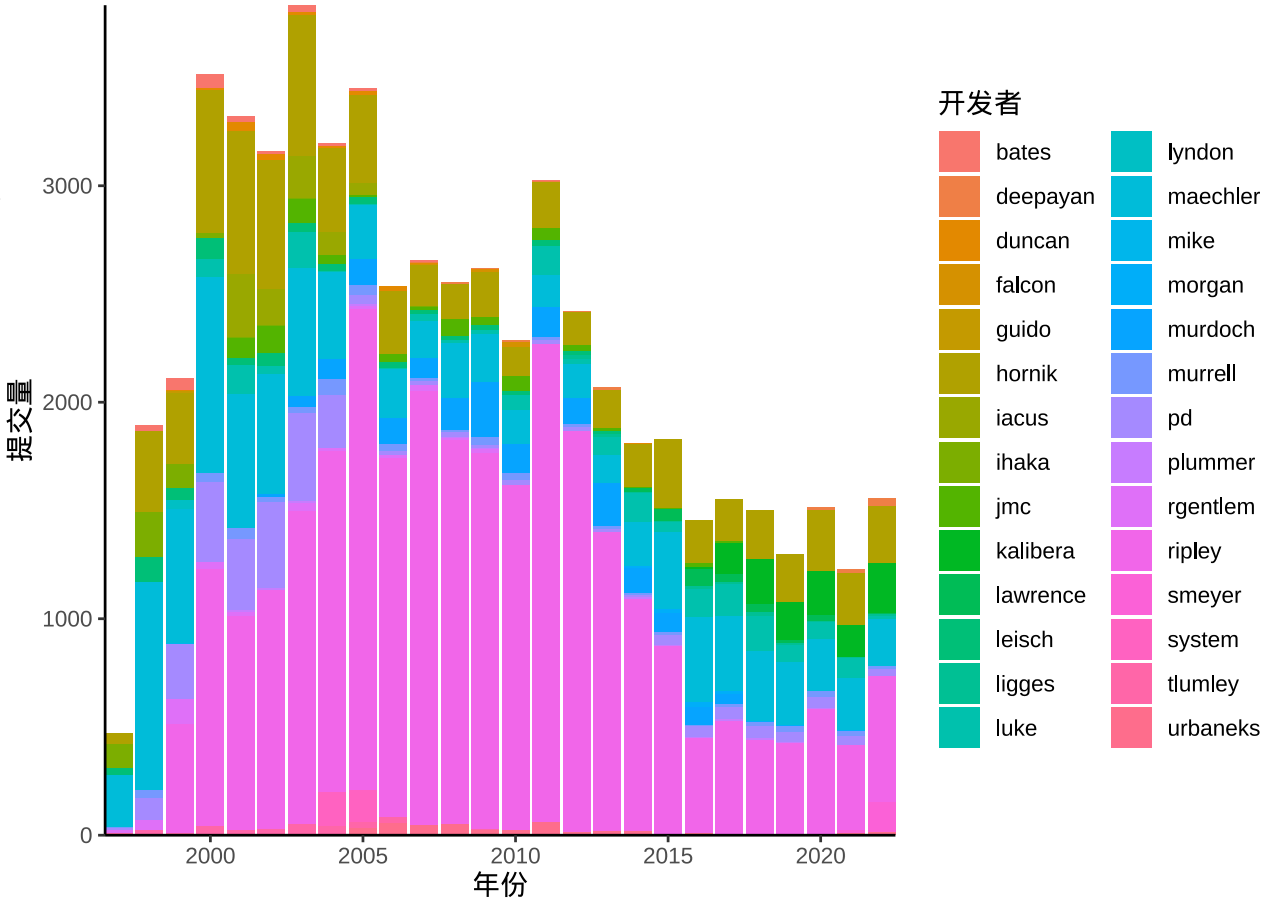


图 8.32: 代码提交量的比例趋势

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 8.33 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) +
  labs(x = "年份", y = "提交量", fill = "开发者")
```

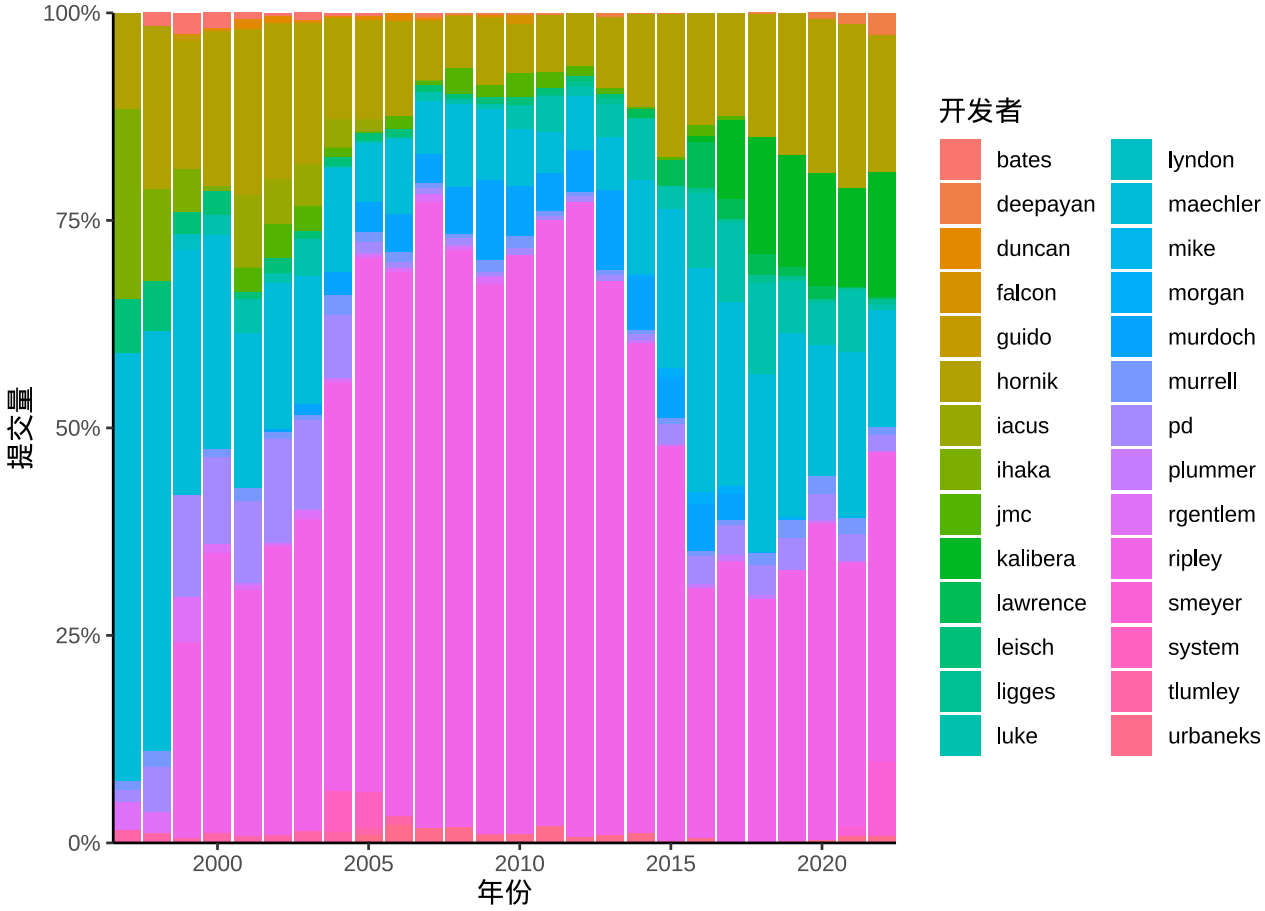


图 8.33: 代码提交量的比例趋势

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 8.34 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() +
  labs(x = "维护者", y = "累计提交量")
```

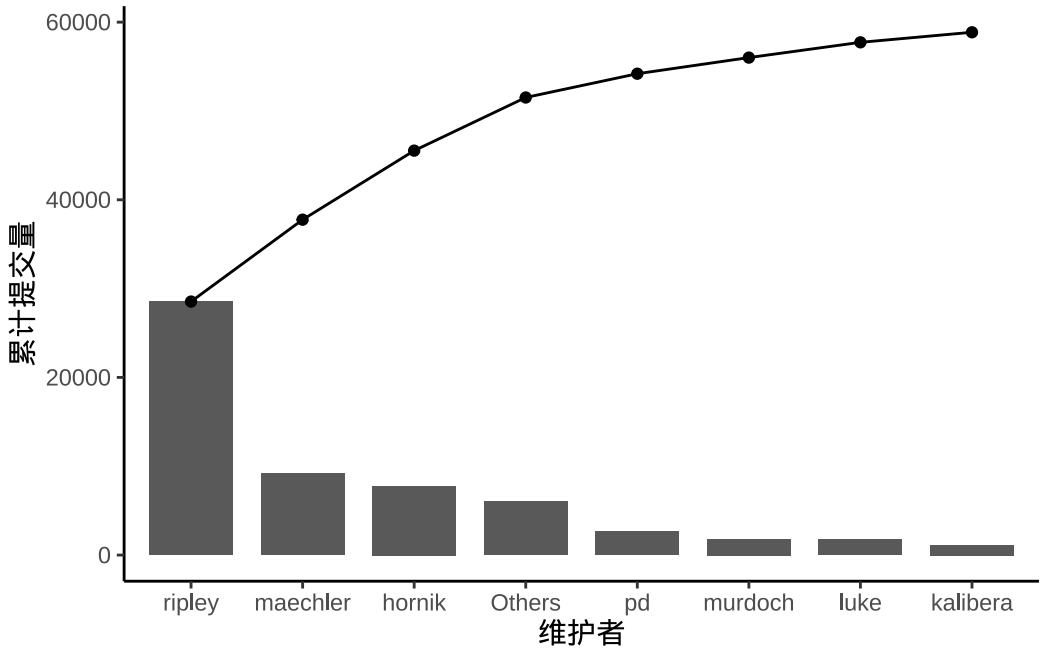


图 8.34: 代码提交量的比例分布

8.3.5 马赛克图

马赛克图常用于展示多个分类数据，如图 8.35 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()
```

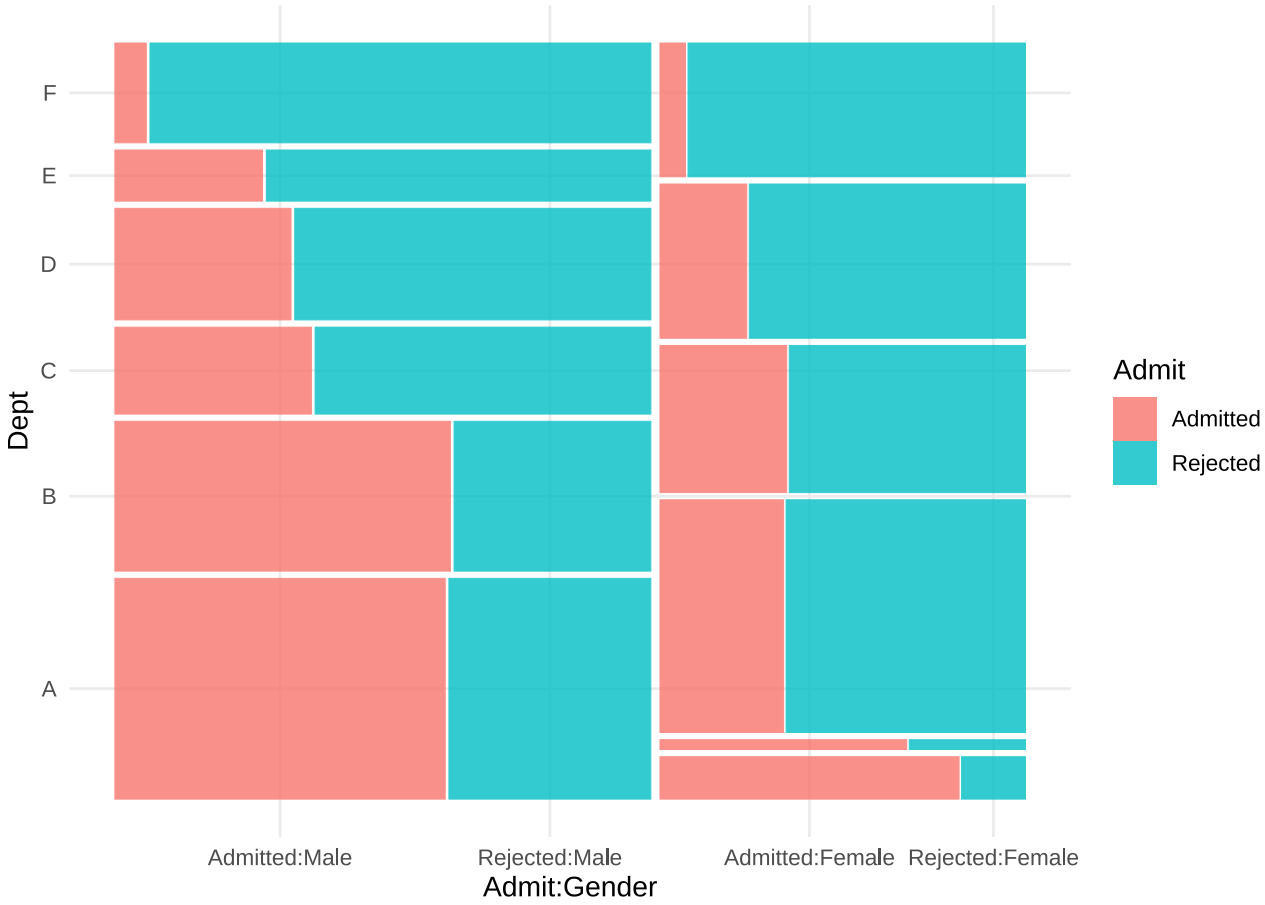


图 8.35: 加州伯克利分校院系录取情况

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。

8.3.6 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 8.36 展示南北半球，各地域内各个国家 GDP 的占比。

表格 8.5: 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")
```

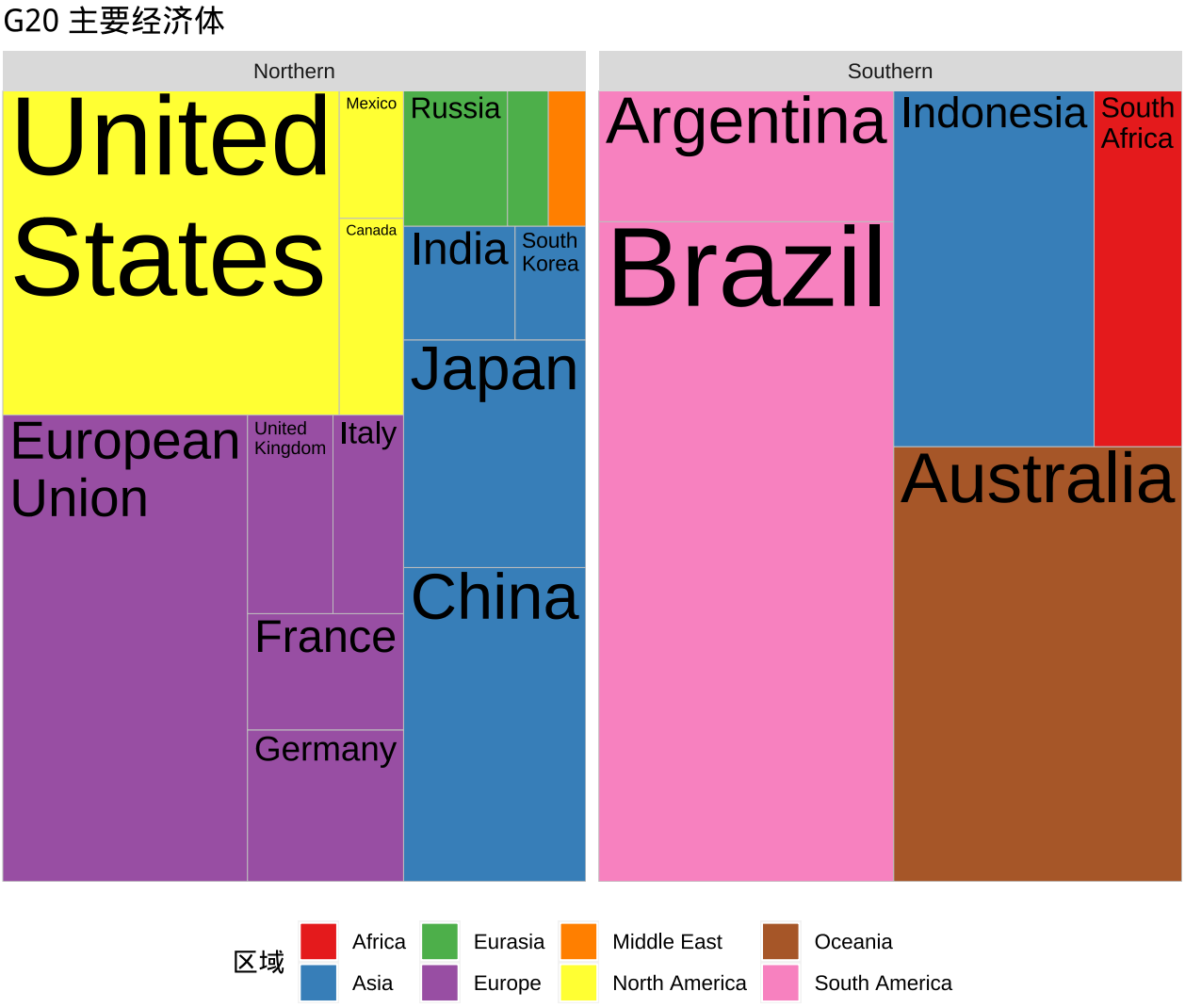


图 8.36: G20 主要经济体的 GDP 占比

8.3.7 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 8.6 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。

表格 8.6: 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 8.7 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 8.7: 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
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 8.37 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")
```

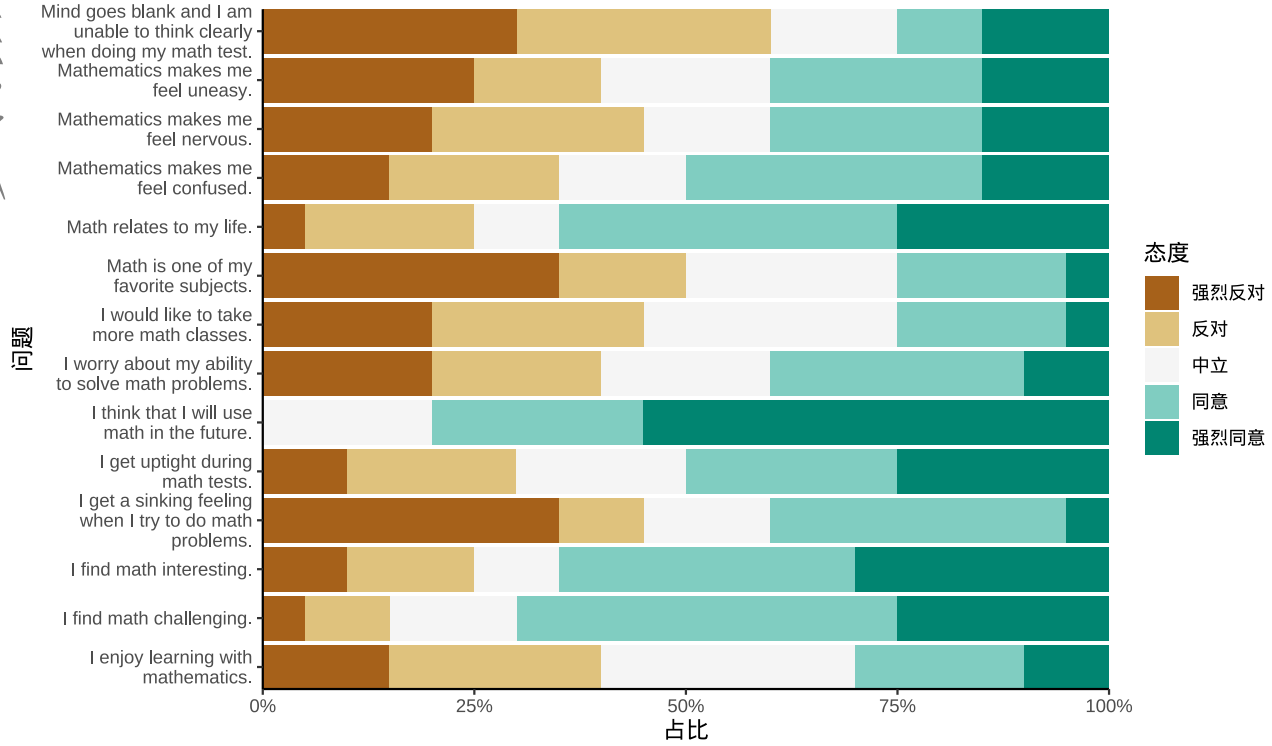


图 8.37: 你喜欢数学吗

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)
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)
```

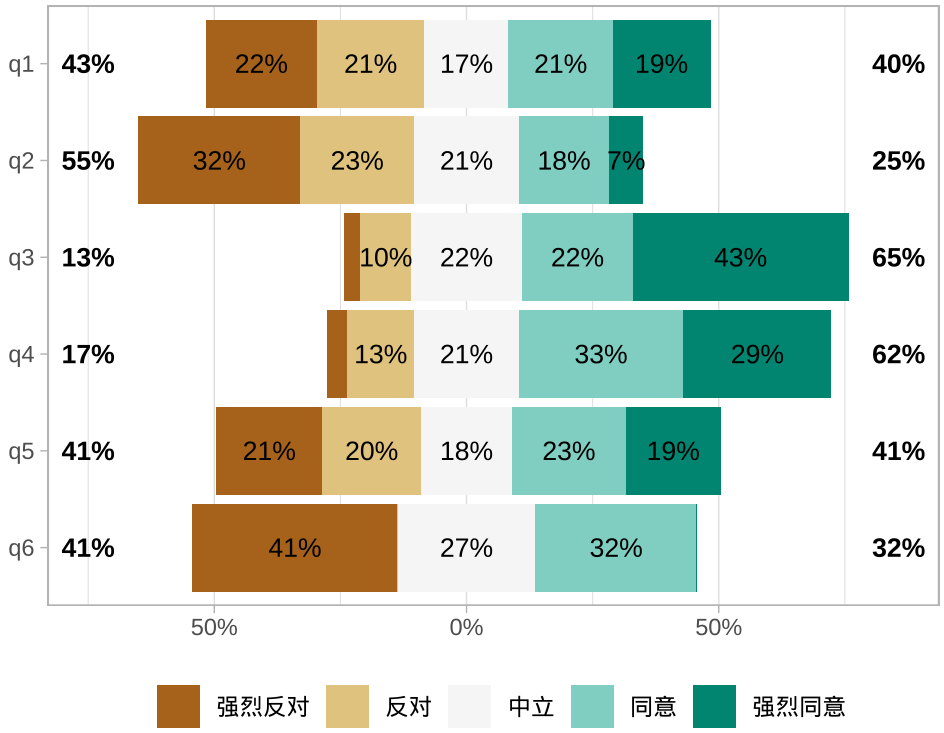


图 8.38: Likert 图

第九章　统计图形

- 章节 9.1 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 9.2 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 9.3 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。

9.1　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 9.1 。

表格 9.1: 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

9.1.1　箱线图

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

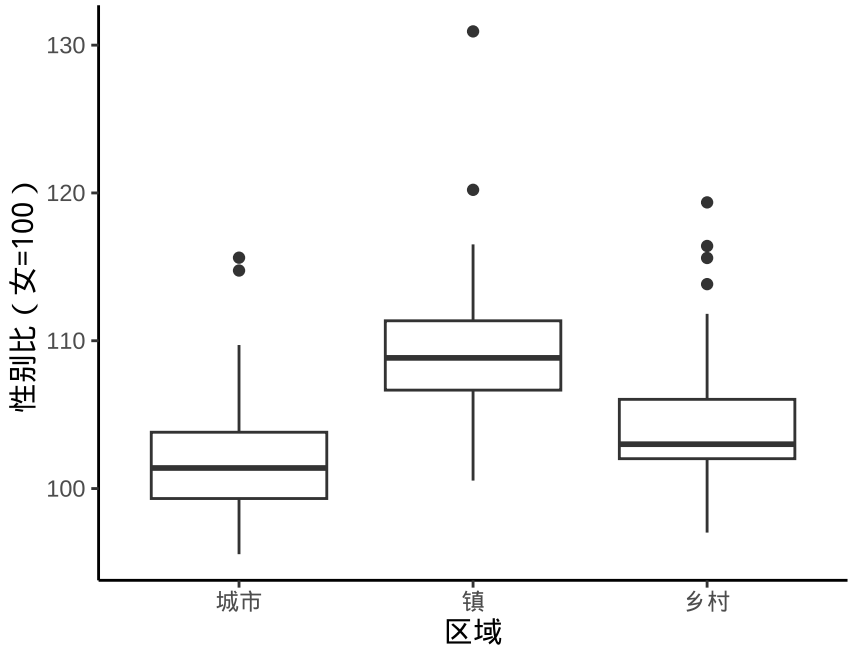


图 9.1: 箱线图

图 9.2 是箱线图的变体 (McGill 和 Larsen 1978)

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

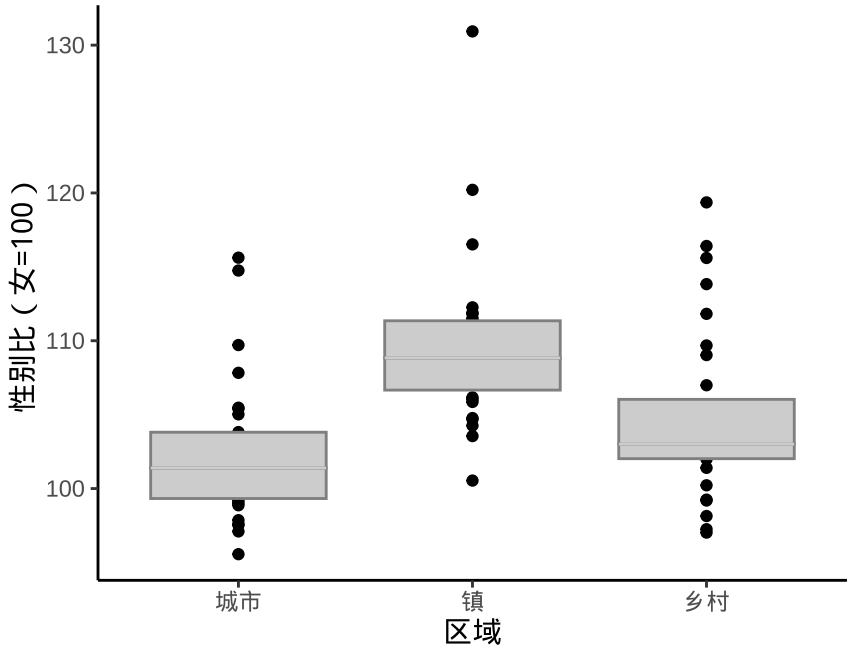


图 9.2: 箱线图的变体

箱线图的历史有 50 多年了，它的变体也有很多。

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

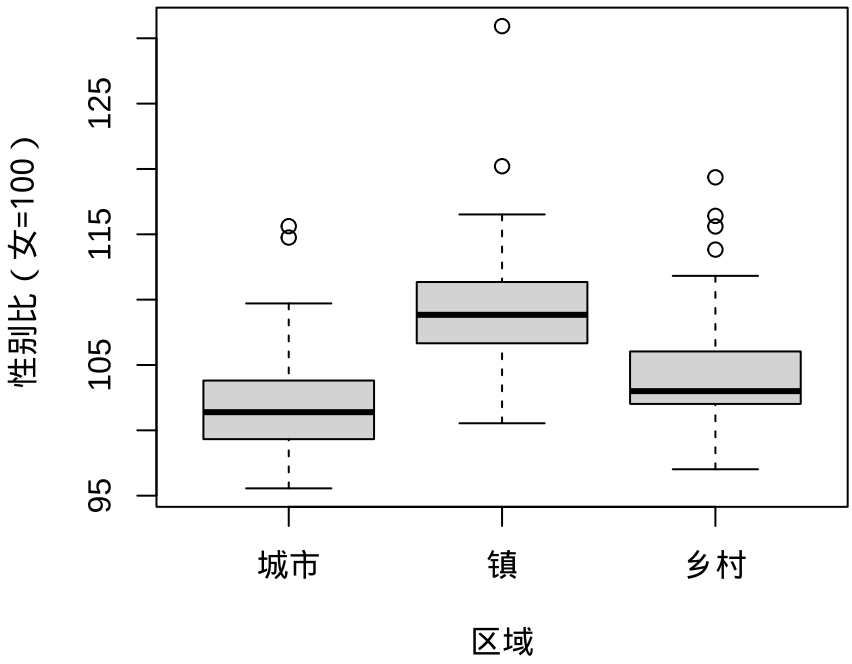


图 9.3: 箱线图

9.1.2 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()
```

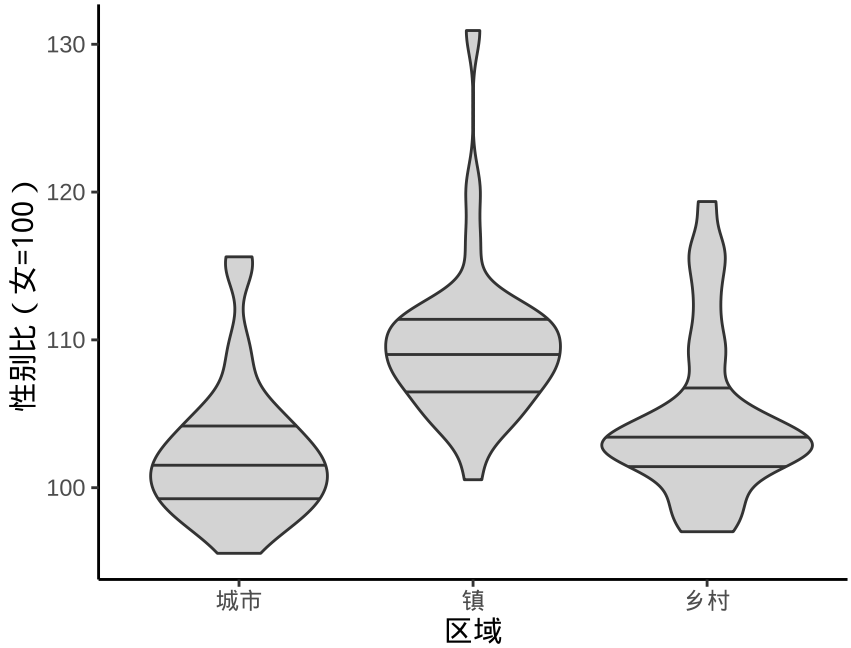


图 9.4: 提琴图

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

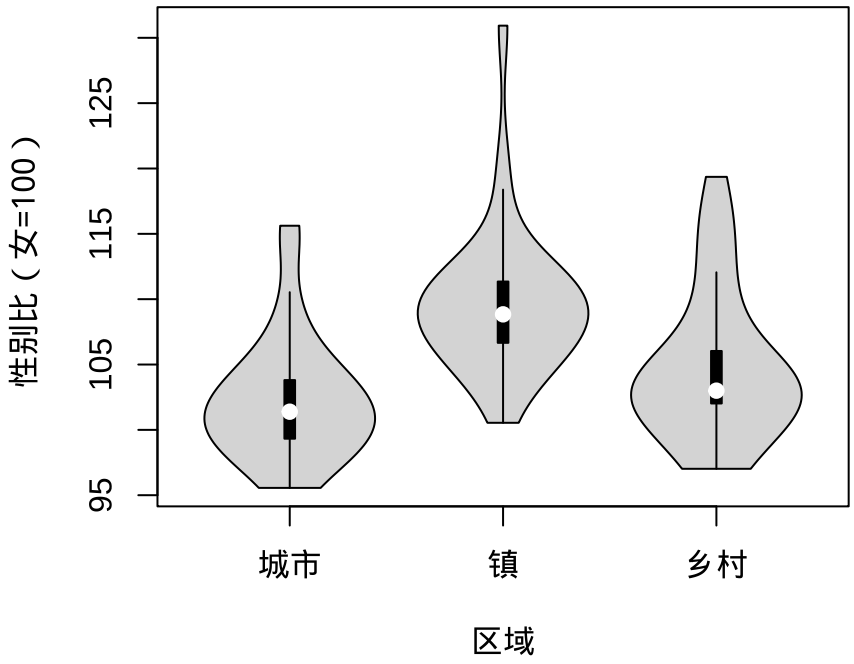


图 9.5: 提琴图

beanplot 的名字是根据图形的外观取的，豌豆，rug 用须线表示数据

```
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）"
)
```

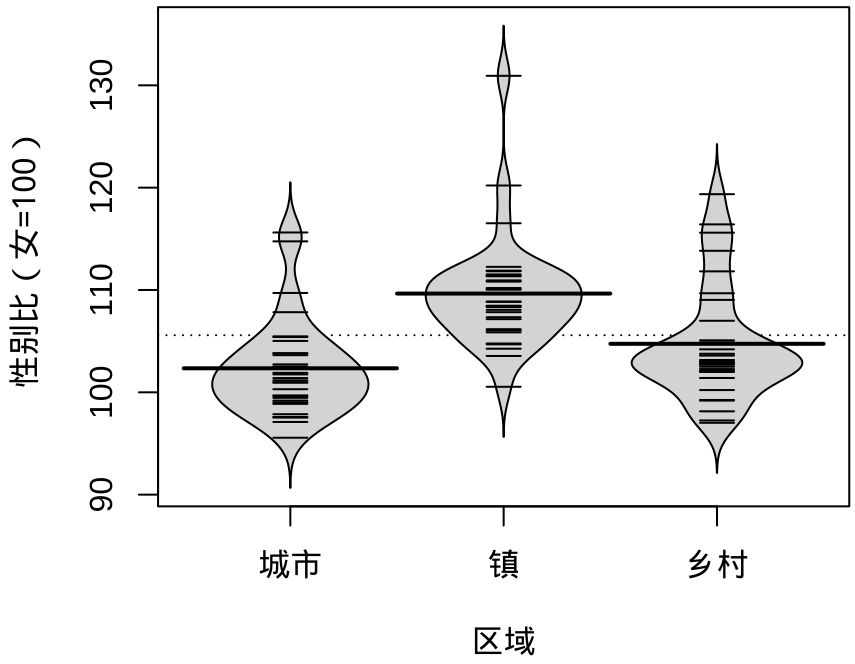


图 9.6: 提琴图

9.1.3 直方图

```
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()
```

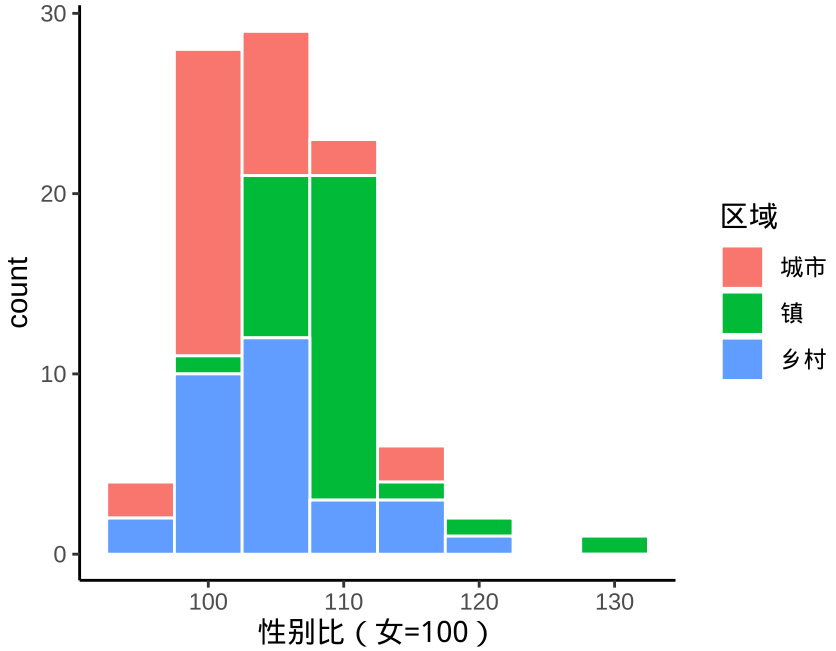


图 9.7: 直方图

```
ggplot(data = province_sex_ratio, aes(x = `性别比（女=100）`, color = `区域`)) +
  geom_freqpoly(binwidth = 5, stat = "bin") +
  theme_classic()
```

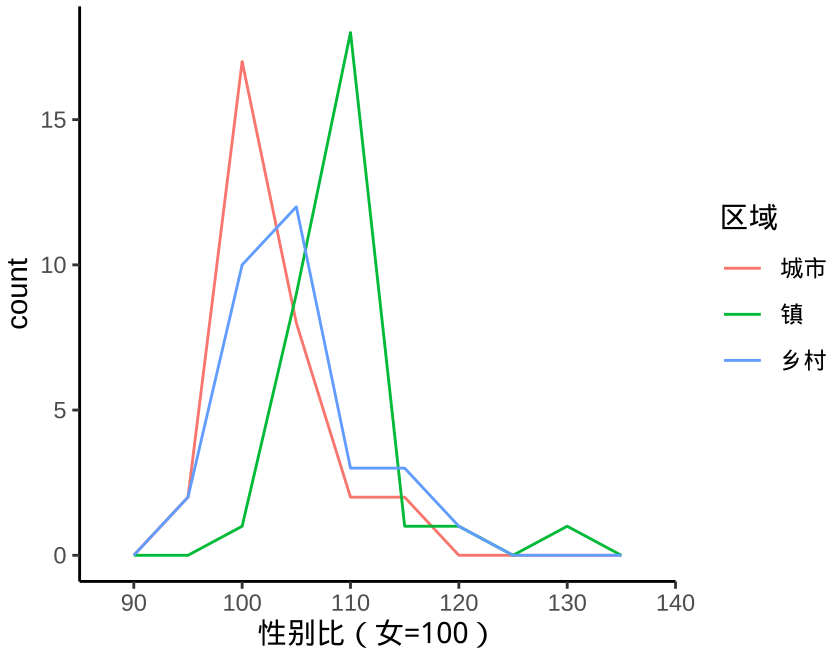


图 9.8: 直方图

9.1.4 密度图

图 9.9 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()
```

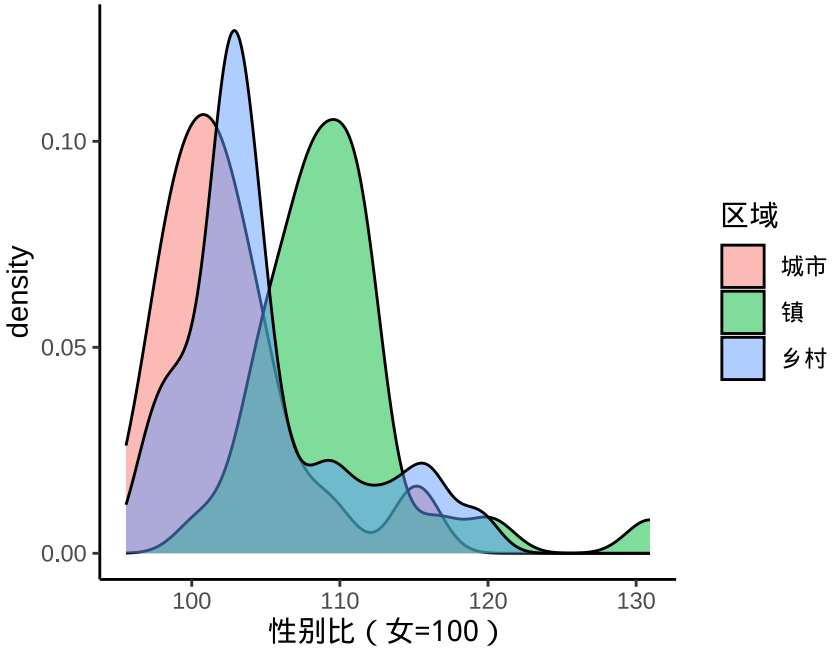


图 9.9: 密度图

图 9.10 丢失边际密度

```
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()
```

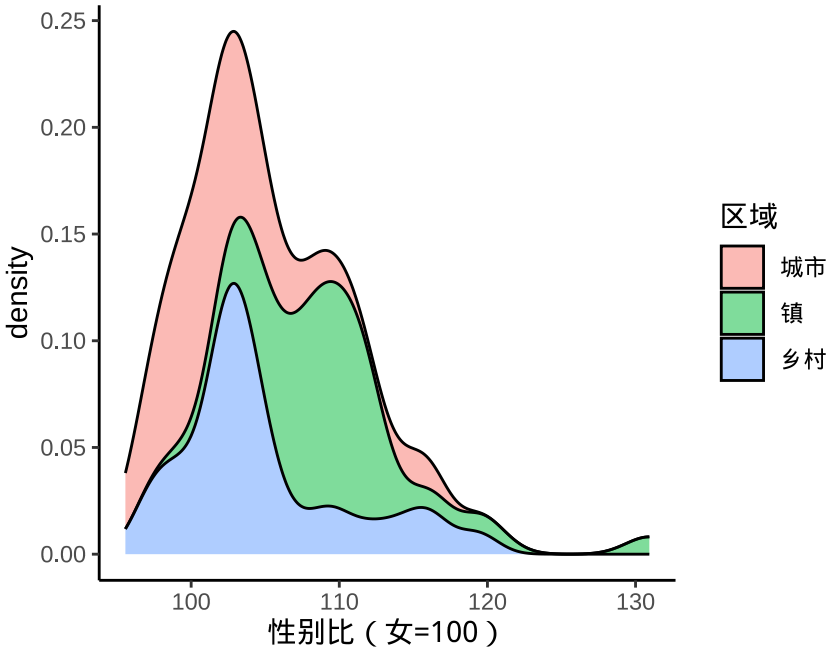


图 9.10: 堆积密度图

> **警告**
>
> 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

图 9.11 保护边际密度

```
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()
```

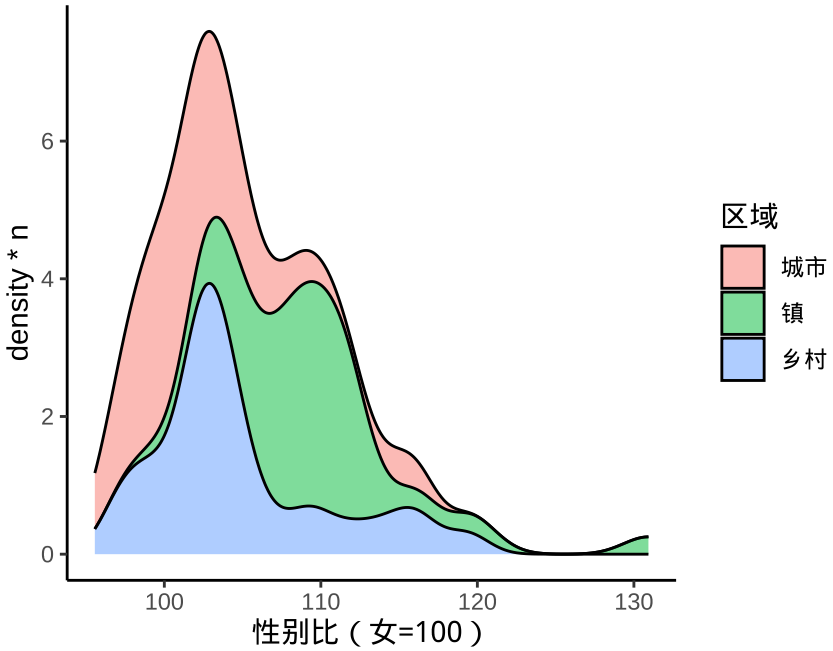


图 9.11: 累积分布密度图

什么原因导致图 9.11 和图 9.10 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，条件密度曲线图 9.11

```
p1 <- ggplot(diamonds, aes(x = carat, y = after_stat(density), fill = cut)) +
  geom_density(position = "stack") +
  theme_classic()

p2 <- ggplot(diamonds, aes(x = carat, y = after_stat(density * n), fill = cut)) +
  geom_density(position = "stack") +
  theme_classic()

p3 <- ggplot(diamonds, aes(x = carat, y = after_stat(count), fill = cut)) +
  geom_density(position = "stack") +
  theme_classic()

library(patchwork)
p1 / p2 / p3
```

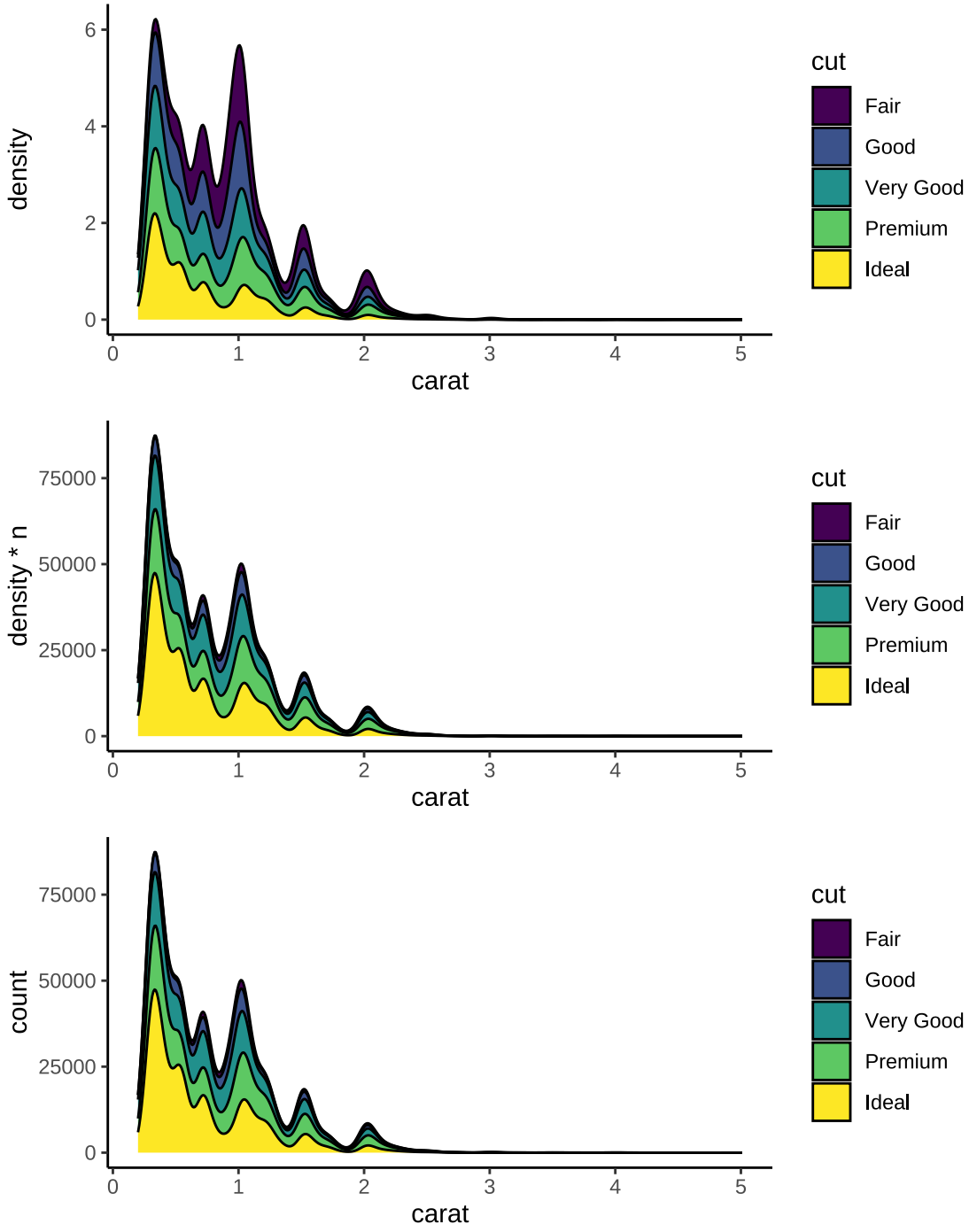


图 9.12: 堆积密度图

联合密度曲线

```
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"
  ) +
  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
```

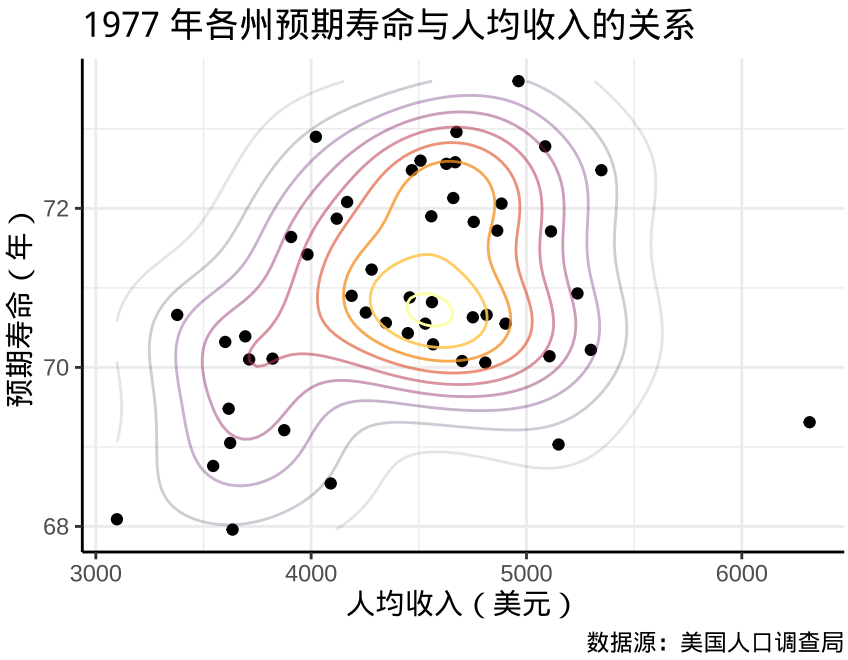


图 9.13: 二维密度图

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")
```

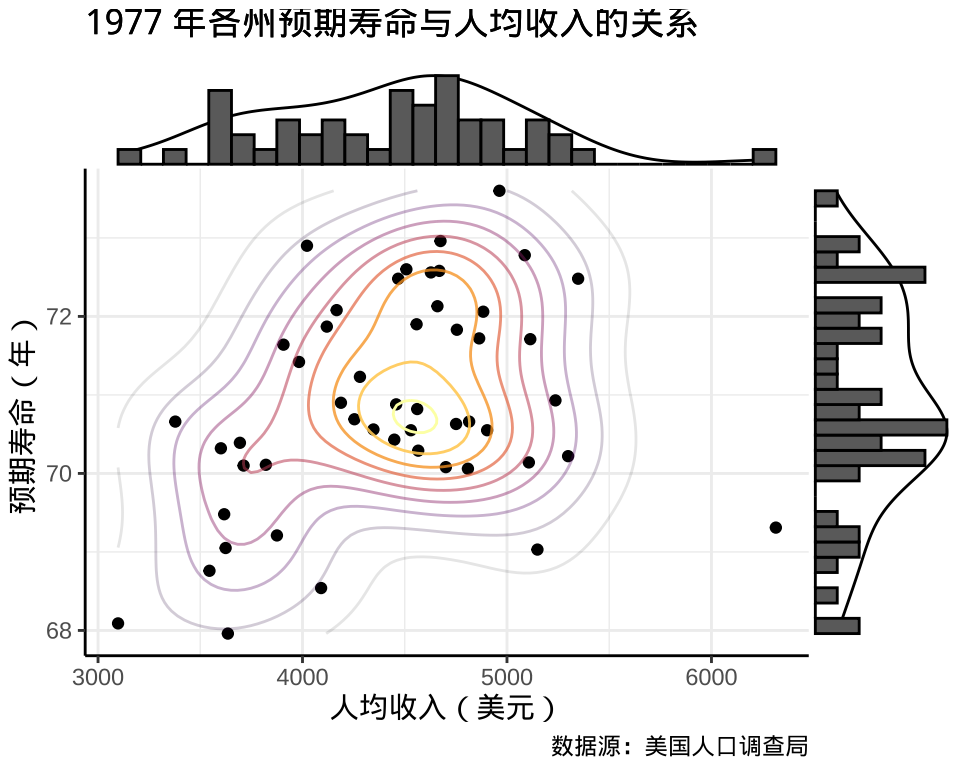


图 9.14: 描述边际分布

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() +
  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"
  )
```

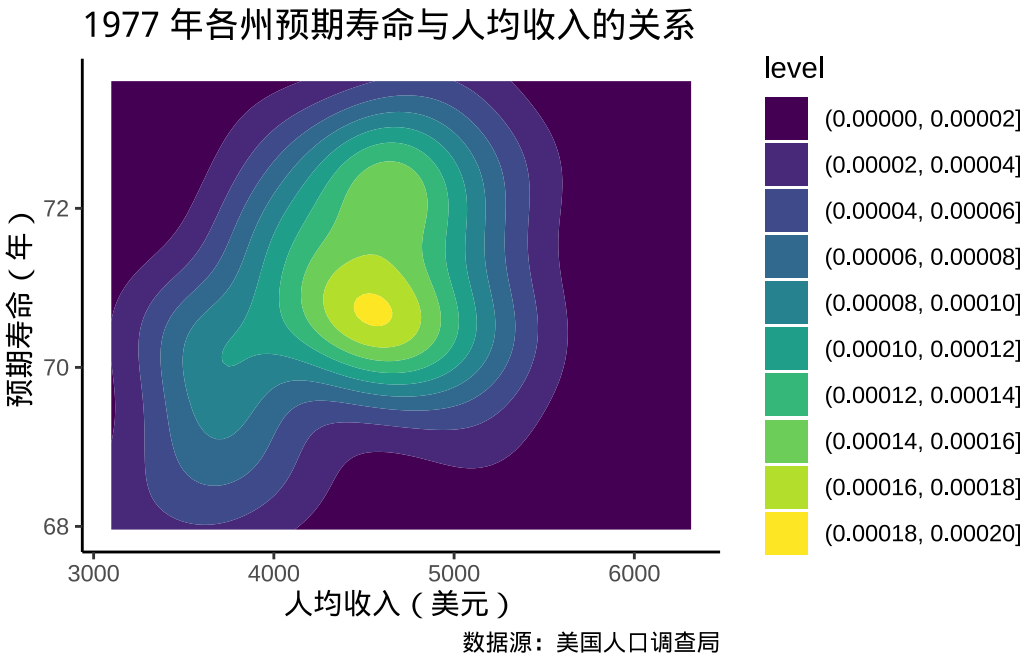


图 9.15: 二维密度图

相比于 **ggplot2** 内置的二维核密度估计，**ggdensity** (Otto 和 Kahle 2022) 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 与 ggdensity 类似，展示 highest density regions (HDR)

```
library(ggdensity)
ggplot(data = state_x77, aes(x = Income, y = `Life Exp`)) +
  geom_point() +
  geom_hdr() +
  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"
  )
```

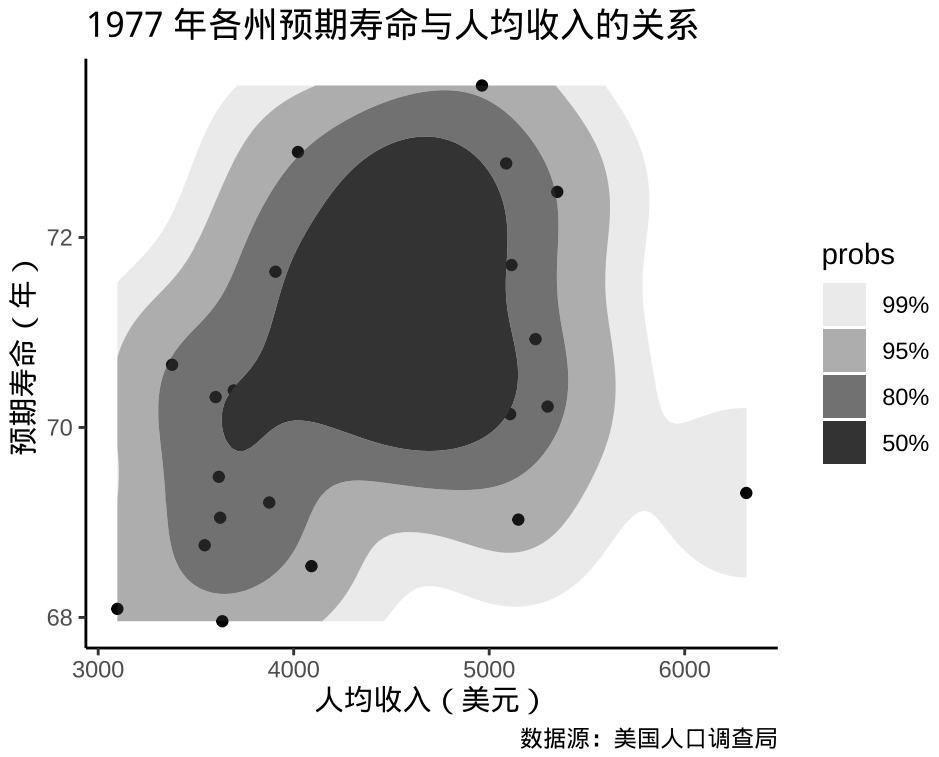


图 9.16: 二维密度图

9.1.5 岭线图

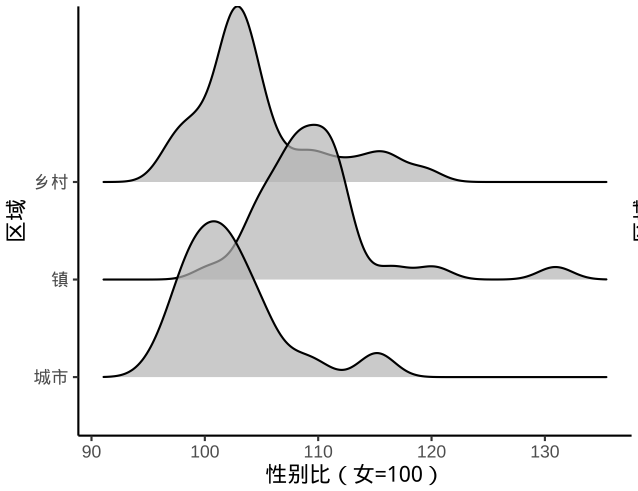
叠嶂图，还有些其它名字，如峰峦图、岭线图等，详情参考统计之都主站《叠嶂图的前世今生》，主要用来描述数据的分布情况，在展示分布的对比上效果非常好。

图 9.17 设置窗宽为 1.5 个百分点

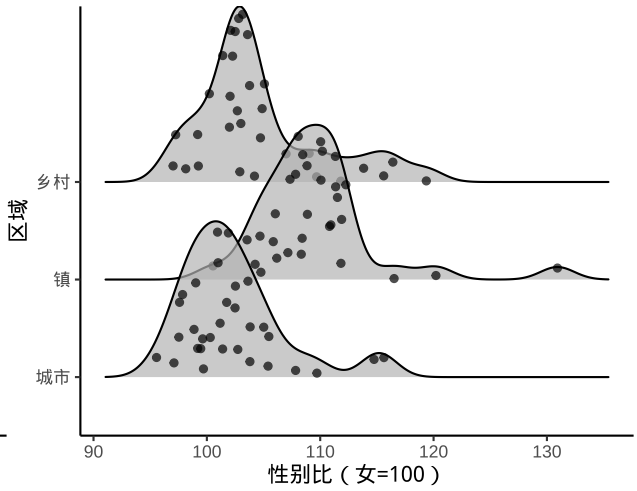
> 提示
>
> 除了中国国家统计年鉴，各省、自治区、直辖市及各级统计局每年都会发布一些统计年鉴、公告等数据，读者可以在此基础上继续收集更多数据，来分析诸多有意思的问题：
>
> 1. 城市、镇和乡村男女性别比呈现差异化分布的成因。
> 2. 城市、镇和乡村男女年龄构成。
> 3. 将上述问题从省级下钻到市、县级来分析。

9.1.6 抖动图

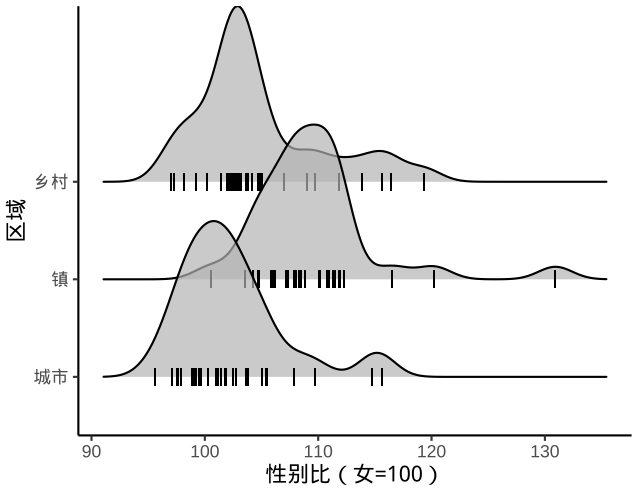
散点图展示原始数据



(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 9.17: 描述数据分布

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

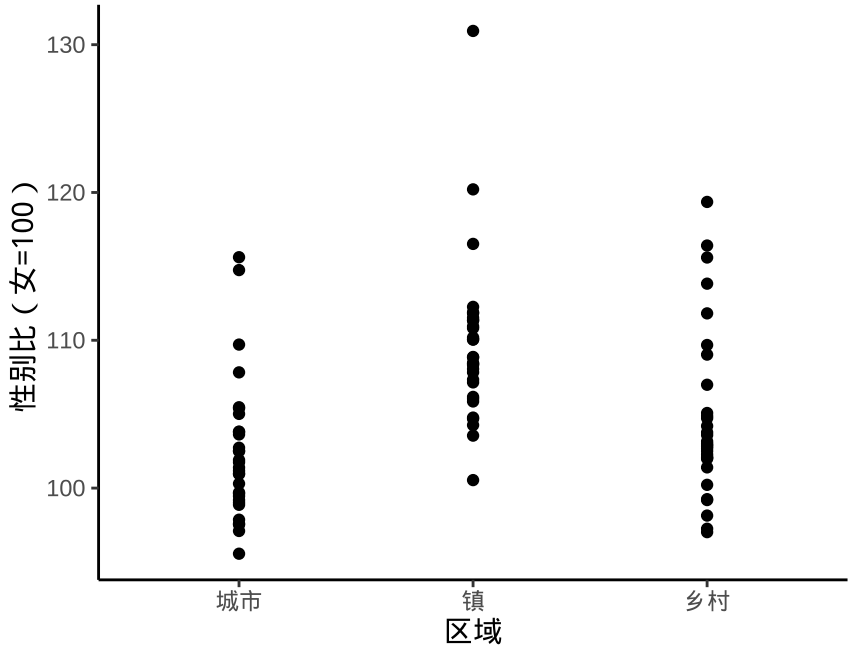


图 9.18: 散点图

提示

Base R 函数 `stripchart()` 可以实现类似图 9.18 的效果。

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

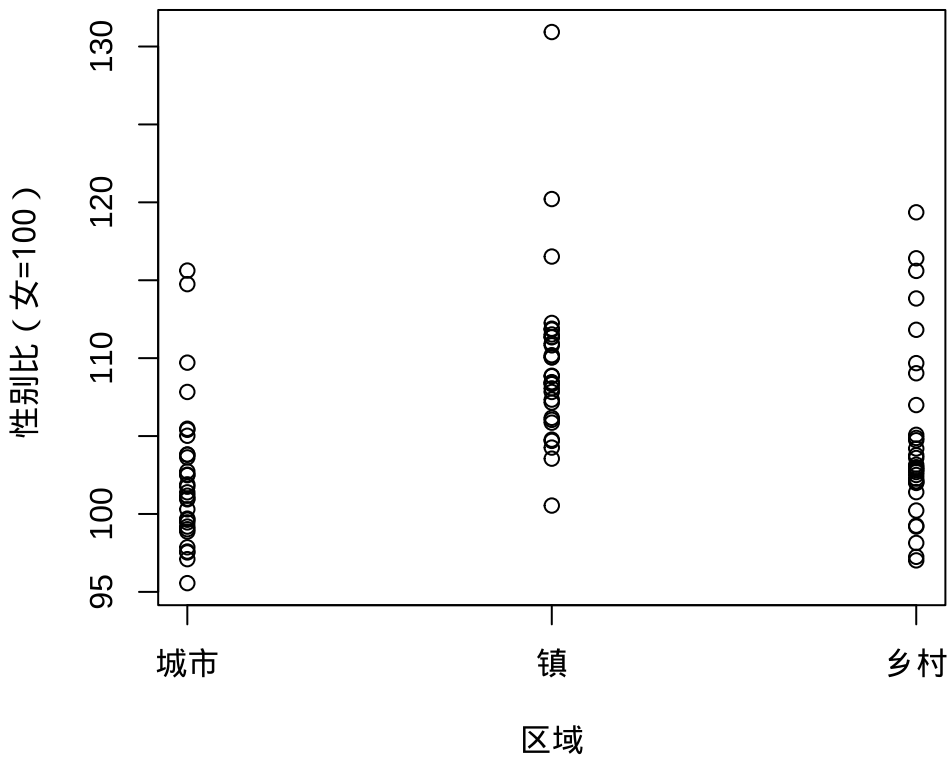


图 9.19: 散点图

抖动图展示原始数据

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

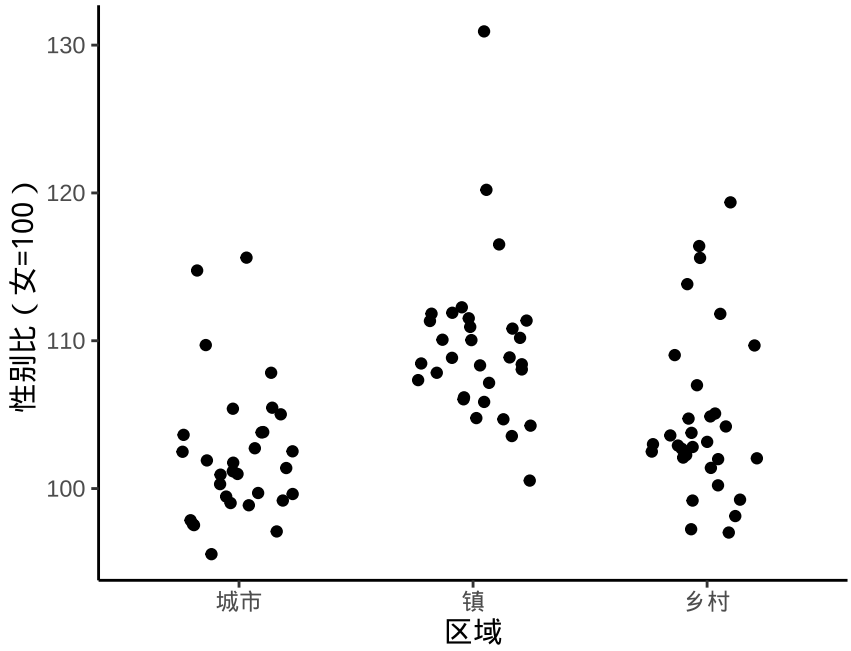


图 9.20: 抖动图

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

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

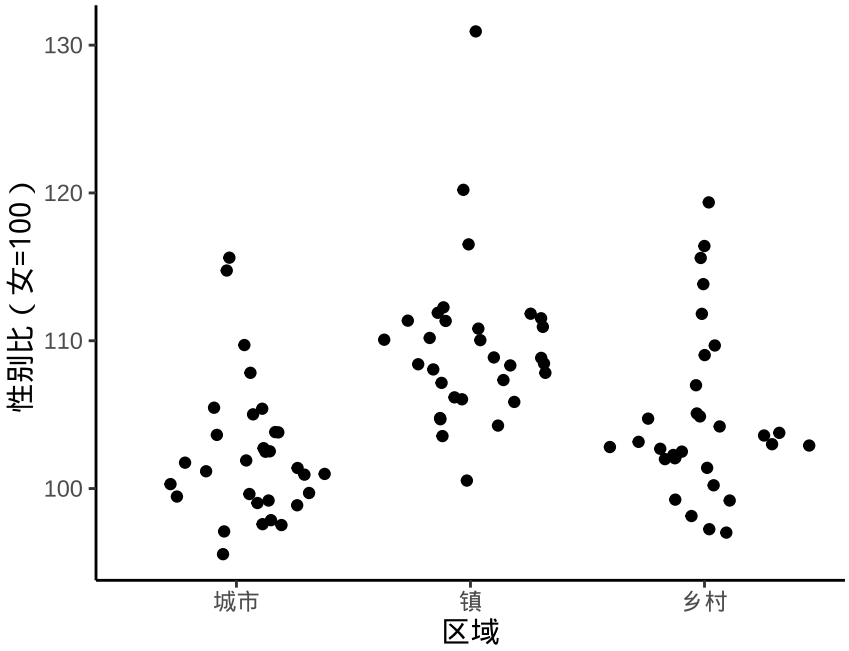


图 9.21: 加强版的抖动图

数据点受 violin 的曲线限制

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

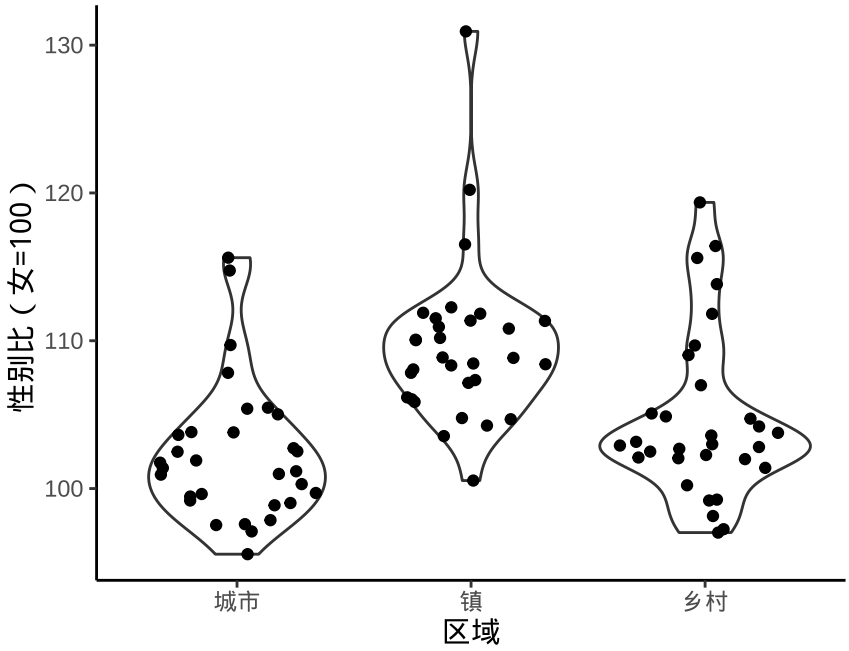


图 9.22: 加强版的抖动图

蜂群图也是某种形式的抖动图

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

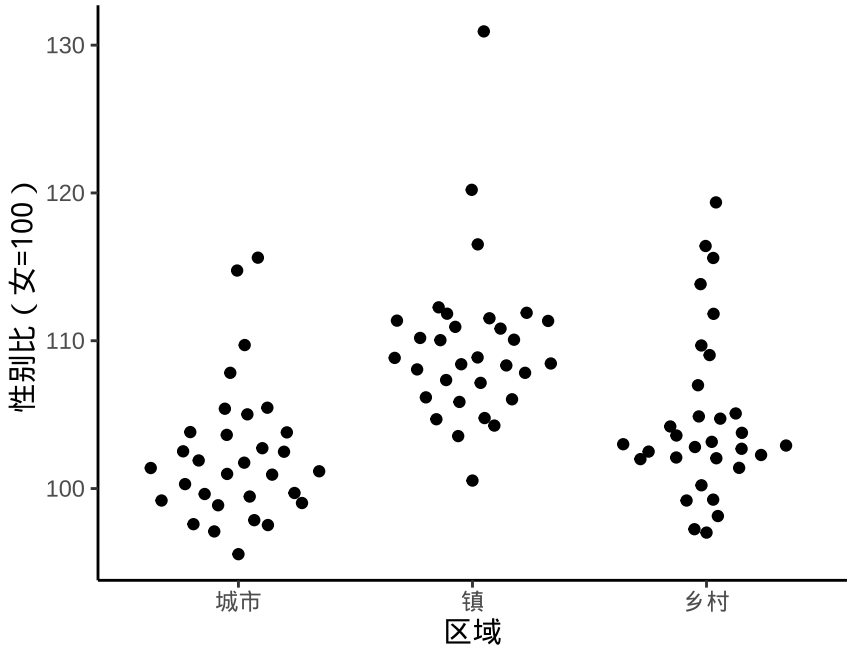


图 9.23: 加强版的抖动图

添加 violin 作为参考边界，与 sina 是非常类似的

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

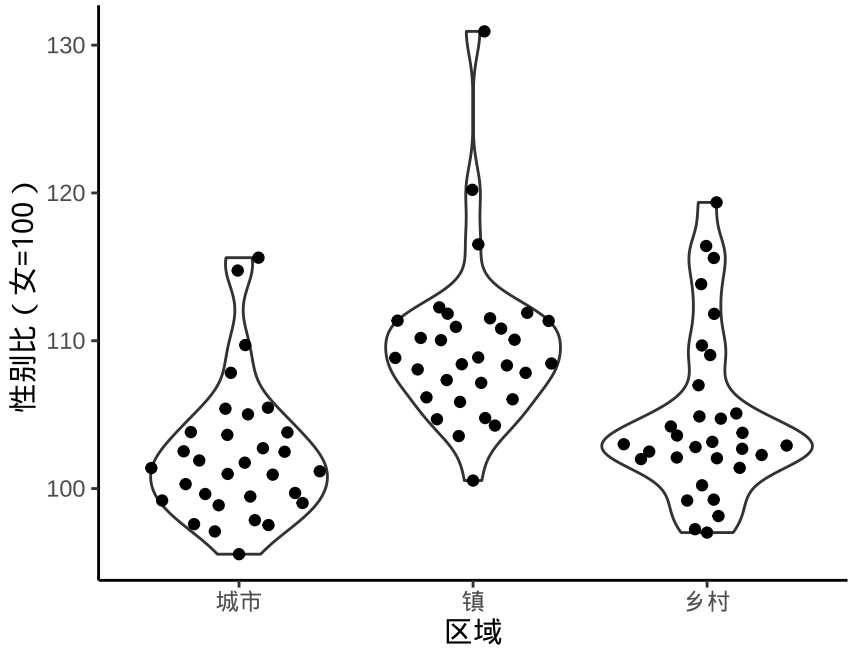


图 9.24: 加强版的抖动图

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() +
  theme_classic()
```

9.2 描述关系

9.2.1 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 9.2: 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

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
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 (%) ")
```

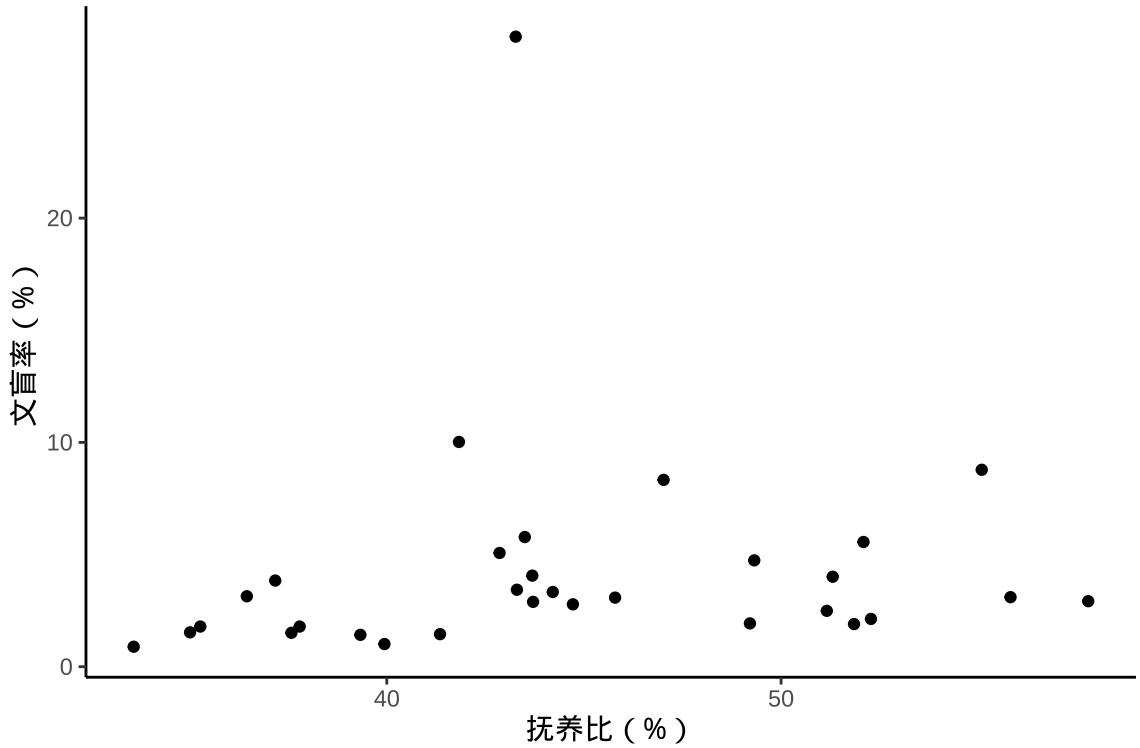


图 9.25: 文盲率与抚养比的关系

9.2.2 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 9.26 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
  ) +
  coord_cartesian(xlim = c(30, 60), ylim = c(0, 10.5), expand = FALSE) +
  theme_classic() +
  labs(x = "抚养比（%）", y = "文盲率（%）")
```

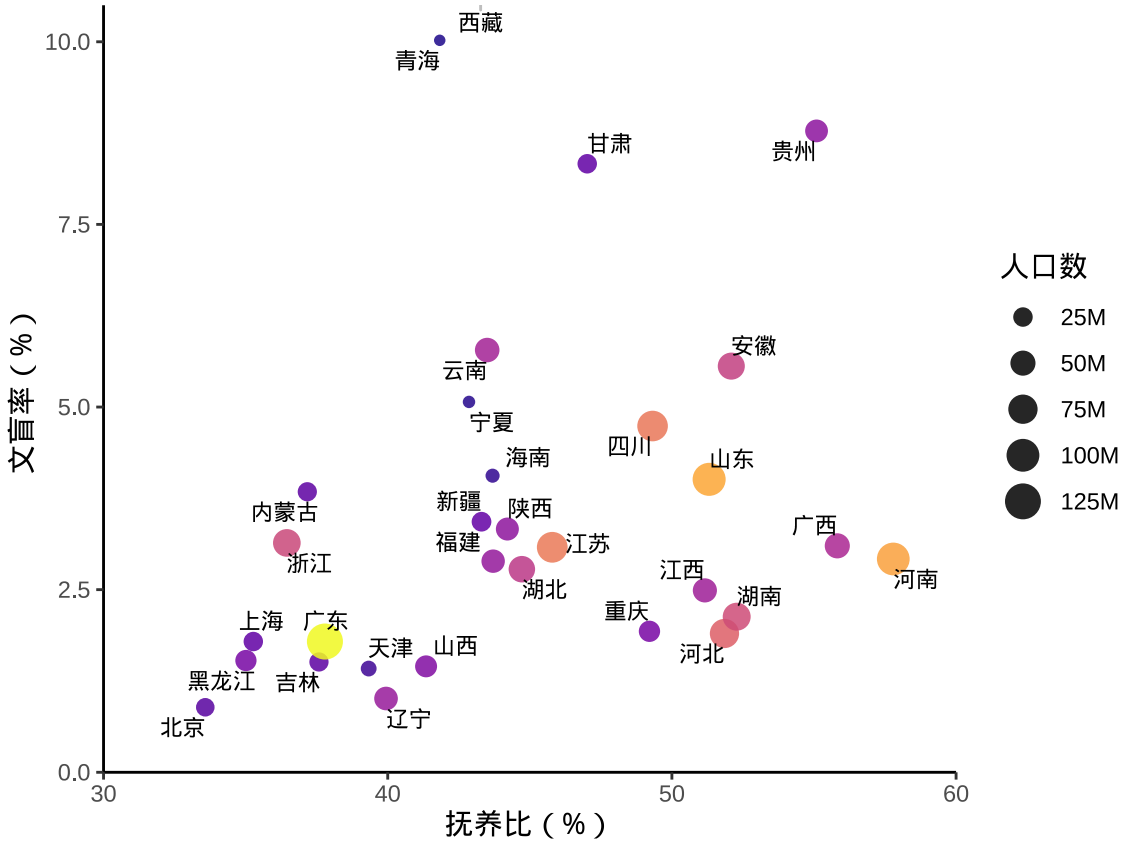


图 9.26: 文盲率和抚养比数据

9.2.3 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 9.27 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(
    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)
```

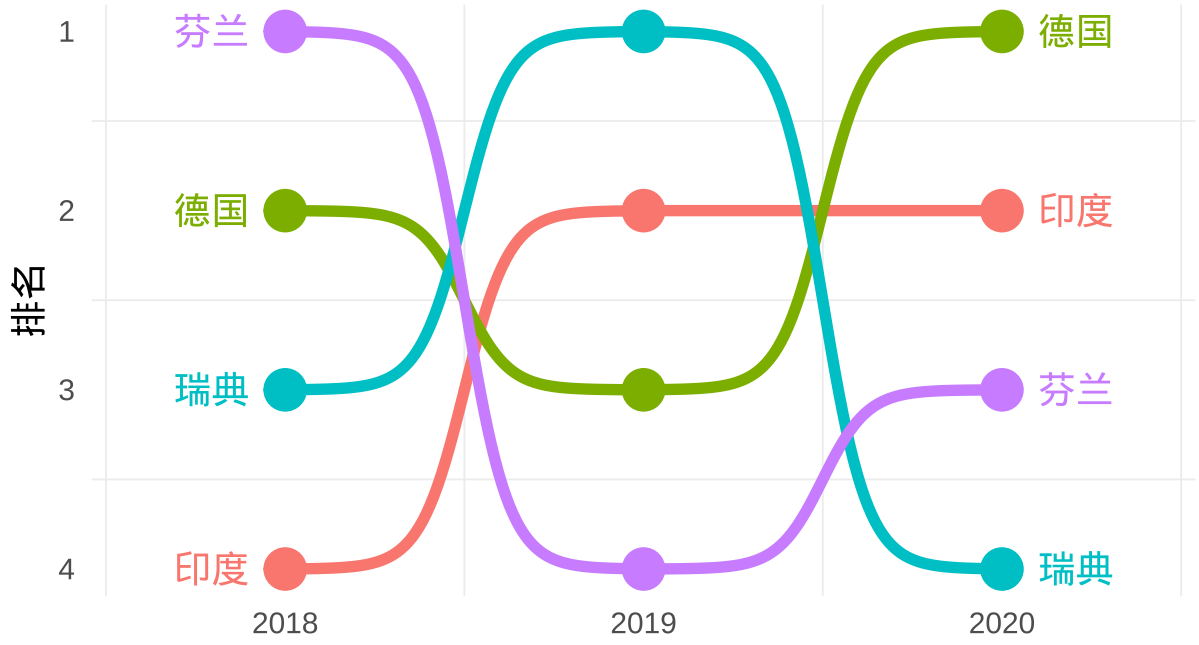


图 9.27: 凹凸图

9.2.4 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 9.28 所示

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

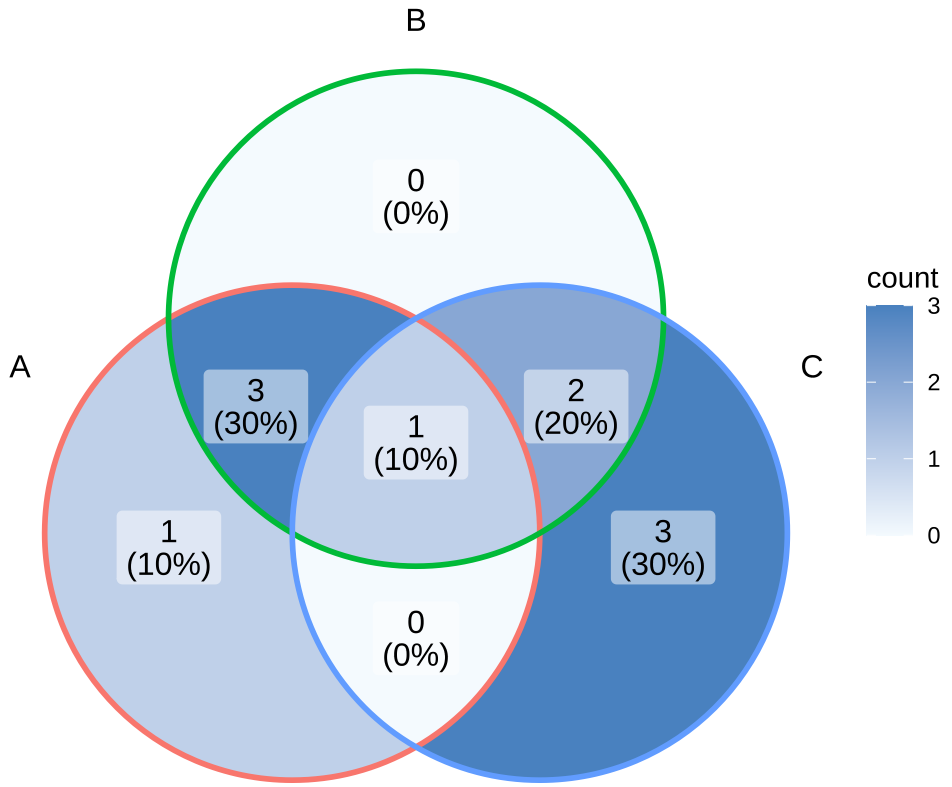


图 9.28: A、B、C 三个集合的交叉关系

9.2.5 网络图

tidygraph包操作图数据，**ggraph**包描述网络图中节点重要性及依赖关系。

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

graph
```

```
#> # A tbl_graph: 70 nodes and 506 edges
#> #
#> # A directed multigraph with 1 component
#> #
#> # A tibble: 70 x 2
#>   name  Popularity
#>   <chr>      <dbl>
#> 1 1              2
#> 2 2              0
```

```
#> 3 3              0
#> 4 4              4
#> 5 5              5
#> 6 6              2
#> # i 64 more rows
#> #
#> # A tibble: 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"
  )
```

```
#> Warning: Using the `size` aesthetic in this geom was deprecated in ggplot2 3.4.0.
#> i Please use `linewidth` in the `default_aes` field and elsewhere instead.
```

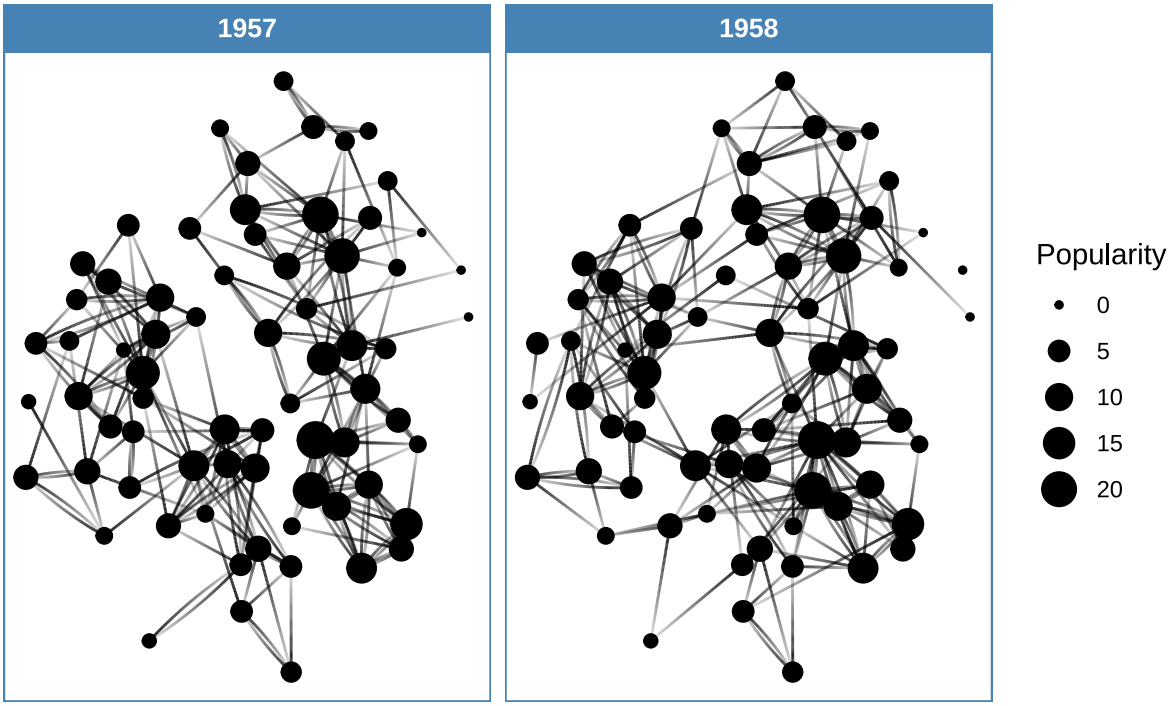


图 9.29: 学校关系

9.3 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

9.3.1 置信区间

表格 9.3: 二项分布的分布列

0	1	2	⋯	n
p_0	p_1	p_2	⋯	p_n

二项分布 Binomial(n, p) 的参数 p 的精确区间估计如下：

$$\left(\text{Beta}(\frac{\alpha}{2}; x, n-x+1), \text{Beta}(1-\frac{\alpha}{2}; x+1, n-x)\right) \tag{9.1}$$

其中，x 表示成功次数，n 表示实验次数，$\text{Beta}(p; v, w)$ 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

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

p_x 表示二项分布 $\text{Binomial}(n, p)$ 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

$$B_x(a, b) = \int_0^x t^{a-1}(1-t)^{b-1}\text{dt}$$
$$I_x(a, b) = \frac{B_x(a, b)}{B(a, b)}, \quad B(a, b) = B_1(a, b)$$

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 9.30 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) {
  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")
```

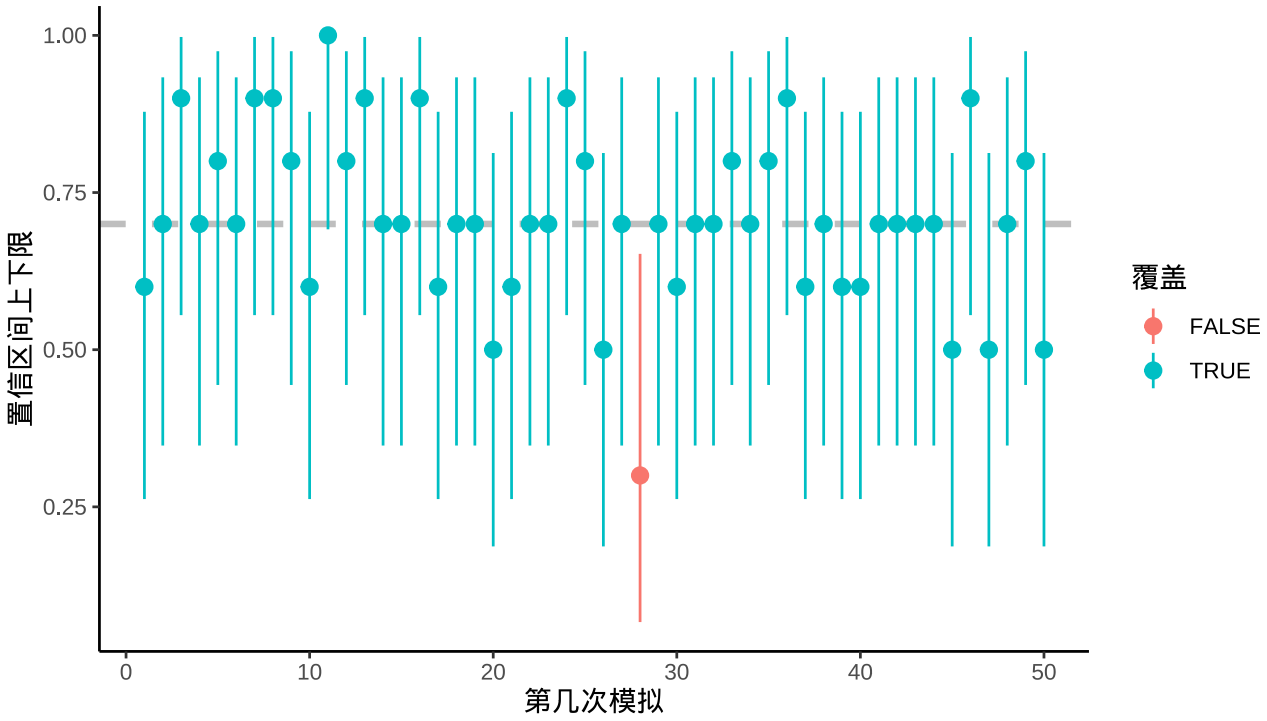


图 9.30: Clopper-Pearson 置信区间

9.3.2 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。

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")
```

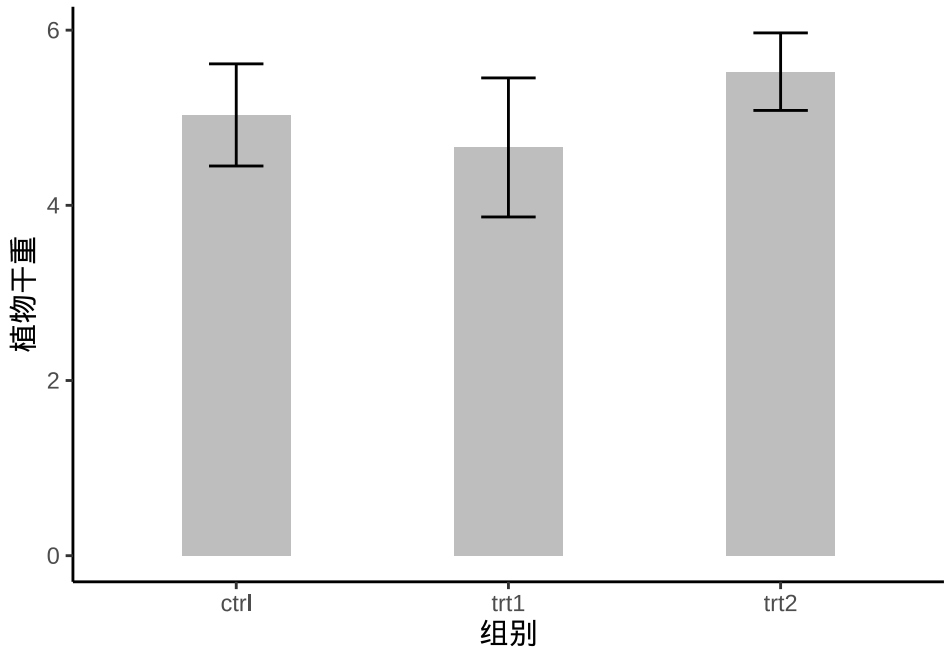


图 9.31: 植物生长

ℹ 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(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
```

图 9.32 展示假设检验的结果

```
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")
```

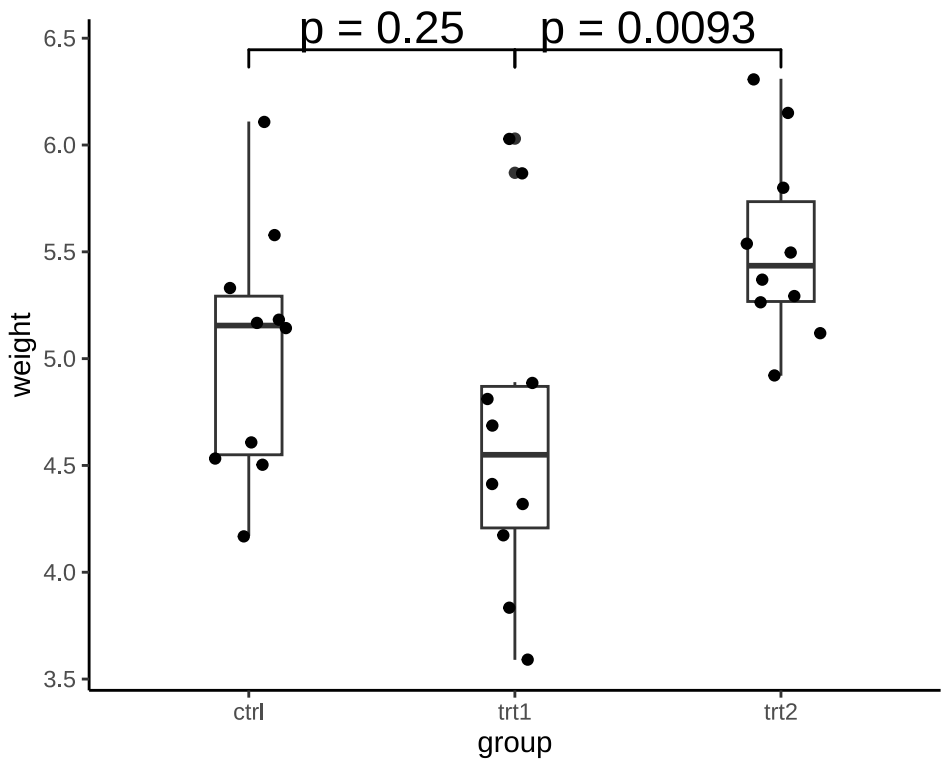


图 9.32: 展示假设检验的结果

9.3.3 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 = T,
    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")
```

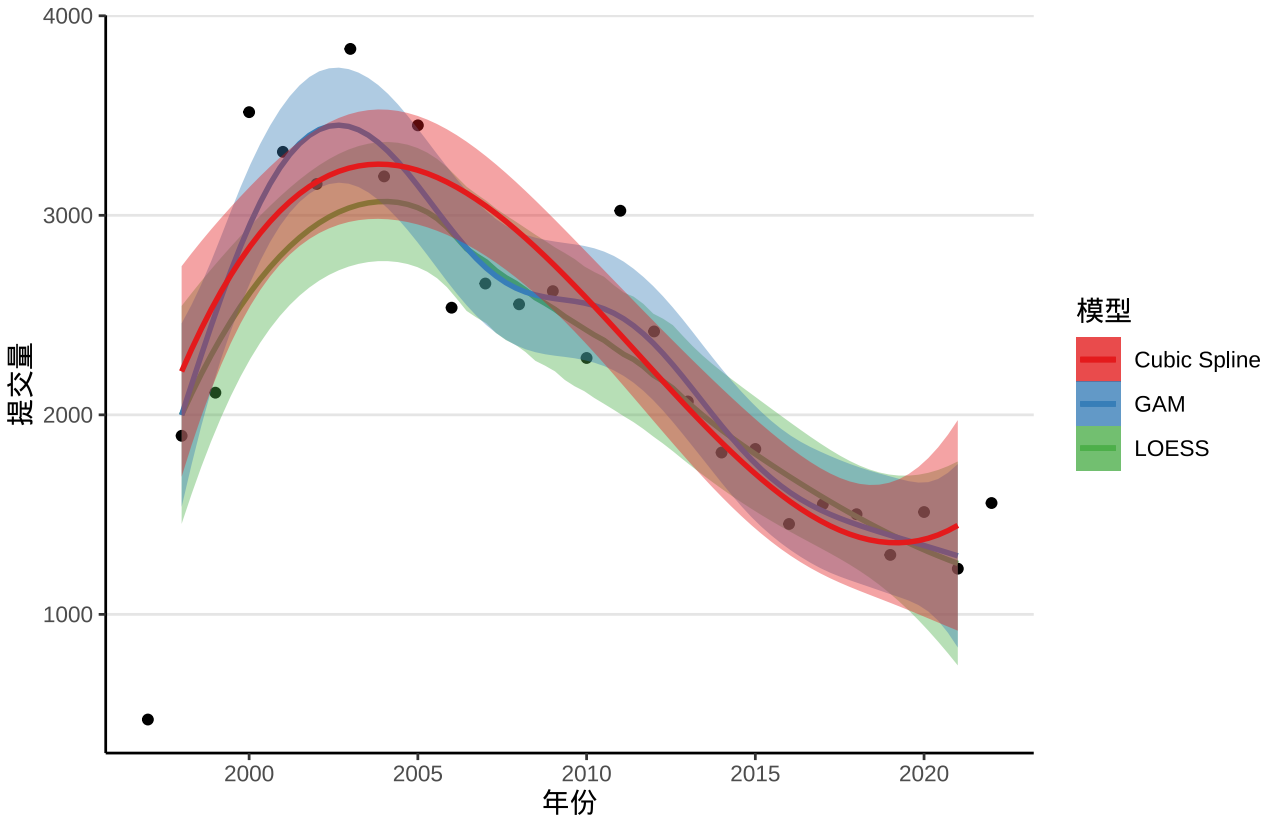


图 9.33: 趋势拟合、预测和推断

9.3.4 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。

```
state_x77 <- data.frame(state.x77,
  state_name = rownames(state.x77),
  state_region = state.region,
  state_abb = state.abb,
  check.names = FALSE
)

# 模型诊断图
fit <- lm(`Life Exp` ~ Income + Murder, data = state_x77)
op <- par(mfrow = c(3, 2), mar = c(4, 4, 3, 1))
plot(fit,
  ask = FALSE, which = c(1, 2, 3, 4, 5, 6),
  caption = list(
    "残差和拟合值", "正态 Q-Q 图",
    "尺度-位置", "Cook 距离", "残差和杠杆值",
    expression("Cook 距离 vs 杠杆值" * h[ii] / (1 - h[ii]))
  )
)
par(op)
```

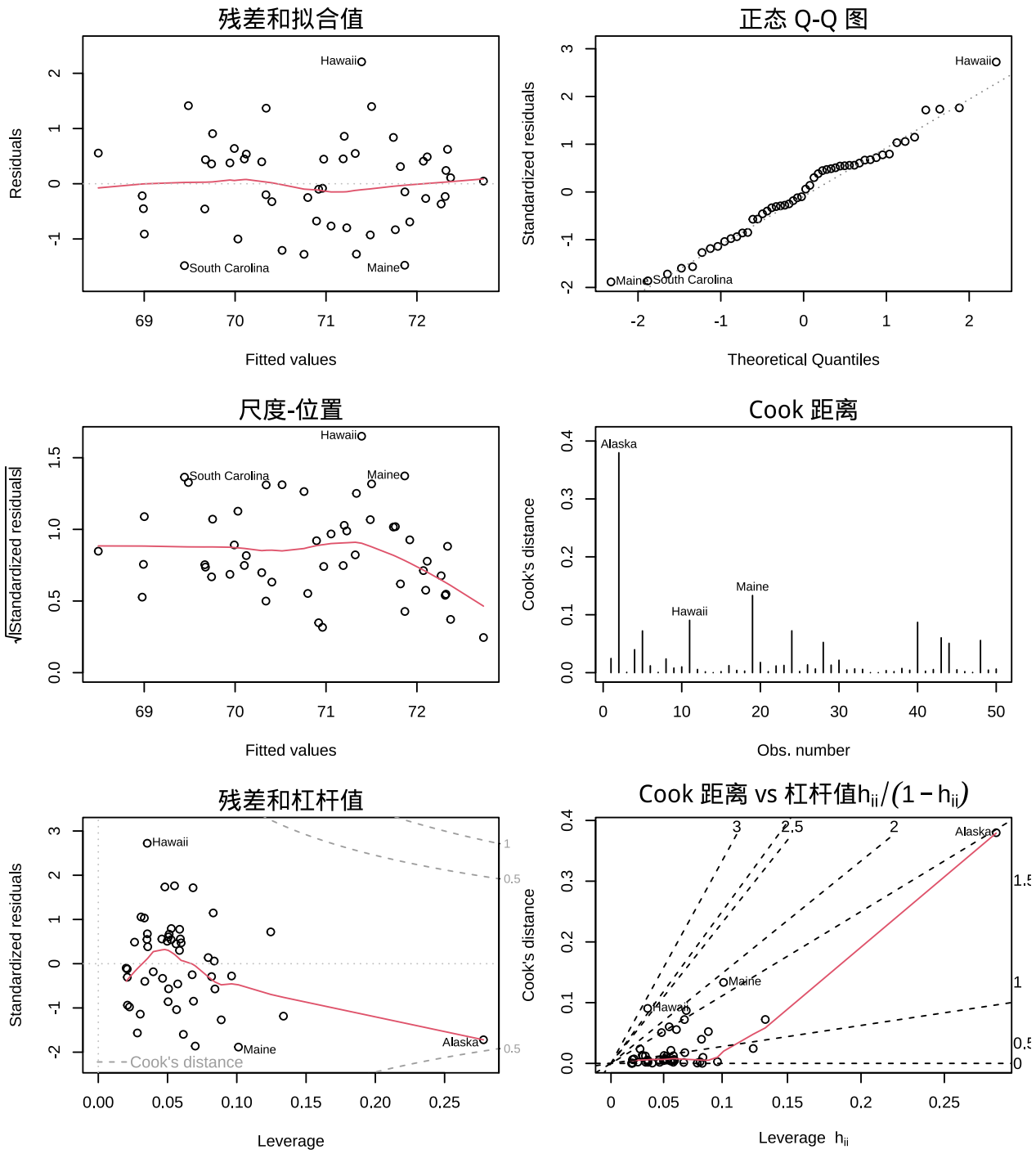


图 9.34: 线性模型的诊断图

对于复杂的统计模型，比如混合效应模型的诊断，**ggPMX** 包。

9.3.5 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：

$$\text{Life Exp} = \alpha + \beta_1 \text{Income} + \beta_2 \text{Murder} + \epsilon \tag{9.2}$$

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")
```

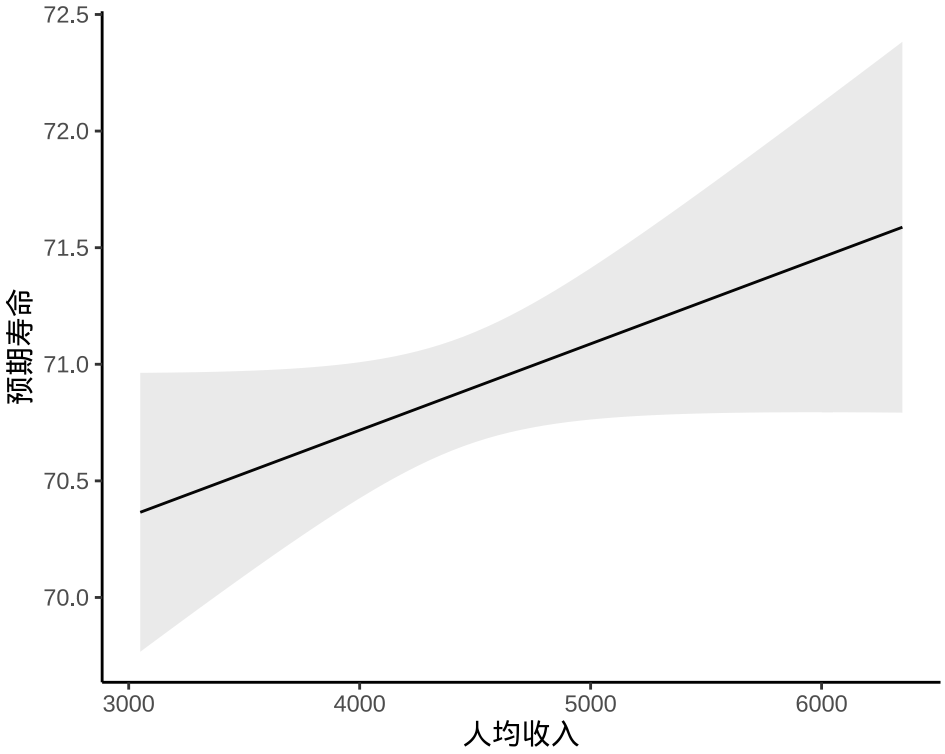


图 9.35: 边际效应

第十章　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)
```

10.1　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。

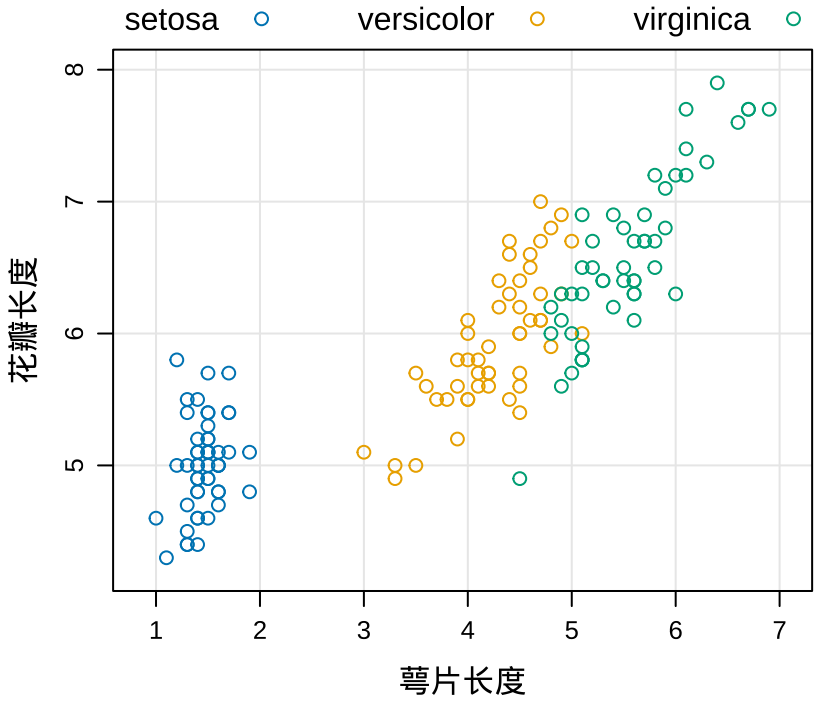


图 10.1: 分组散点图

- 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 10.2b 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",
  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。

10.2 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。

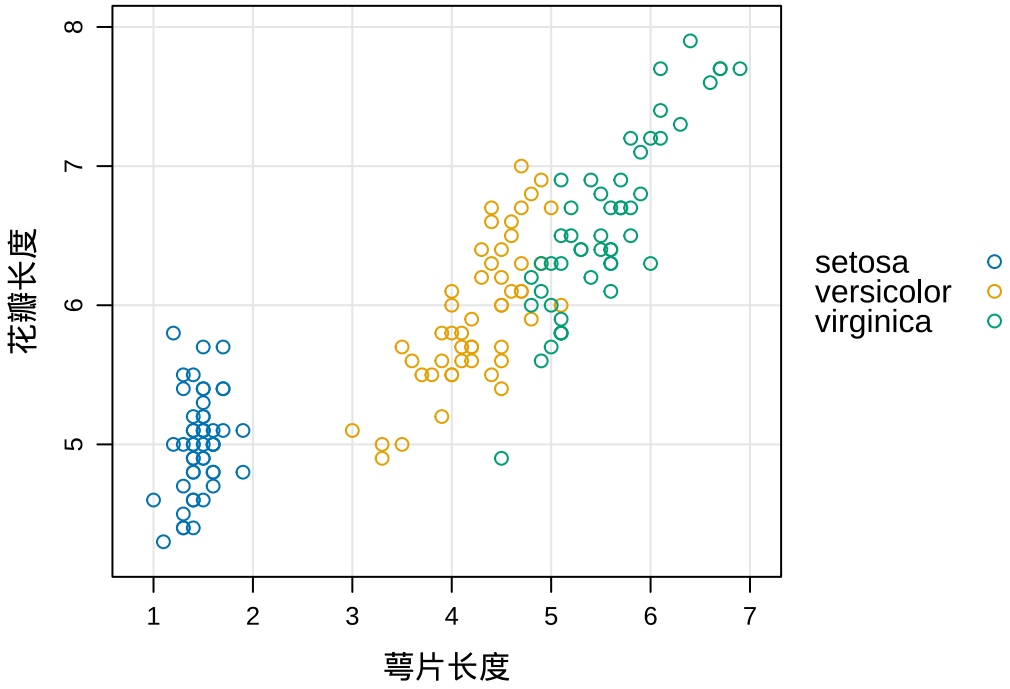
```
tp <- trellis.par.get()
names(tp)
```

```
#>  [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"
```

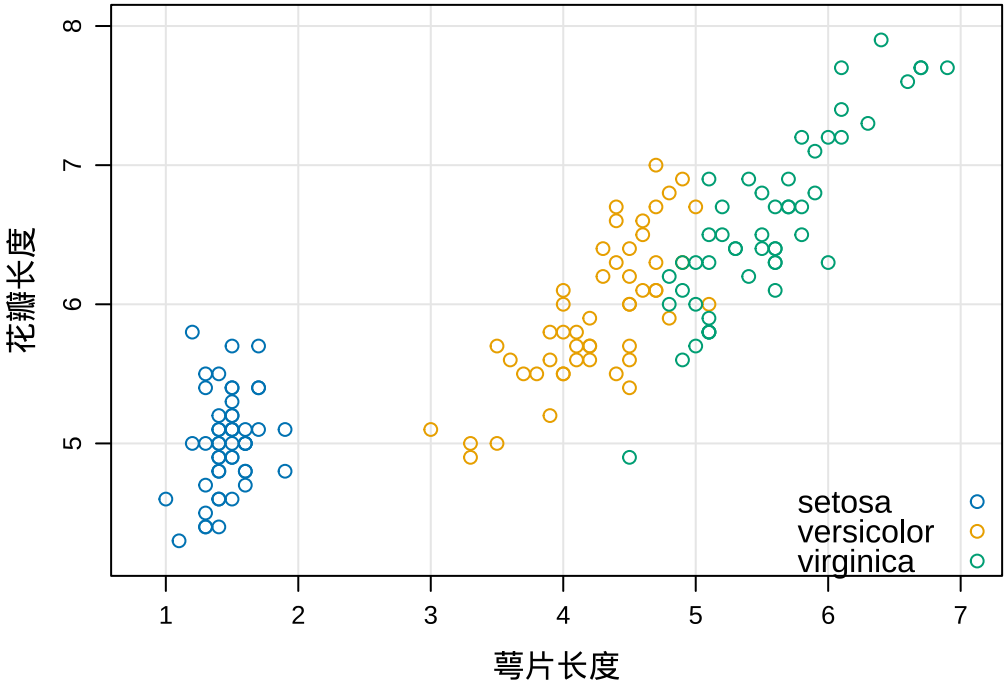
可以看到，图形参数着实非常多，知道了这么多图形参数，而每个参数又有哪些选项可取呢？不忙，再看看图形参数的结构，比如 `superpose.symbol`。

```
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
#>
```



(a) 图例位于图右侧



(b) 图例位于内内部

图 10.2: 调整图例位置

```
#> $col
#>          blue        orange   bluishgreen    vermillion       skyblue
#>     "#0072B2"     "#E69F00"     "#009E73"     "#D55E00"     "#56B4E9"
#>        yellow reddishpurple
#>     "#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 10.3 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 10.4 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 10.1 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)
  )
)
```

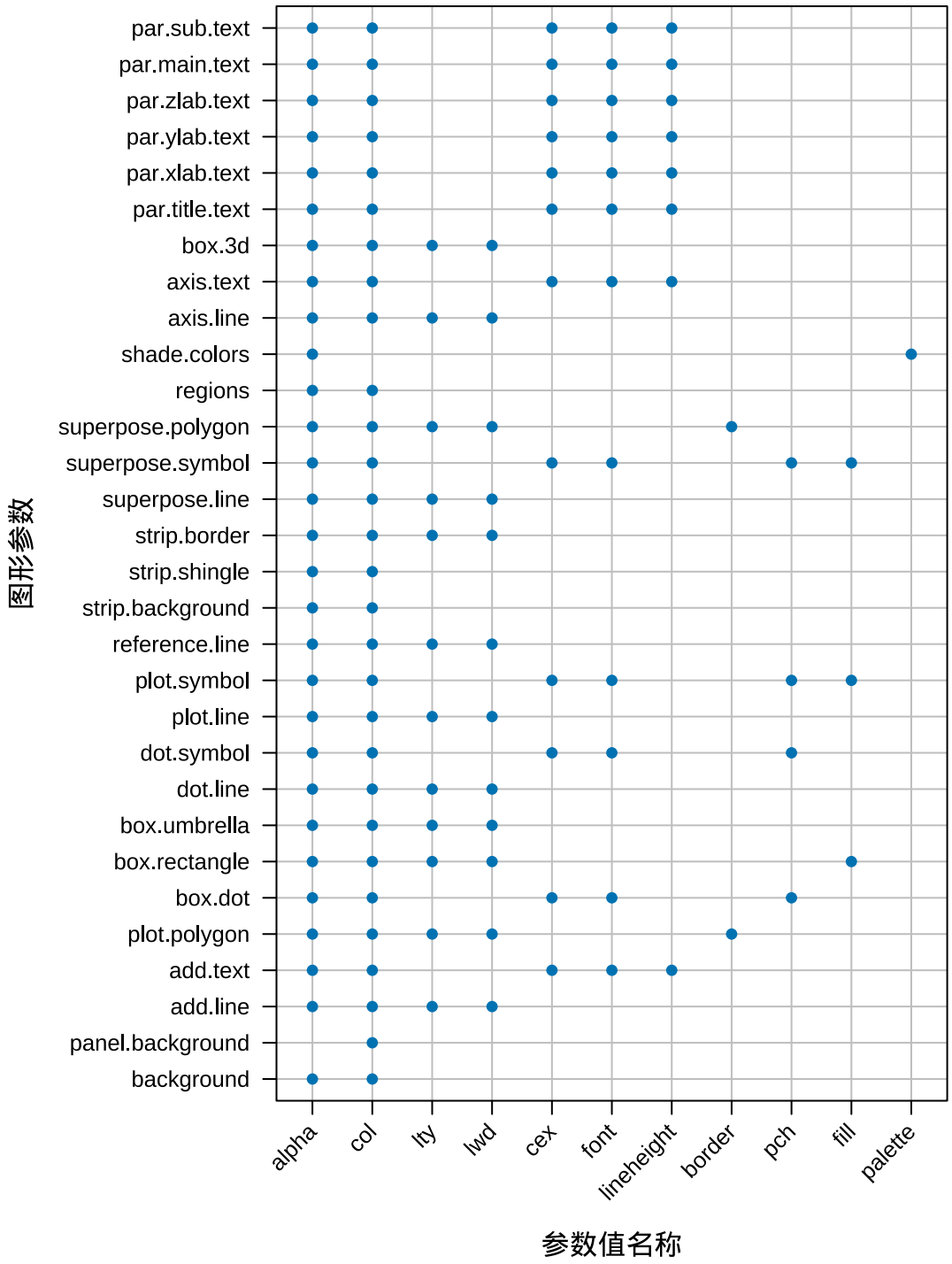


图 10.3: 常用图形参数

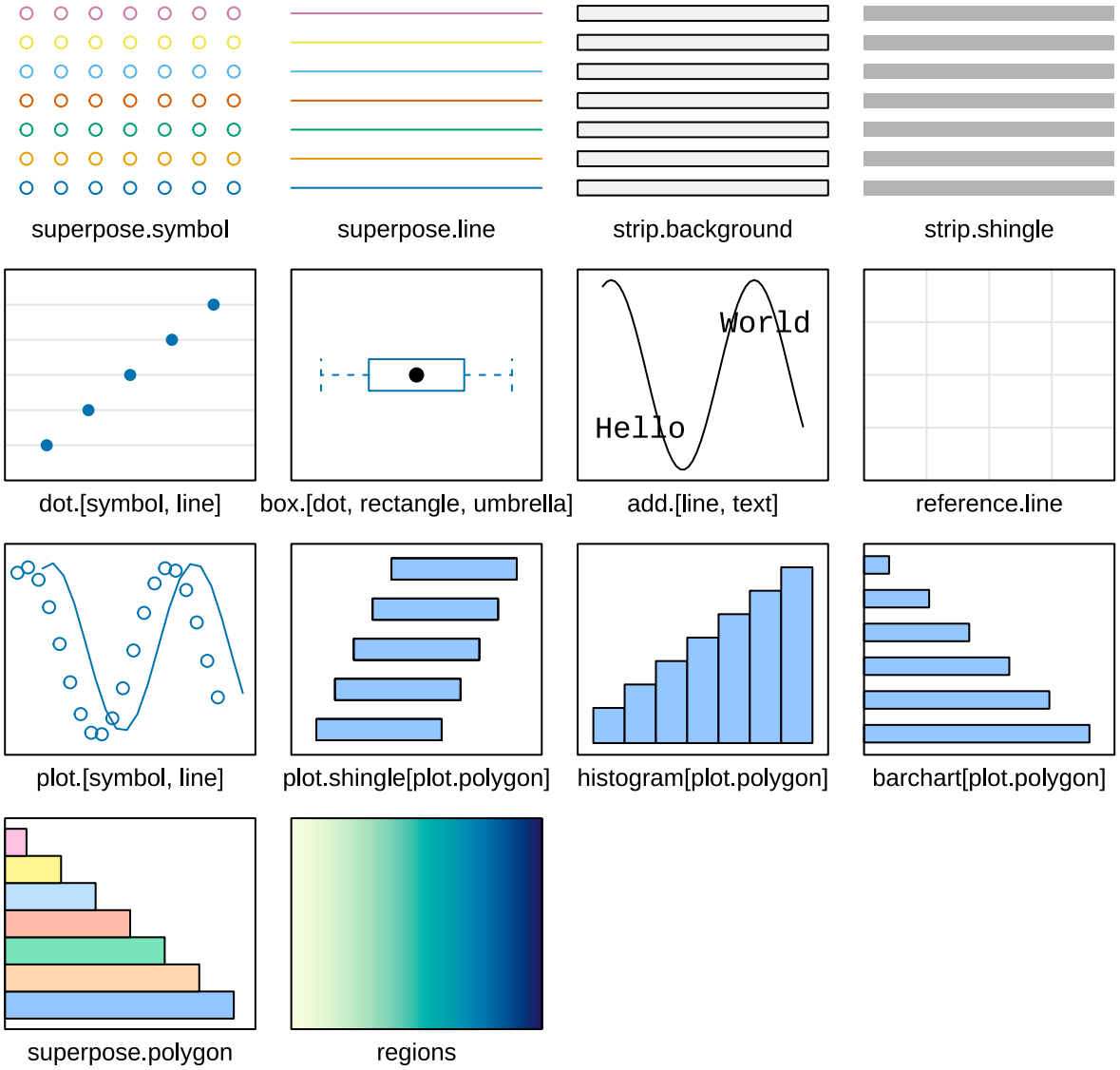


图 10.4: 图形参数效果预览

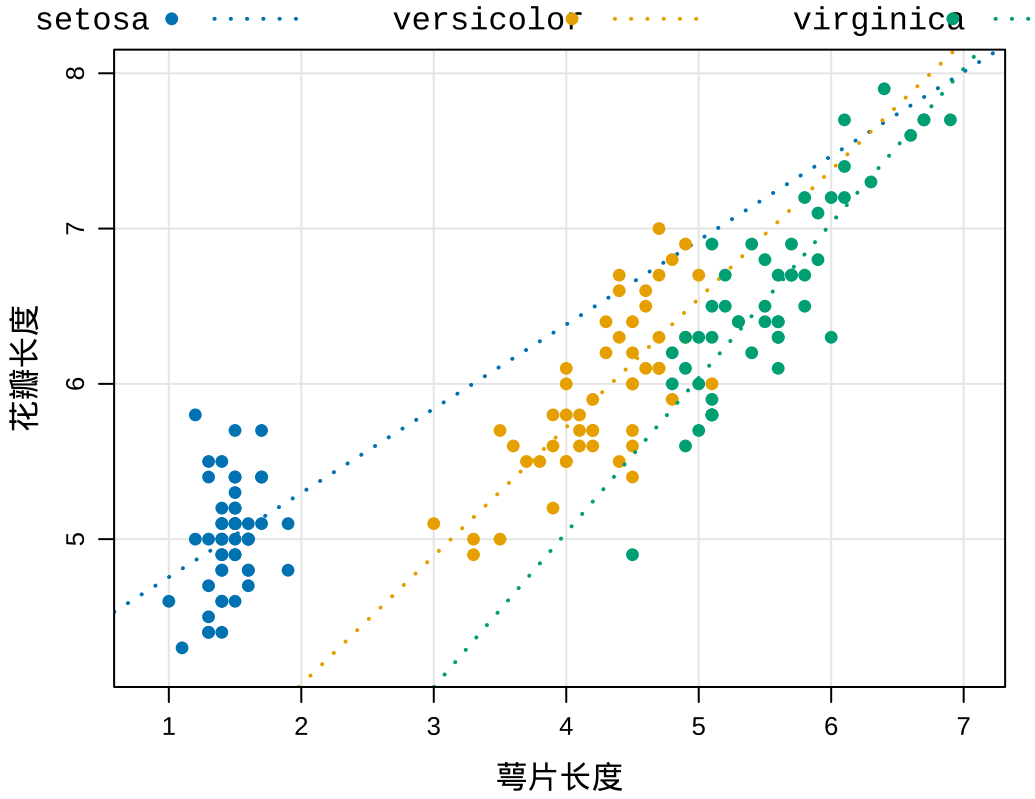
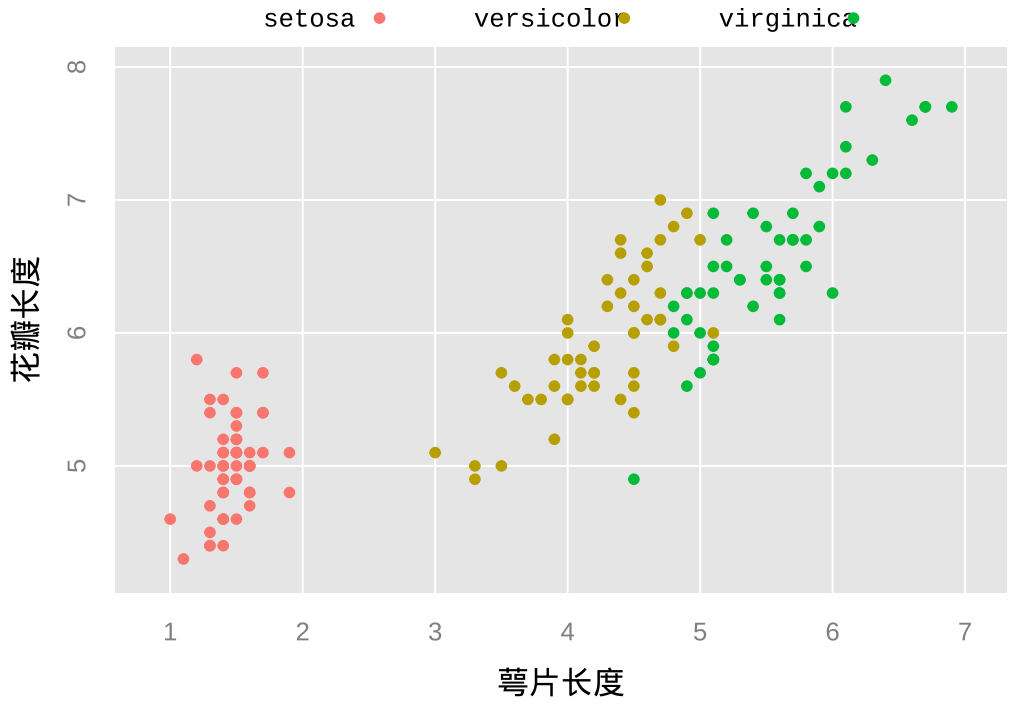


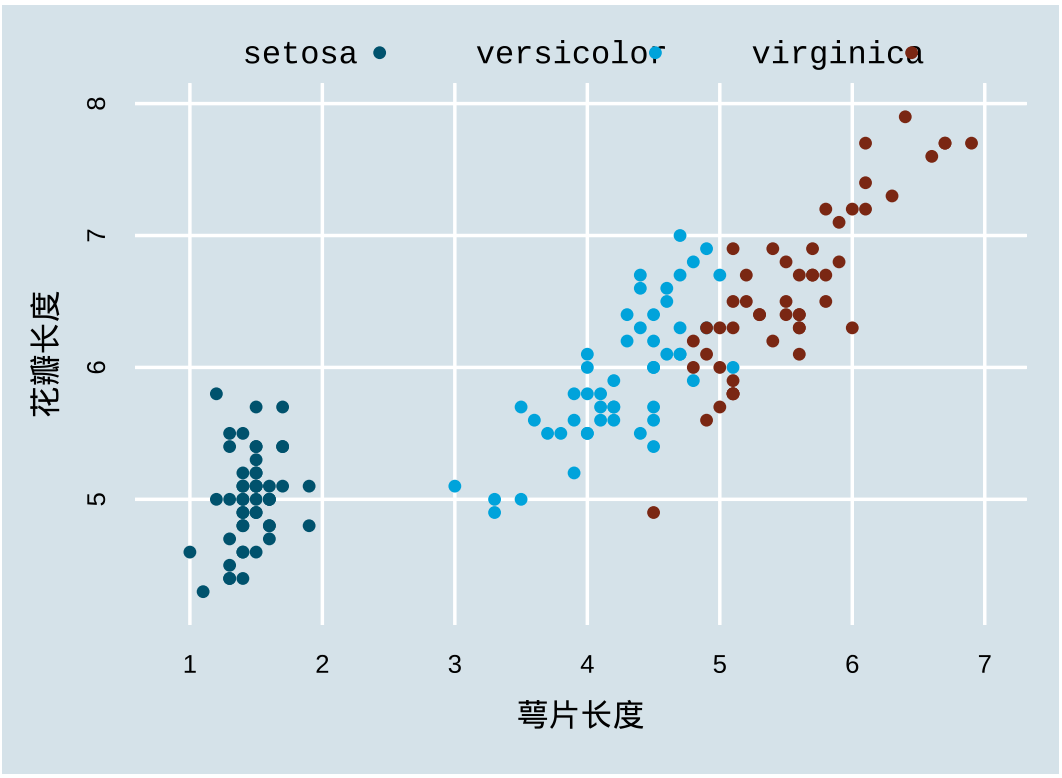
图 10.5: 调整点、线、图例元素

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")
)
```



(a) ggplot2 包默认的绘图主题



(b) 《经济学人》杂志的绘图主题

图 10.6: latticeExtra 内置的两个主题

10.3 常见图形

10.3.1 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 10.7 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)
)
```

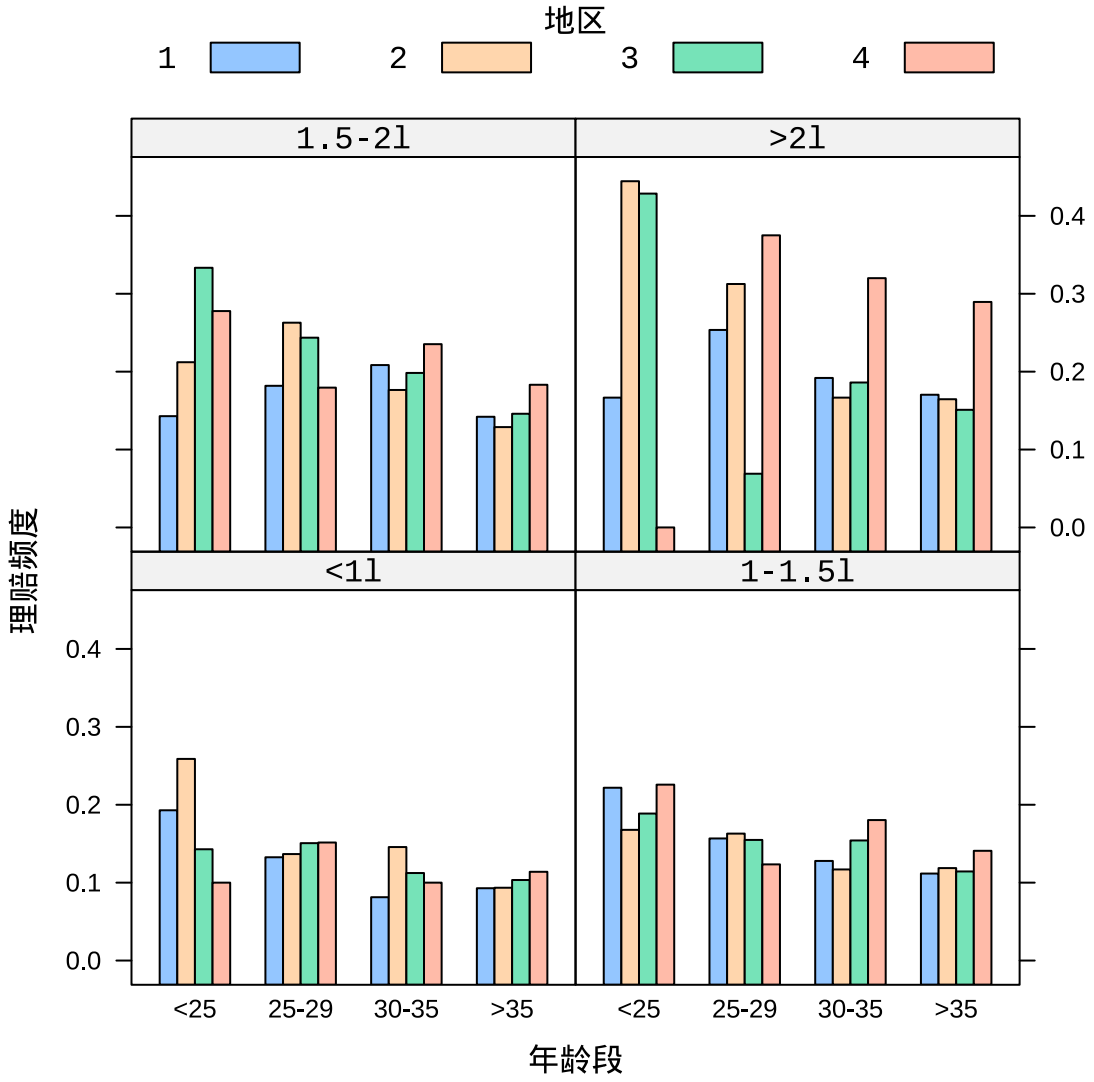


图 10.7: 分组柱形图

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。

10.3.2 分组箱线图

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

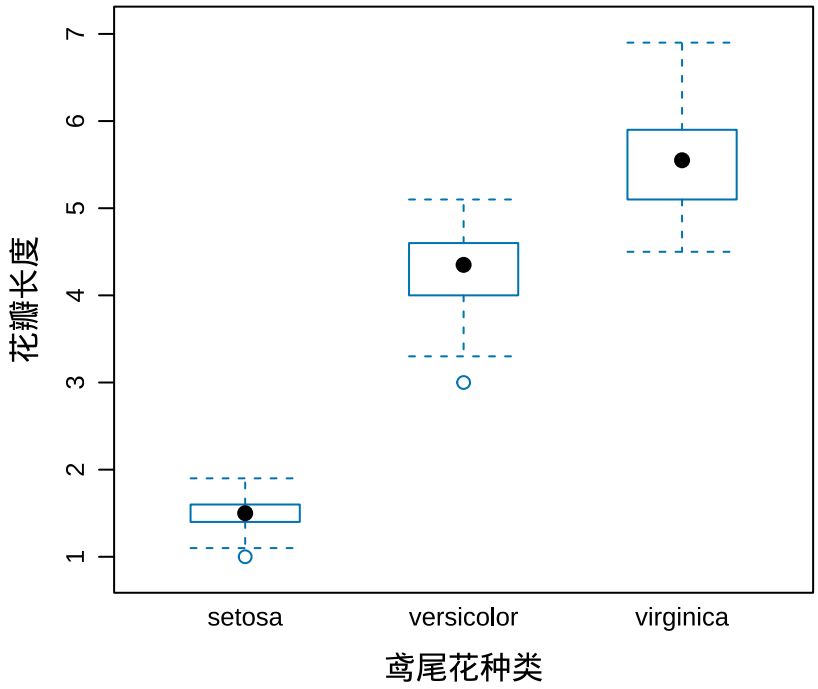


图 10.8: 分组箱线图

10.3.3 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")
```

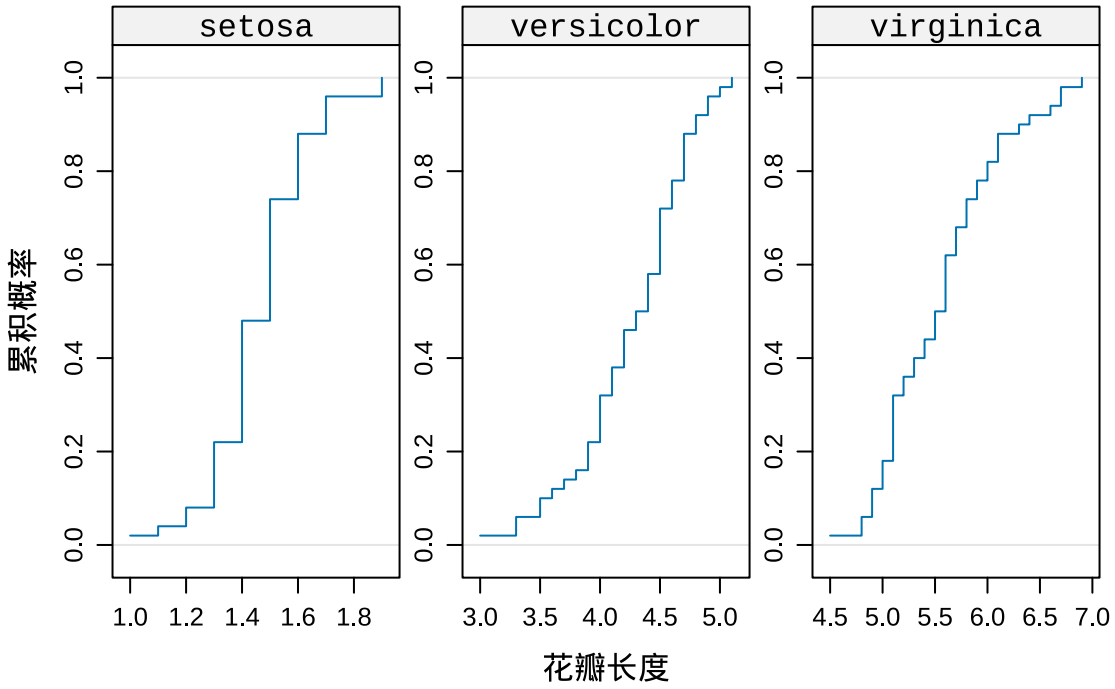


图 10.9: 经验分布图

10.3.4 回归曲线图

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

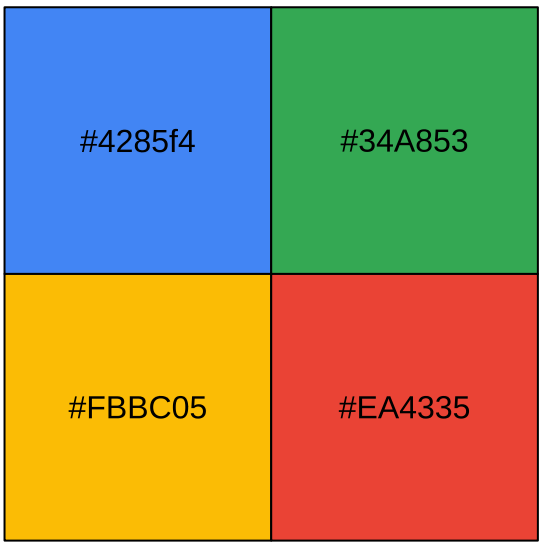
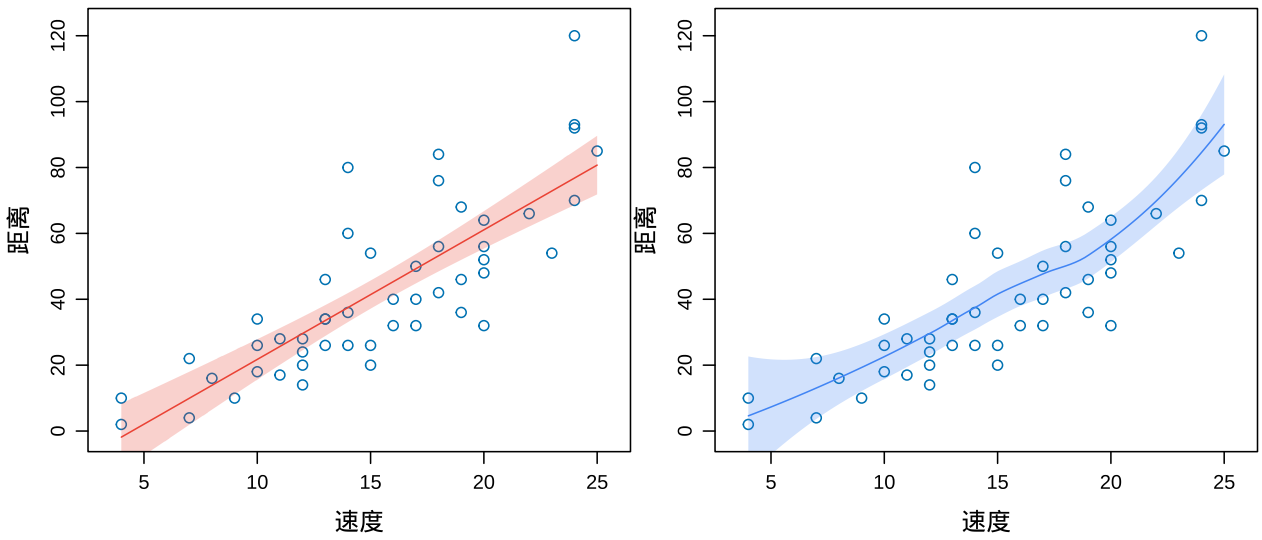


图 10.10: 调色板

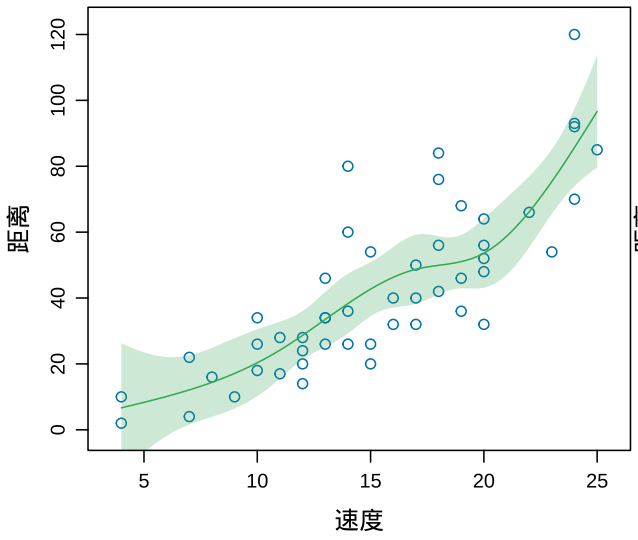
图 10.11 中用不同的回归模型拟合数据中的趋势。1920s 汽车行驶距离和速度的关系图。函数 `panel.smoother()` 来自 **latticeExtra** 包

```
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", ...
    )
  }
)
```

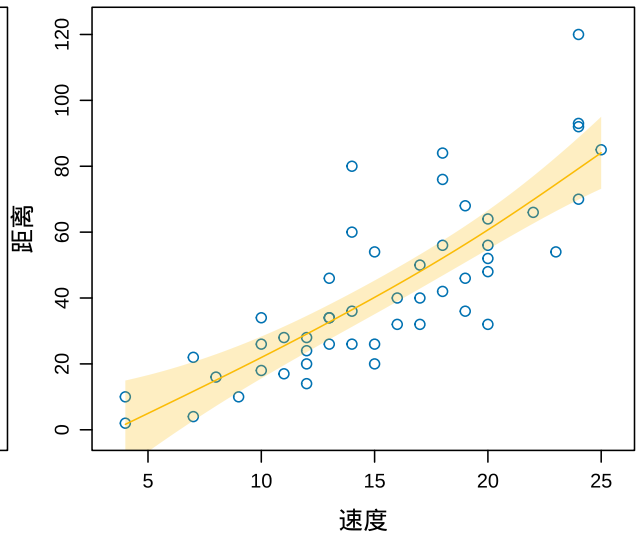


(a) 线性回归

(b) 局部多项式回归



(c) 自然样条回归



(d) 广义可加回归

图 10.11: 回归曲线图

10.3.5 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"
)
```

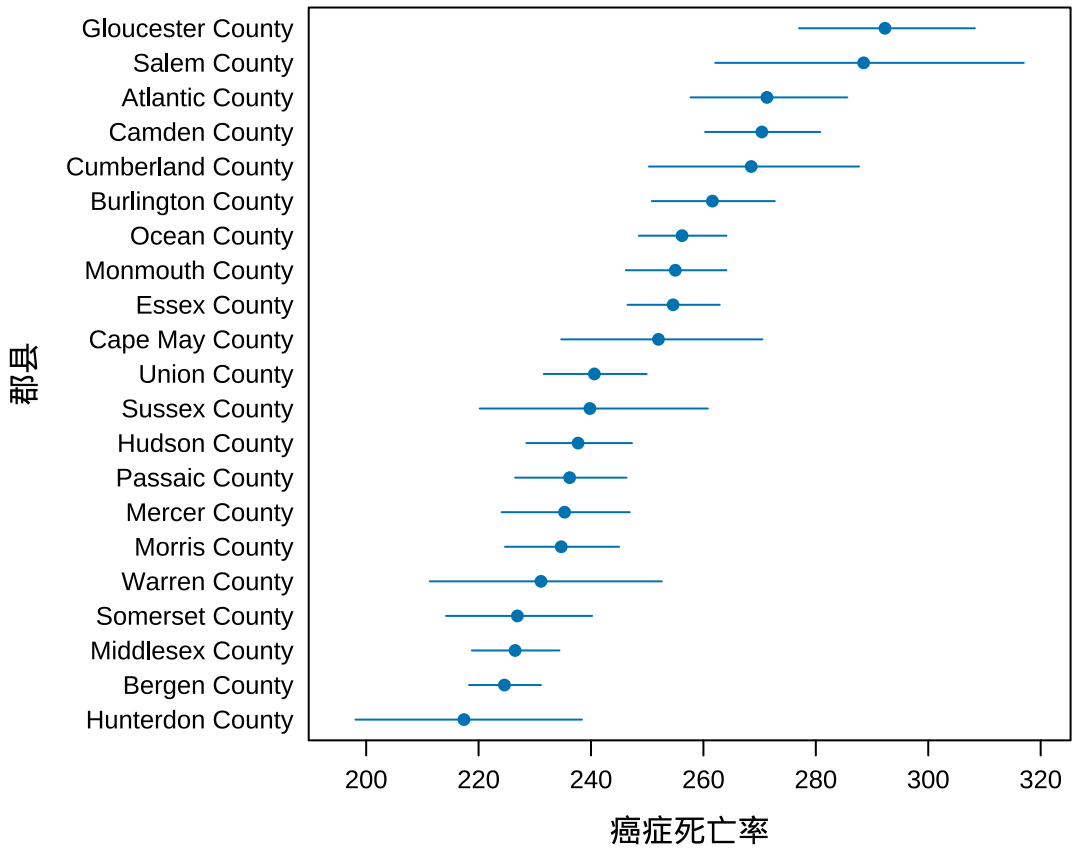


图 10.12: 置信区间图

10.3.6 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",
  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)
)
```

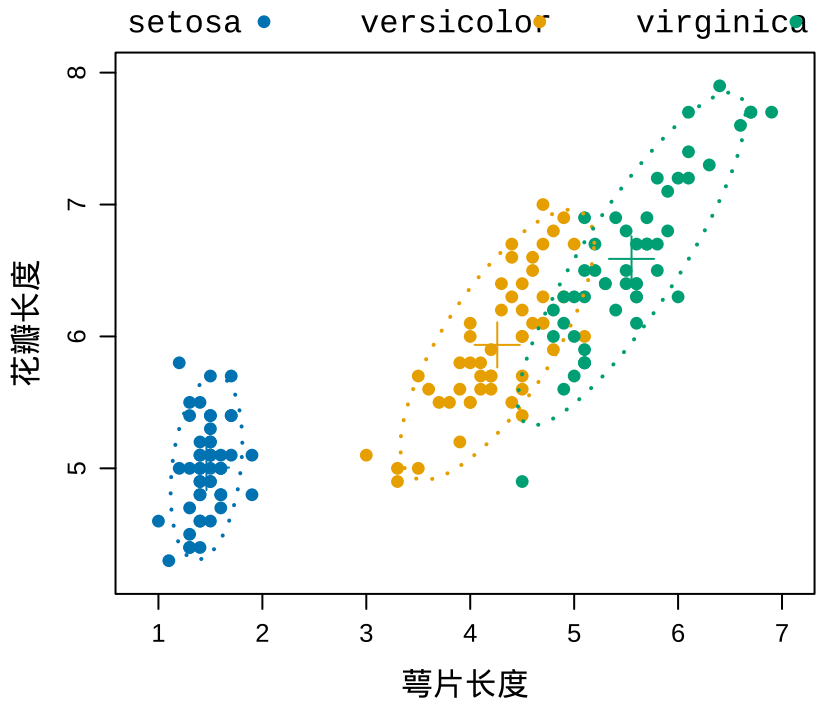


图 10.13: 分组置信椭圆图

10.3.7 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)
)
```

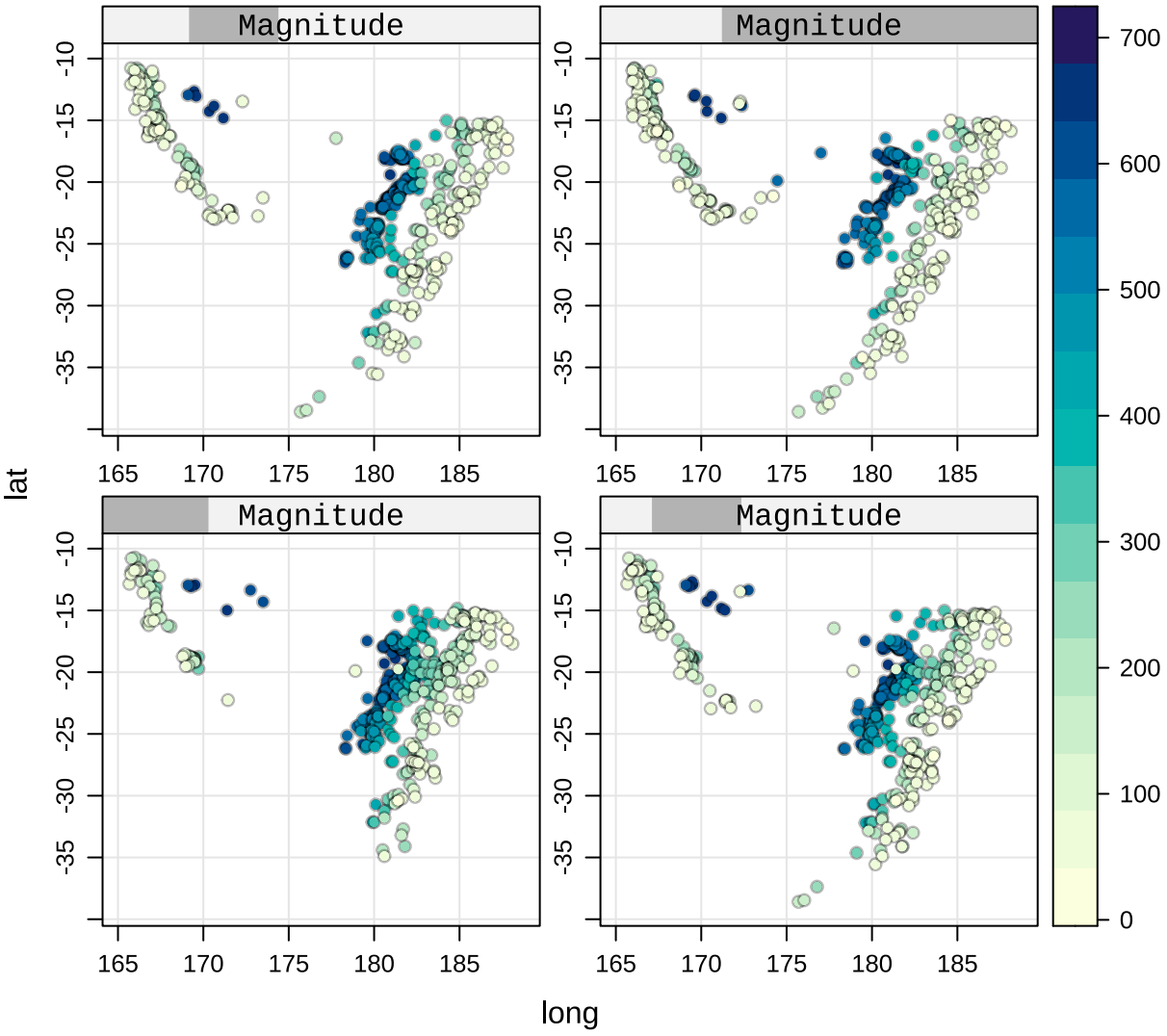


图 10.14: 分面水平图

10.3.8 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(
    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")
  )
)
```

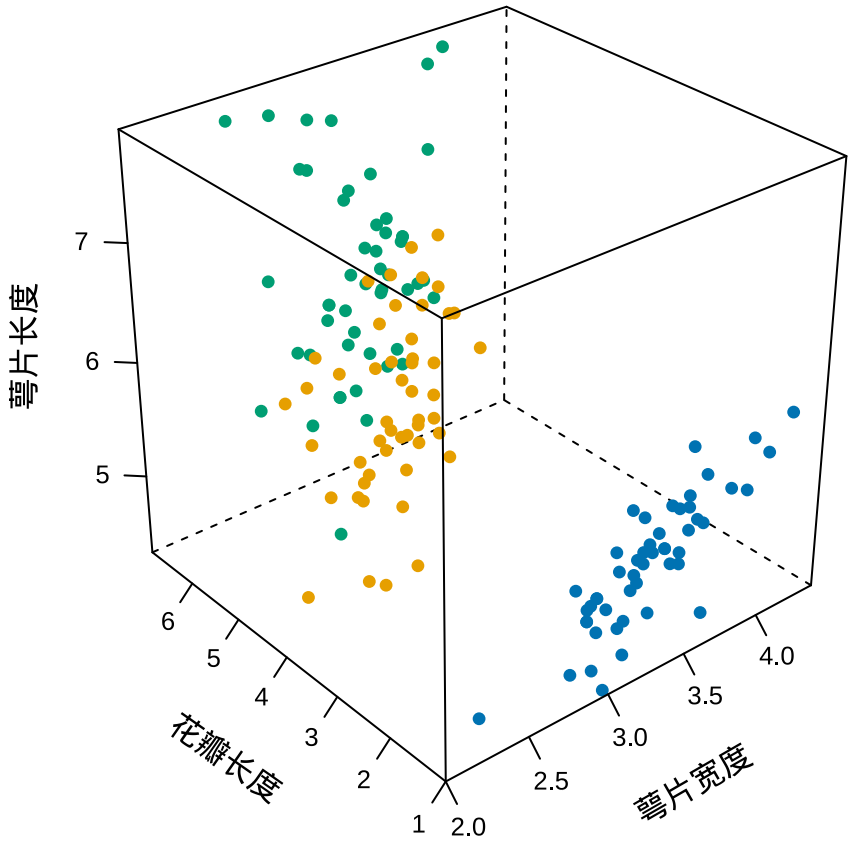


图 10.15: 三维散点图

10.3.9 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 $t=1$ 。

```
# 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)
)
```

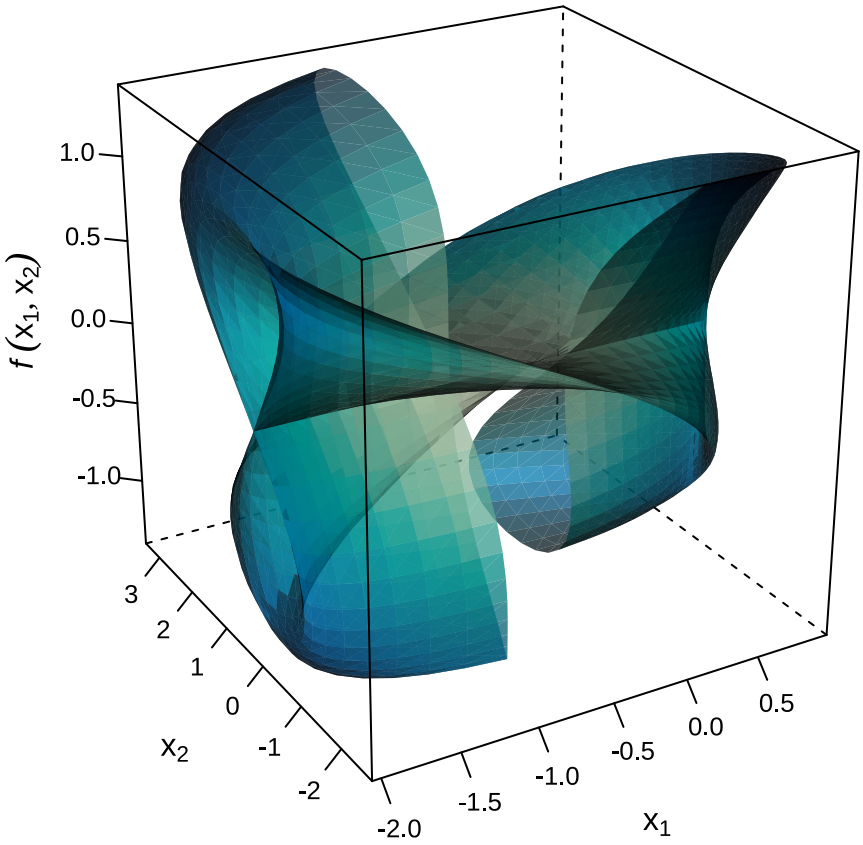


图 10.16: 三维透视图

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")
    )
  ),
  # 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)
)
```

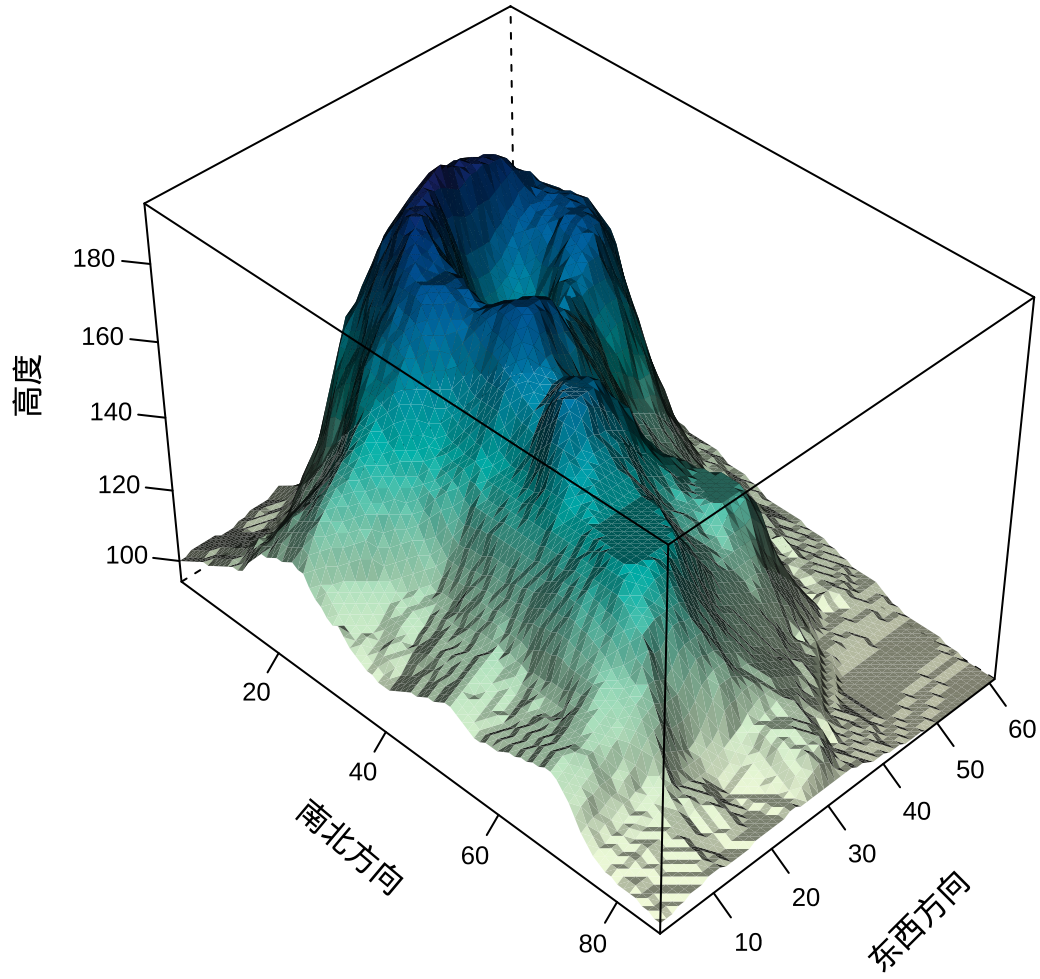


图 10.17: 奥克兰火山地形图

10.3.10 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
  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"
)
```

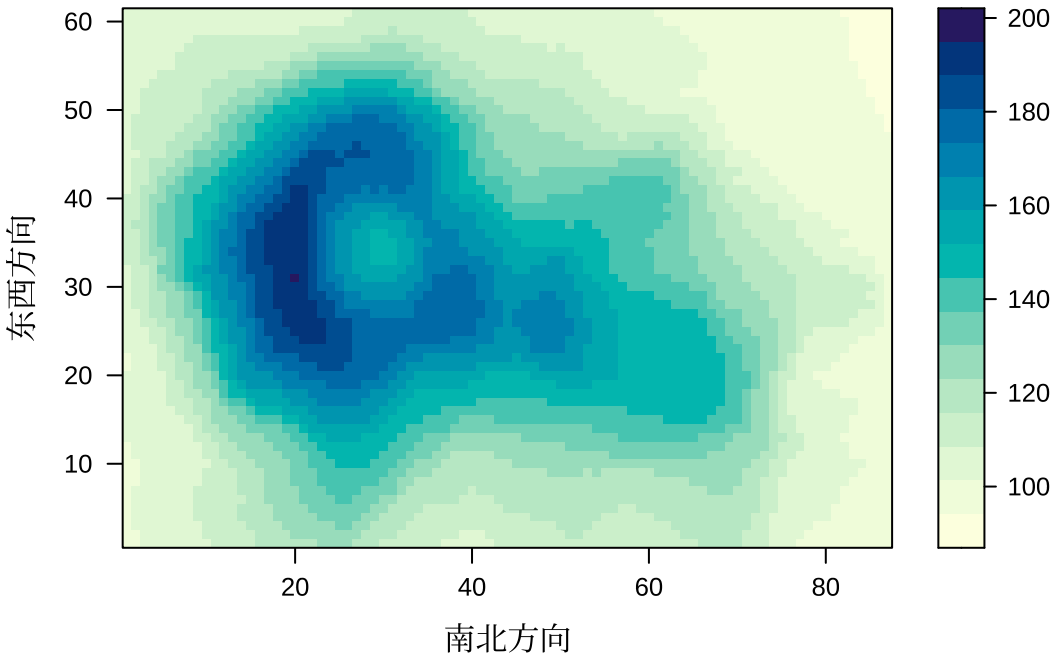


图 10.18: 奥克兰火山的地形轮廓图

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"
)
```

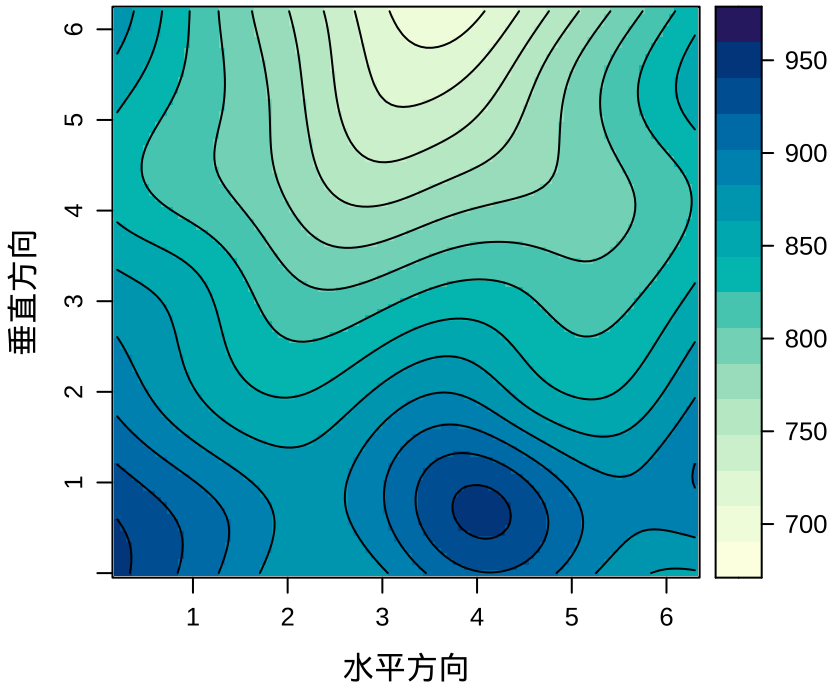


图 10.19: 奥克兰火山的地形轮廓图

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。

10.3.11 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")
# 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 = ""
)
```

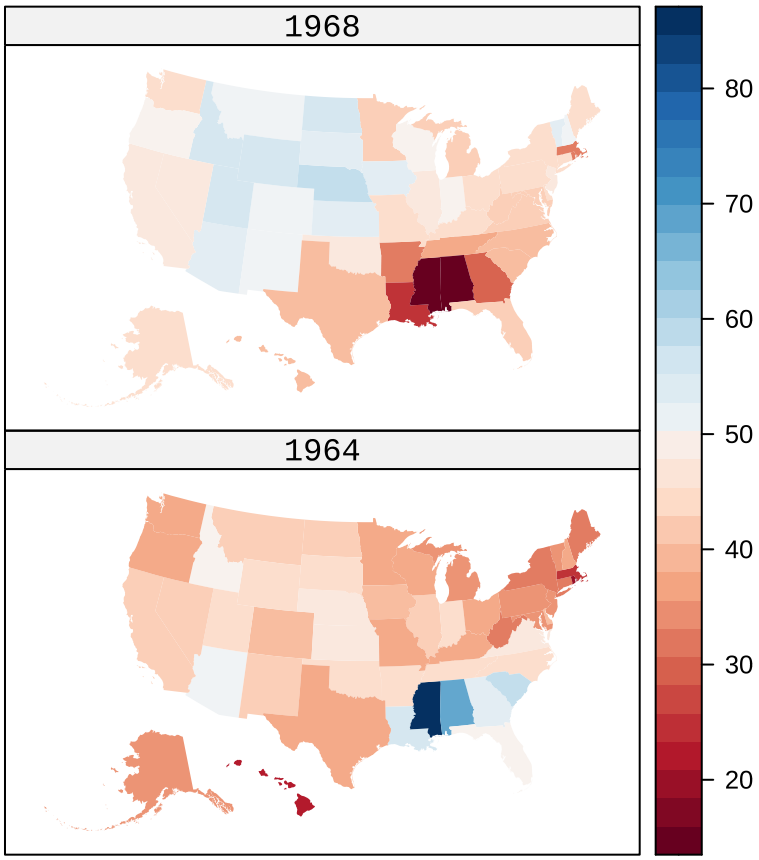


图 10.20: 共和党在各州的得票率

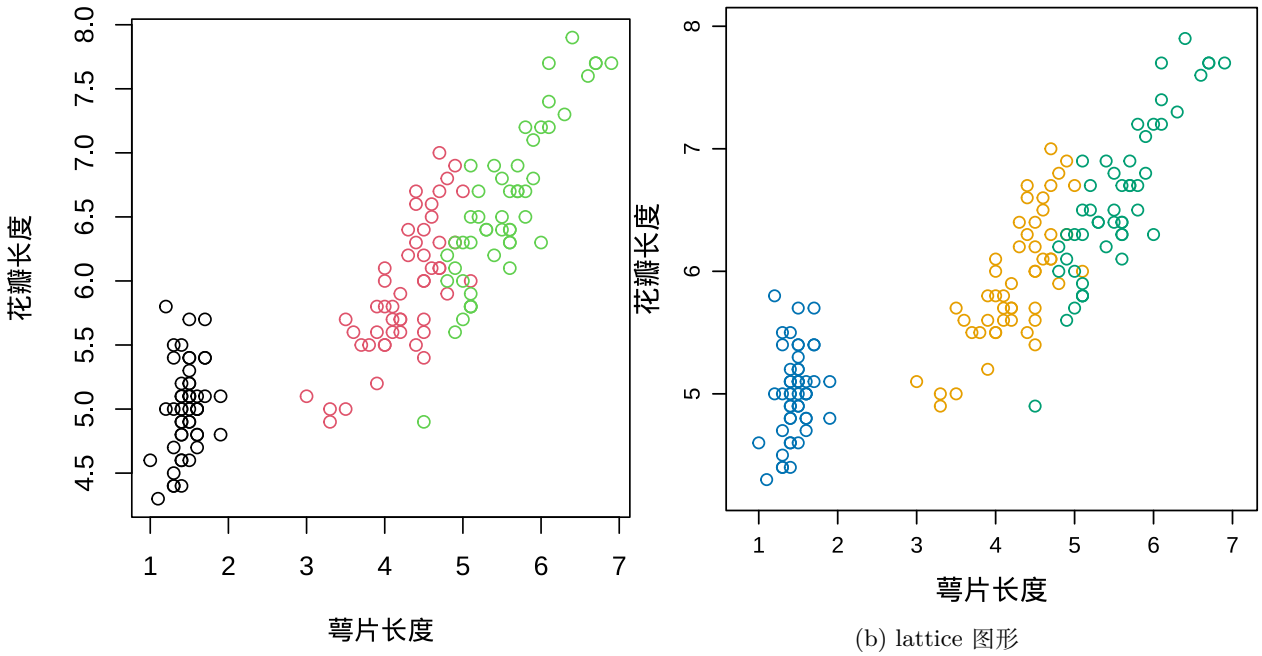
参数 `border` 设置州边界线的颜色，NA 表示去掉边界线。参数 `map` 设置州边界地图数据。参数 `colramp` 设置一个调色板，用于将得票率与调色板上的颜色映射起来。美国历届大选，共和党和民主党竞争总统职位，最终由得票率决定，用红蓝对抗型调色板表现竞争关系。基于 **RColorBrewer** 包的 `RdBu` 调色板，用函数 `colorRampPalette()` 构造一个新的红蓝调色板。参数 `layout` 将多个子图按照一定顺序排列，图中设置 2 行 1 列的多图布局。参数 `scales` 用来调整刻度，设置 `list(draw = FALSE)` 将图中的刻度去掉了。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。

10.4 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,
  scales = "free", xlab = "萼片长度", ylab = "花瓣长度")
```



(a) Base R 图形

(b) lattice 图形

图 10.21: 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 10.1: 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

R 包	函数	图形	作用
lattice	`cloud()`	三维散点图	描述分布
latticeExtra	`panel.smoother()`	回归曲线图	描述趋势
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

第十一章　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。

11.1　绘图基础

利用点、线等基础元素从零开始绘图。

11.1.1　`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。

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
```

(a) versicolor 杂色鸢尾 (b) setosa 山鸢尾 (c) virginica 弗吉尼亚鸢尾

图 11.1: 三种鸢尾花

```
#>  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。

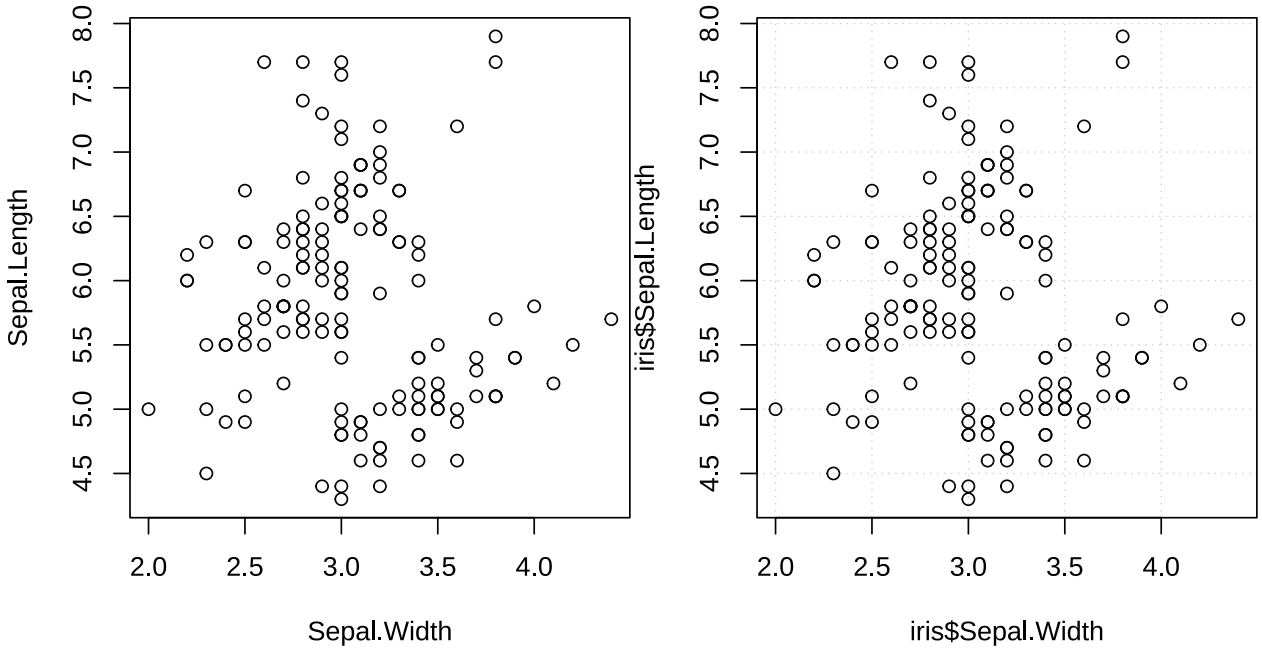
```
plot(Sepal.Length ~ Sepal.Width, data = iris)
plot(iris$Sepal.Width, iris$Sepal.Length, panel.first = grid())
```

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。

11.1.2 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"
)
```



(a) 公式语法绘制散点图

(b) 带背景参考线的散点图

图 11.2: 快速作图函数 `plot()`

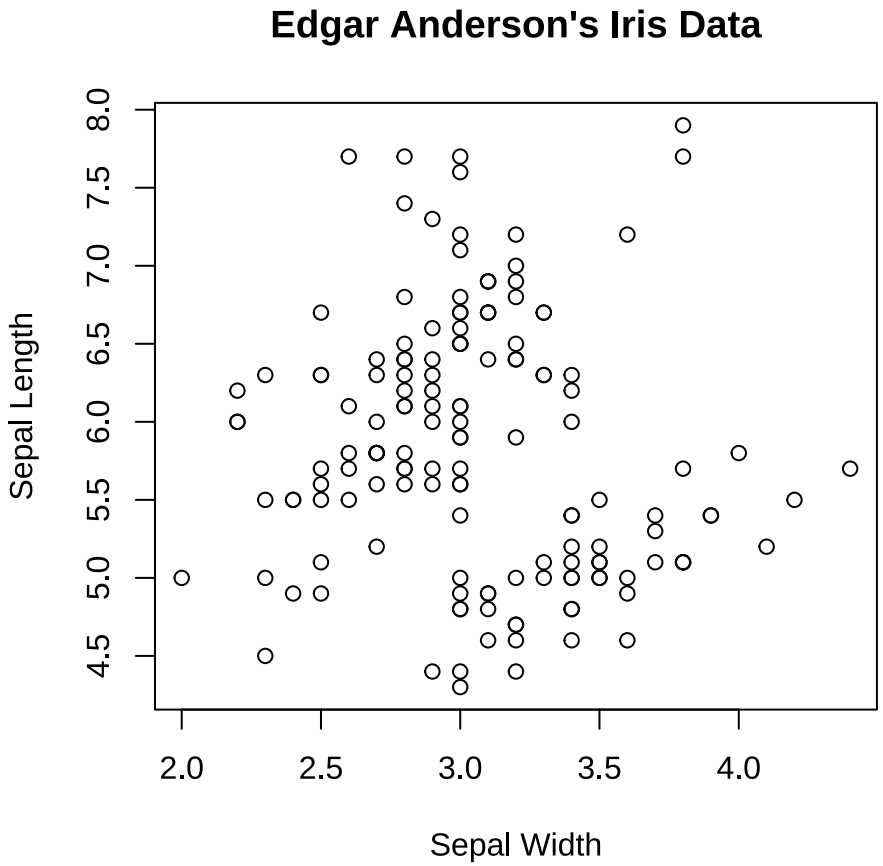


图 11.3: 标签

11.1.3 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 11.4 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"
)
```

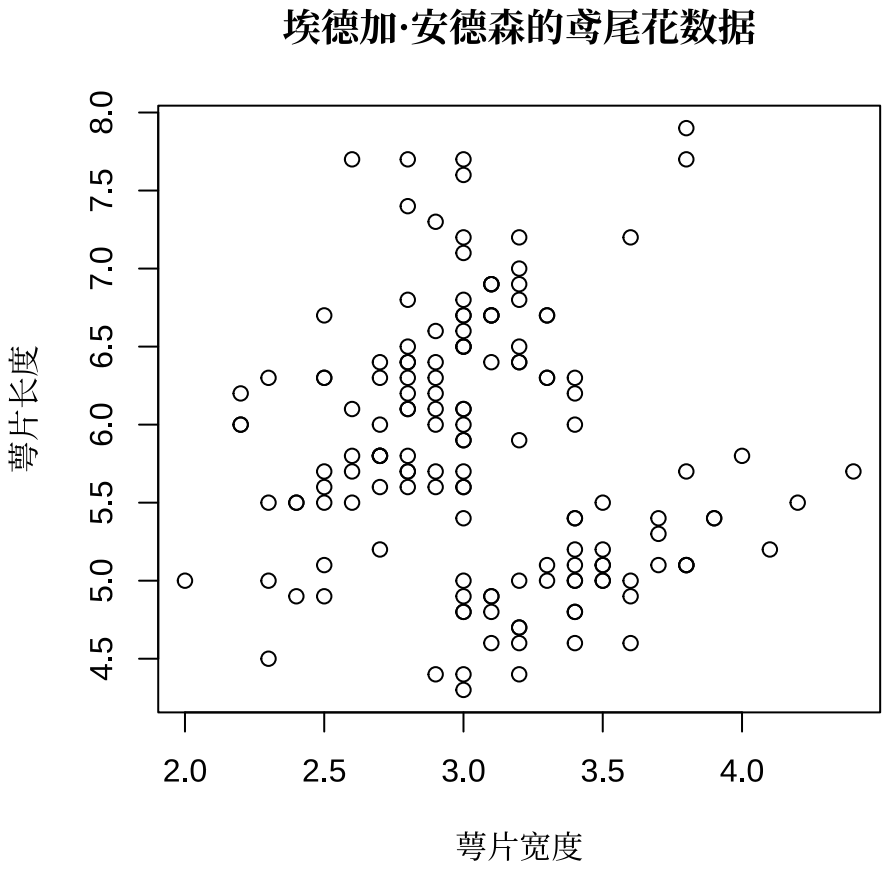


图 11.4: 字体

11.1.4 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)
```

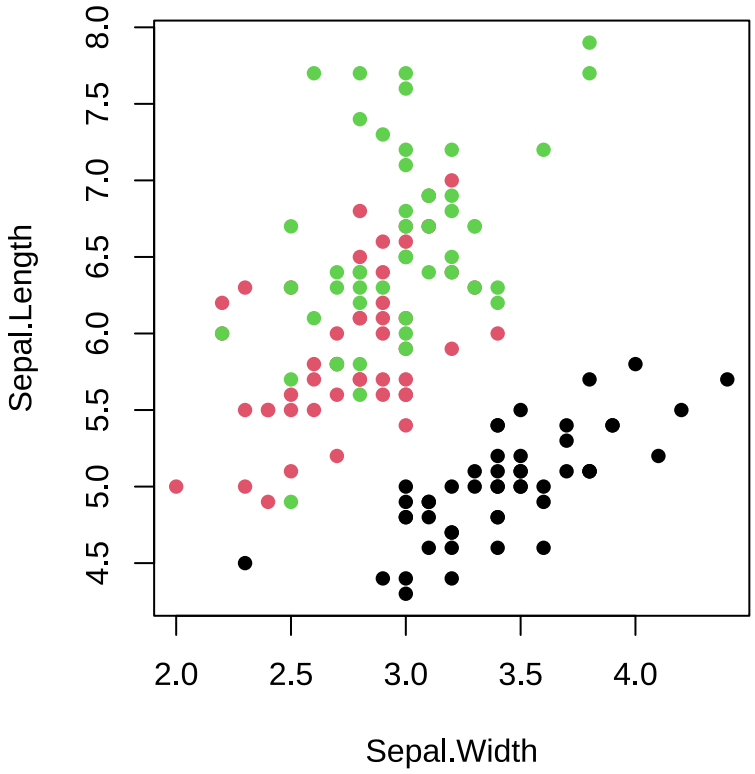


图 11.5: 分组

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
)
```

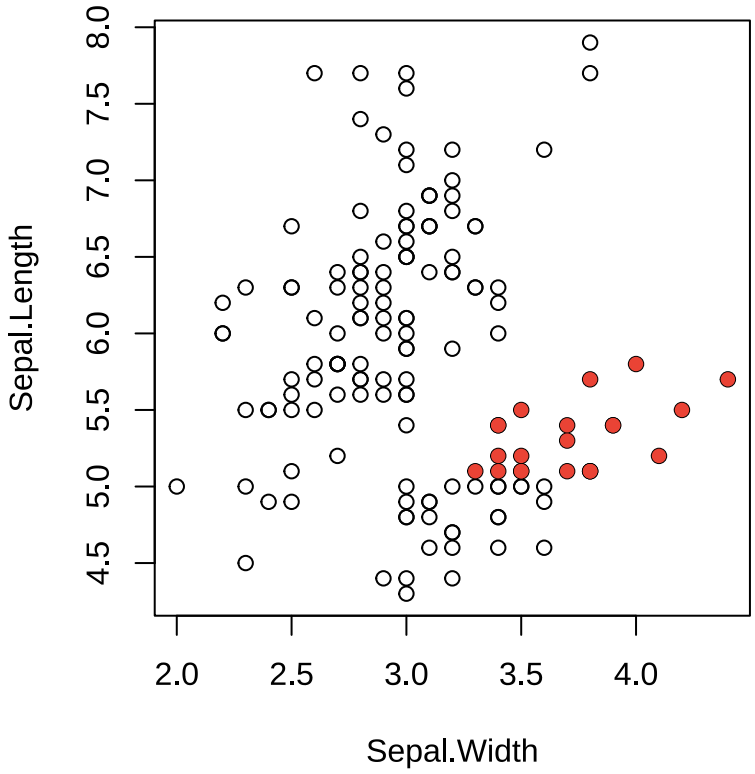


图 11.6: 分组

11.1.5 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 11.7 。

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 11.5 变成图 11.8 。

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

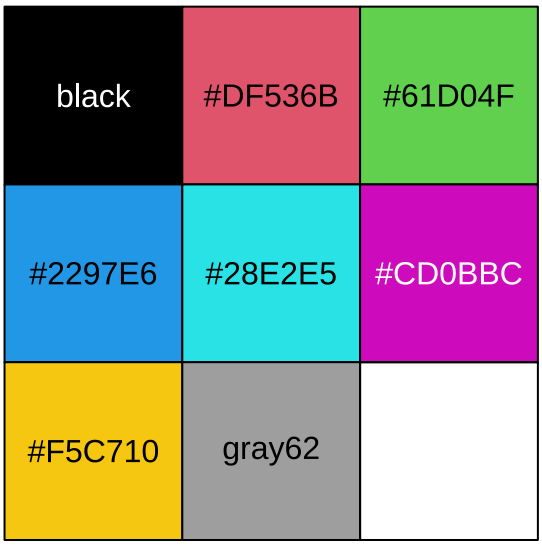


图 11.7: 默认调色板

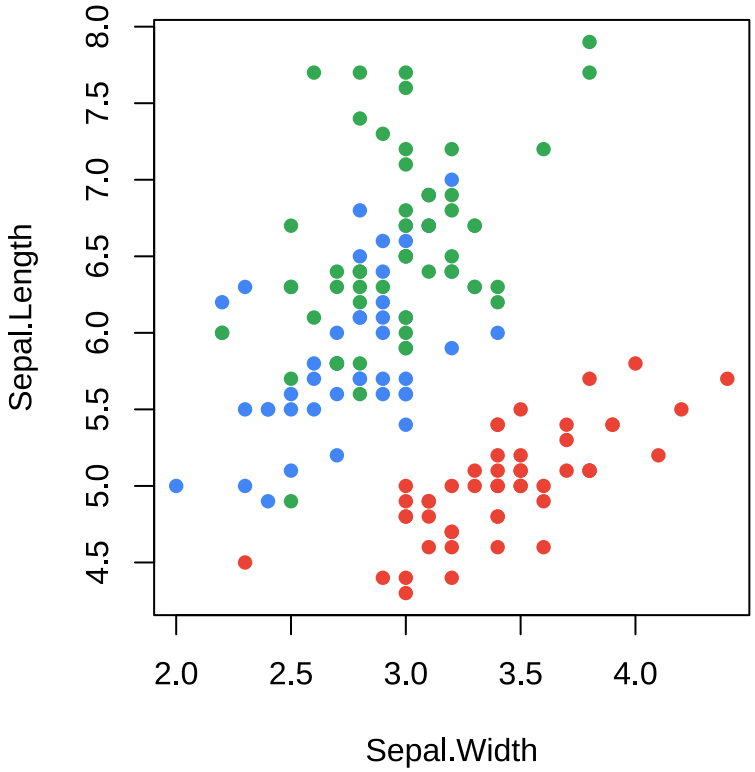


图 11.8: 配色

11.1.6 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。

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

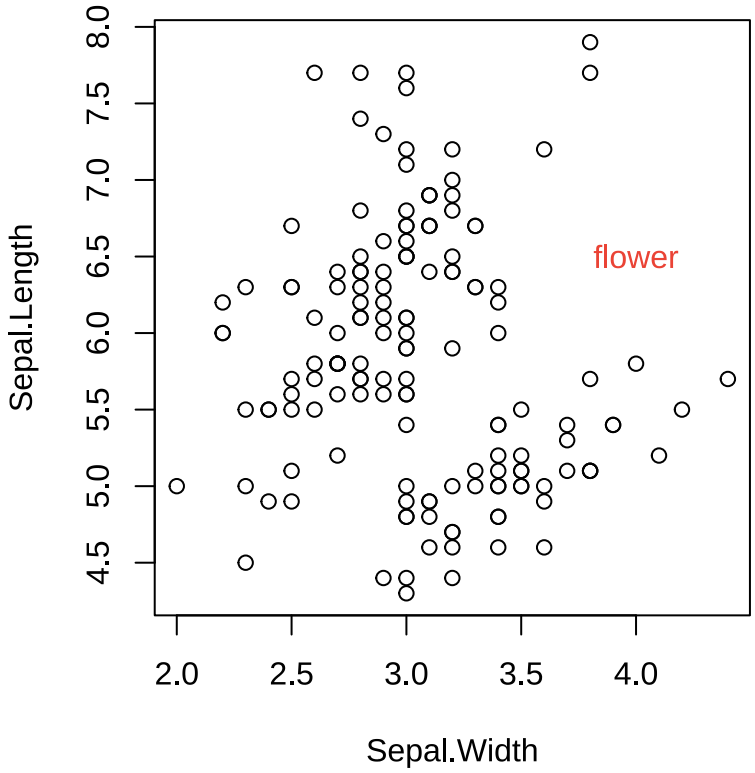


图 11.9: 注释

11.1.7 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")
)
```

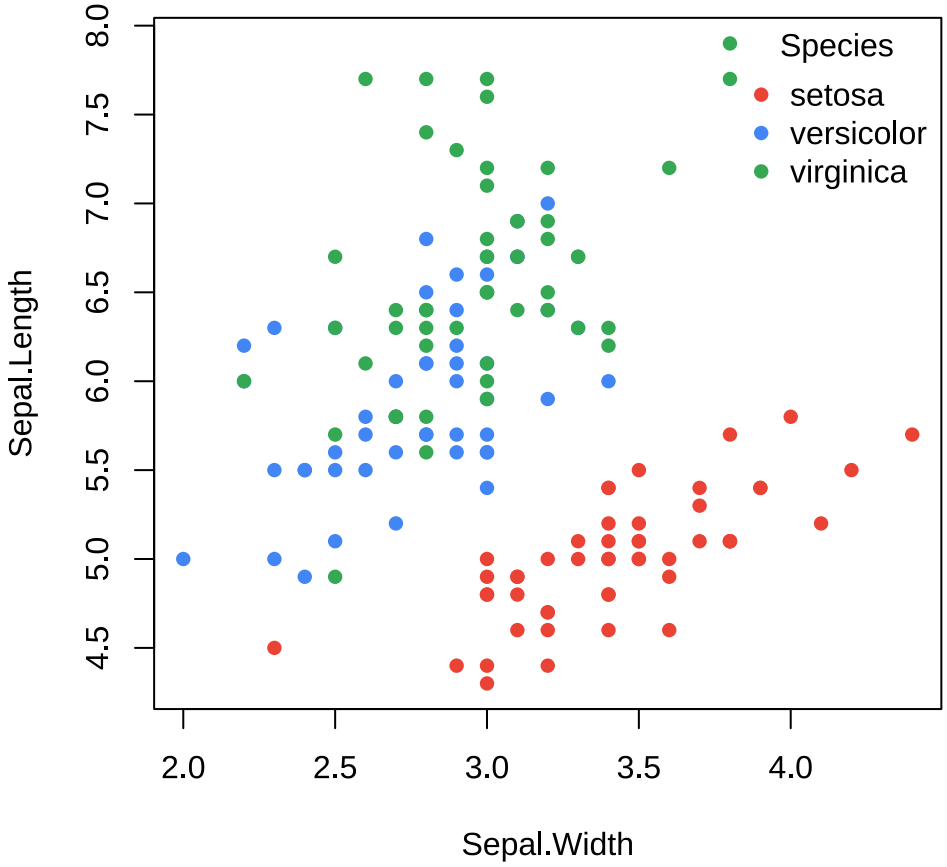


图 11.10: 图例

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)
```

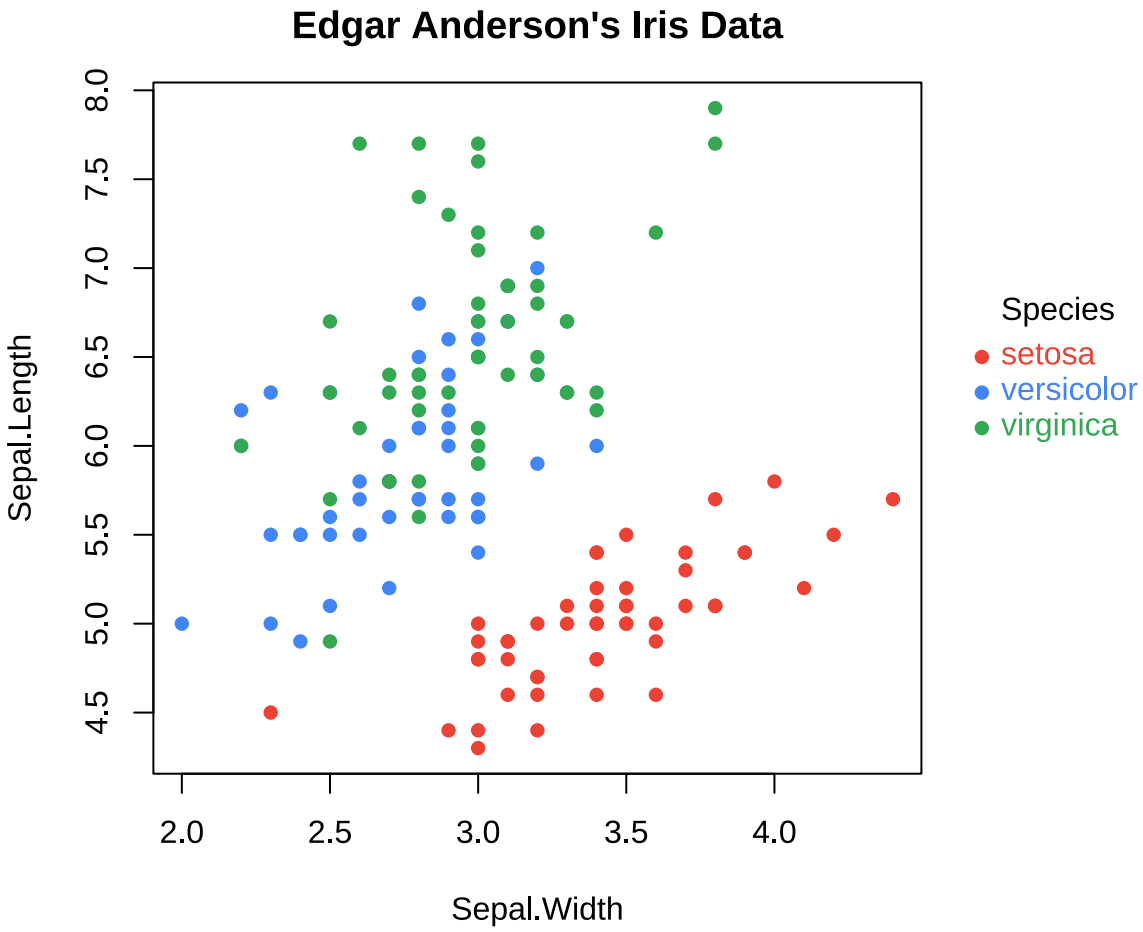


图 11.11: 图例

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")
)
```

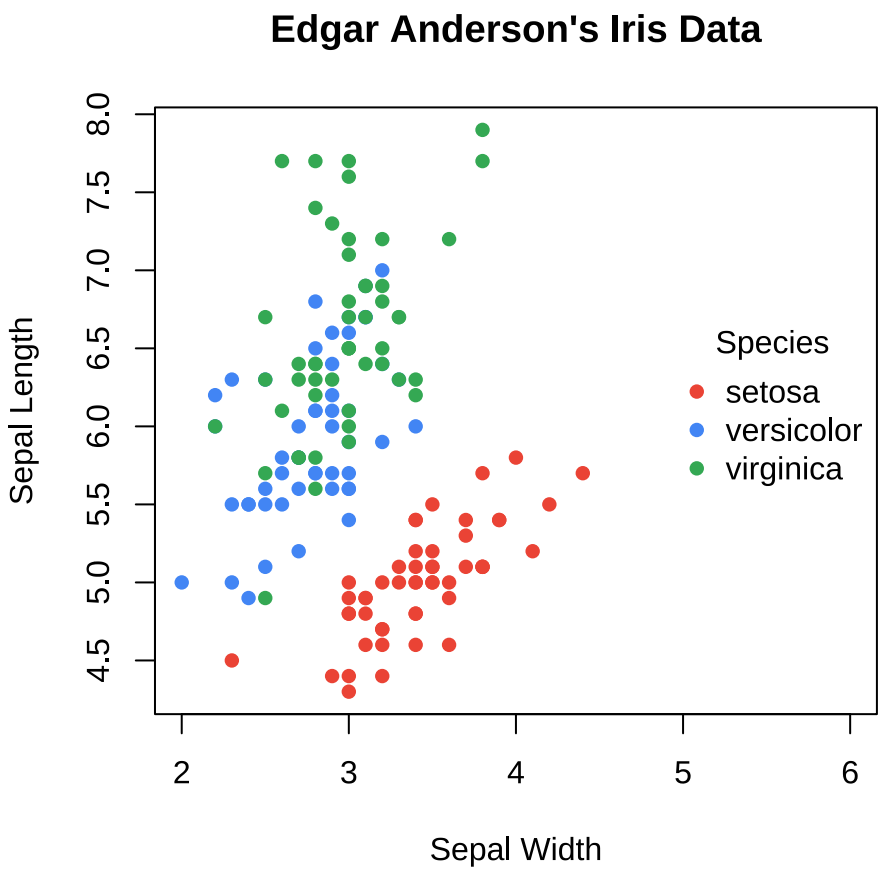


图 11.12: 图例

11.1.8 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
```

```
#>
#> $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。

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

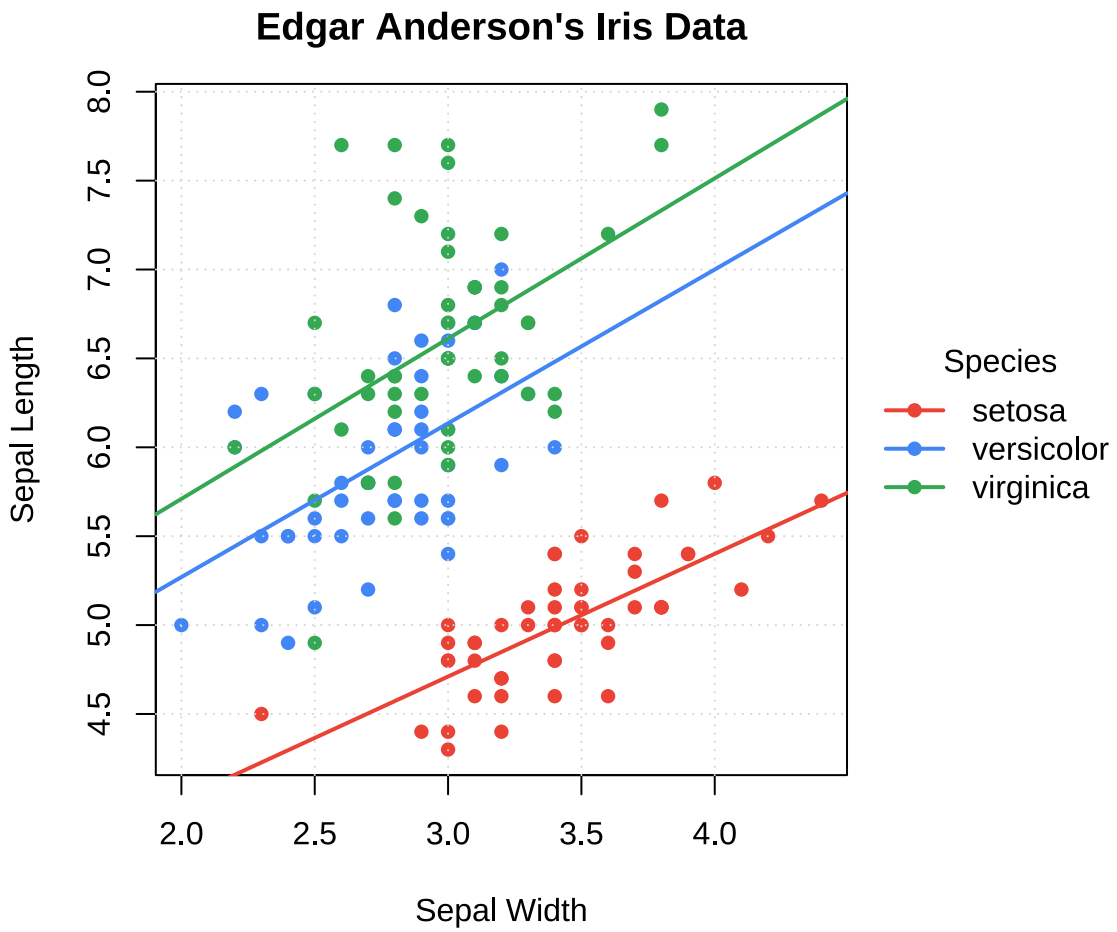


图 11.13: 分组线性回归

11.2 绘图进阶

11.2.1 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)
```

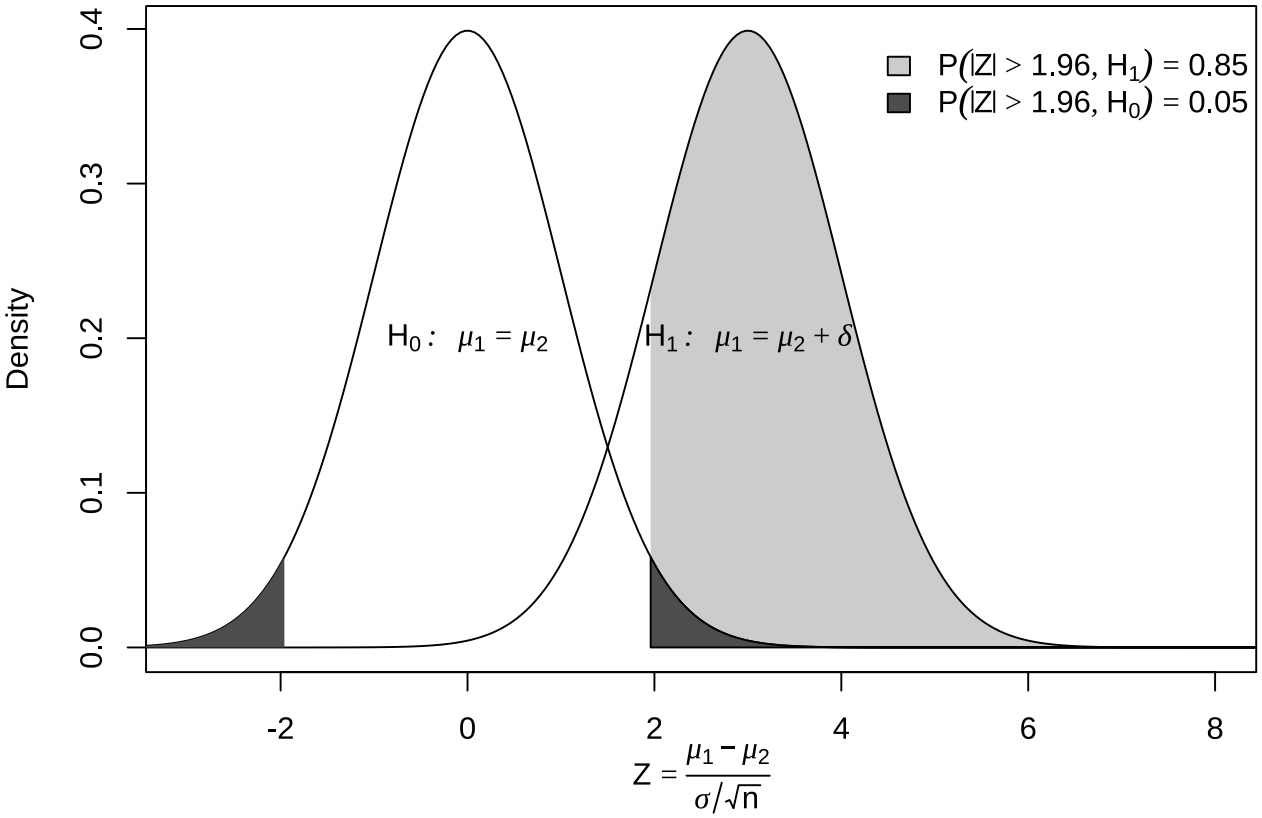


图 11.14: 正态总体下两样本均值之差的检验

11.2.2 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(
    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)
```

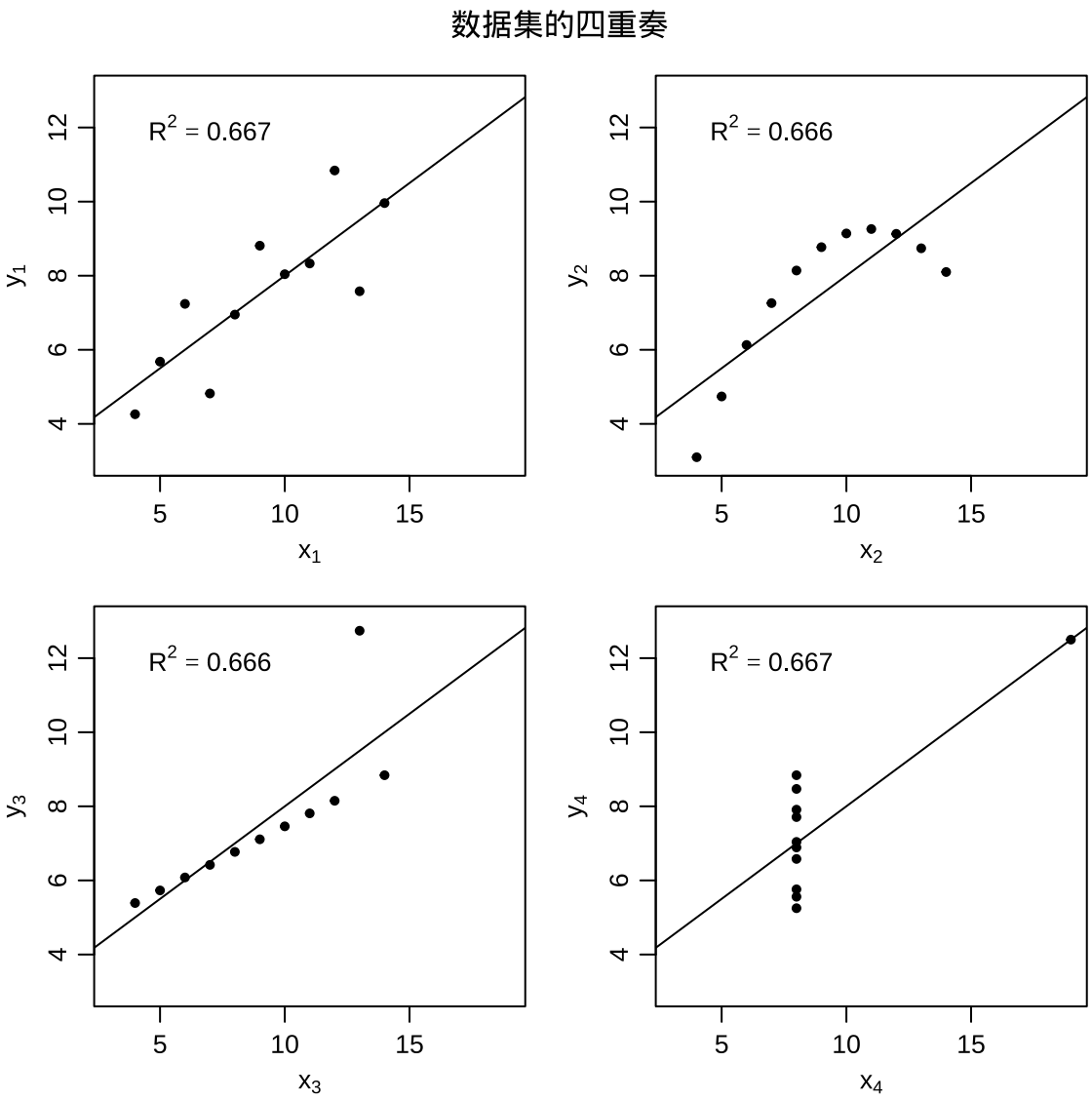


图 11.15: 数据可视化很重要

11.3 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。

11.3.1 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 = "$y$")
```

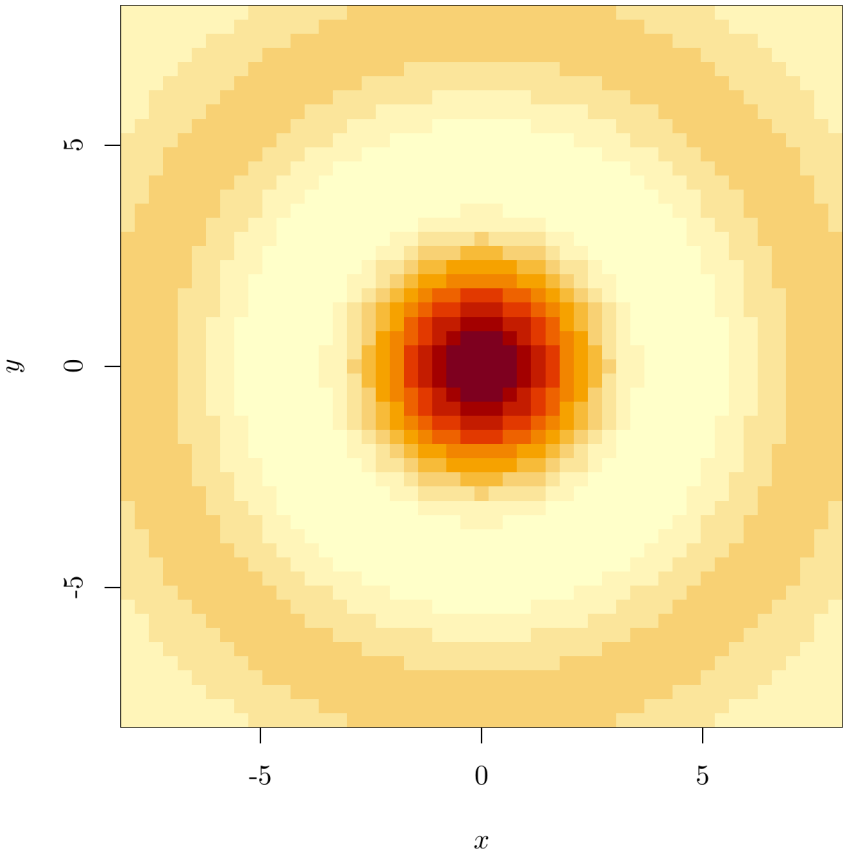


图 11.16: 颜色图

11.3.2 透视图

函数 persp() 绘制透视图

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

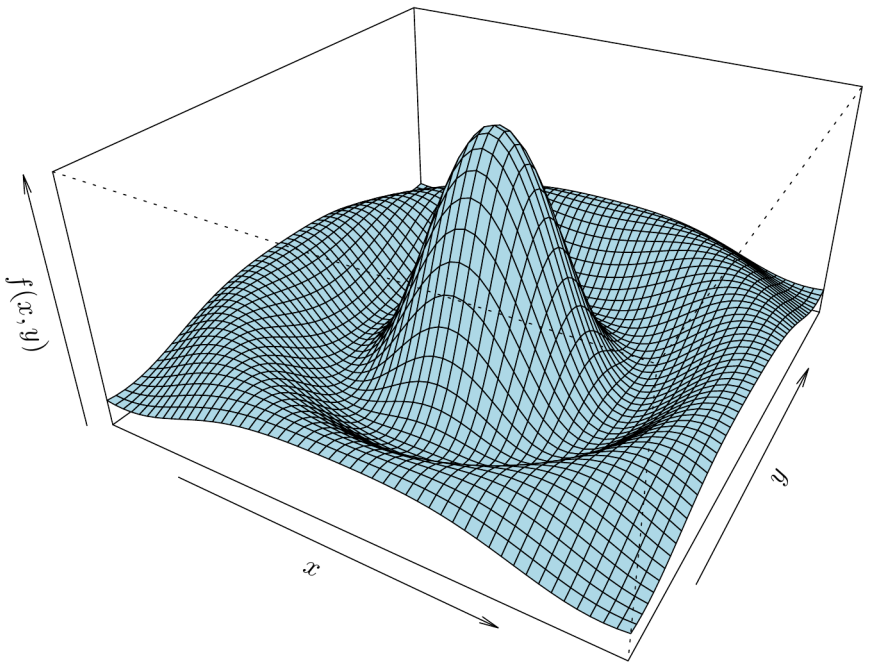


图 11.17: 透视图

11.3.3 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 = "$y$")
```

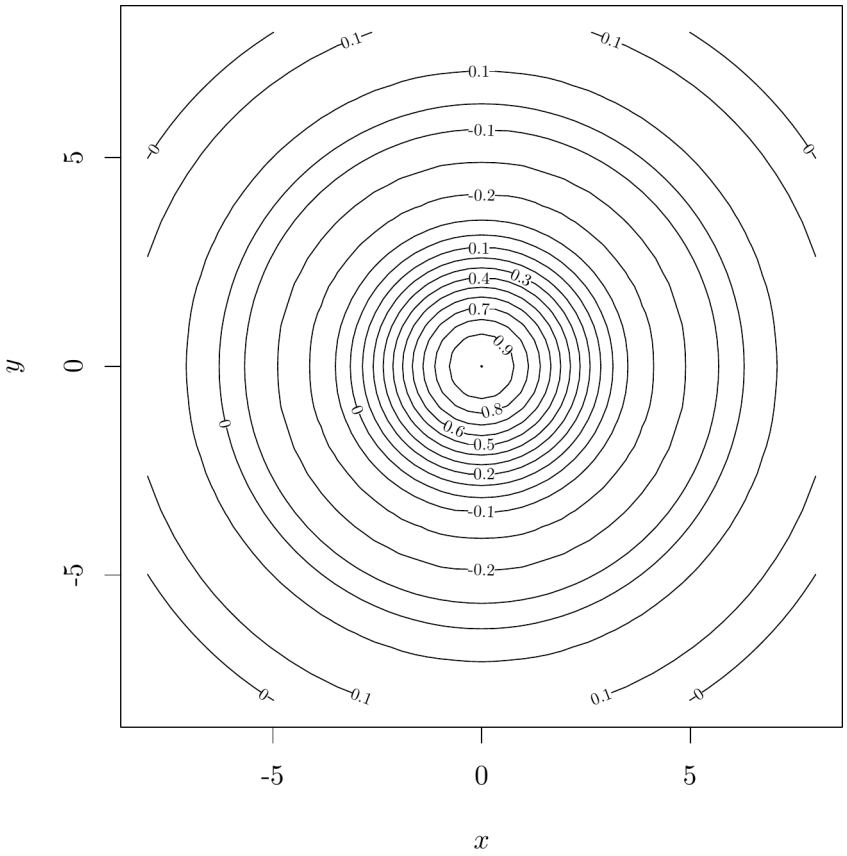


图 11.18: 等值线图

11.3.4 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)
  },
  key.axes = {
    axis(4, seq(-0.2, 1, length.out = 9))
  }
)
```

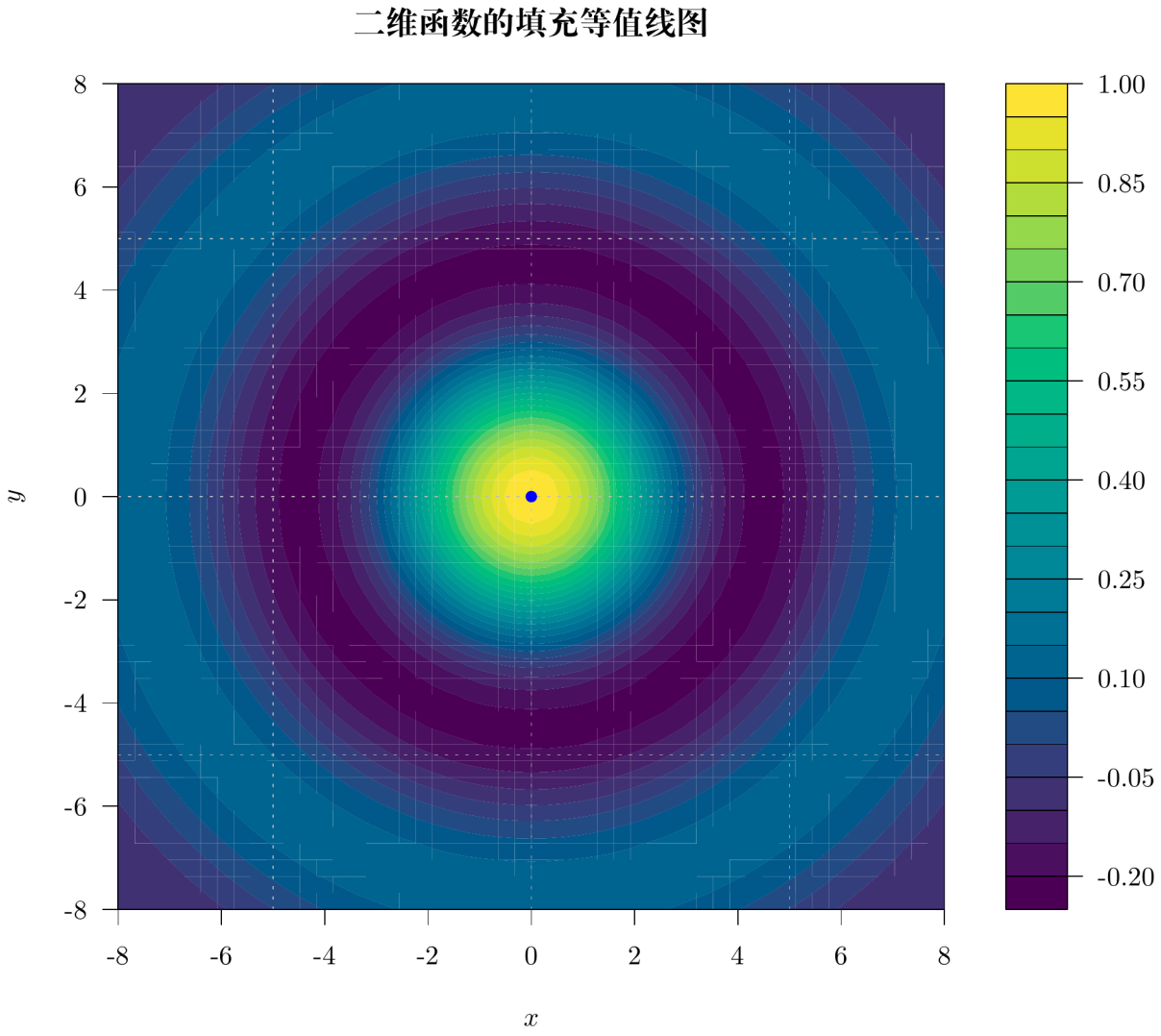


图 11.19: 填充等值线图

11.4 总结

11.4.1 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)
```

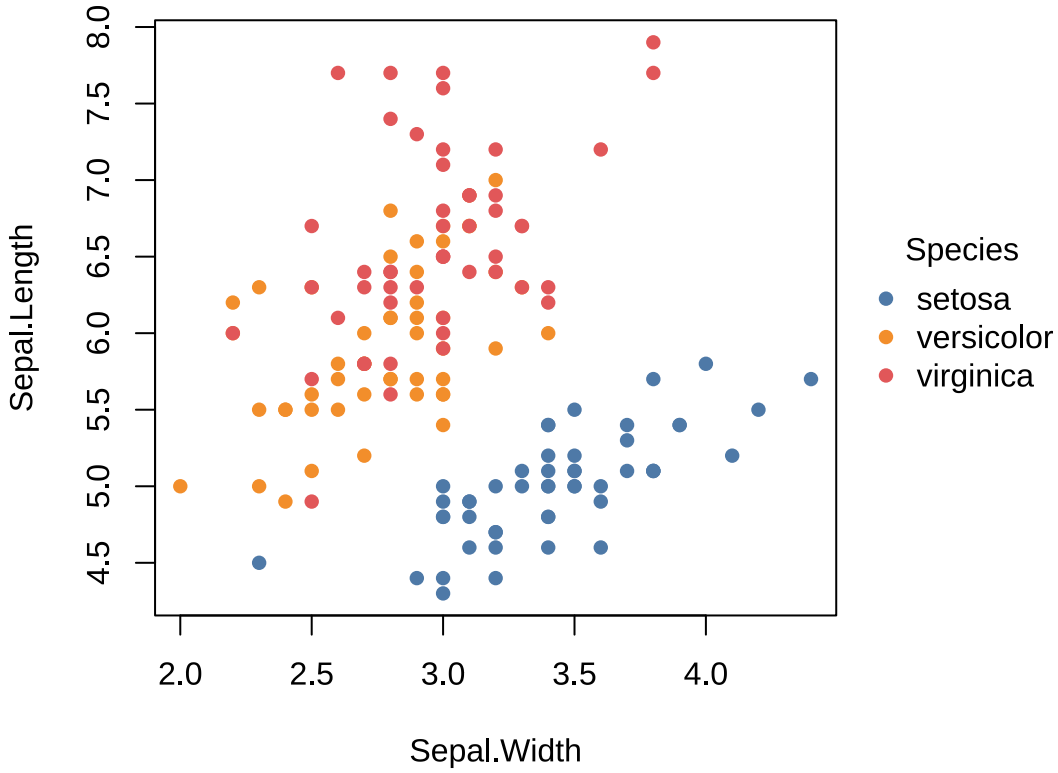


图 11.20: plot2 包绘制分组散点图

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)
```

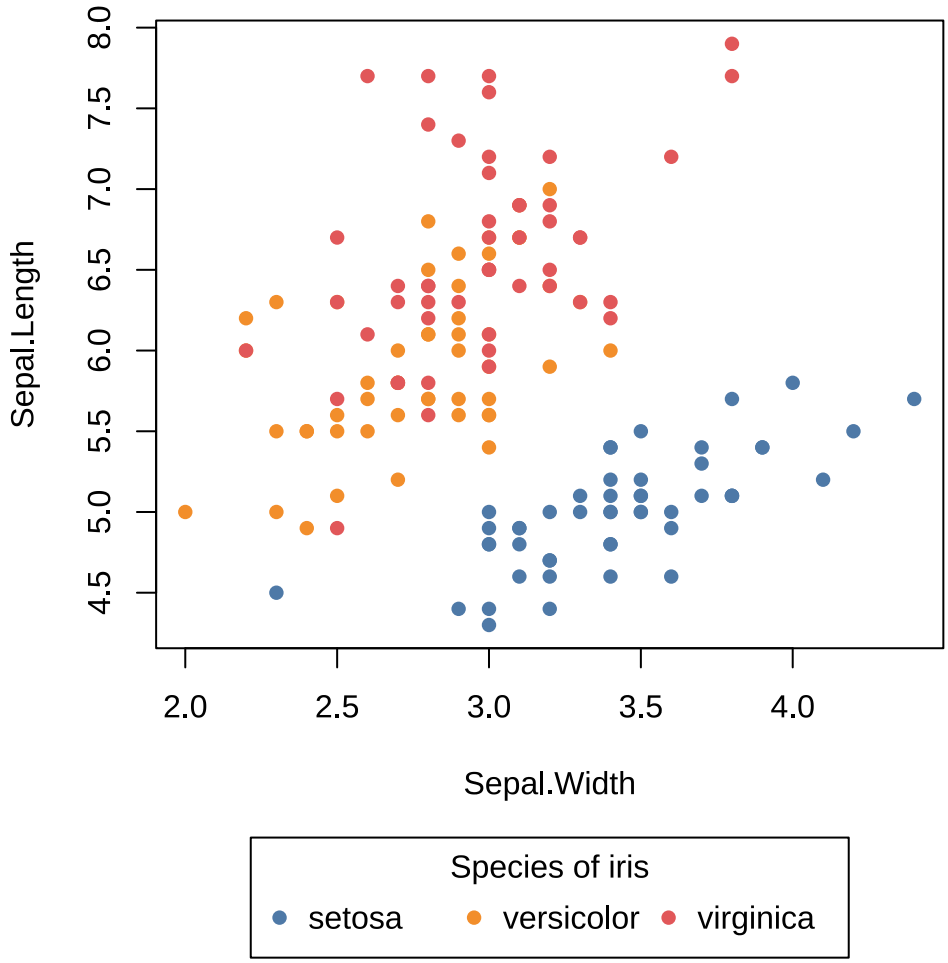


图 11.21: plot2 包调整图例位置

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
))
```

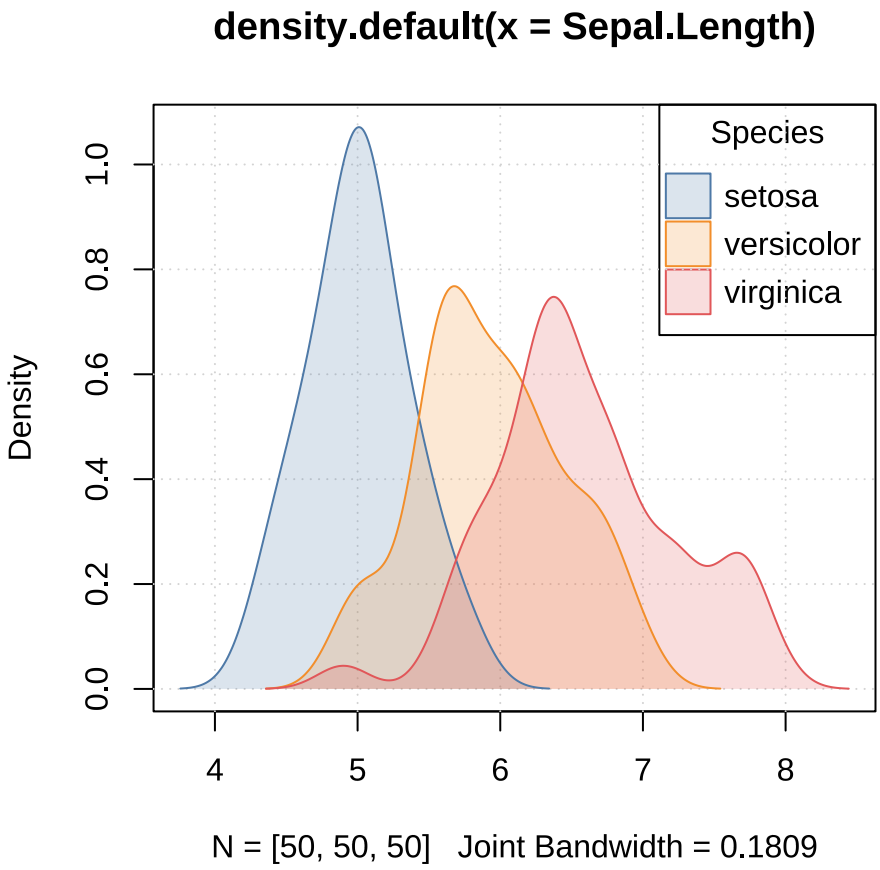


图 11.22: plot2 包绘制密度曲线图

11.4.2 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
  )
)
```

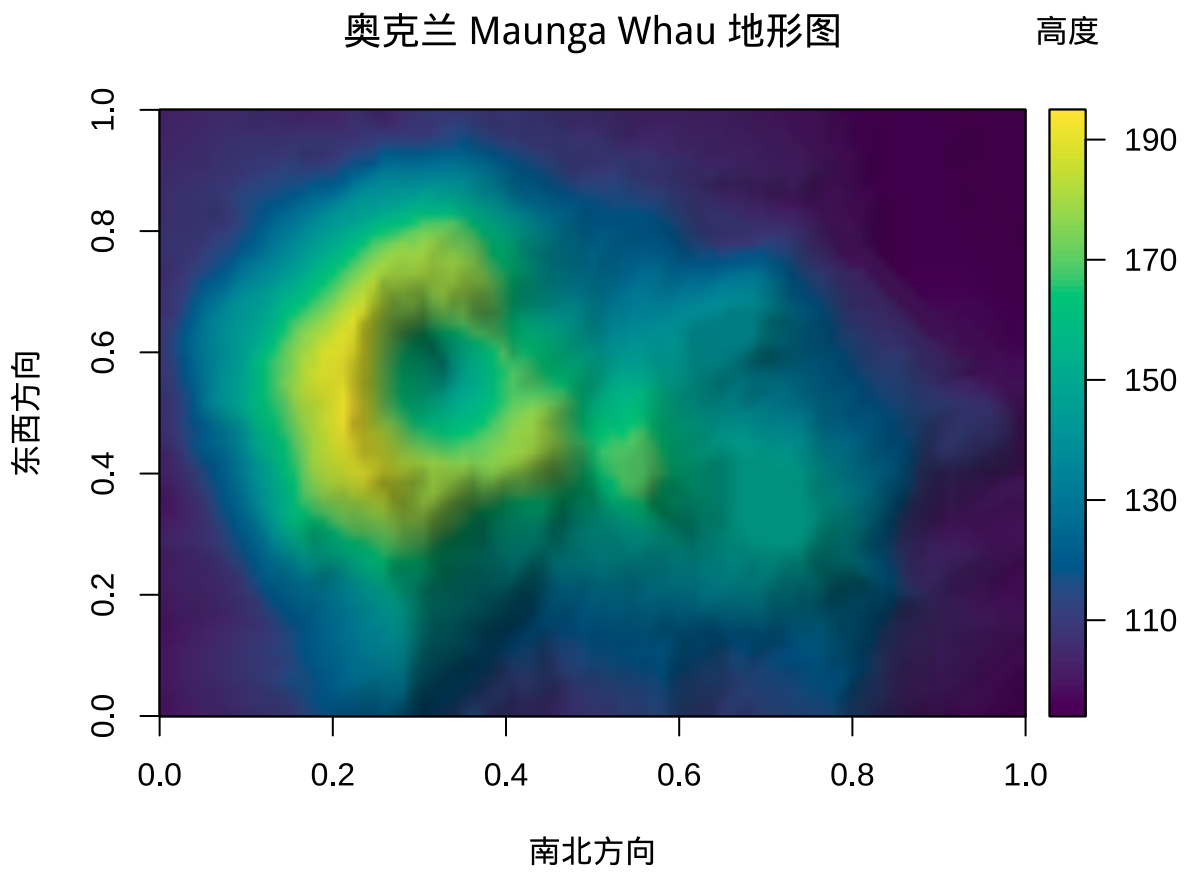


图 11.23: 奥克兰火山地形图

```
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)
```

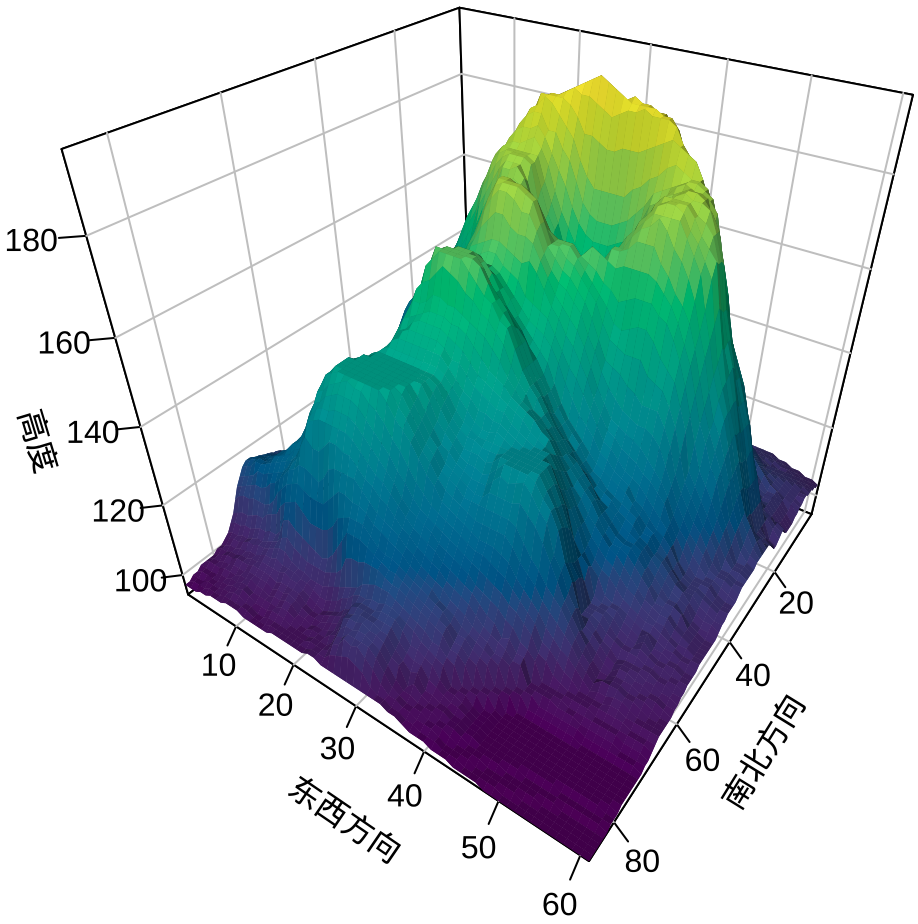


图 11.24: 奥克兰火山地形图

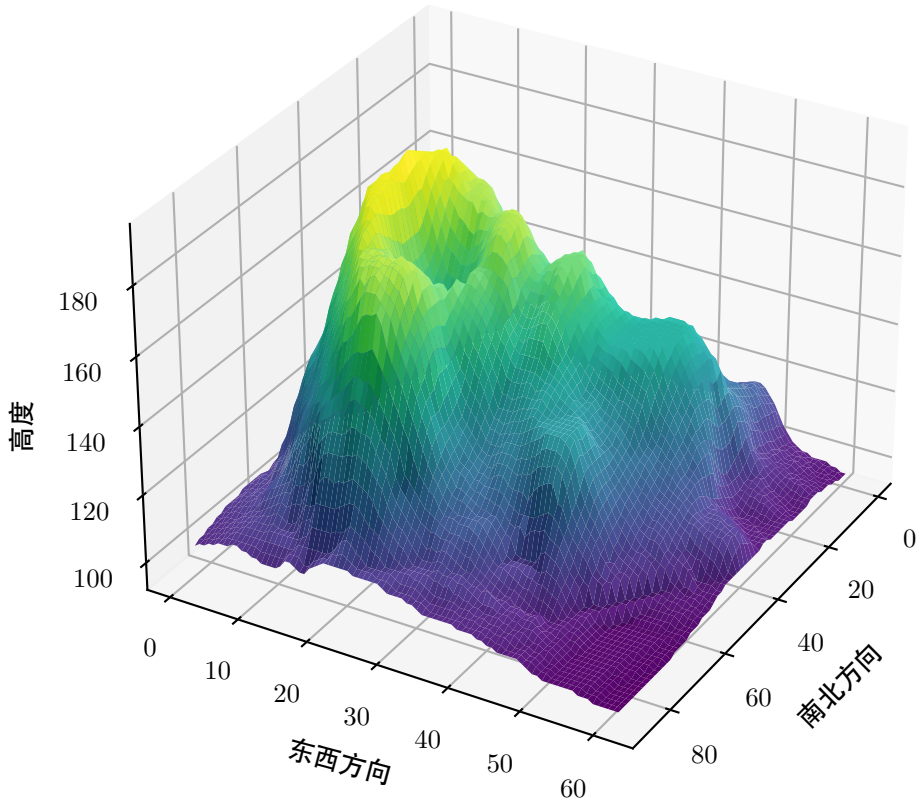


图 11.25: matplotlib 绘制三维透视图

第十二章　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。

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！三维图图 12.6 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。

12.1　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) -- (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 12.1 所示。

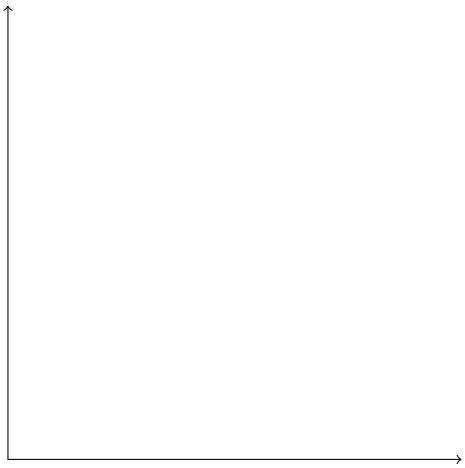


图 12.1: TikZ 绘图

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。
- `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 12.2 所示。

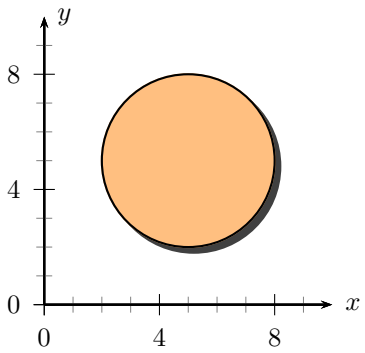


图 12.2: PSTricks 绘图

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 12.2 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 12.2 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。

12.2 PGF 宏包

TikZ 是 TikZ ist kein Zeichenprogramm 的简写，命名有 Linux 哲学意味。提供一套易于学习和使用的绘图语法，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) -- (0,6);
\end{tikzpicture}
```

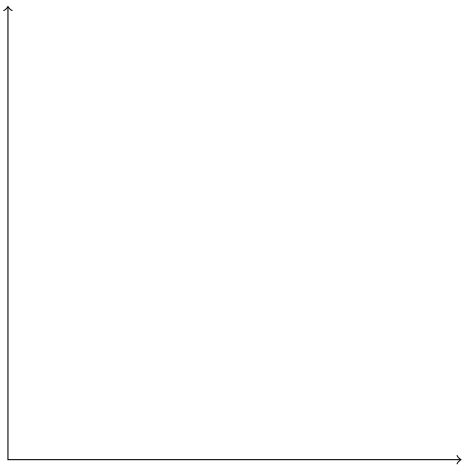


图 12.3: PGF 绘制曲线图

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) -- (0,6) node[left]{$V(q)$};
\end{tikzpicture}
```

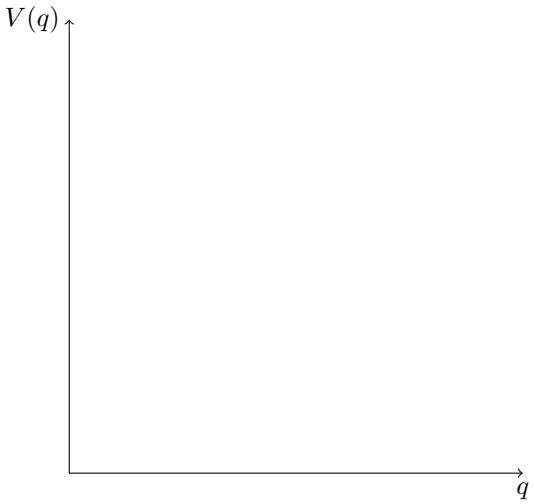


图 12.4: PGF 绘制曲线图

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

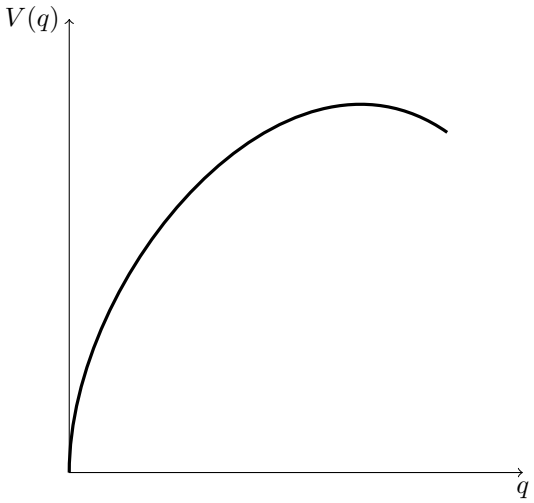


图 12.5: PGF 绘制曲线图

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。

12.3 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=1.18}
\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=1.18` 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}
```

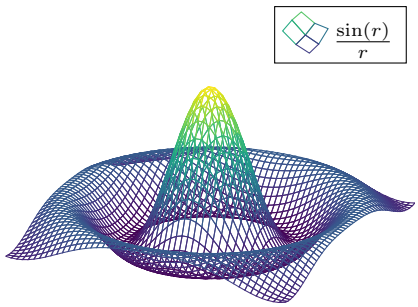


图 12.6: TikZ 绘制三维图 viridis 调色板

```
\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}
```

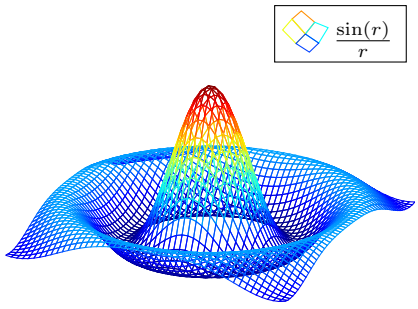


图 12.7: TikZ 绘制三维图 jet 调色板

```
\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}
```

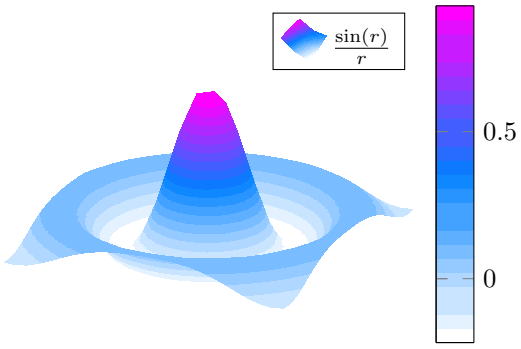


图 12.8: TikZ 绘制三维图 cool 调色板

- `\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。

12.4 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}
```

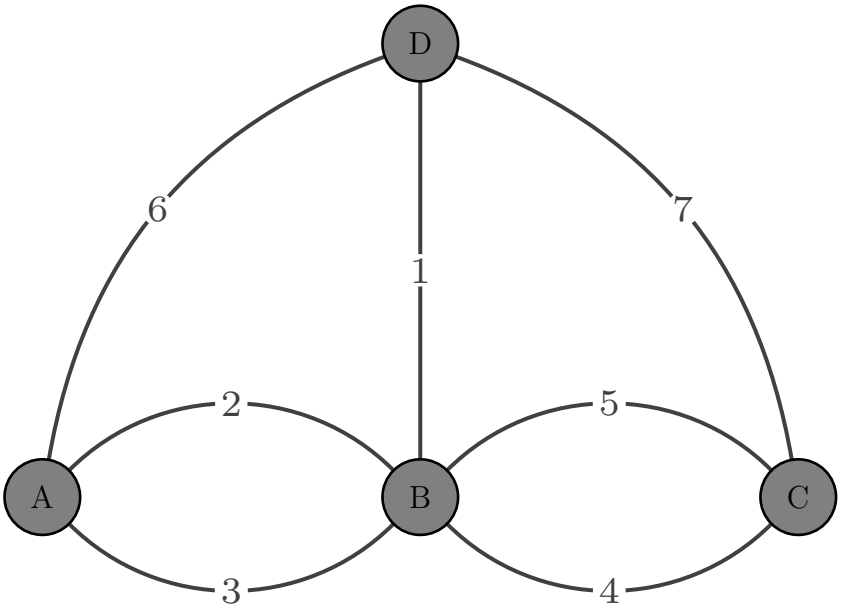


图 12.9: 柯尼斯堡七桥

- `\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。

12.5 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 12.10 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},
    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}}
    };
\end{tikzpicture}
```

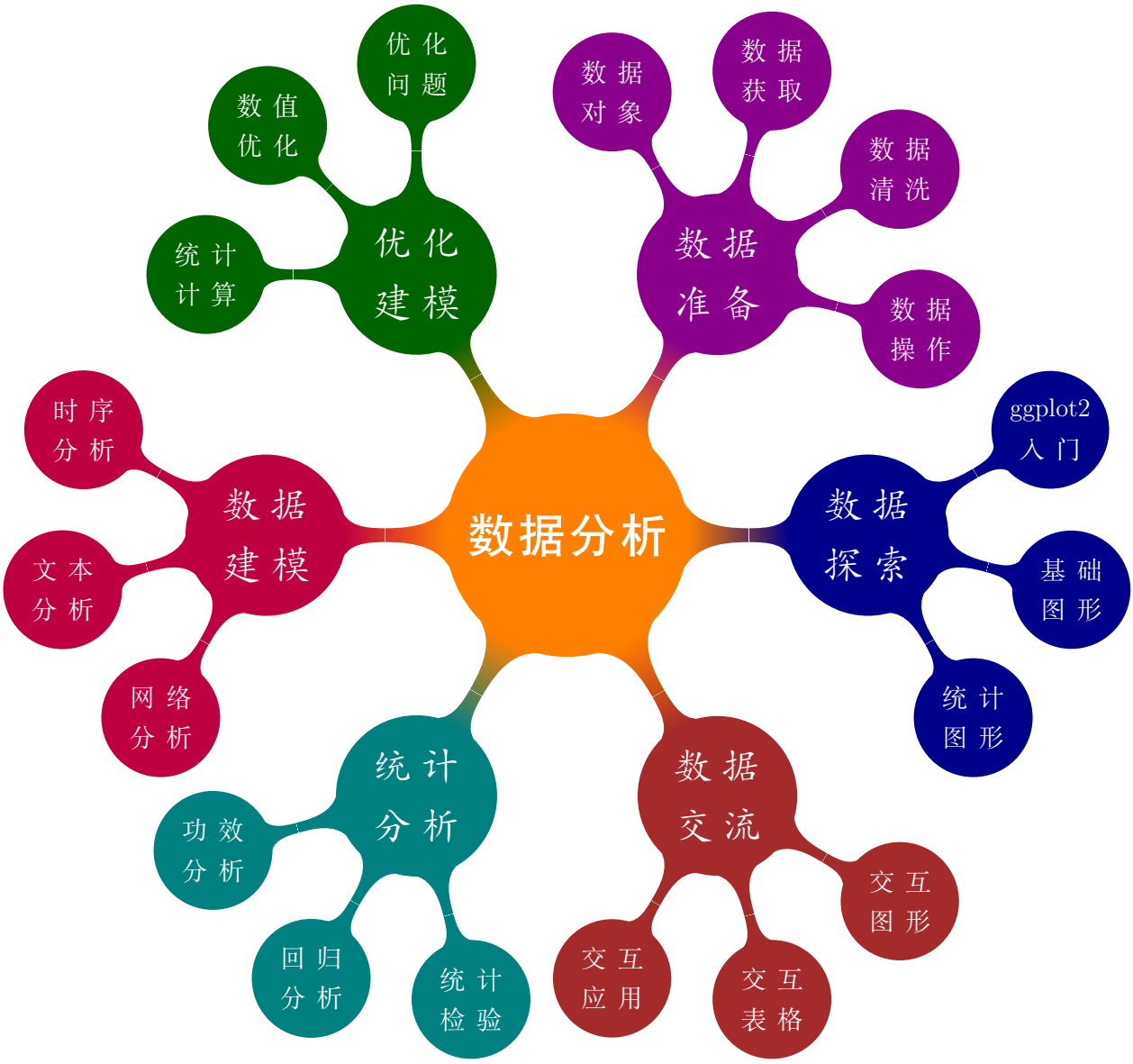


图 12.10: TikZ 绘制思维导图

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] {
      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」。

12.6 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)
}
```

图 12.11: 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}},
}
```

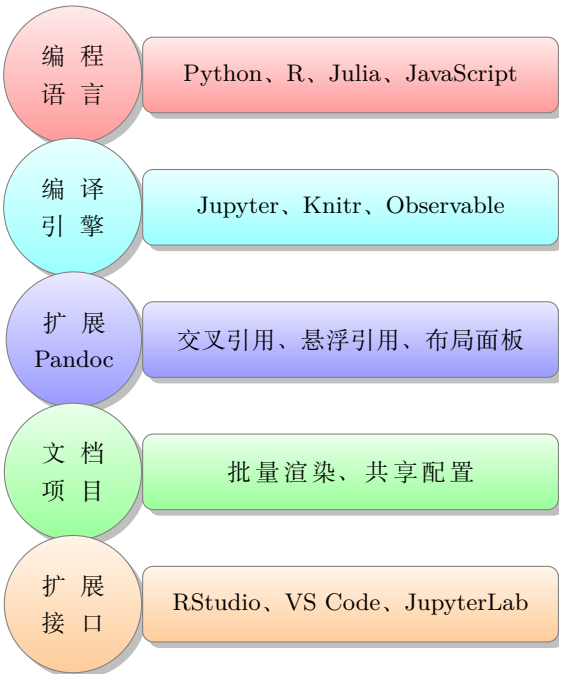


图 12.12: 描述图

第三部分

数据交流

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。

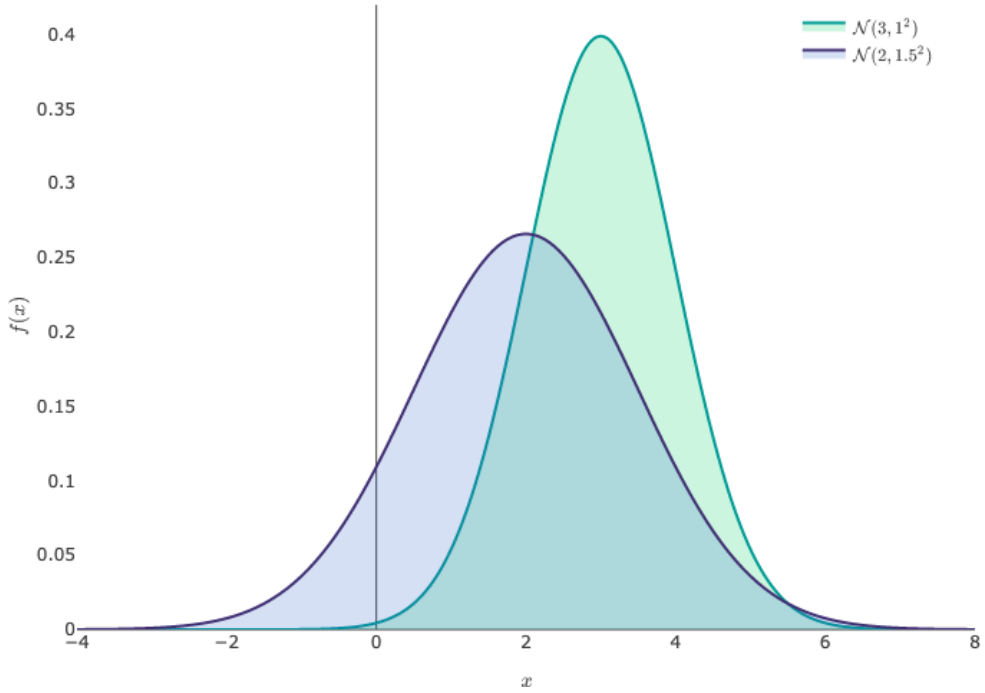


图 13.1: 默认风格的简单散点图

```
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)
```

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"))
  )
) |>
  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(
    xaxis = list(title = "经度"),
    yaxis = list(title = "纬度")
  )
```

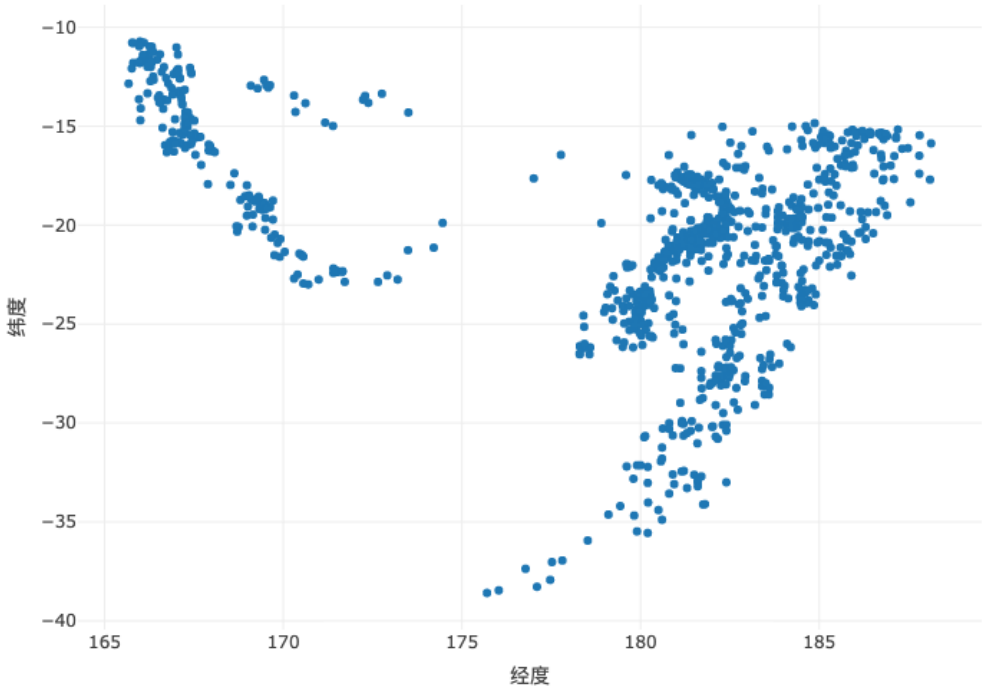


图 13.2: 普通散点图

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")
) |>
  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")
  )
```

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",
  marker = list(
    color = "lightblue",
    line = list(color = "white", width = 2)
  )
) |>
  plotly::layout(
    xaxis = list(title = "震级"),
    yaxis = list(title = "频率")
  )
```

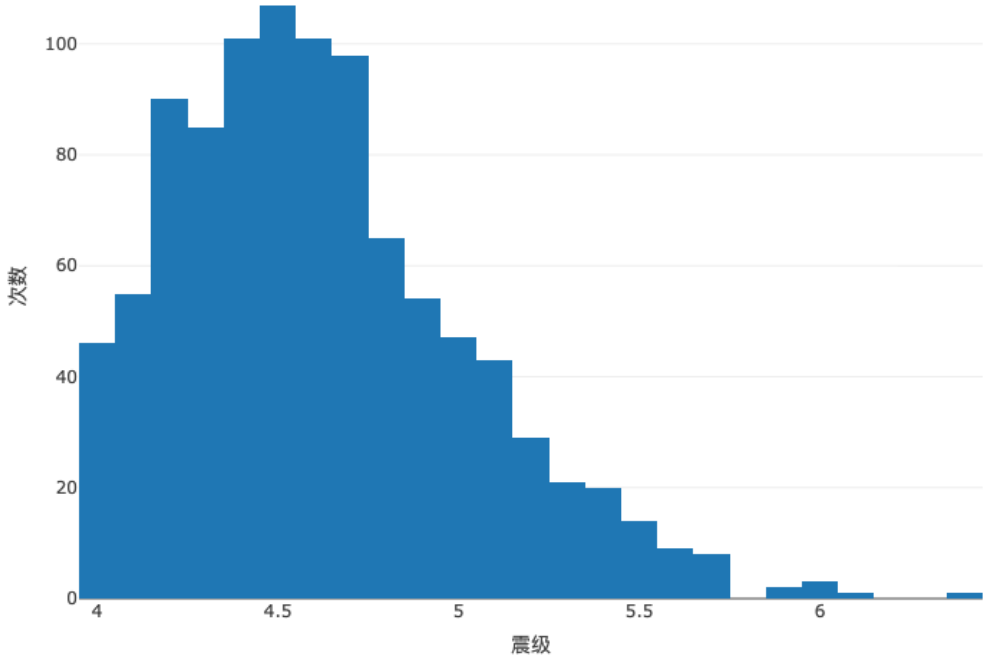


图 13.3: 地震震级的频数分布图

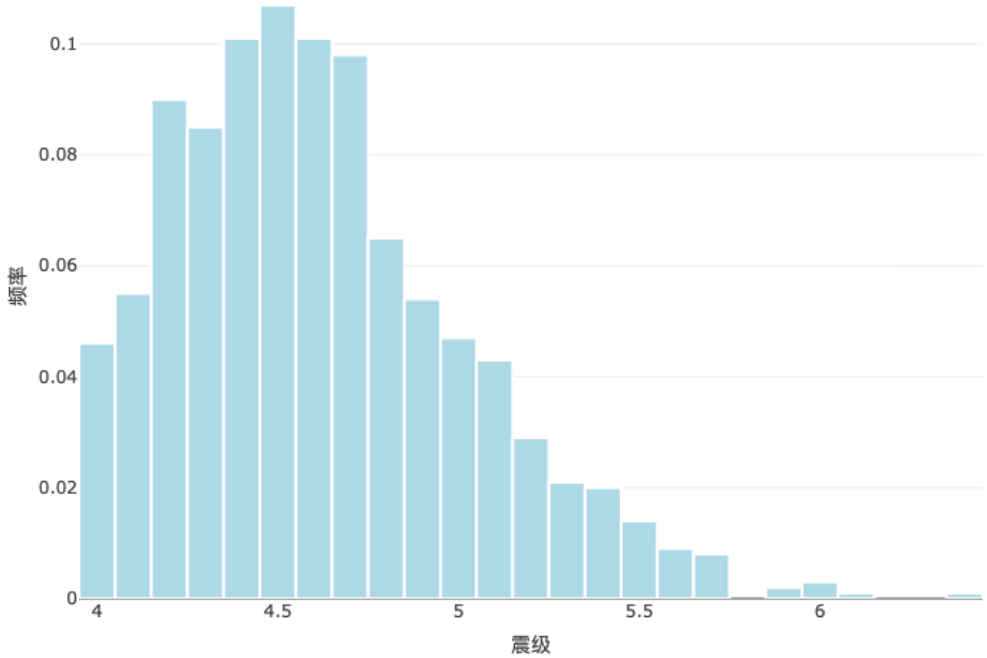


图 13.4: 地震震级的频率分布图

`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")
  )
```

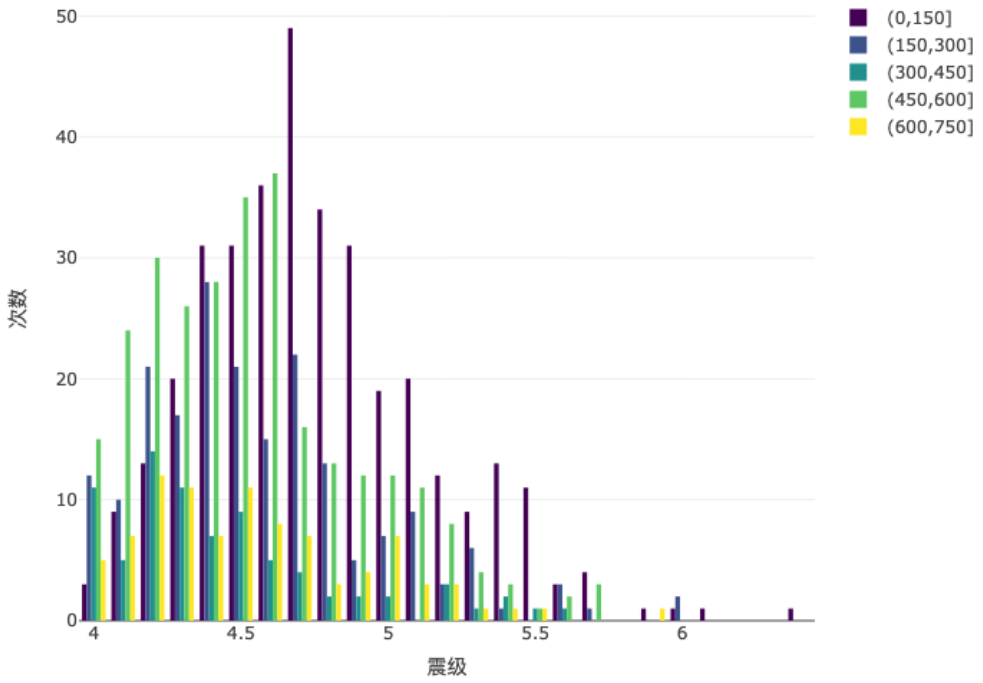


图 13.5: 地震震级的频率分布图

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")
```

```
)
```

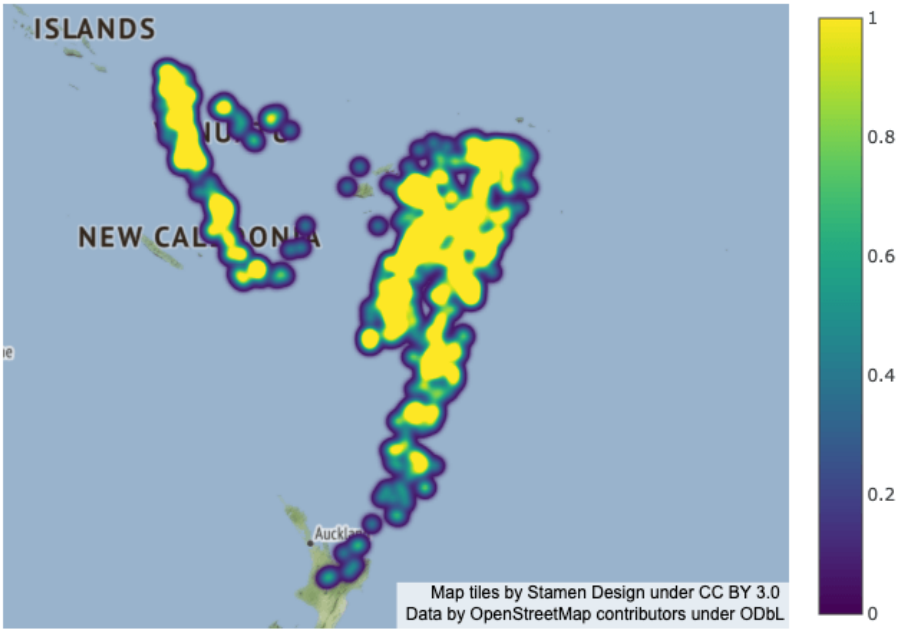


图 13.6: 空间点数据的核密度估计

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"
  )
```

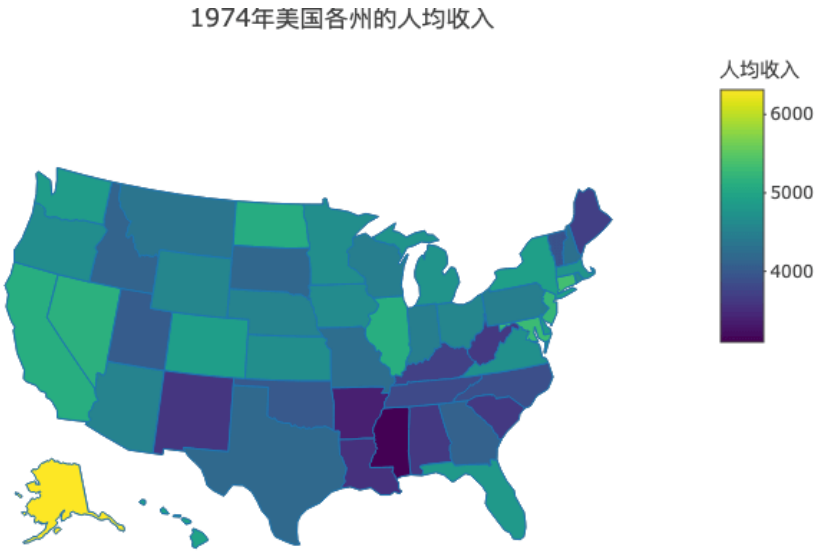


图 13.7: 1974 年美国各州的人均收入

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 **?@fig-plotly-animation** 展示 1999-2022 年 Martin Maechler 和 Brian Ripley 的代码提交量变化。

```r
# 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)
  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
  ) |>
  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)"),
  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
  )
```

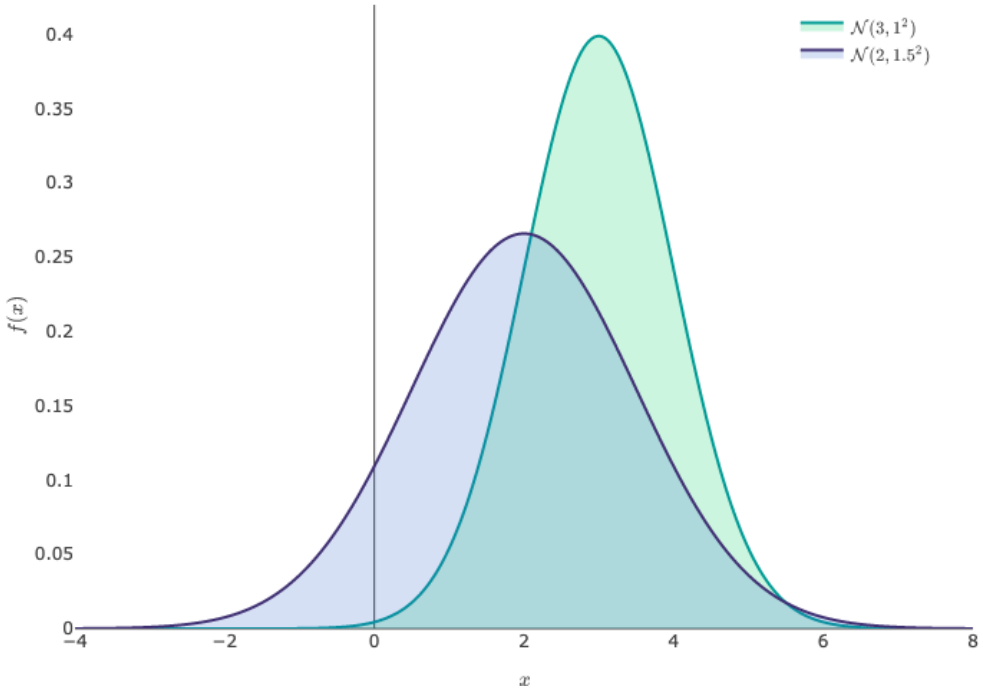


图 13.8: 设置数学公式

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
```

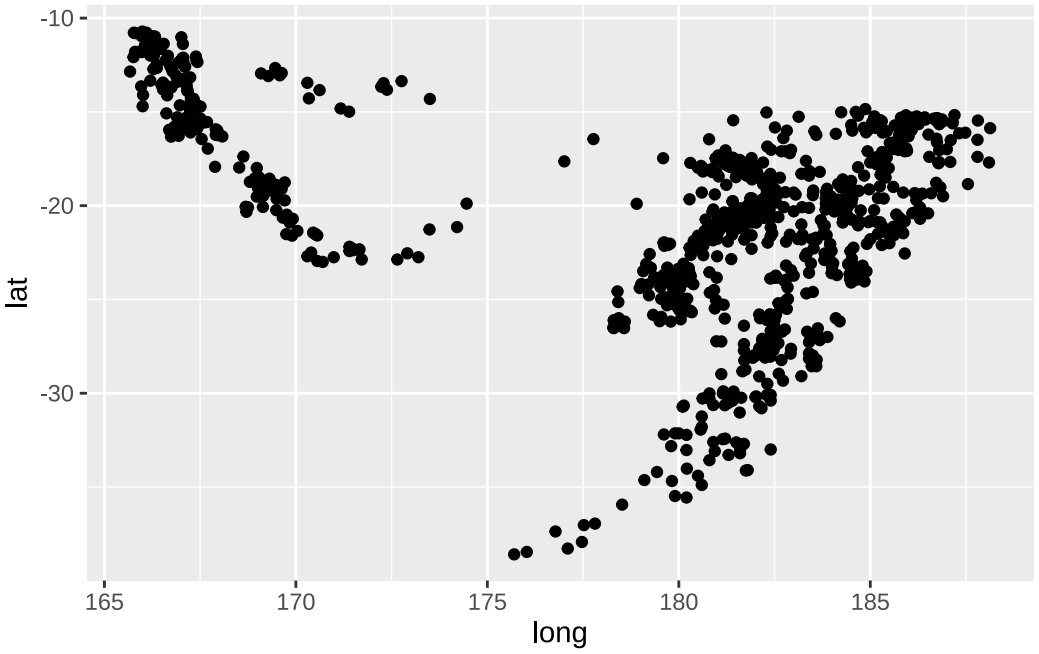


图 13.9: ggplot2 绘制的静态图形

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") |>
  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,
      range = c(160, 190),
      dtick = 5
    ),
    lataxis = list(
      showgrid = TRUE,
      gridwidth = 0.5,
      range = c(-40, -10),
      dtick = 5
    )
  ))
```

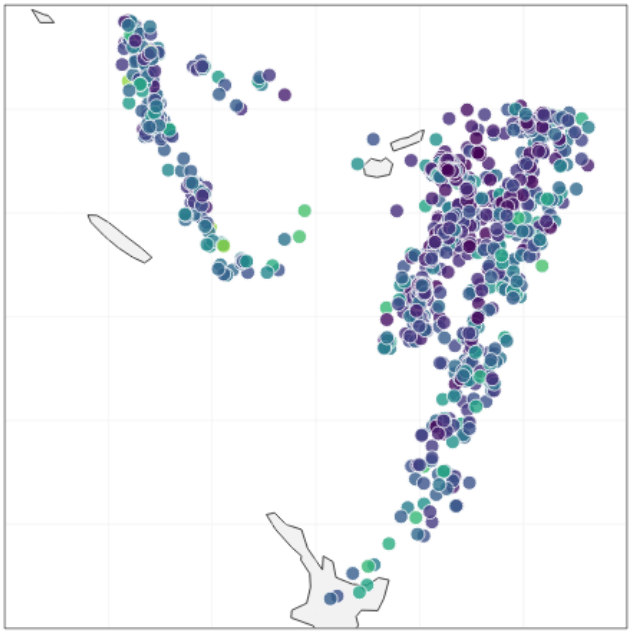


图 13.10: 空间点数据图

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(
    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 **?@fig-subplot** 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 **?@fig-subplot-2** 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))
```

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)

dat <- tribble(
  ~name1, ~name2,
  as.character(htmltools::tags$b("加粗")), as.character(htmltools::a(href = "https://rstudio.com", "走
  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) {
```

表格 14.1: **?(caption)**

```
  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 = F, # 设置 escape = F
  colnames = c(colorize_text("第1列", "red"), as.character(htmltools::em("第2列"))),
  caption = htmltools::tags$caption(
    style = "caption-side: top; text-align: center;",
    "表格 2: ", htmltools::em("表格标题")
  ), # 在表格底部显示标题，默认在表格上方显示标题
  # filter = "top", # 过滤框
  options = list(
    pageLength = 5, # 每页显示5行
    dom = "t"
  )
)
```

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：

表格 14.2: **?(caption)**

```
# 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
    )
  ),
  caption = "Base R 包内置的数据集"
)
```

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

第十五章　交互应用

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)
```

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"
```

```
    )
  })
}

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
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
      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

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) {
  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,
              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)
      )
    )
  )
)

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",
      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)
  })
}

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$h2("第一个tab页")
  ),
  material_side_nav_tab_content(
    side_nav_tab_id = "tab_2",
    tags$h2("第二个tab页")
  )
)

server <- function(input, output) {

}
shinyApp(ui = ui, server = server)
```

15.6 Shiny 部署

15.6.1 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 secon
  ),
  hr(),
  checkboxInput("throw", "Throw on download?"),
  downloadButton("download", "下载 (等待5秒)"),
  plotOutput("plot")
)

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)
```

15.7 Shiny 替代品

R Markdown 文档

- crosstalk 交互
- flexdashboard 布局
- DT 交互表格
- leaflet 交互地图
- ggiraph 交互图形

Quarto 文档

15.8 Shiny 案例

- radiant 探索性数据分析解决方案

15.9 总结

- 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。
- 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。
- 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。
- Shiny 权限。shinymanager 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。

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。

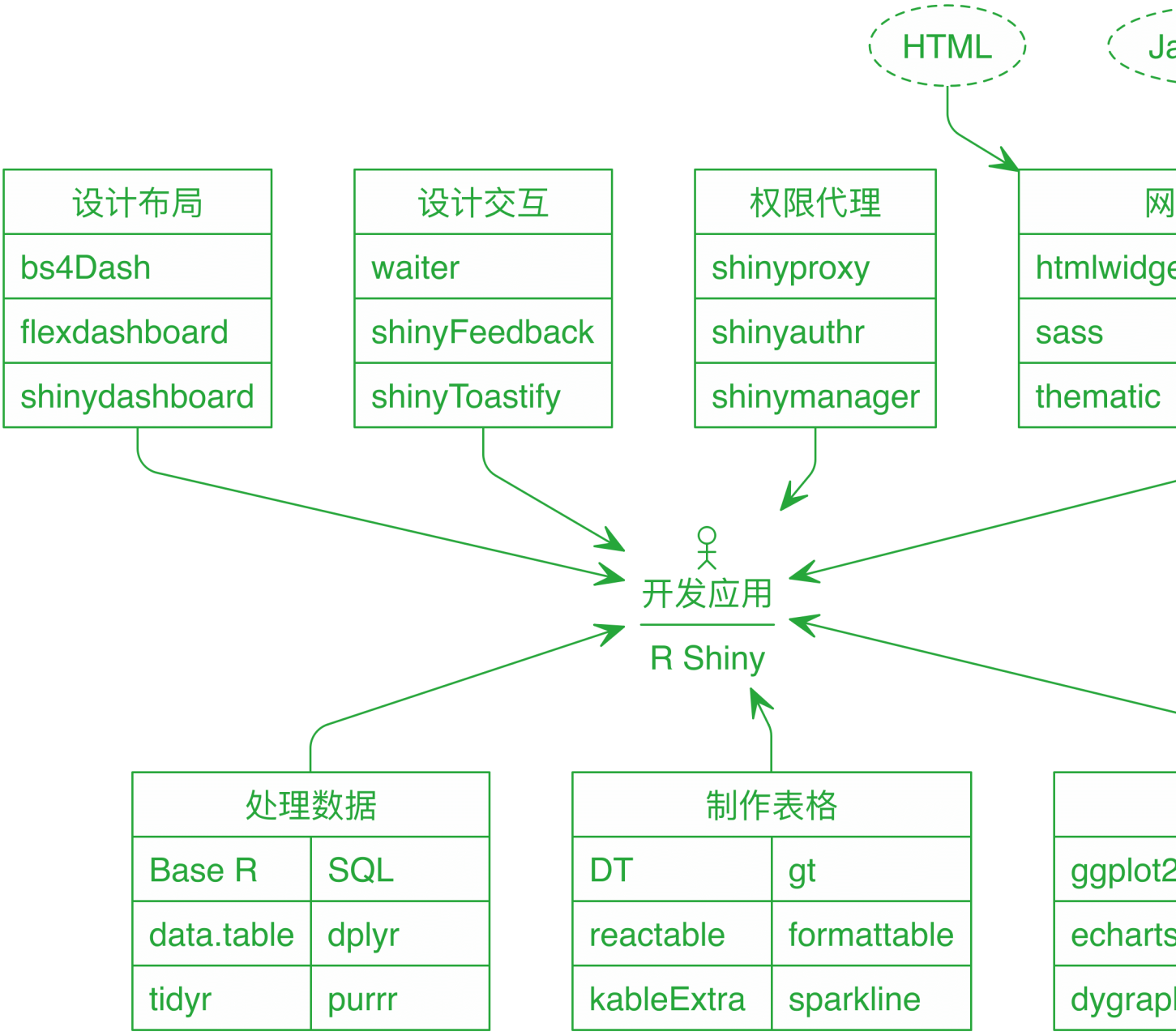


图 15.1: Shiny 生态系统

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。

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

(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]
```

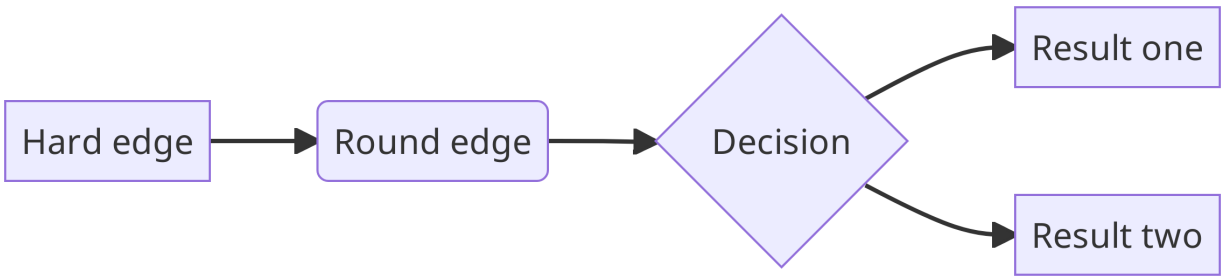


图 16.2: 流程图

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()
```

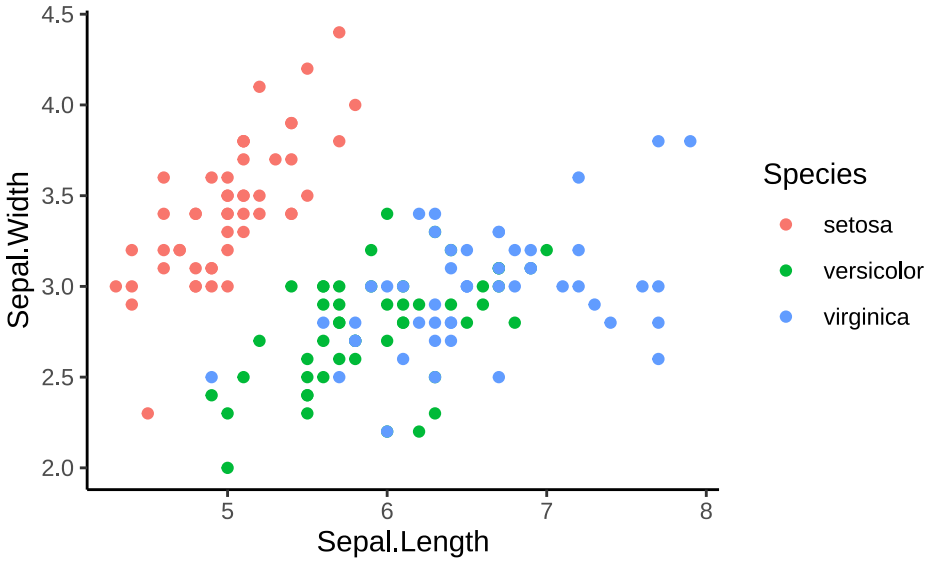


图 16.3: 一幅简单的 ggplot2 图形

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

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`
	• 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）`
	• 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 。

[1] (on the difference of Lattice (which eventually was called grid) and Trellis) DSC 2001, Wien (March 2001)

$$
\begin{aligned}
\mathsf{Var}\{\hat{\beta}\} &= \mathsf{Var}\{(X^\top X)^{-1}X^\top \mathbf{y}\} \\
&= (X^\top X)^{-1}X^\top \mathsf{Var}\{\mathbf{y}\}((X^\top X)^{-1}X^\top)^\top \\
&= (X^\top X)^{-1}X^\top \mathsf{Var}\{\mathbf{y}\}X(X^\top X)^{-1} \\
&= (X^\top X)^{-1}X^\top \sigma^2 I X(X^\top X)^{-1} \\
&= (X^\top X)^{-1}\sigma^2
\end{aligned} \tag{16.2}
$$

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`。
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

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}}
```

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：

$$\left(\sqrt{\frac{M}{1-\left(\frac{r}{\widetilde{x_1+\cdots+u_N}}\right)^2}\left(\sum_{\beta=1}^{N}\sum_{i=1}^{n}\frac{\partial u_\beta}{\partial x_i}+1\right)}+\sqrt{XY}\right)^3$$

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：

$$\left(\sqrt{\frac{M}{1-\left(\frac{r}{\widetilde{x_1+\cdots+u_N}}\right)^2}\left(\sum_{\beta=1}^{N}\sum_{i=1}^{n}\frac{\partial u_\beta}{\partial x_i}+1\right)}+\sqrt{XY}\right)^3$$

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：

```
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

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

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 \\
```

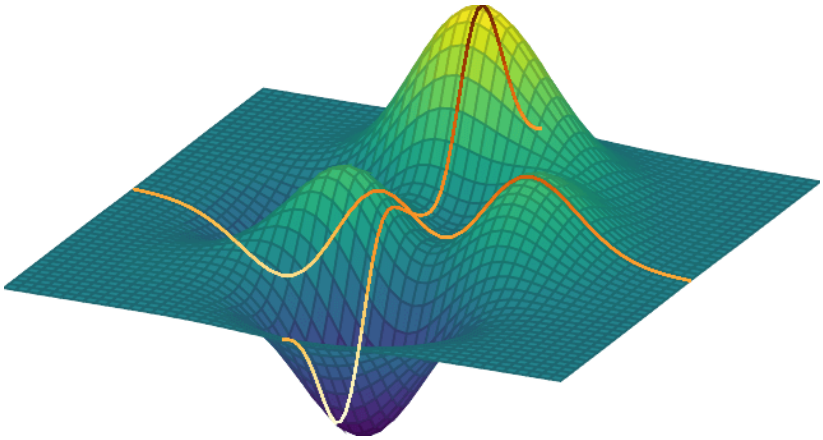


图 17.4: 图片的标题

```
      \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。

17.2 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"
---

# 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

17.3 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：

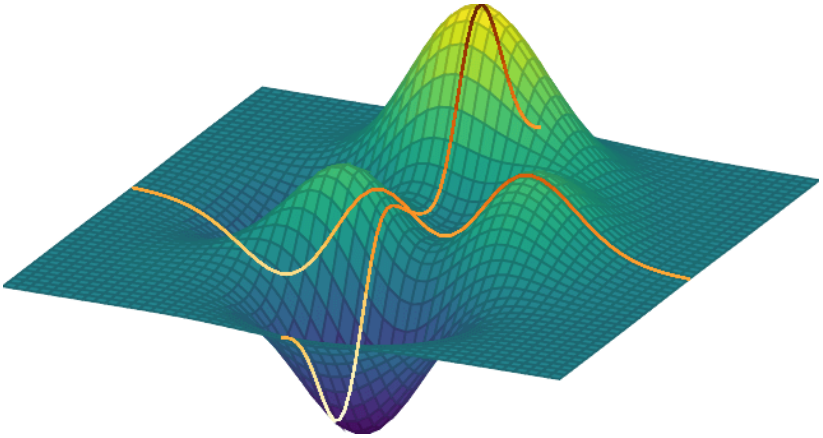


图 17.5: Peaks 函数图像

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：

表格 17.4: 制作表格的管道语法

Default	Left	Right	Center
12	12	12	12

Default	Left	Right	Center
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。

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

18.2　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
```

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")
)
```

第四部分

统计分析

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？杨灿老师在讨论帖提出检验的作用和实际应用问题。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，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 费舍尔提出，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。

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。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

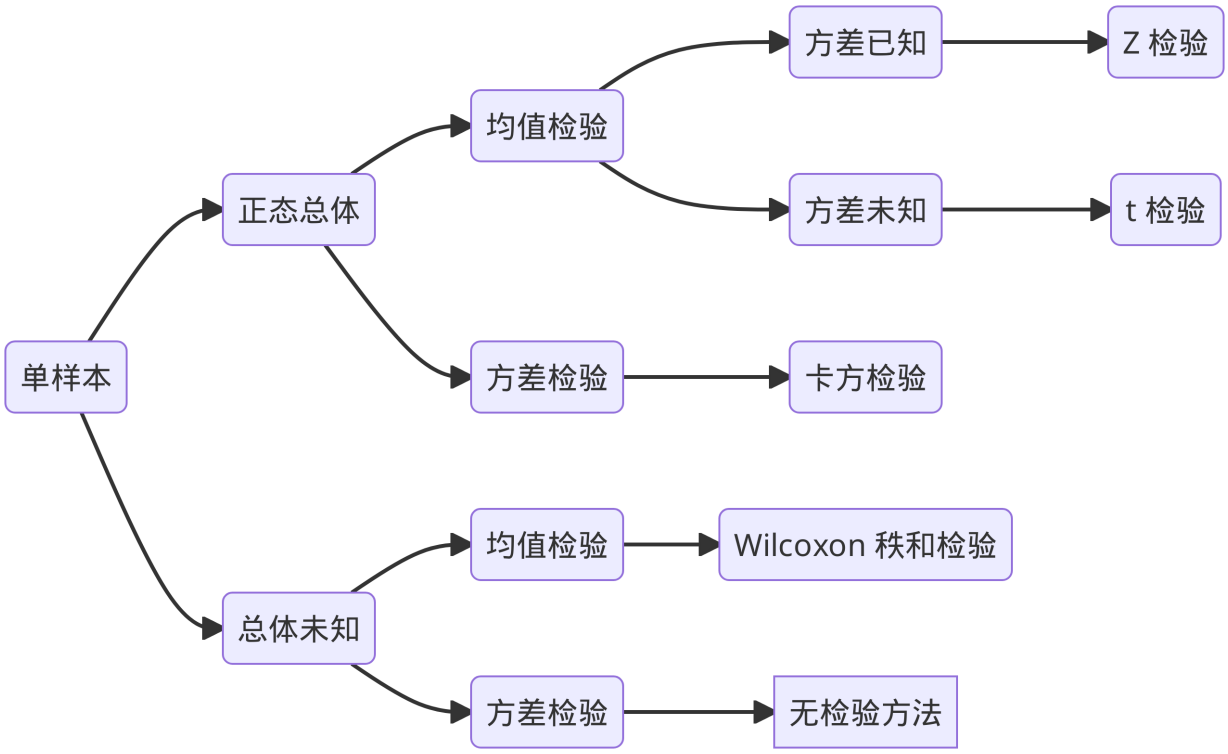


图 19.1: 单样本检验

$$u = \frac{\bar{x} - \mu_0}{\sigma/\sqrt{n}}$$

假定 $\mu_0 = 1$ 对于检验问题 I 拒绝域 $\{u \geq u_{1-\alpha}\}$

```
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
```

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 $\{t \geq t_{1-\alpha}(n-1)\}$

```r
# 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
```

```r
# 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)。

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

```r
# 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
```

```r
# P 值
1 - pchisq(q = chi, df = n -1)
```

```
#> [1] 0.002183653
```

19.1.3 总体未知均值检验

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。

```r
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
```

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

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

常见检验问题

$$
\begin{array}{lllll}
\text{I} & H_0: \mu_1 - \mu_2 \leq 0 & vs. & H_1: \mu_1 - \mu_2 > 0 \\
\text{II} & H_0: \mu_1 - \mu_2 \geq 0 & vs. & H_1: \mu_1 - \mu_2 < 0 \\
\text{III} & H_0: \mu_1 - \mu_2 = 0 & vs. & H_1: \mu_1 - \mu_2 \neq 0
\end{array}
$$

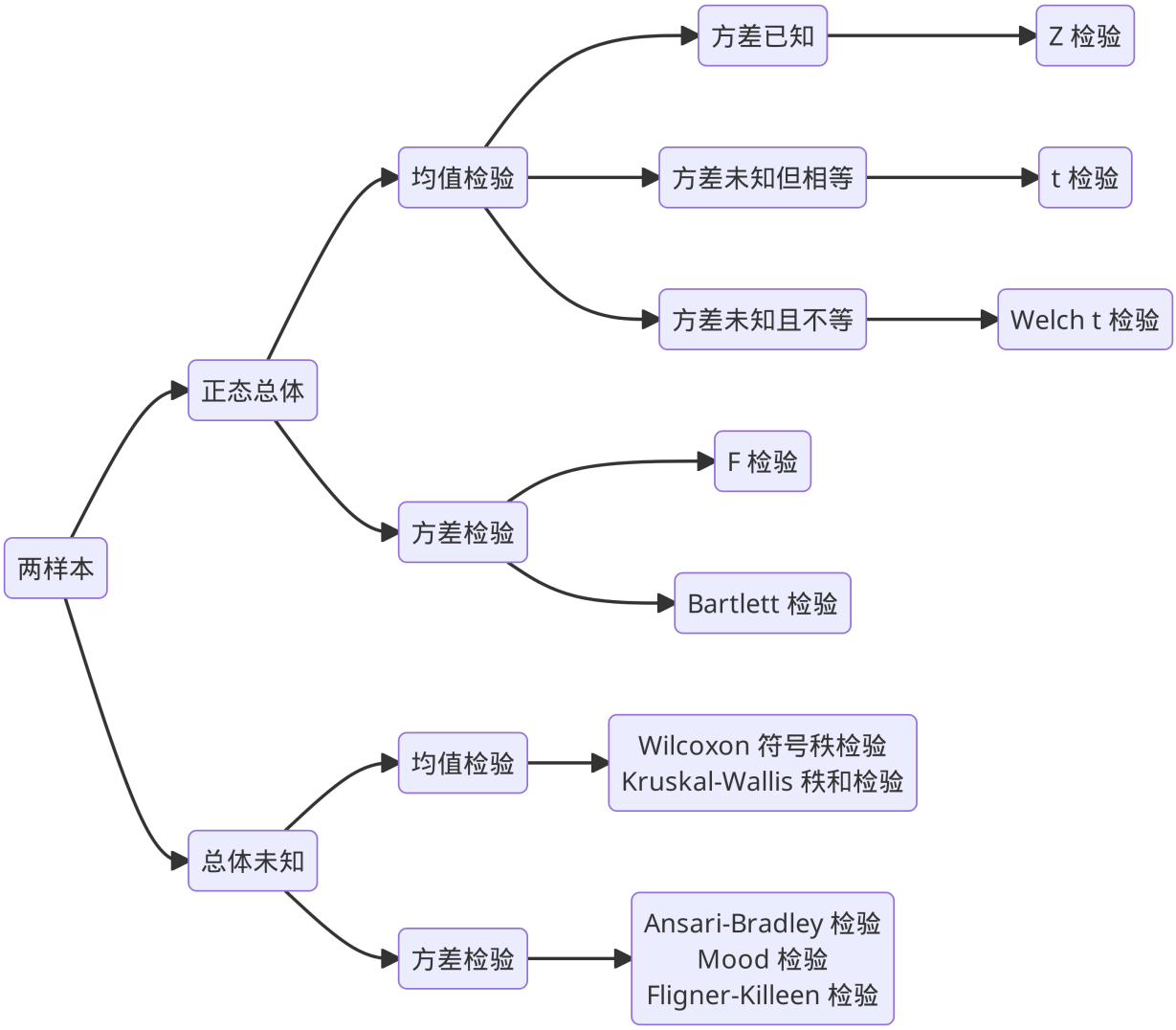


图 19.2: 两样本检验

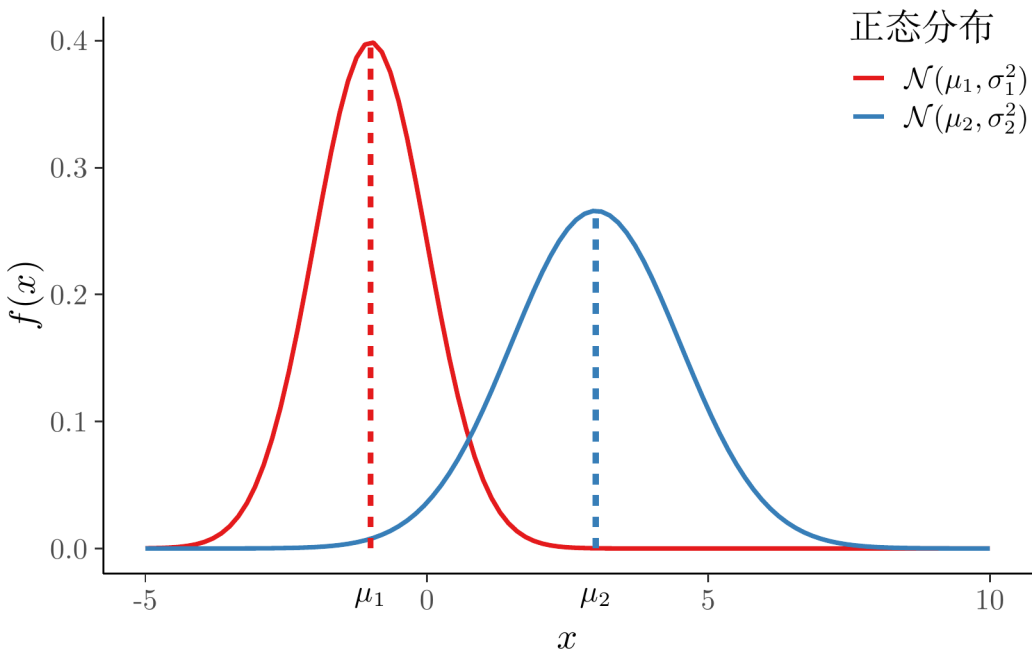


图 19.3: 两样本均值之差的检验

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

$$W_1 = \{u \geq u_{1-\alpha}\}, \quad p_1 = 1 - \Phi(u_0)$$

```
n_1 <- 100
n_2 <- 80
mu_1 <- 10
sigma_1 <- 2.5
mu_2 <- 6
sigma_2 <- 4.5

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

```
# 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

```
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。

```
# 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

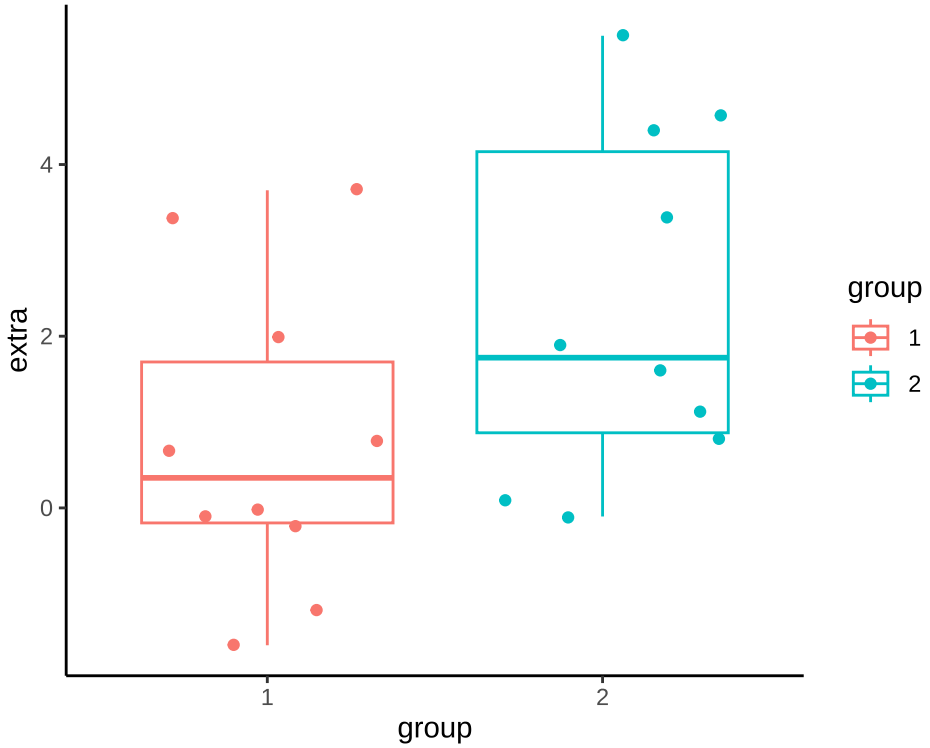


图 19.4: 学生睡眠数据的分布

```r
# 方差未知且不等
t.test(extra ~ group, data = sleep, var.equal = FALSE)
```

```
#>
#>  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。许宝𨱇（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𨱇的生平事迹和学术成就，见《许宝𨱇先生的生平和学术成就》。钟开莱（Kai-Lai Chung）将许宝𨱇的论文集整理出版 (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。

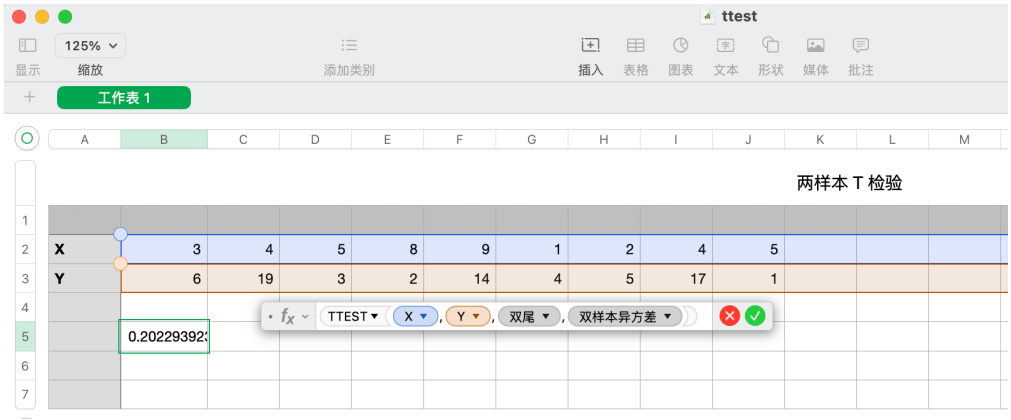


图 19.5: 办公软件 Numbers 的两样本 t 检验

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。

```
# 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
```

```
# 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。

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()`。

```r
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

```r
# 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
)
```

```
#>
#>  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 19.1 。

表格 19.1: 检验方法分类

两个样本	多个样本
ansari.test() / mood.test()	fligner.test()

Ansari-Bradley 检验 `ansari.test()` 和 Mood 检验 `mood.test()` 属于两样本的非参数检验，检验尺度参数是否相同（齐性）。Fligner-Killeen 检验 `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)
```

```
#> 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。

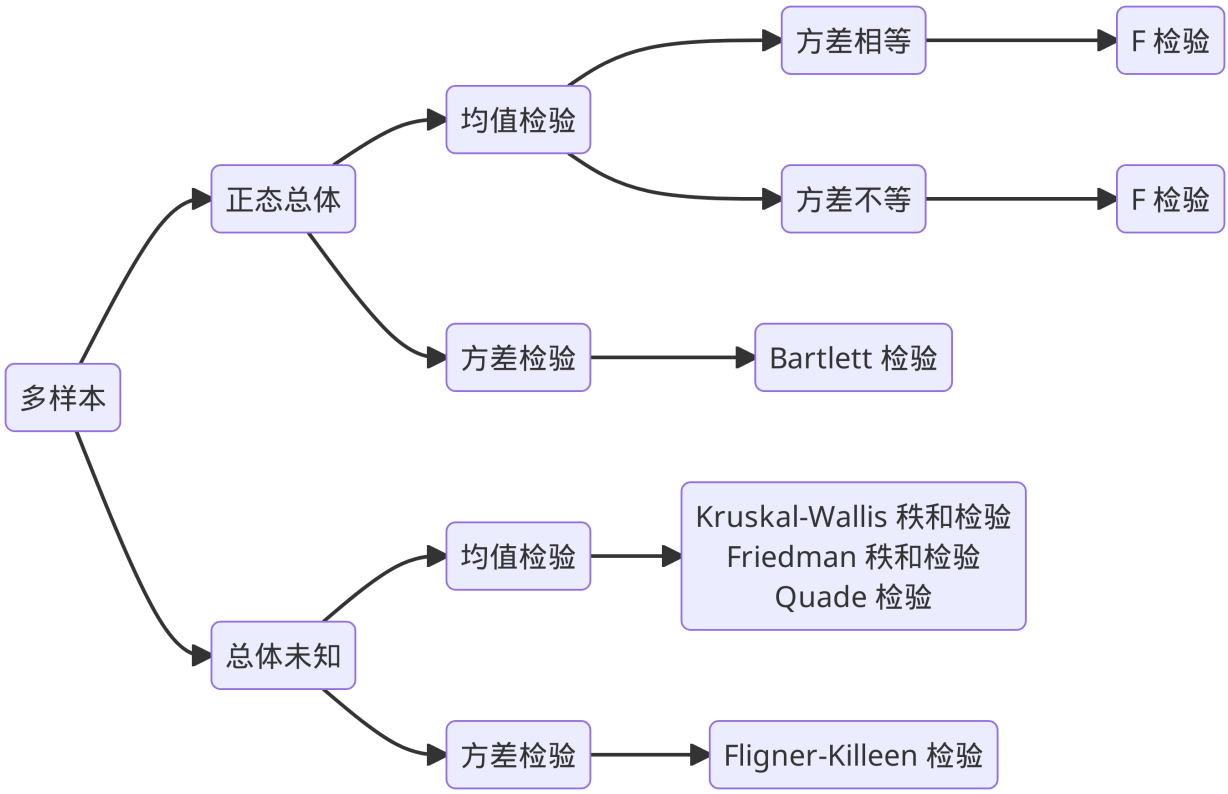


图 19.6: 多样本检验

```
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)
```

```
#>      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()
```

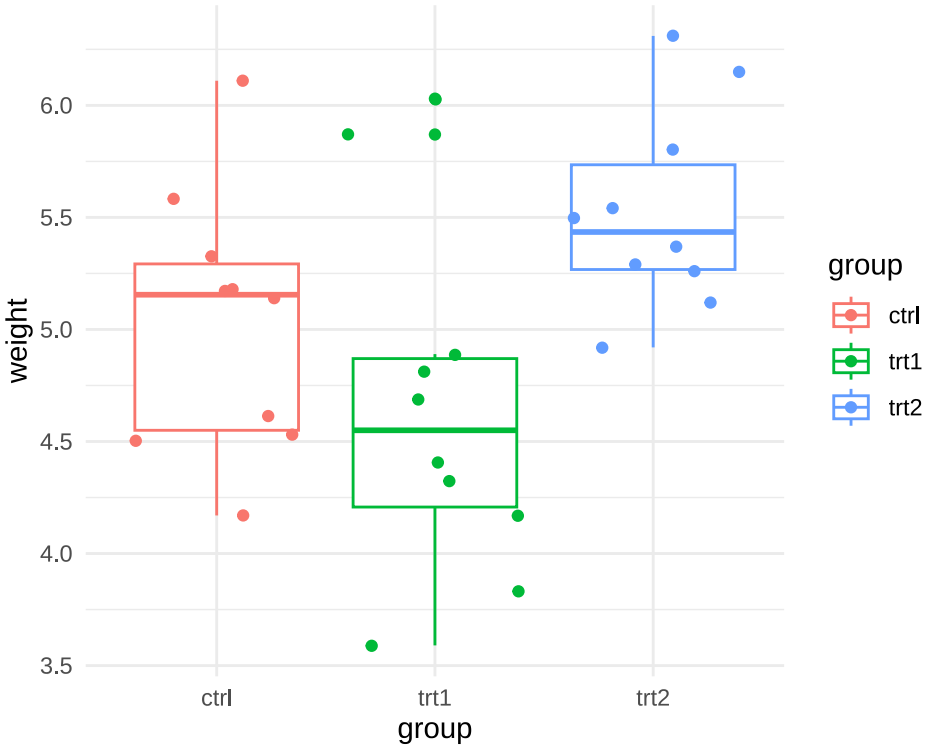


图 19.7: 植物干重

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 19.2

表格 19.2: 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

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
#> 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

后面总体分布未知的情况下的方差检验也都可以用在这里。

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。相比于 Bartlett 检验，Levene 检验更加稳健。

```
# 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
```

```
# 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。

```
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

```
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 。

```
# 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
```

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。

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
```

```
#> group2       0.7001140 1.58 2.459886
```

group2 对应的 95% 的置信区间是 $(0.7001140, 2.459886)$ 。

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，介绍做多个组两两配对的 t 检验。

```
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。

19.4.2 逐对比例检验

对于离散数据，没有配对检验，但可以两两比较，采用逐对比例检验方法。如下示例含有 4 个组，采用函数 pairwise.prop.test() 做两两比较。

```
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
#> 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
```

> **重要**
>
> 配对检验和两两比较检验是不同的，前者要求两个组的样本量相同，样本点之间是对应关系，后者无此要求。

19.4.3 配对 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
```

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
```

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。

```
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
```

19.4.4 配对相关性检验

配对样本的相关性检验 `cor.test()`：Pearson's 相关系数，Kendall's τ 检验或者 Spearman's ρ 检验

```
# cor.test(method = "pearson")  # lm(y ~ 1 + x)
# cor.test(method = "kendall")
# cor.test(method = "spearman") # lm(rank(y) ~ 1 + rank(x))
```

19.5 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。

19.5.1 正态性检验

> 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]

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.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.

[1] https://stat.ethz.ch/pipermail/r-help/2005-April/070508.html

— 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

$$T_{EP} = 1 + \frac{n}{\sqrt{3}} + \frac{2}{n} \sum_{i=2}^{n} \sum_{j=1}^{i-1} \exp\{ - \frac{(x_j - x_i)^2}{2s_\star^2} \} - \sqrt{2} \sum_{i=1}^{n} \exp\{ - \frac{(x_i - \bar{x})^2}{4s_\star^2} \}$$

其中 $\bar{x}, s_\star^2$ 分别是样本均值和（除以 n 的）样本方差。

19.5.2 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]

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
```

19.5.3 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。

[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

19.5.4 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)
```

19.5.5 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 $\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_m}$，$\boldsymbol{x_i} \in \mathbb{R}^p$，$\boldsymbol{x_i} \overset{i.i.d}{\sim} \text{MVN}(0, \Sigma)$，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 $\text{MVN}(0, \Sigma)$，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

$$f(X) = \frac{1}{2^{\frac{mp}{2}} |\Sigma|^{\frac{m}{2}} \Gamma_p(\frac{m}{2})} |X|^{(m-p-1)/2} \exp\{-\frac{1}{2}\text{tr}(\Sigma^{-1}X)\}$$

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，$m = \text{df}, \Sigma = \text{Sigma}$。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 $m = \text{df}, \Sigma = \text{Sigma}$。

```
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。

```
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
```

```
# 随机矩阵 M 的期望理论值
2 * S
```

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

19.6 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 提出卡方统计量和卡方检验以来，陆续涌现出来一批人，其中，最重要的统计学家及其传承关系见下图 19.8 。

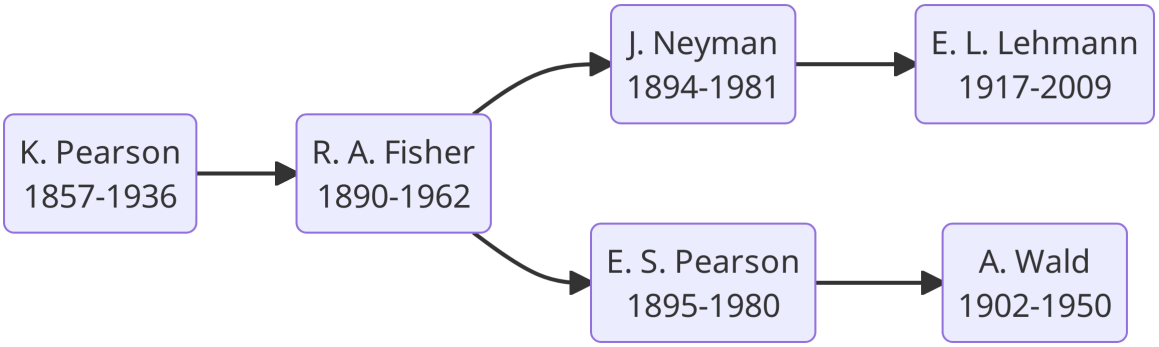


图 19.8: 假设检验理论的主要贡献者

19.6.1 假设检验和多重比较的关系

FDR 是 False Discovery Rate 的简称

19.6.2 假设检验和方差分析的关系

19.6.2.1 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)
#>
#> $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)
```

```
#>
#>  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。

19.6.2.2 双因素一元方差分析

```
with(ToothGrowth, interaction.plot(supp, dose, len))
```

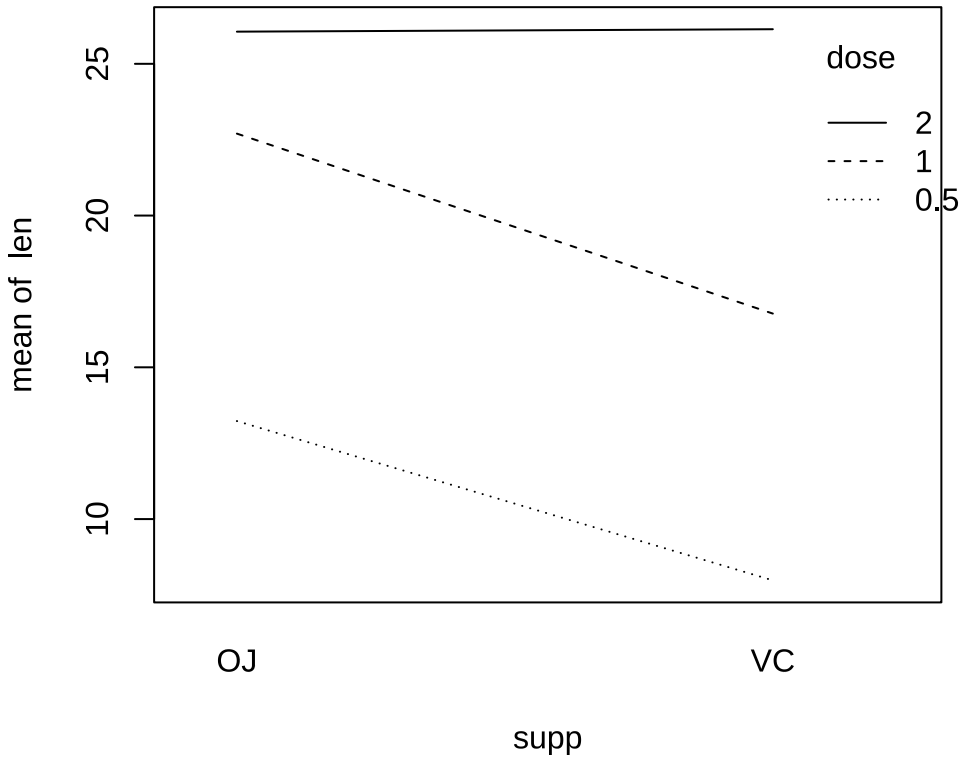


图 19.9: OJ 和 VC 的交互作用

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
```

19.6.2.3 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")
```

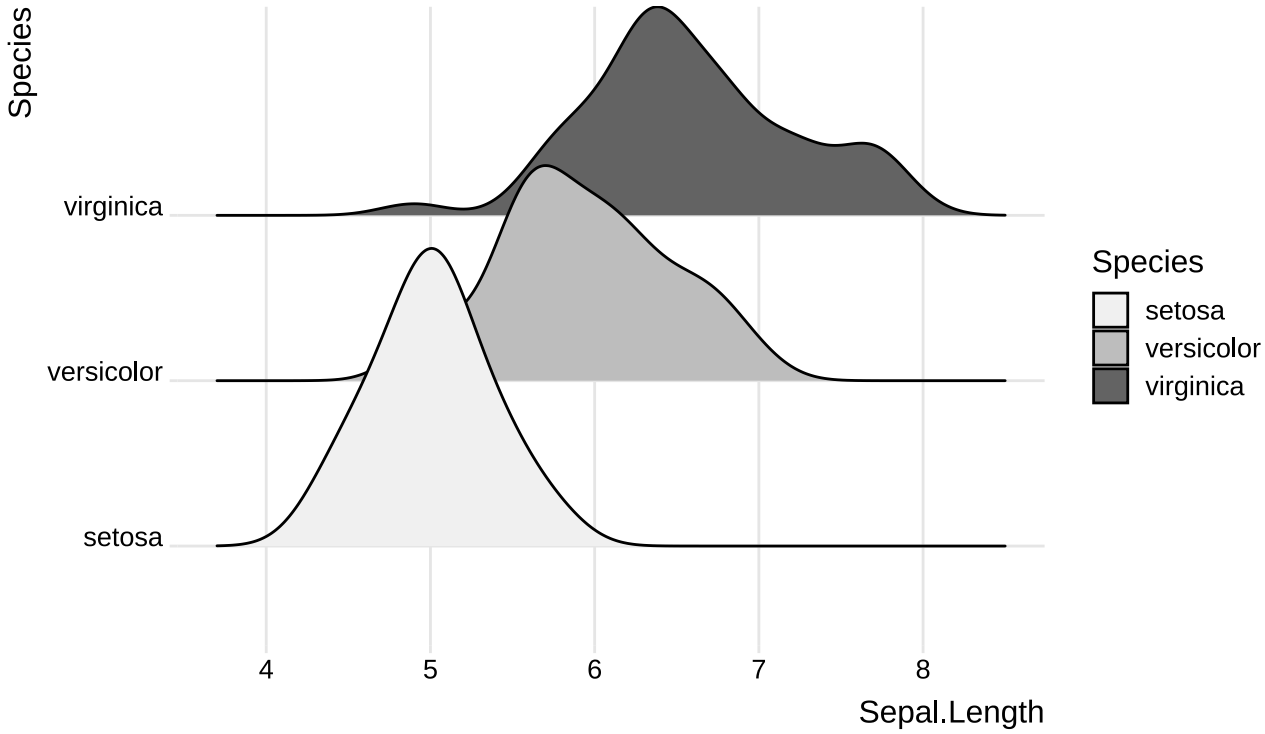


图 19.10: 鸢尾花萼片长度的分布

```
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` 。

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
```

19.6.3 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)
```

```
#>
#>  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。

19.6.4 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

19.6.4.1 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
```

19.6.4.2 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
```

19.6.5 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，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。

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)

$$\text{height}_{children} = \alpha + \beta * \text{height}_{midparent} + \epsilon$$

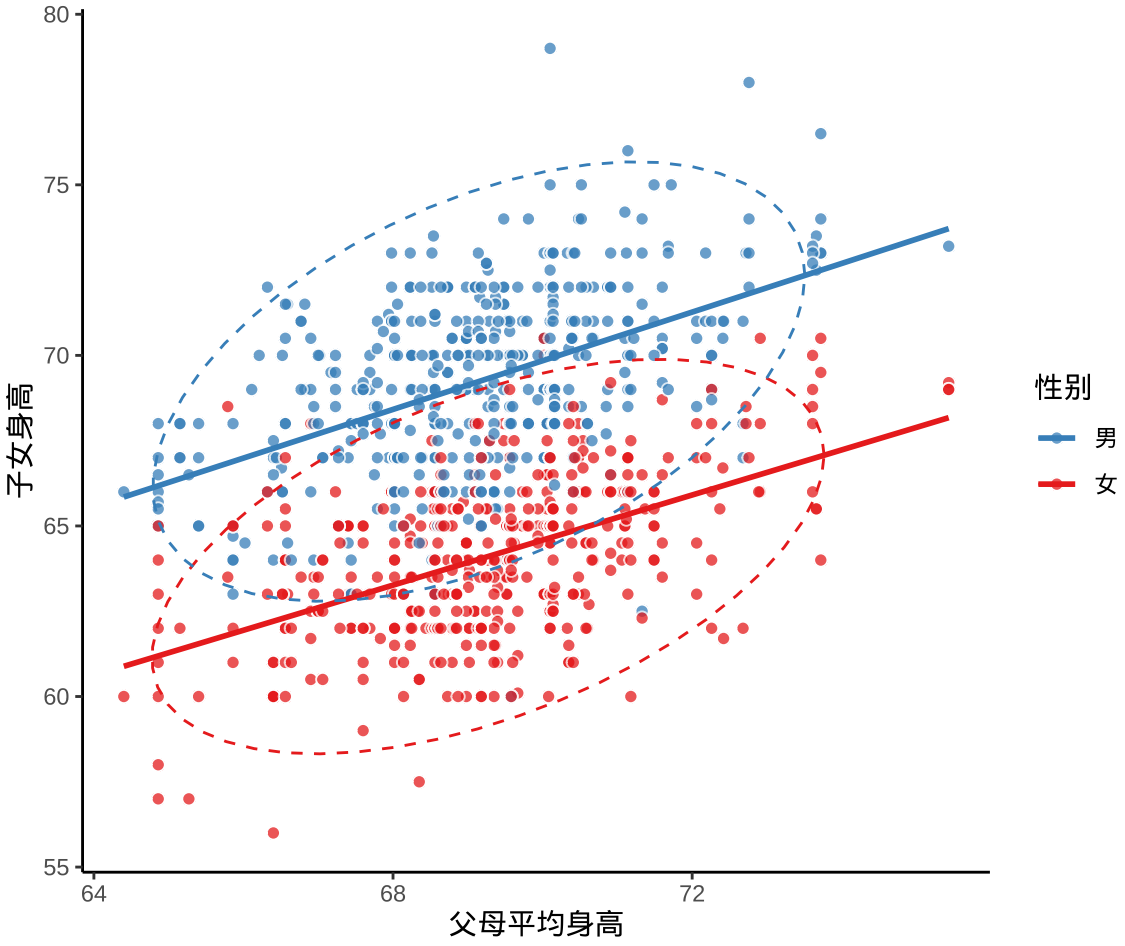


图 20.1: 子代身高与亲代身高的关系

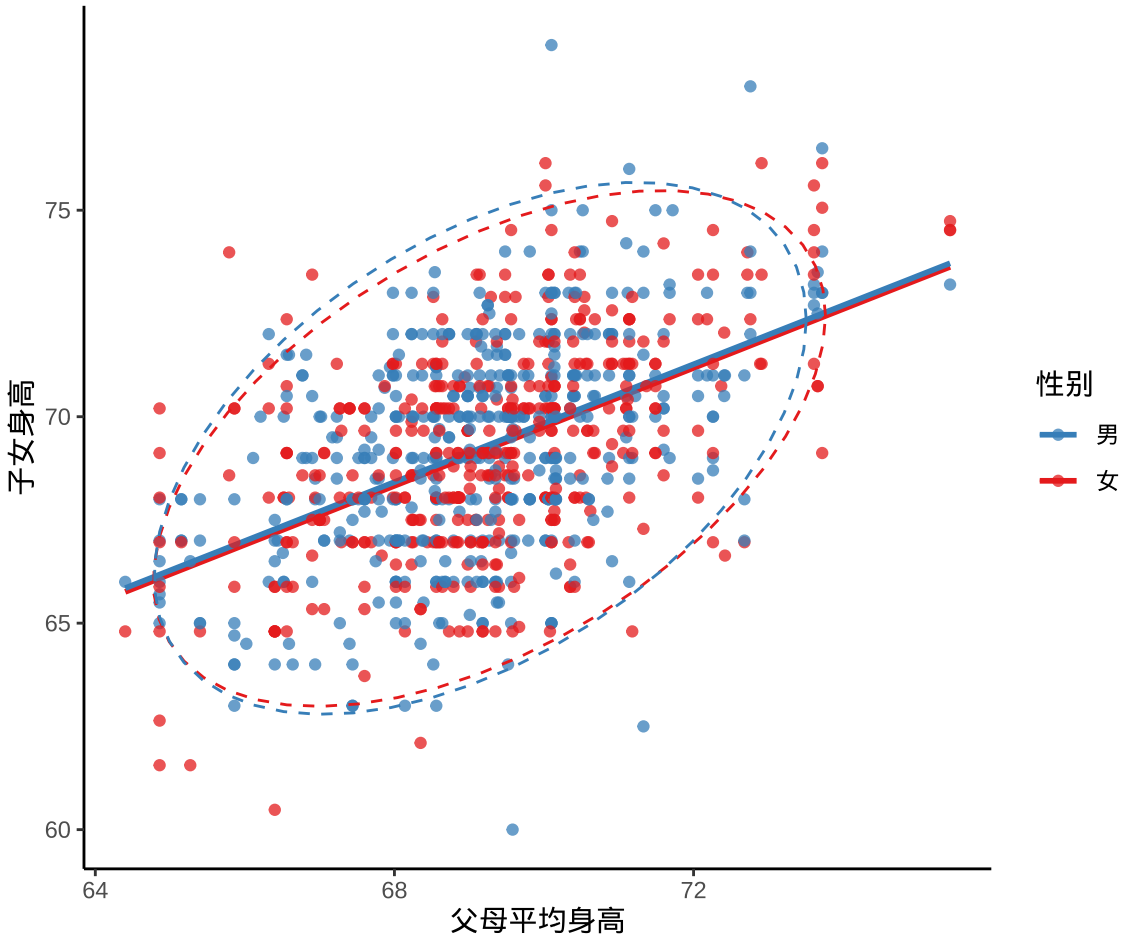


图 20.2: 子代身高与亲代身高的关系

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

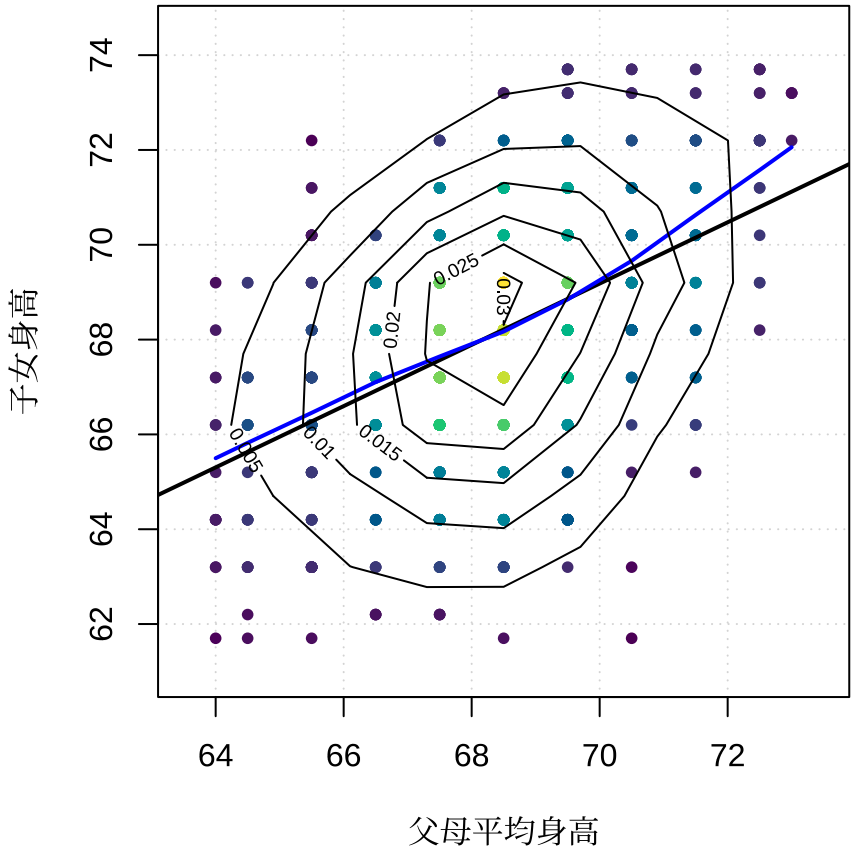


图 20.3: 二维核密度估计与二元正态分布

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。

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。

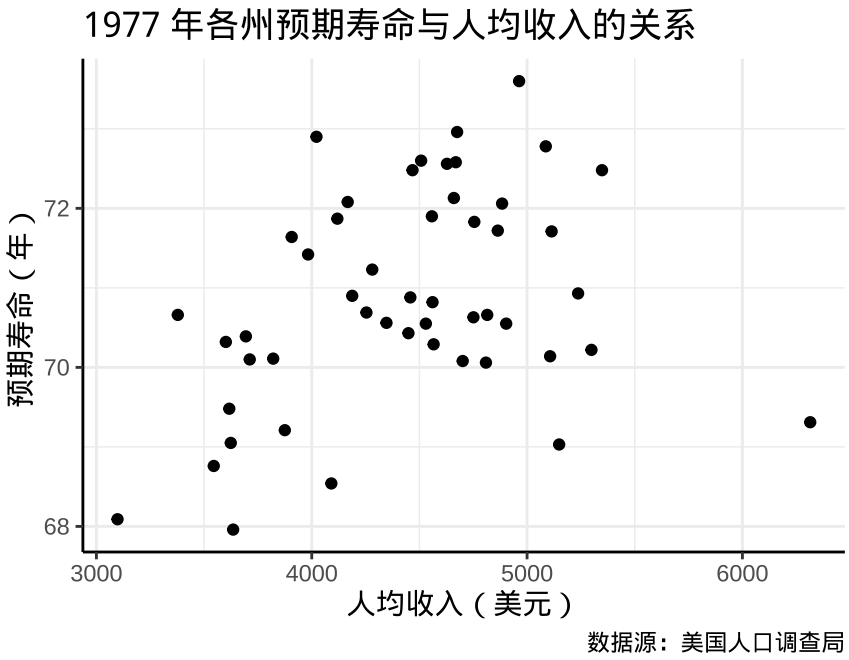


图 20.4: 预期寿命与人均收入的关系图

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 *
```

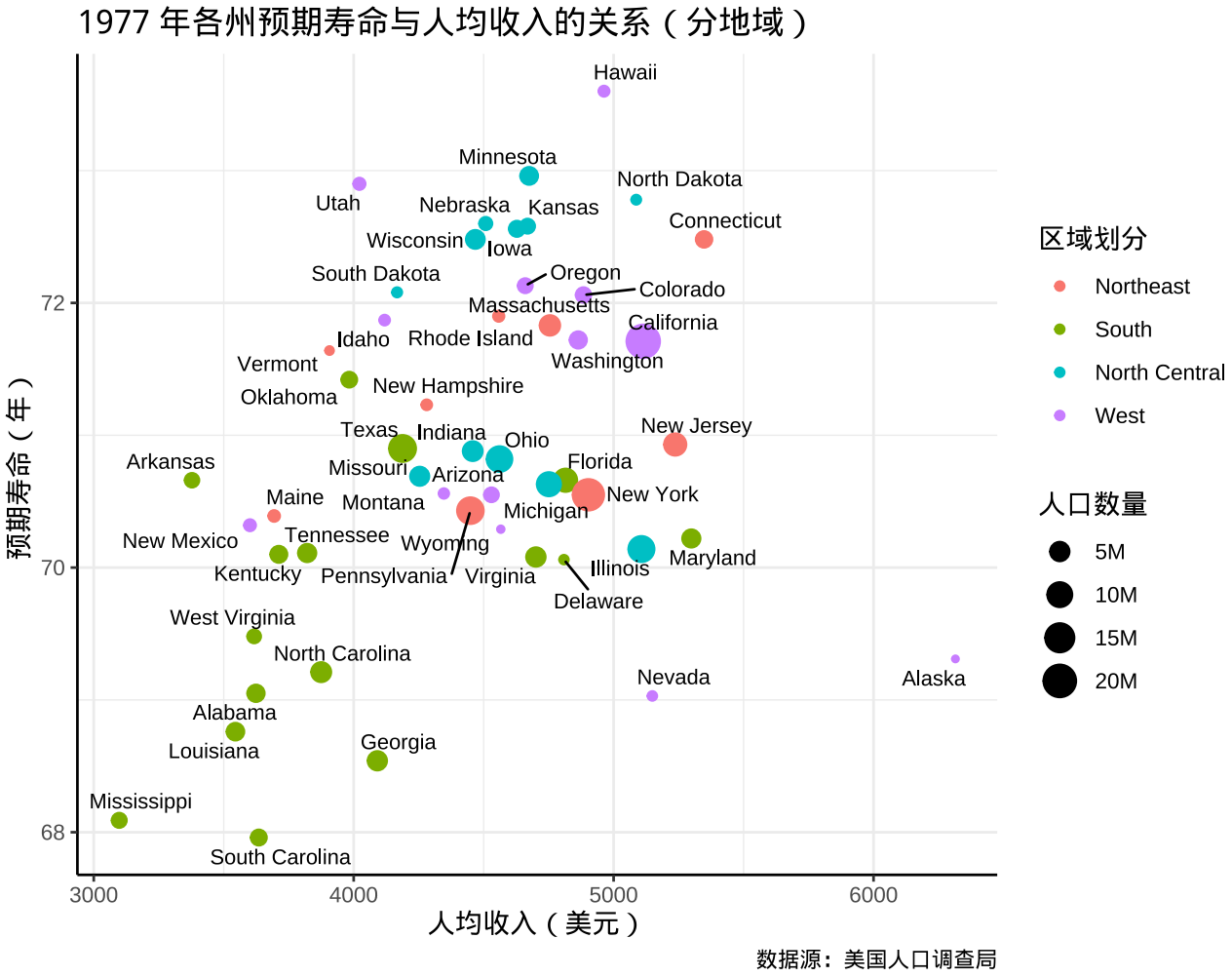


图 20.5: 分地域预期寿命与人均收入的气泡图

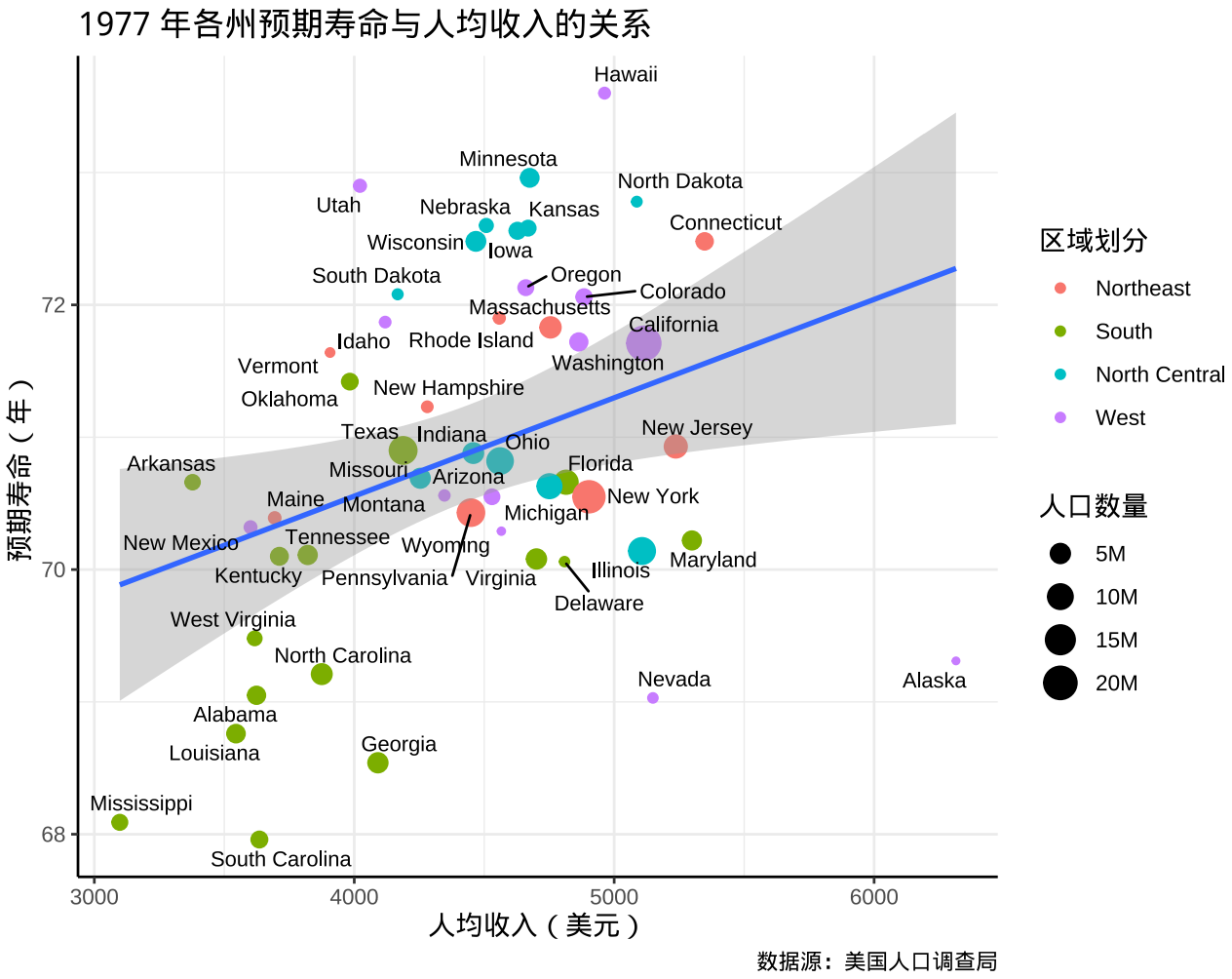


图 20.6: 1977 年美国各州预期寿命与人均收入的关系：回归分析

```
#> ---
#> 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 ...
#>  $ 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。

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 $\sum_{i=1}^{n} X_i$。

```
# 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

```
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

```
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

$$P(|X-EX|\geq\epsilon)\leq\frac{Var(X)}{\epsilon^2}$$
$$P(|X-EX|\leq\epsilon)\geq 1-\frac{Var(X)}{\epsilon^2}$$

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

$$T^{\star}=\frac{\bar{X}}{\bar{Y}}$$

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)
```

```
#>
#>  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
#>   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

```r
# 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 = "_"
)

# 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。

```
library(ggplot2)
library(ggstats)
ggplot(as.data.frame(Titanic)) +
  aes(x = Class, fill = Survived, weight = Freq, by = Class) +
  geom_bar(position = "fill") +
  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")
```

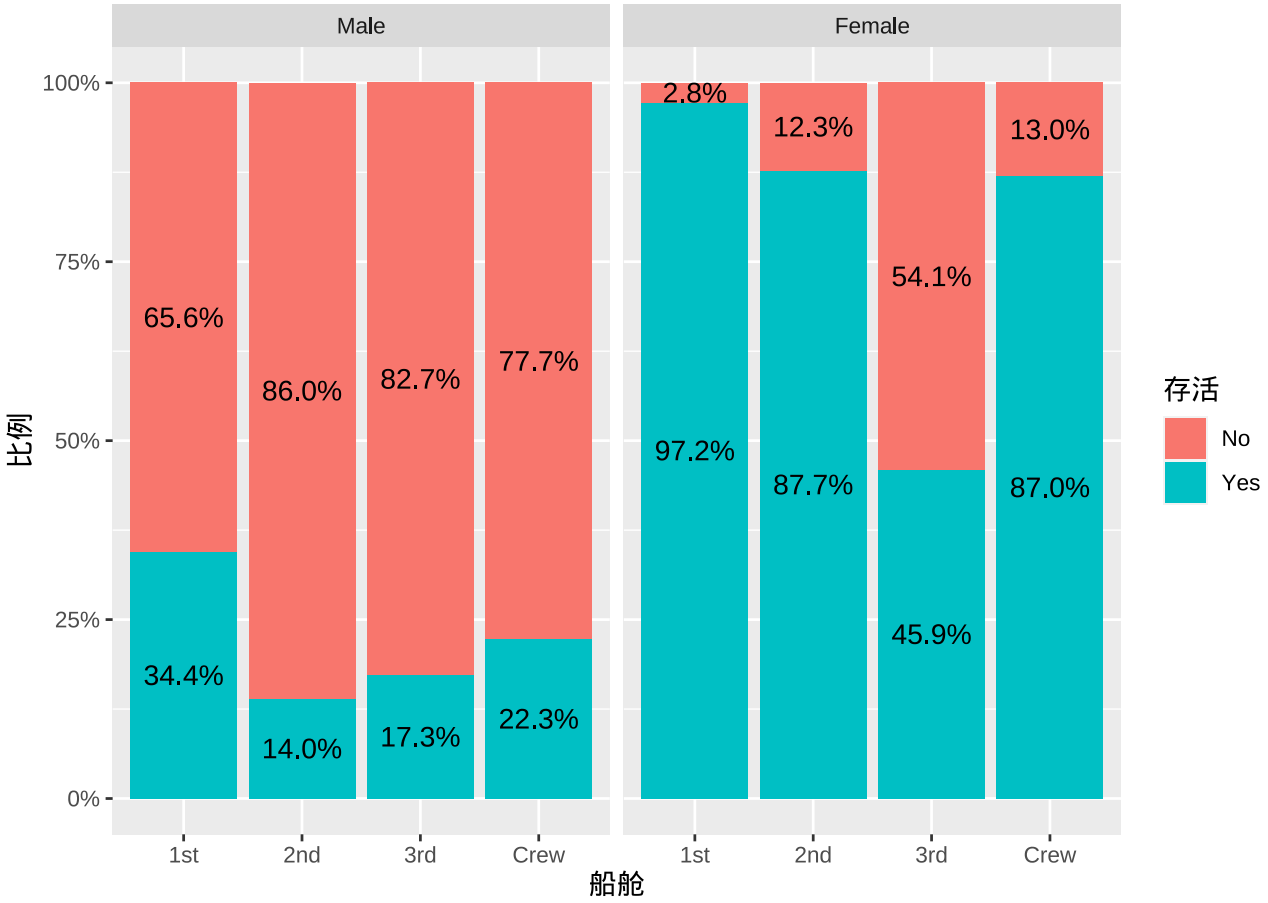


图 21.1: 百分比堆积柱形图展示多维分类数据

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),
  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"
  )
```

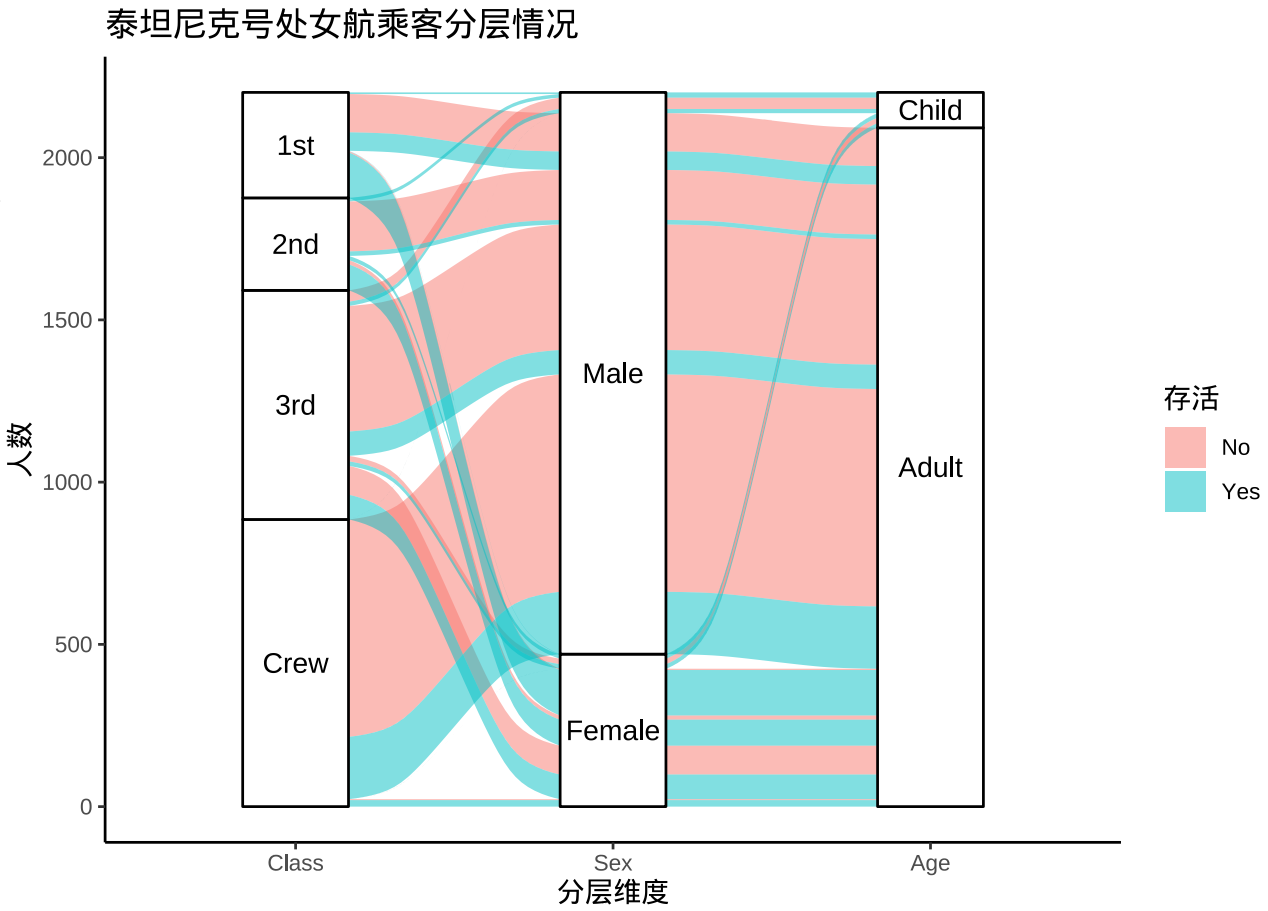


图 21.2: 桑基图展示多维分类数据

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)
```

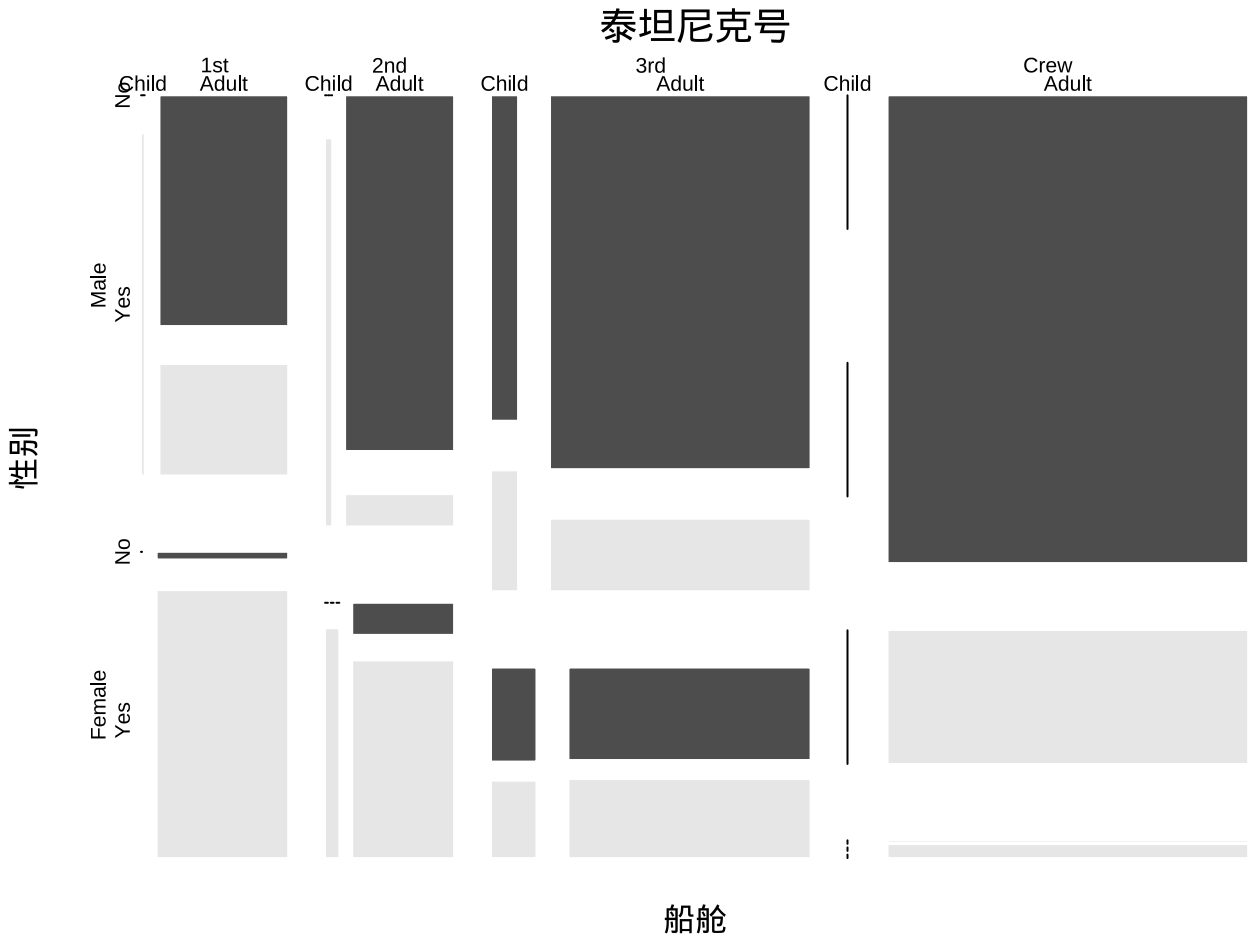


图 21.3: 马赛克图展示多维分类数据

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
)
```

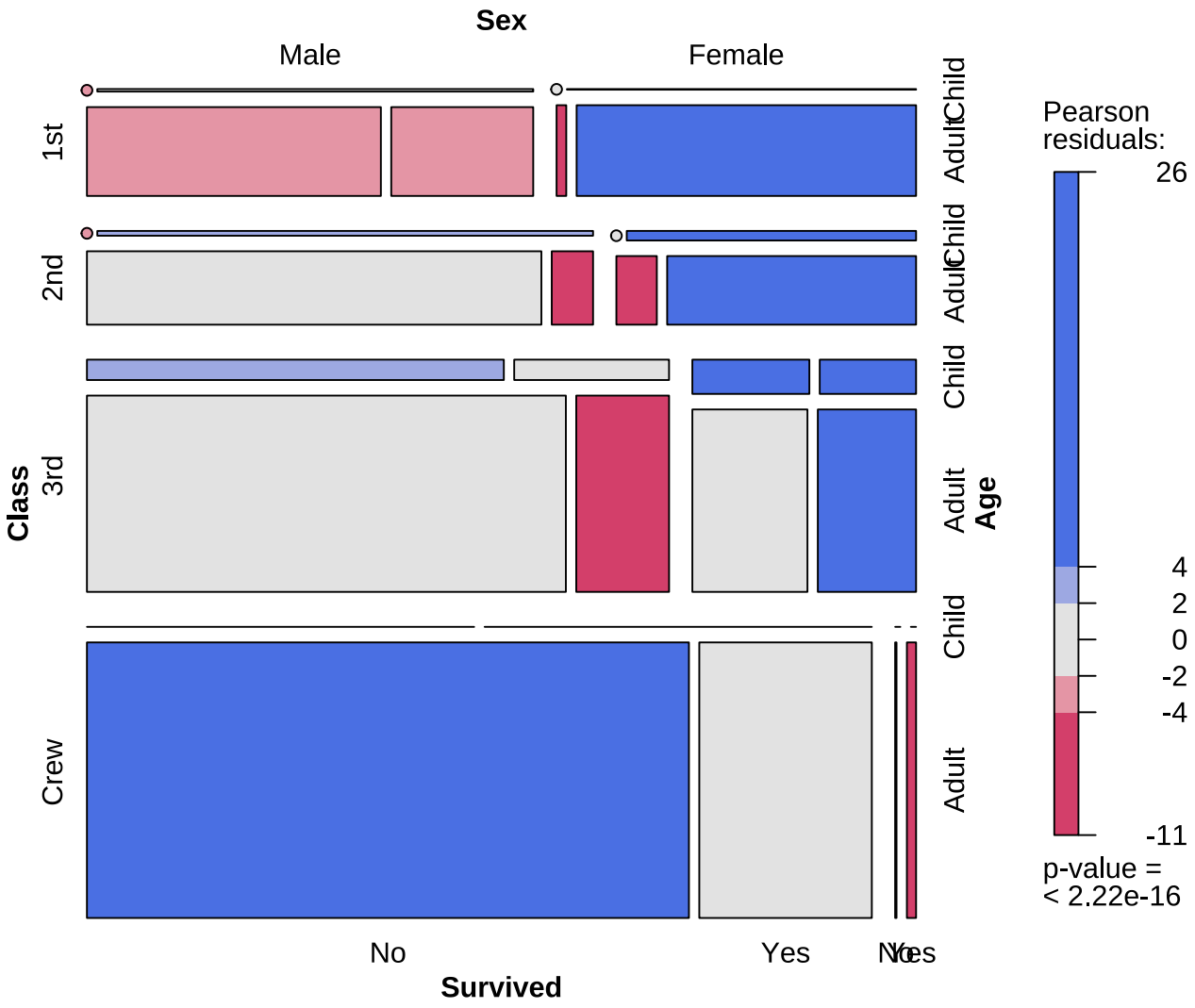


图 21.4: 马赛克图展示多维分类数据

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。

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
#>
#> 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
#>   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。

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
```

[1] https://en.wikipedia.org/wiki/Ronald_Fisher

```
# 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:
#>    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 $2\times2\times6$ 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
```

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
```

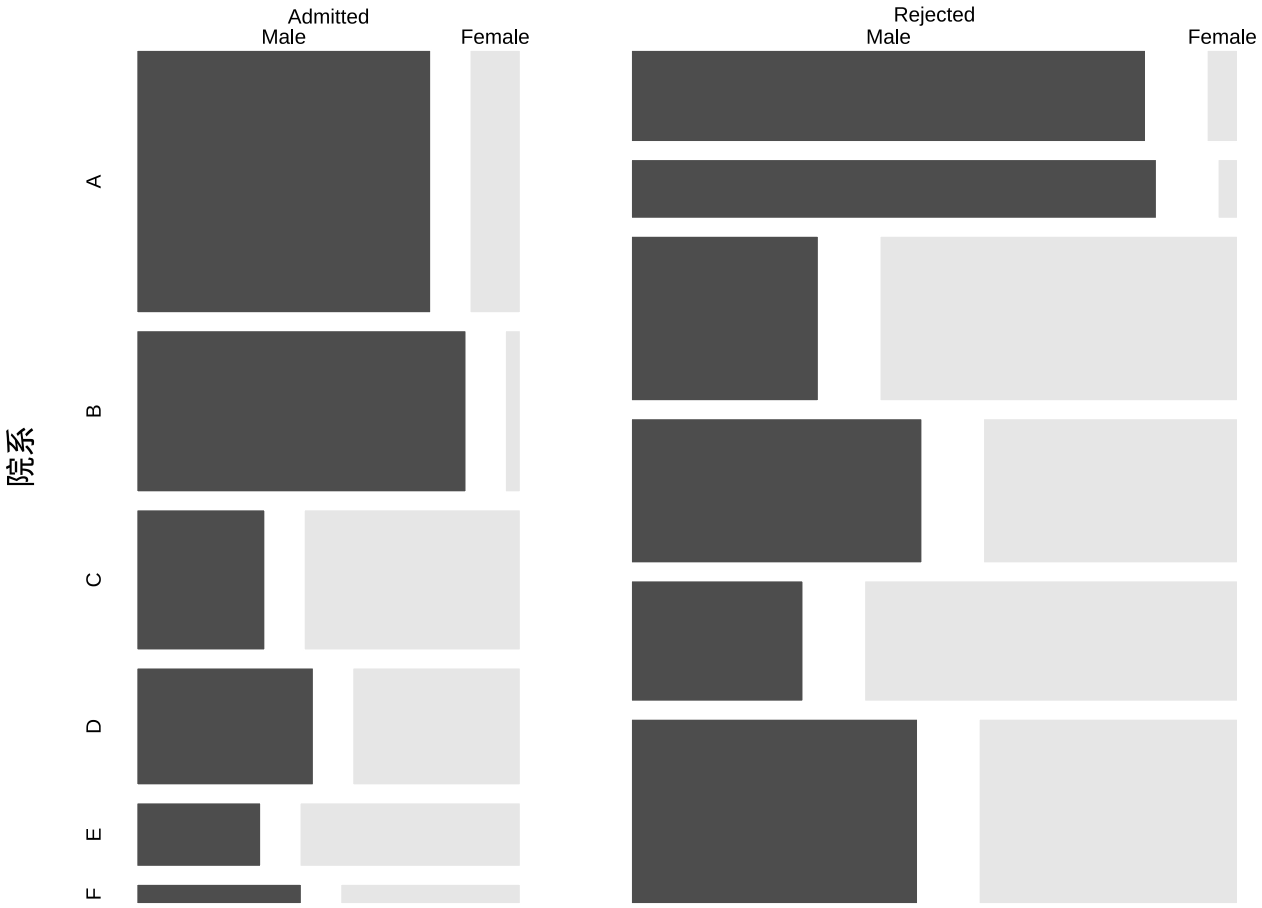


图 21.5: 加州伯克利分校院系录取情况

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：

```
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
```

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",
  data = as.data.frame(UCBAdmissions)
)
chisq.test(ma, correct = FALSE)
```

```
#>
#>  Pearson's Chi-squared test
```

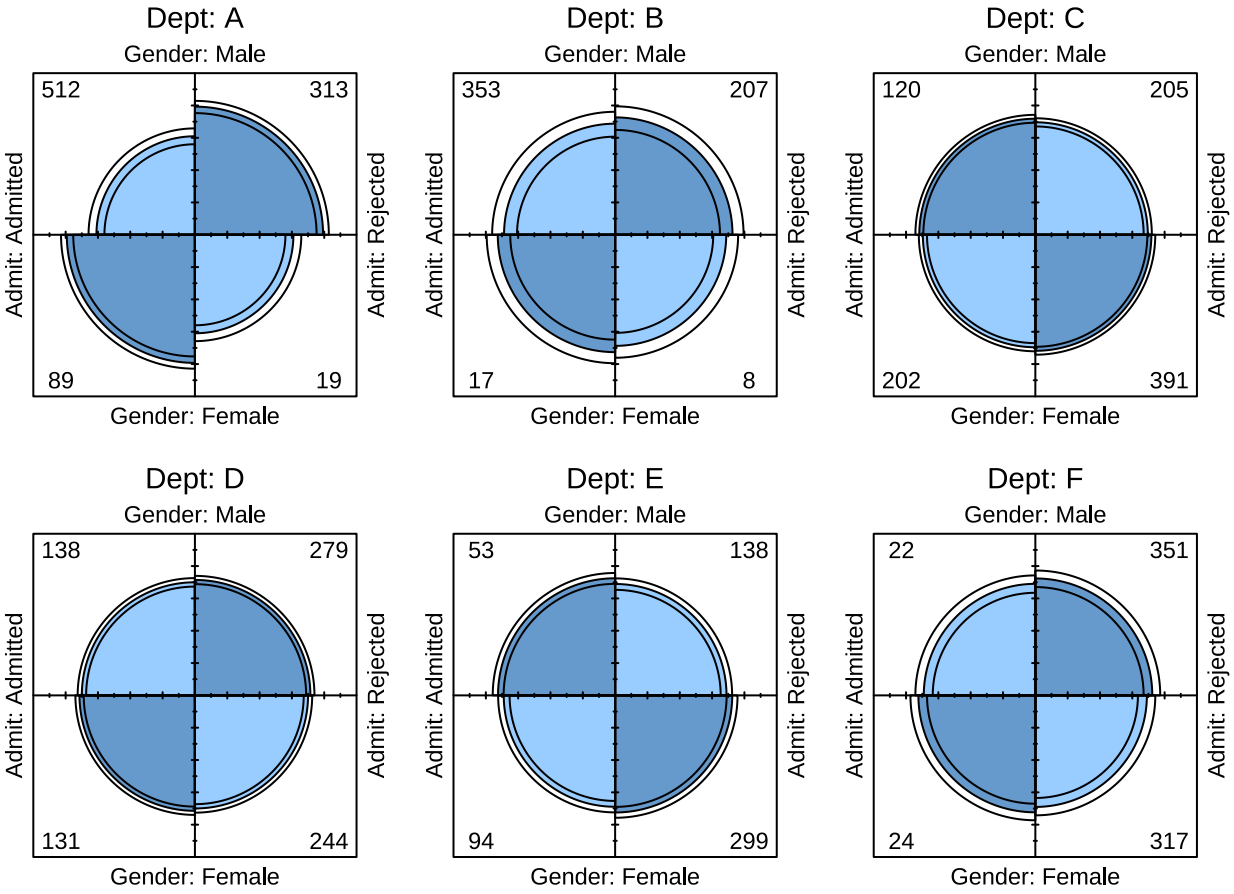


图 21.6: 加州伯克利分校各院系录取情况

```
#>
#> 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
#>
#> 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
#>
#> 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
```

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"
)
```

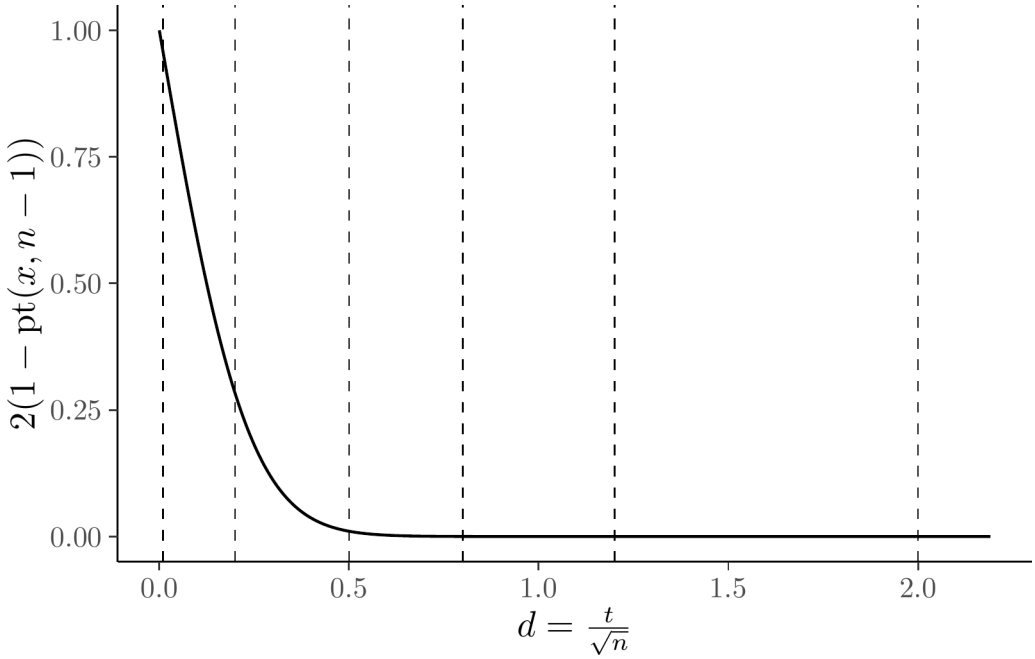


图 22.1: t 检验的功效

```
#>
#>      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

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
```

```r
# 代入计算功效
power.t.test(
  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

```r
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㬎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

```
# 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
```

```
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

```
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"
)
```

```
#>
#>     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
  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
```

第五部分

数据建模

第二十三章　网络分析

23.1　R 语言社区规模

从 CRAN 上的 R 包及其开发者数量来看看目前 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 2022-12-31 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", "License",
                         "Title", "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。

```
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，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。还可以换个说法，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%。

```
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")
```

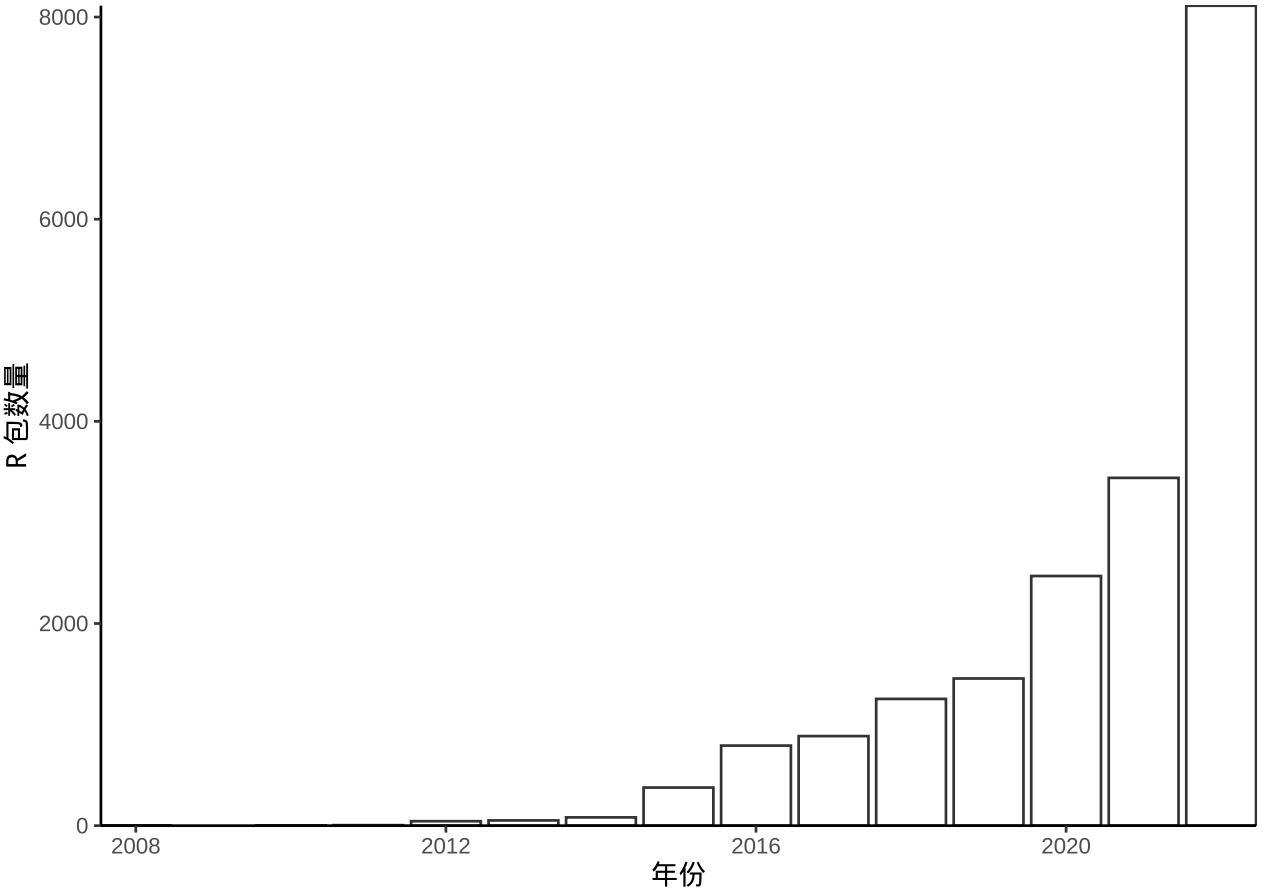


图 23.1: CRAN 上 R 包的更新情况

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
pdb$Maintainer2 <- sub(pattern = "<.*?>", replacement = "", x = pdb$Maintainer)
pdb$Maintainer2 <- trimws(pdb$Maintainer2, which = "both", whitespace = "[ \t\r\n]")
pdb$Maintainer2 <- tolower(pdb$Maintainer2)
# 维护者总数
# length(unique(pdb$Maintainer2))
```

```
aggregate(
  data = pdb, Maintainer2 ~ published_year,
  FUN = function(x) {
    length(unique(x))
  }
) |>
  ggplot(aes(x = published_year, y = Maintainer2)) +
  geom_col(fill = NA, color = "gray20") +
  theme_classic() +
  coord_cartesian(expand = F) +
  labs(x = "年份", y = "开发者数量")
```

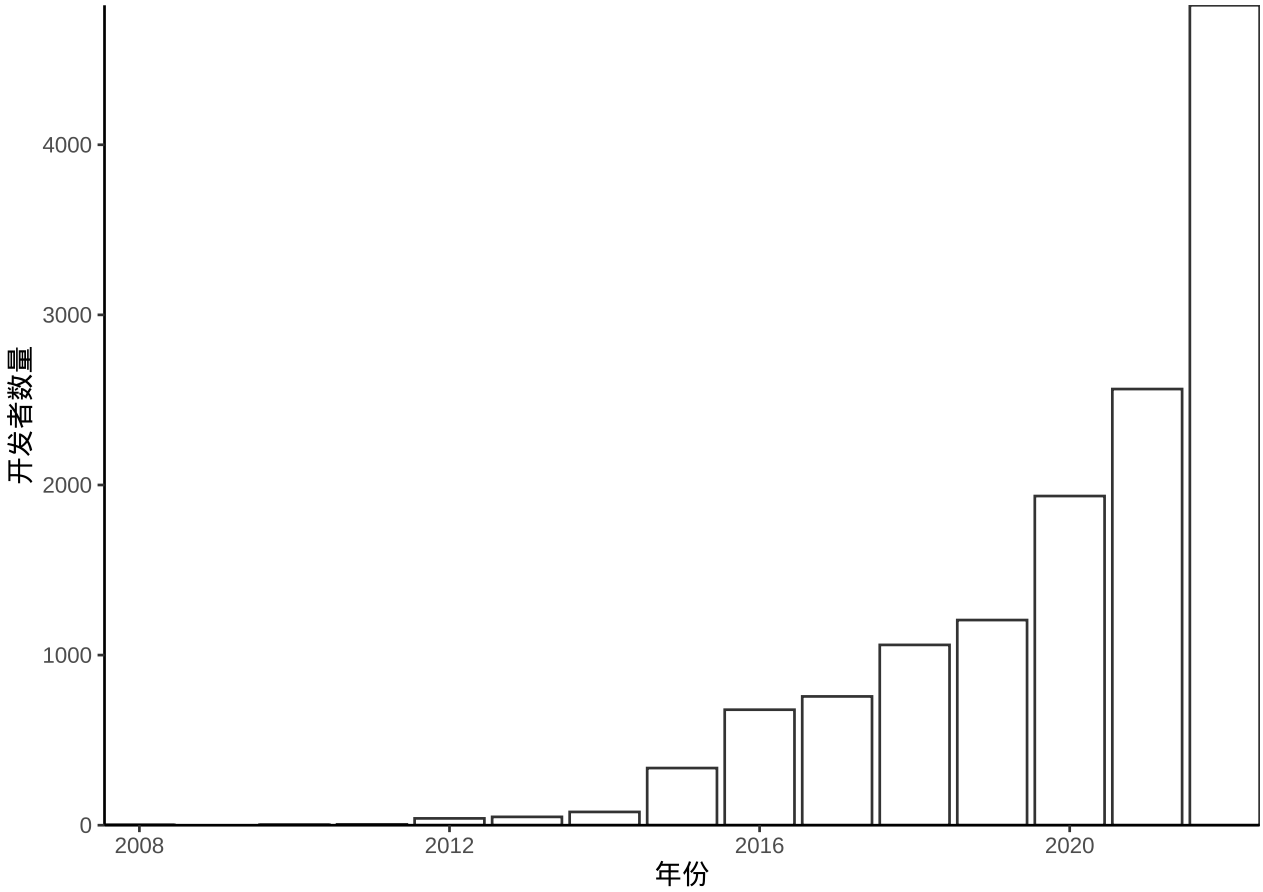


图 23.2: CRAN 上的维护者活跃情况

第二十四章　文本分析

第二十五章　预测股价的变化趋势

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(fGarch)      # ARCH GARCH
library(dygraphs)    # 交互展示

library(nnet)         # 前馈神经网络
library(tensorflow)   # 深度学习框架
library(keras)        # LSTM 模型
```

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。

25.1　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 2022-05-27 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 / 2023-07-14 containing:
  Data:    double [1185, 6]
 Columns: 3690.HK.Open, 3690.HK.High, 3690.HK.Low, 3690.HK.Close, 3690.HK.Volume ... with 1 more column
  Index:   Date [1185] (TZ: "UTC")
  xts Attributes:
    $ src    : chr "yahoo"
    $ updated: POSIXct[1:1], format: "2023-07-15 03:19:34"
```

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 2023-07-14，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 2023-07-15 11:19:34。

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。

```
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。

```
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
xts(x, order.by = idx)
```

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

25.2 数据探索

25.2.1 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")
```

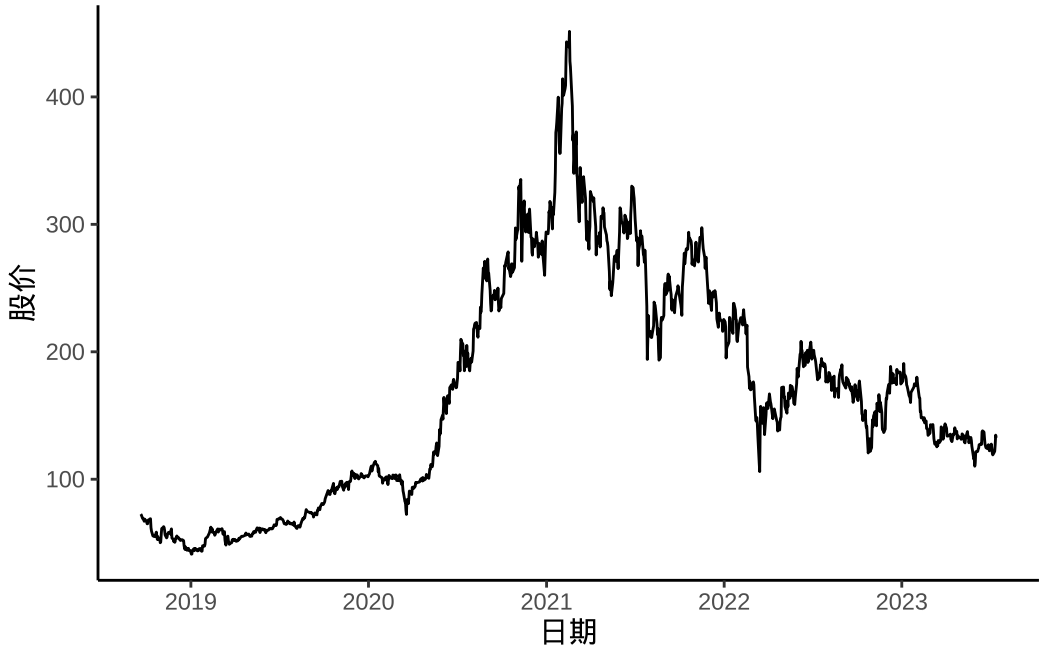


图 25.1: 美团在香港上市以来的股价走势

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。

```
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")
```

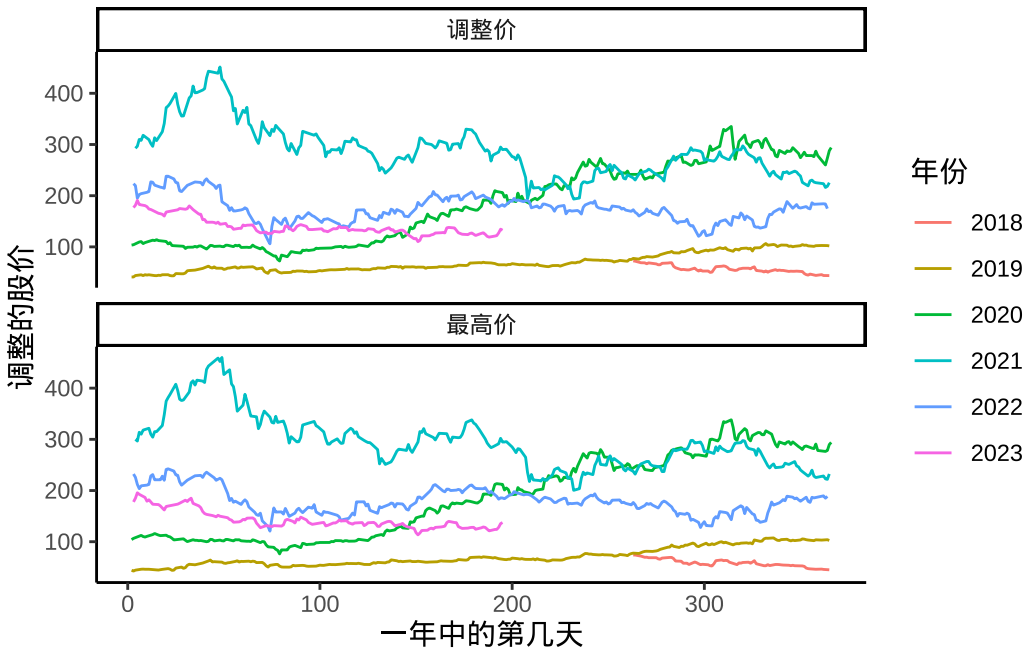


图 25.2: 美团调整的股价逐年走势

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。

25.2.2 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")
```

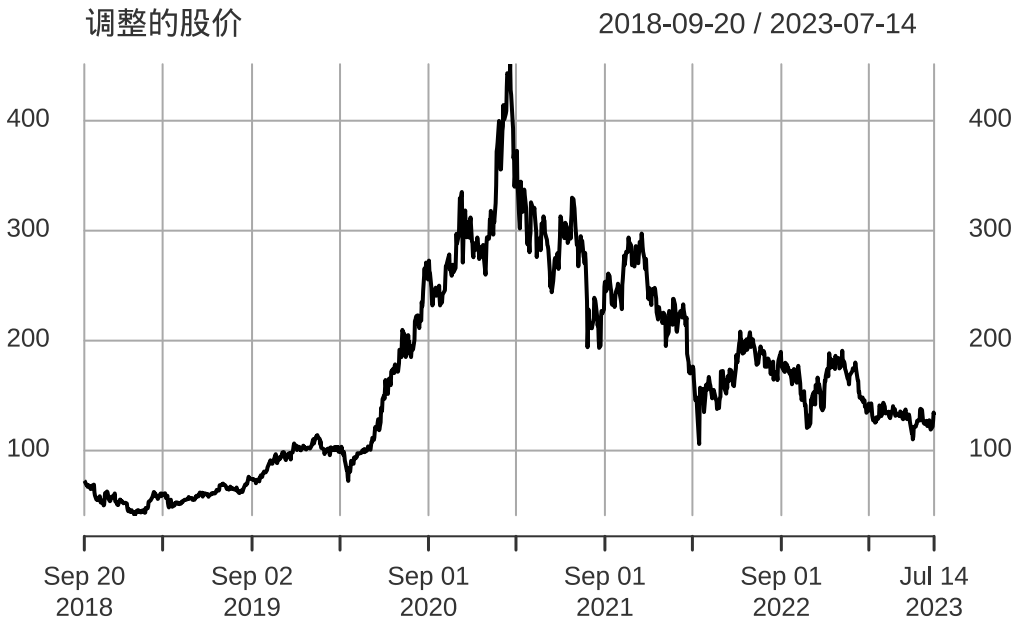


图 25.3: 美团在香港上市以来的股价走势

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"
)
```

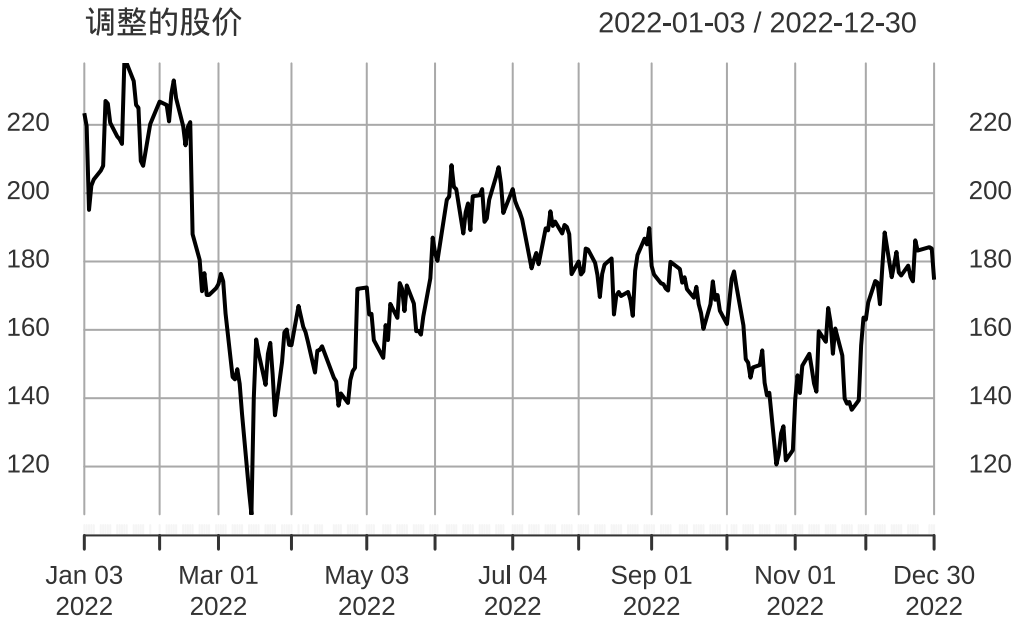


图 25.4: 美团 2021 年的股价走势

元旦节三天不开市，所以假期没有数据。

25.2.3 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()
```

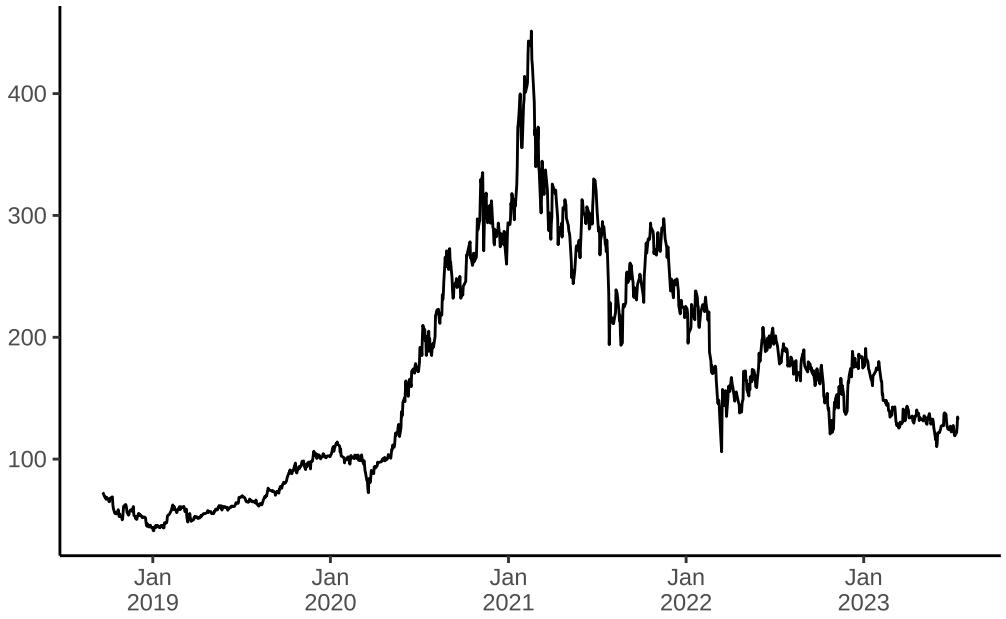


图 25.5: 美团股价走势

25.2.4 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()];
  }'

# 绘图
dygraph(meituan[, "3690.HK.Adjusted"], main = "美团股价走势") |>
  dyRangeSelector(dateWindow = c(format(Sys.Date(), "%Y-01-01"), as.character(Sys.Date()))) |>
  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()
```

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。

图 25.6: 美团股价变化趋势

25.3 平稳性诊断

25.3.1 自相关图

```
autoplot(acf(AirPassengers, plot = FALSE)) +
  theme_classic()
```

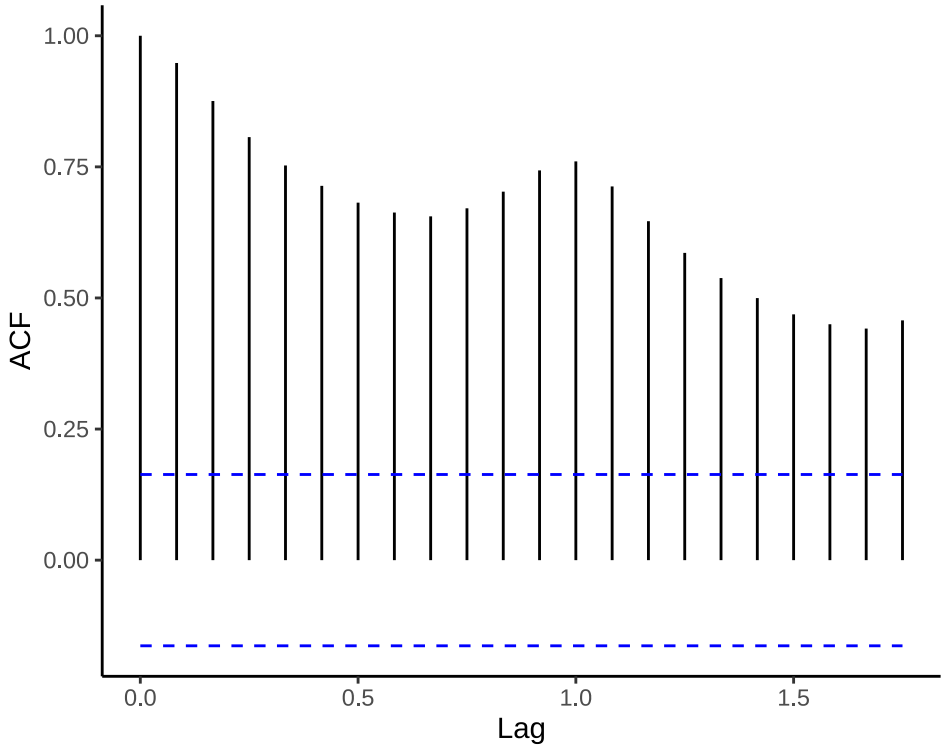


图 25.7: 乘客数量自相关图

25.3.2 偏自相关图

```
autoplot(pacf(AirPassengers, plot = FALSE)) +
  theme_classic()
```

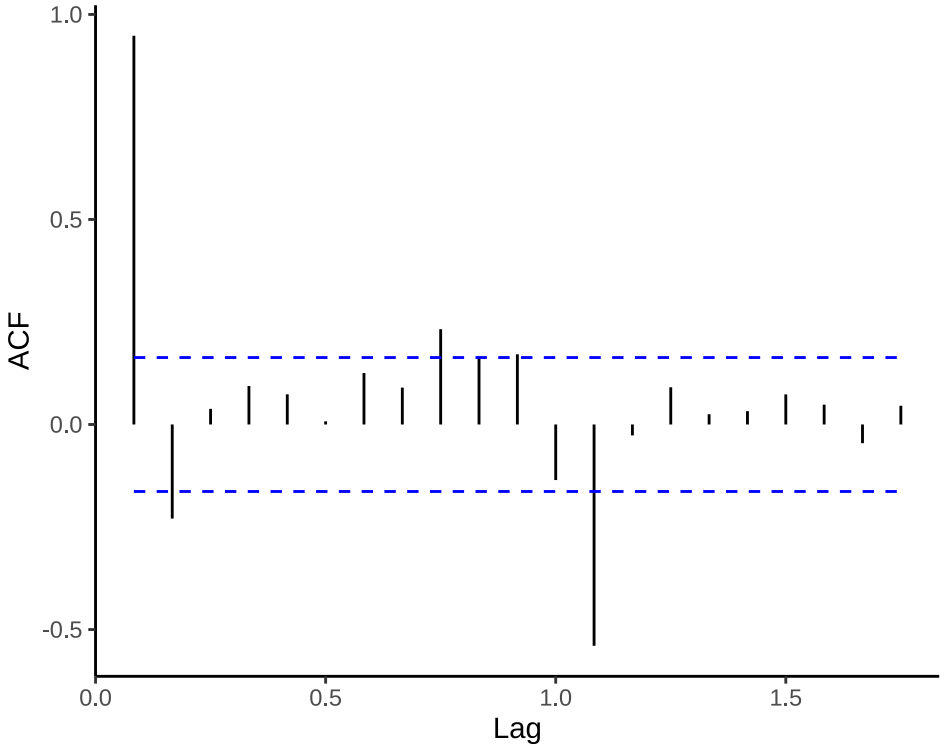


图 25.8: 乘客数量偏自相关图

25.3.3 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)
```

```
     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
```

25.3.4 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。

```
# 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
```

25.3.5 单位根检验

25.3.6 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。

25.4 指数平滑模型

25.4.1 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。

25.4.2 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。

- 递归过滤（自回归）

$$y_i = x_i + f_1 y_{i-1} + \cdots + f_p y_{i-p} \tag{25.1}$$

- 卷积过滤（移动平均）

$$y_i = f_1 x_{i+o} + \cdots + f_p x_{i+o-(p-1)} \tag{25.2}$$

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 25.1 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
```

```
# 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：

$$
\begin{aligned}
&y_1 \text{ 不存在} \\
&y_2 \text{ 不存在} \\
&y_3 = \frac{2}{3}x_3 + \frac{1}{6}x_2 + \frac{1}{6}x_1 \\
&y_i = \frac{2}{3}x_i + \frac{1}{6}x_{i-1} + \frac{1}{6}x_{i-2}, \quad i \geq 3
\end{aligned}
$$

比照模型方程式 25.2 ，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
```

25.4.3 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 (`geom_line()`).
```

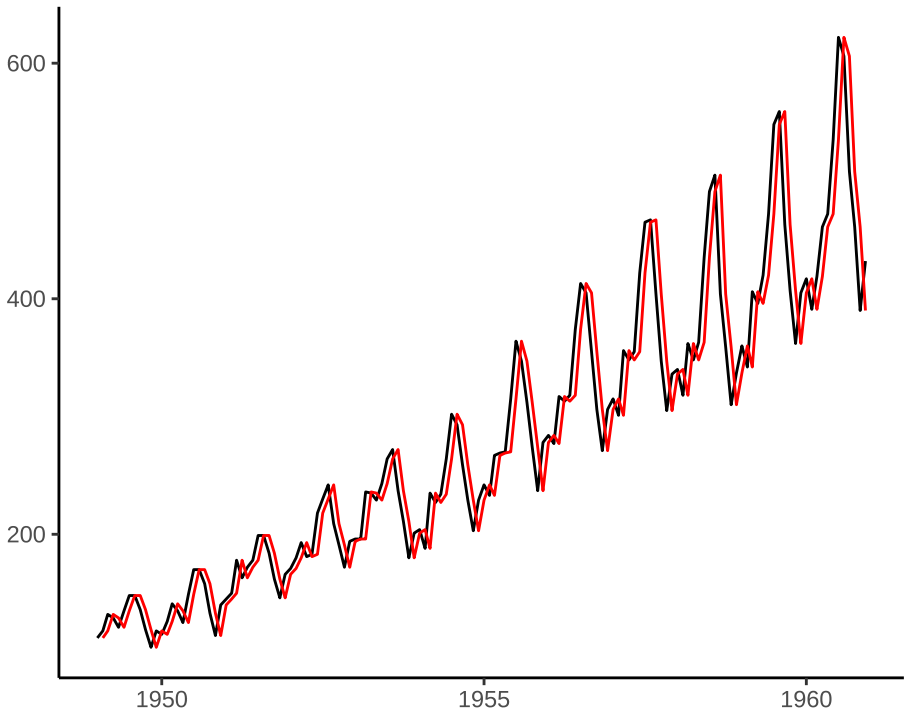


图 25.9: 简单指数平滑模型

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)
```

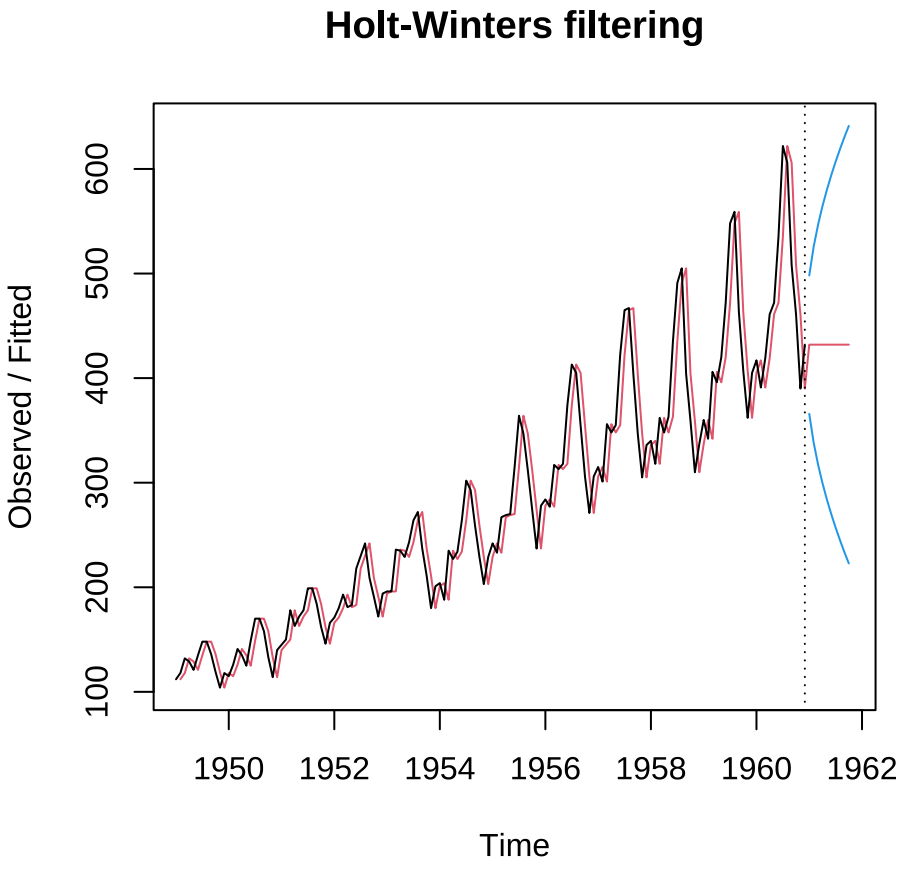


图 25.10: 简单指数平滑模型预测

25.4.4 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:
 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)
```

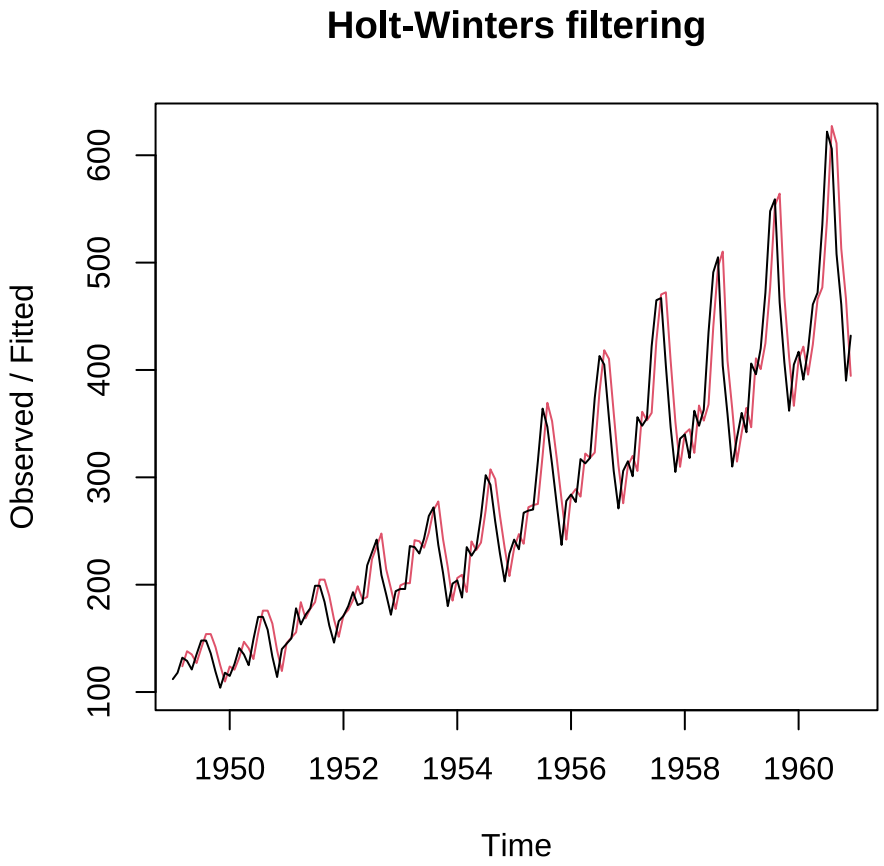


图 25.11: holt 指数平滑模型

25.4.5 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：

$$\begin{aligned}\hat{y}_{t+h} &= a_t + h \times b_t + s_{t-p+1+(h-1) \mod p} \\ a_t &= \alpha(y_t - s_{t-p}) + (1-\alpha)(a_{t-1} + b_{t-1}) \\ b_t &= \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1} \\ s_t &= \gamma(y_t - a_t) + (1-\gamma)s_{t-p}\end{aligned}$$

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，$\alpha = 0.248, \beta = 0.0345, \gamma = 1$

```
autoplot(air_passengers_add) +
  theme_classic()
```

```
Warning: Removed 12 rows containing missing values (`geom_line()`).
```

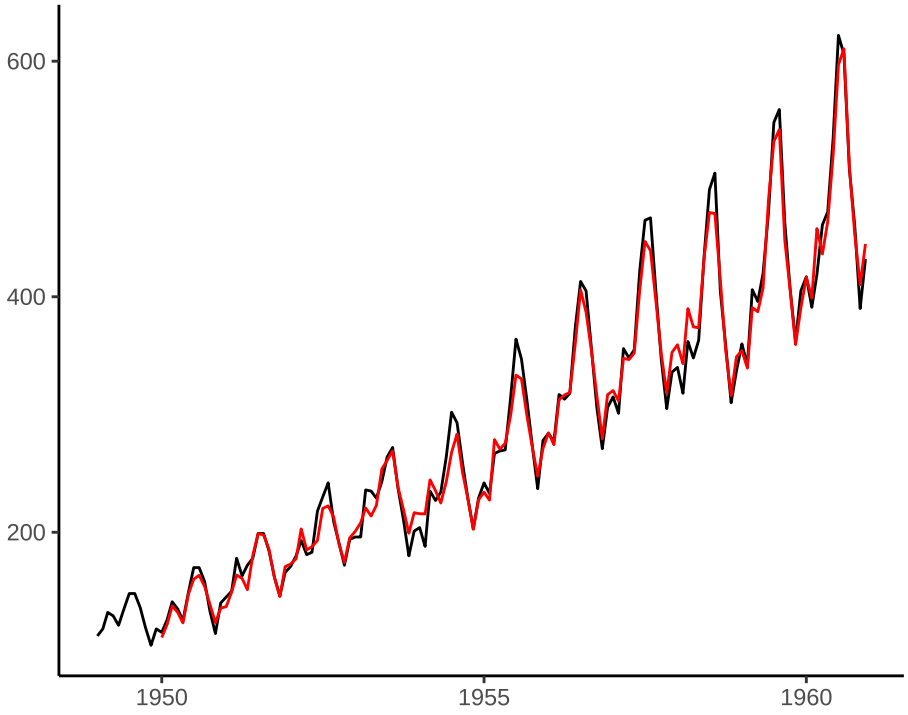


图 25.12: 可加 Holt-Winters 平滑模型拟合

```
air_passengers_mult <- HoltWinters(AirPassengers, seasonal = "mult")
```

```
autoplot(air_passengers_mult) +
  theme_classic()
```

```
Warning: Removed 12 rows containing missing values (`geom_line()`).
```

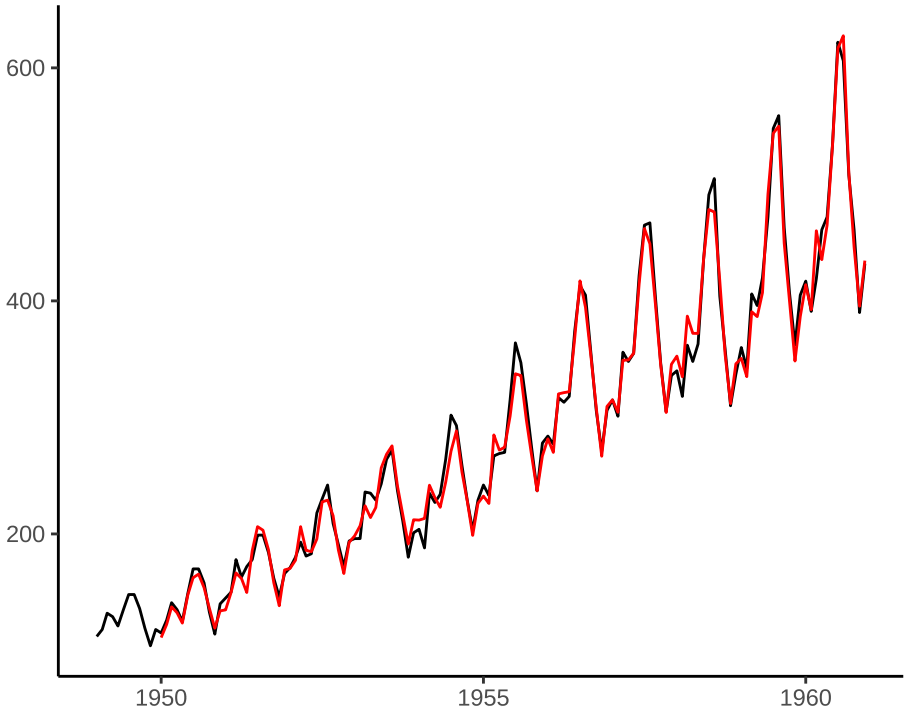


图 25.13: 可乘 Holt-Winters 平滑模型拟合

做一个 Shiny 应用展示参数 α, β, γ 对 Holt-Winters 平滑预测的影响。

25.5 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

25.5.1 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 (`geom_line()`).
```

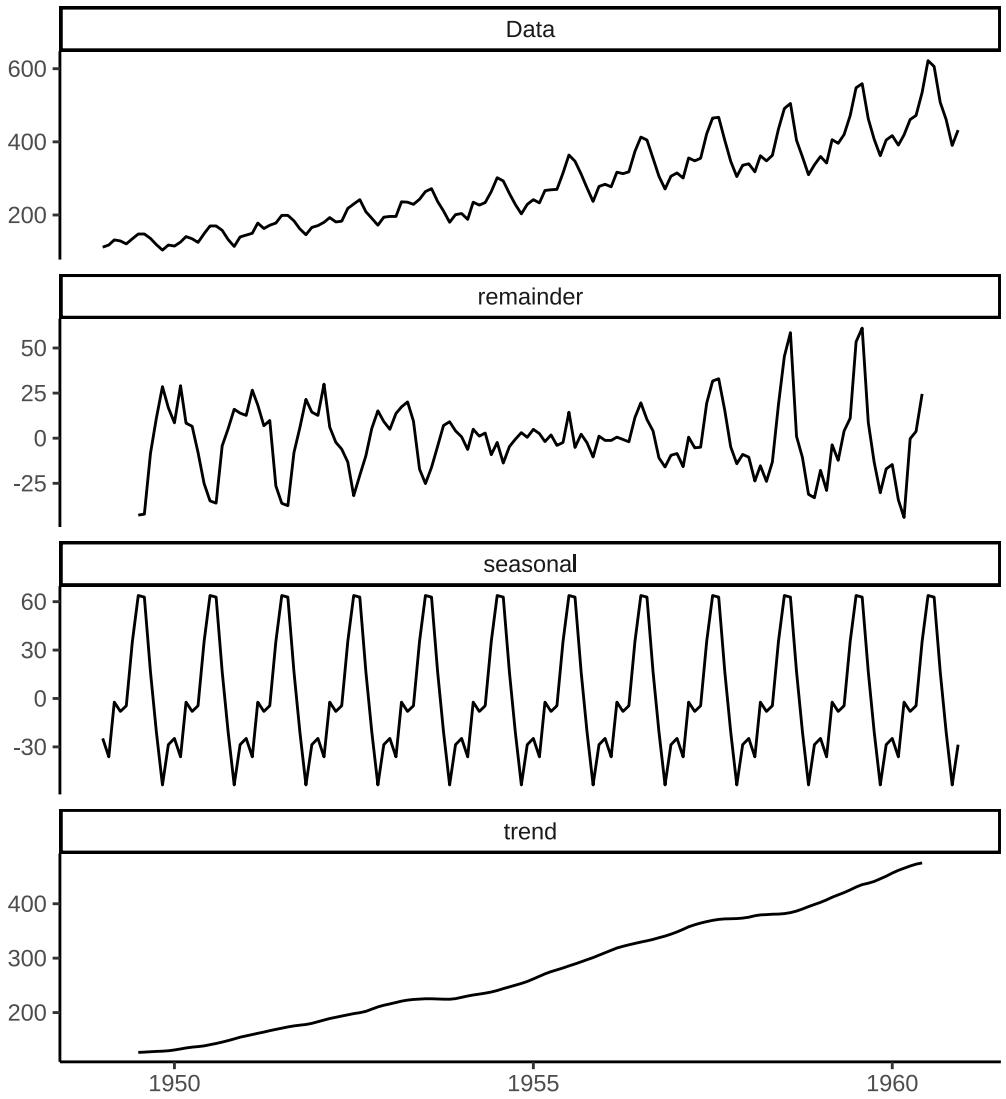


图 25.14: 变化趋势的分解

去掉季节性部分

```
AirPassengers_adjusted <- AirPassengers - air_decomp_add$seasonal
plot(AirPassengers_adjusted)
```

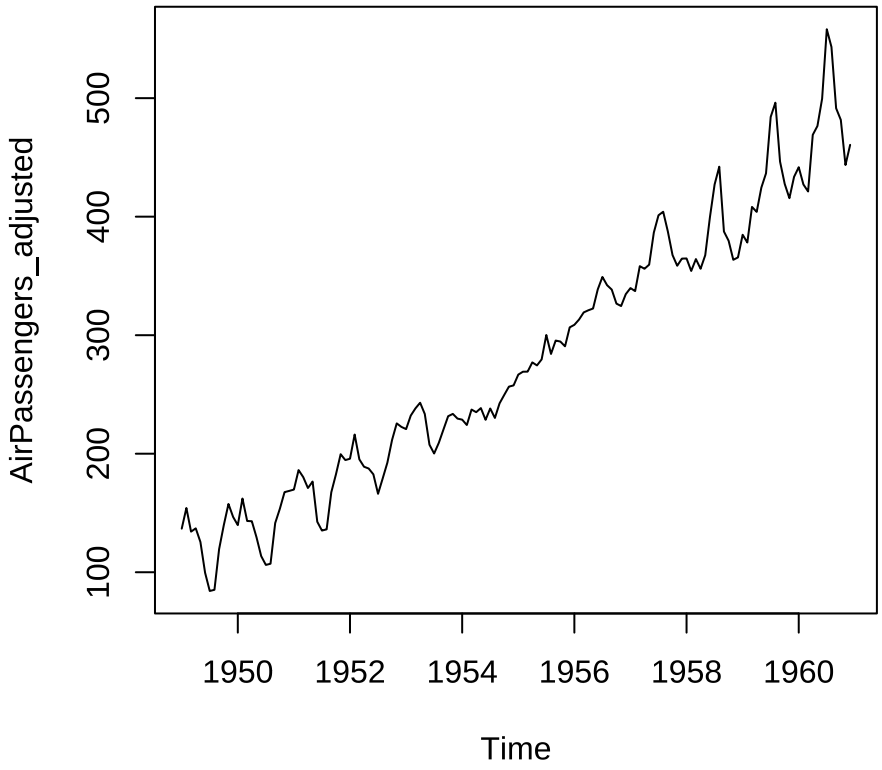


图 25.15: 季节性调整

25.5.2 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()
```

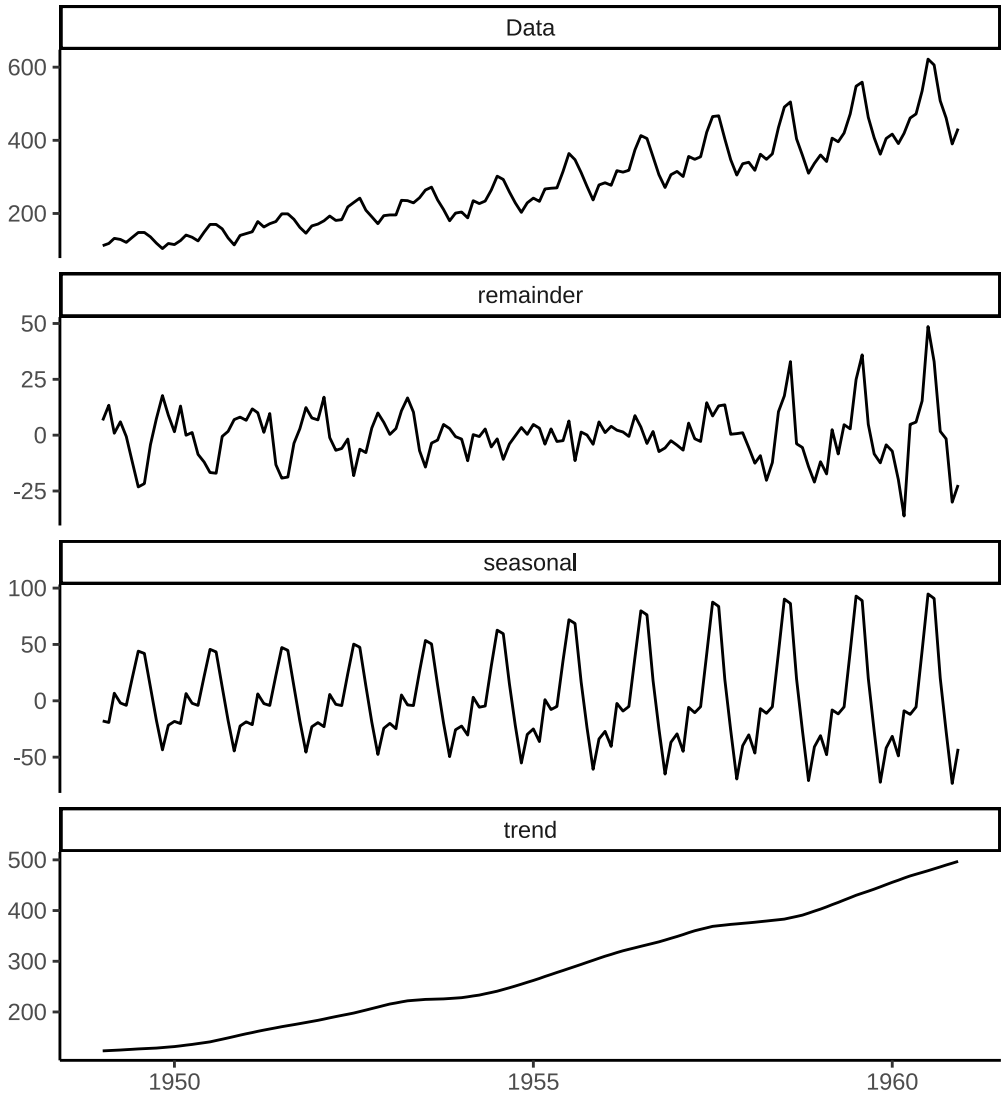


图 25.16: 变化趋势的分解

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
```

25.6 经典时间序列模型

25.6.1 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
```

25.6.2 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
```

25.6.3 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
```

25.6.4 自回归条件异方差模型

《金融时间序列分析讲义》

自回归条件异方差模型（Autoregressive Conditional Heteroskedasticity，简称 ARCH）

```
# library(fGarch)
```

25.6.5 广义自回归条件异方差模型

广义自回归条件异方差模型（Generalized Autoregressive Conditional Heteroskedasticity，简称 GARCH）

```
# library(fGarch)
# garchFit
```

25.7 机器学习算法

forecastML 结合机器学习方法的自回归模型，可以一次向前预测多期。

25.8 神经网络算法

25.8.1 多层感知机

多层感知机是一种前馈神经网络，**nnet** 包的函数 `nnet()` 实现了单隐藏层的简单神经网络。

```
library(nnet)
```

25.8.2 长短期记忆神经网络

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。下面用长短期记忆神经网络来训练美团股价数据，预测未来一周的股价趋势。

```
# library(tensorflow)
# library(keras)
# tf$abs(x = c(-1, 1, 2))
```

25.9 干预效应处理

25.9.1 双重差分法

CausalImpact 借助贝叶斯分析方法推断时间序列中的因果关系，比如广告促销带来的点击效果。根据时间序列数据，度量干预作用的影响。

25.9.2 倾向性得分

MatchIt

25.10 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。

第二十六章　预测核辐射强度的分布

本章内容属于空间分析的范畴，空间分析的内容十分广泛，本章仅以一个模型和一个数据简略介绍。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。

26.1　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 26.1 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 26.2a 右上角的位置。采样点的编号见下图 26.2b，基本上按照从下（南）到上（北），从左（西）到右（东）的顺序依次测量。

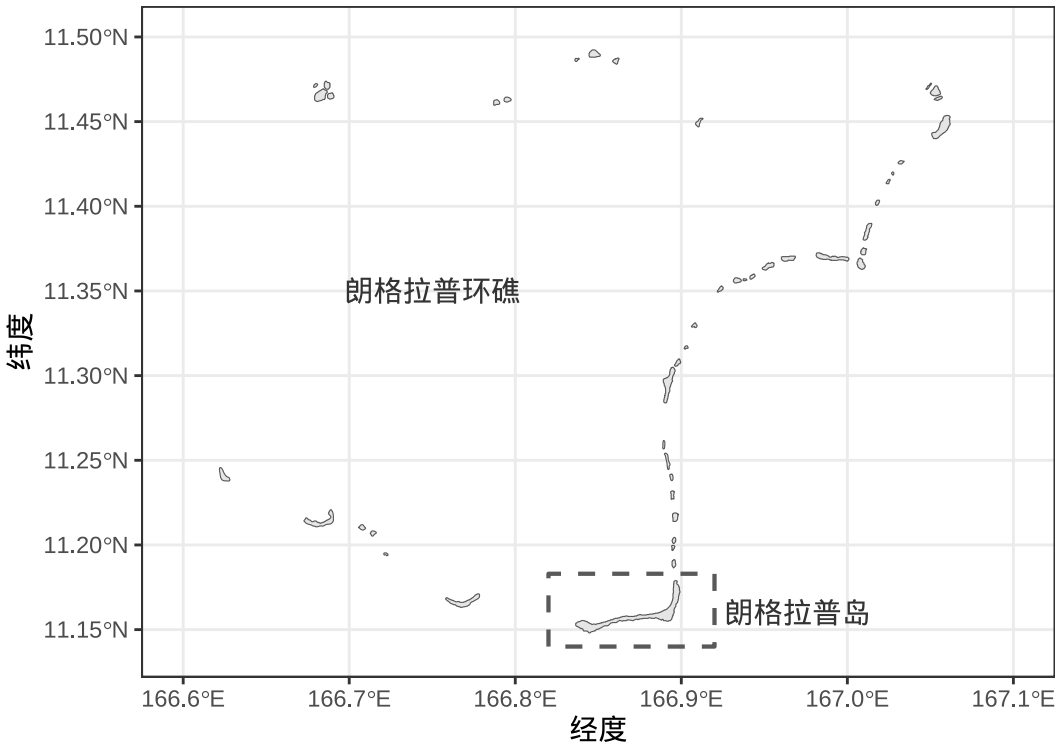


图 26.1: 朗格拉普环礁和朗格拉普岛

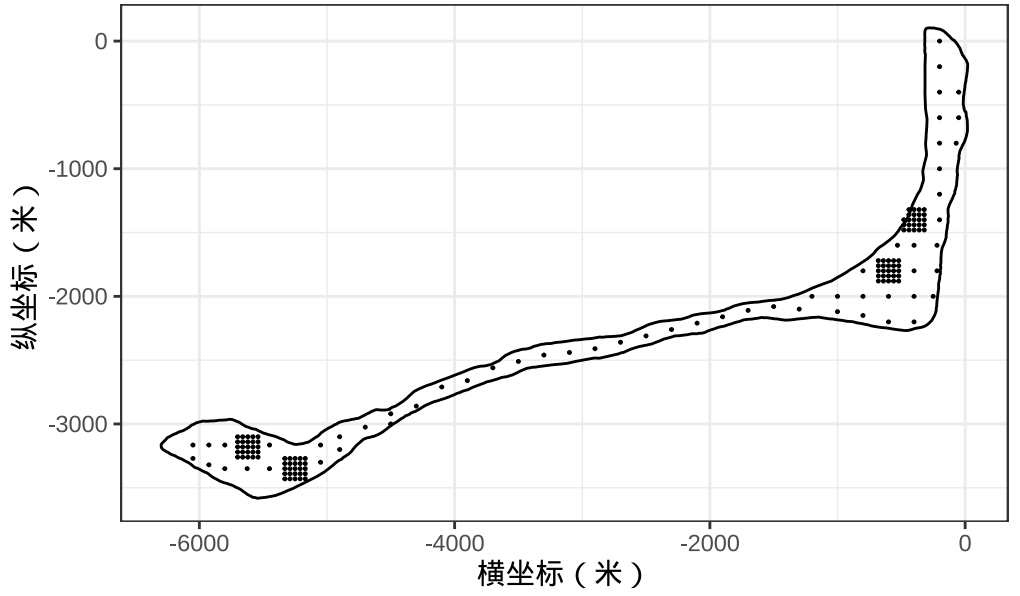
表格 26.1: 朗格拉普岛核辐射检测数据及海岸线坐标数据

(a) 核辐射检测数据

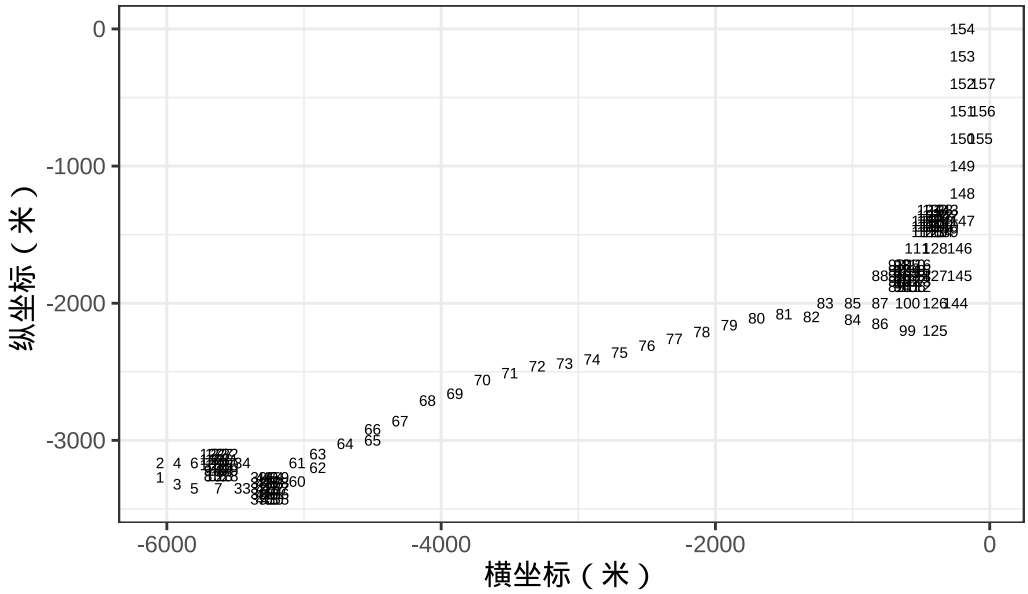
cX 横坐标	cY 纵坐标	counts 数目	time 时间
-6050	-3270	75	300
-6050	-3165	371	300
-5925	-3320	1931	300
-5925	-3165	4357	300
-5800	-3350	2114	300
-5800	-3165	2318	300

(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



(a) 采样分布



(b) 采样顺序

图 26.2: 采样点在岛上的分布

26.2 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 26.2 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 26.3 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。

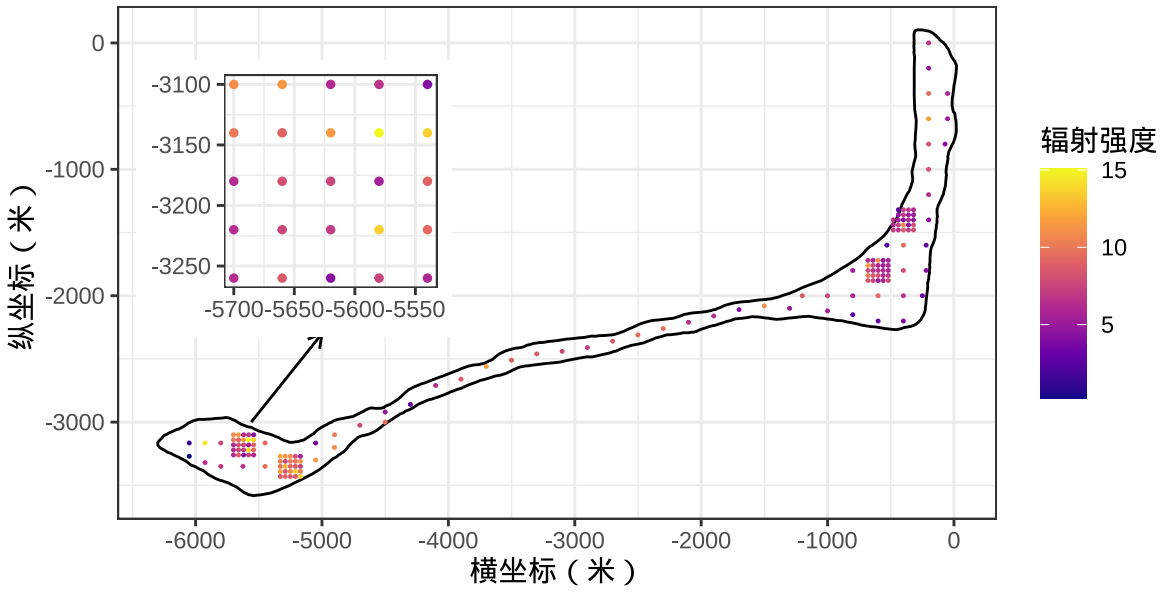


图 26.3: 岛上各采样点的核辐射强度

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 26.4 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。

26.3 数据建模

26.3.1 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。

$$\log(\lambda_i) = \beta$$
$$y_i \sim \text{Poisson}(t_i \lambda_i)$$

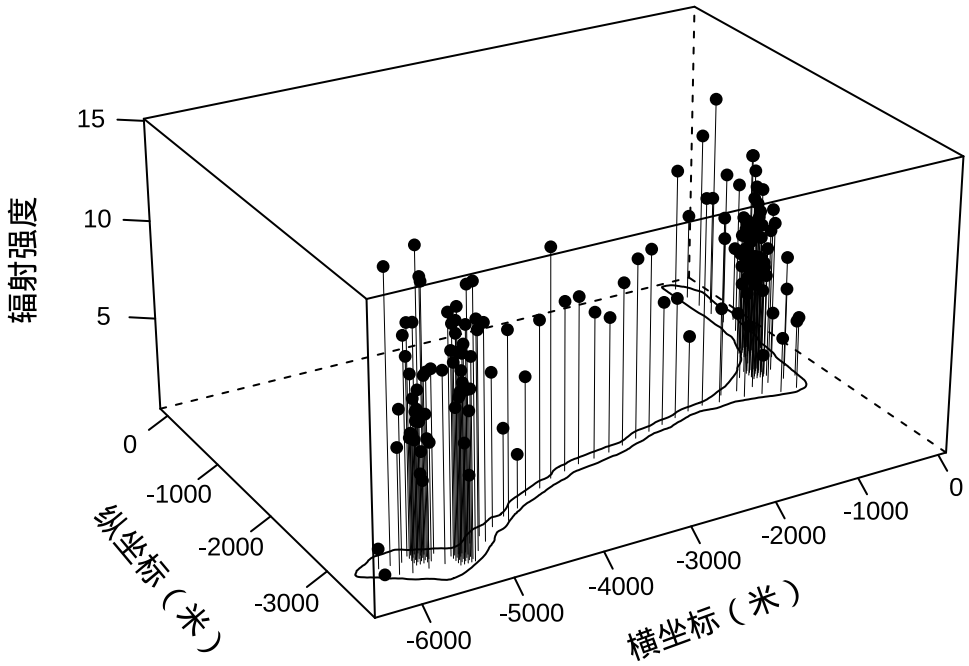


图 26.4: 岛上各采样点的辐射强度

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
#> 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 26.5 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")
```

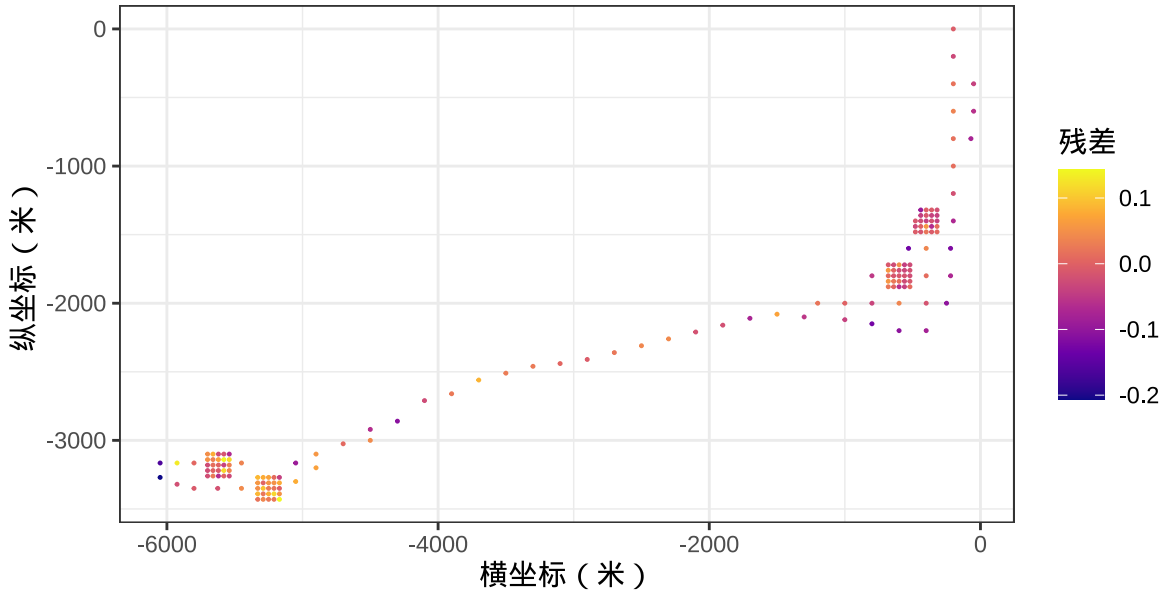


图 26.5: 残差的空间分布

图 26.6 描述残差的分布，从图 26.6a 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 26.6b 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 26.6c 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。

```
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")
```

26.3.2 空间线性混合效应模型

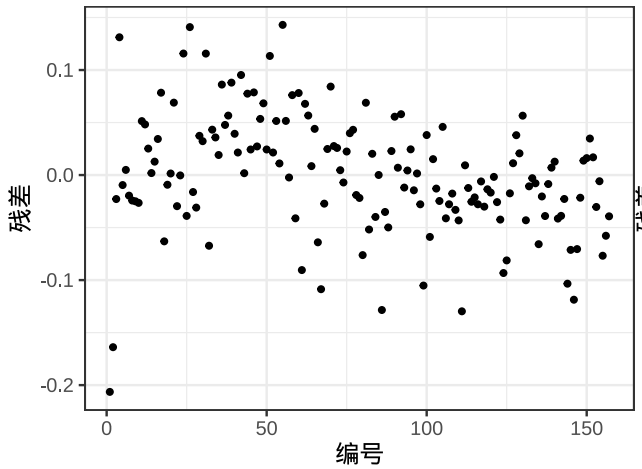
从实际场景出发，也不难理解，位置信息是非常关键的。进一步，充分利用位置信息，精细建模是很有必要的。相邻位置的核辐射强度是相关的，离得近的比离得远的更相关。下面对辐射强度建模，假定随机效应之间存在相关性结构，去掉随机效应相互独立的假设，这更符合位置效应存在相互影响的实际情况。

$$\log\left(\lambda(x_i)\right) = \beta + S(x_i) + Z_i \tag{26.1}$$

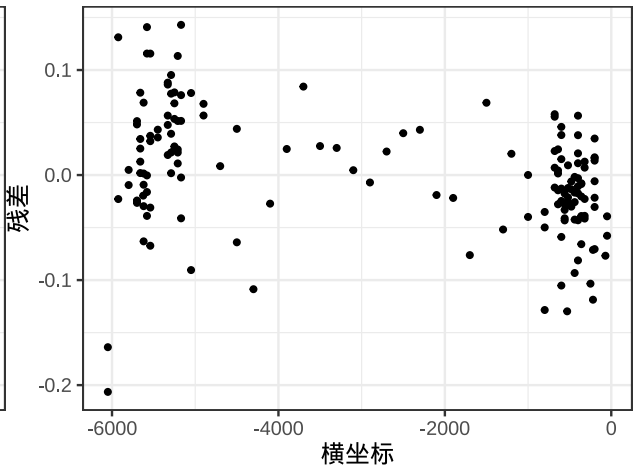
其中，β 表示截距，相当于平均水平，$\lambda(x_i)$ 表示位置 x_i 处的辐射强度，$S(x_i)$ 表示位置 x_i 处的空间效应，$S(x), x \in \mathcal{D} \subset \mathbb{R}^2$ 是二维平稳空间高斯过程 $\mathcal{S}$ 的具体实现。$\mathcal{D}$ 表示研究区域，可以理解为朗格拉普岛，它是二维实平面 $\mathbb{R}^2$ 的子集。Z_i 之间相互独立同正态分布 $\mathcal{N}(0, \tau^2)$，Z_i 表示非空间的随机效应，在空间统计中，常称之为块金效应，可以理解为测量误差、空间变差或背景辐射。值得注意，此时，块金效应和模型残差是合并在一起的。

26.3.2.1 自协方差函数

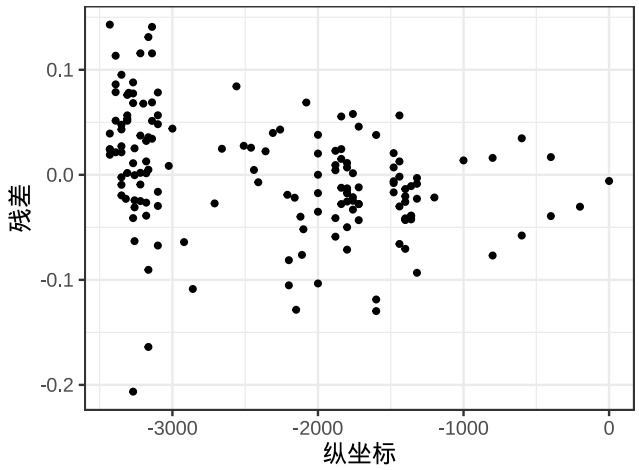
随机过程 $S(x)$ 的自协方差函数常用的有指数型、幂二次指数型（高斯型）和梅隆型，形式如下：



(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 26.6: 残差分布图

$$
\begin{aligned}
\mathrm{Cov}\{S(x_i), S(x_j)\} &= \sigma^2 \exp\left(-\frac{\|x_i - x_j\|_2}{\phi}\right) \\
\mathrm{Cov}\{S(x_i), S(x_j)\} &= \sigma^2 \exp\left(-\frac{\|x_i - x_j\|_2^2}{2\phi^2}\right) \\
\mathrm{Cov}\{S(x_i), S(x_j)\} &= \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{\|x_i - x_j\|_2}{\phi}\right)^{\nu} K_\nu \left(\sqrt{2\nu}\frac{\|x_i - x_j\|_2}{\phi}\right) \\
K_\nu(x) &= \int_0^\infty \exp(-x\cosh t)\cosh(\nu t)\mathrm{dt}
\end{aligned}
\tag{26.2}
$$

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。

$$
\mathrm{Cov}\{S(x_i), S(x_j)\} = \sigma^2 \rho(u_{ij})
$$

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 26.7 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。

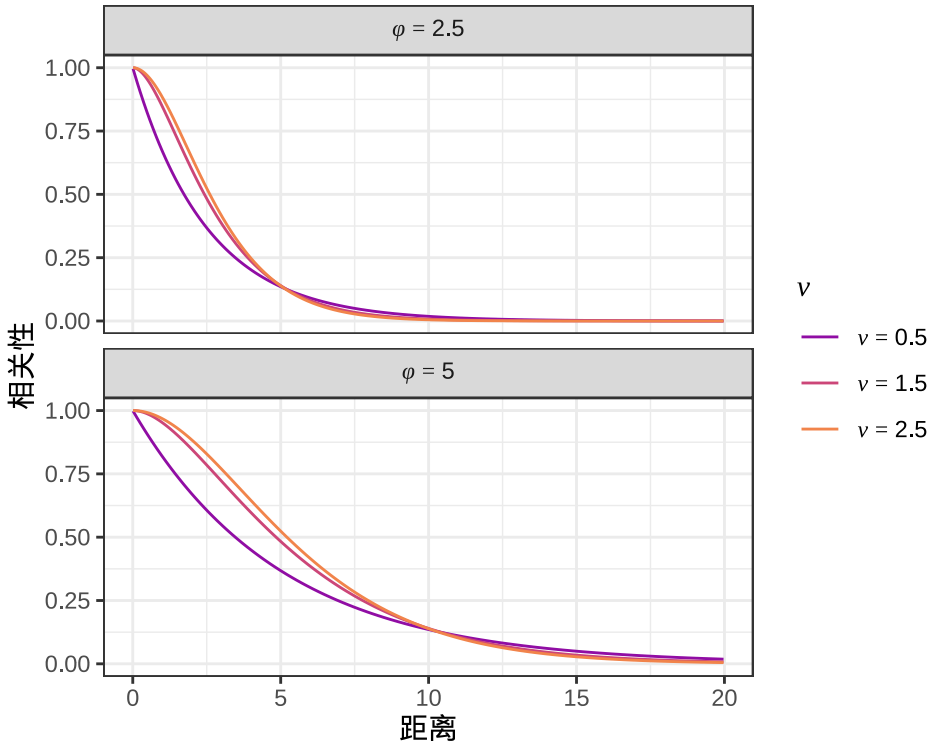


图 26.7: 梅隆型自相关函数曲线

26.3.2.2 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：

$$\rho(u;\phi,\tau_{rel}^2)=\tau_{rel}^2+(1-\tau_{rel}^2)(1-\exp(-\frac{u}{\phi}))$$

为了方便参数估计，**nlme** 包对参数做了一些重参数化的操作。

$$\begin{aligned}\tau_{rel}^2&=\frac{\tau^2}{\tau^2+\sigma^2}\\ \sigma_{tol}^2&=\tau^2+\sigma^2\end{aligned}\tag{26.3}$$

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]
#> [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
```

26.3.2.3 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。

```
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
#> 169.7472088   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 26.3 ，可以得到 τ^2 和 σ^2 的估计。

$$\tau^2 = \tau^2_{rel} \times \sigma^2_{tol} = 0.1092496 \times 0.3294383 = 0.035991$$
$$\sigma^2 = \sigma^2_{tol} - \tau^2_{rel} \times \sigma^2_{tol} = 0.5739672^2 - 0.1092496 \times 0.3294383 = 0.2934473$$

26.3.2.4 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 26.1 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

$$\gamma(u_{ij}) = \frac{1}{2}\mathsf{E}\big\{\big[T(x_i) - T(x_j)\big]^2\big\} = \tau^2 + \sigma^2\big(1 - \rho(u_{ij})\big) \tag{26.4}$$

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 26.4 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 26.8 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
#> 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。

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
```

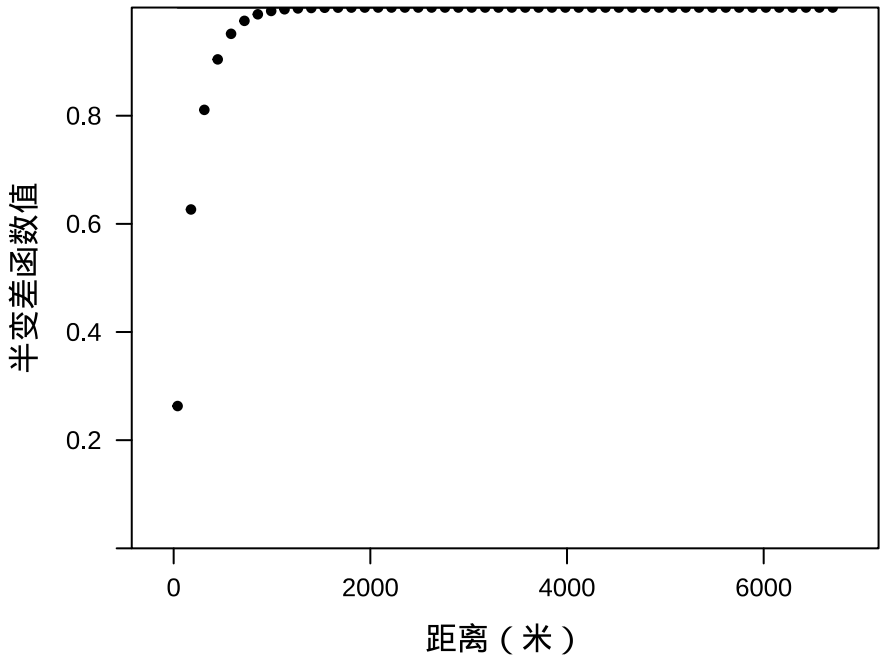


图 26.8: 理论半变差函数图

```
#> 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
#> 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 26.9 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。

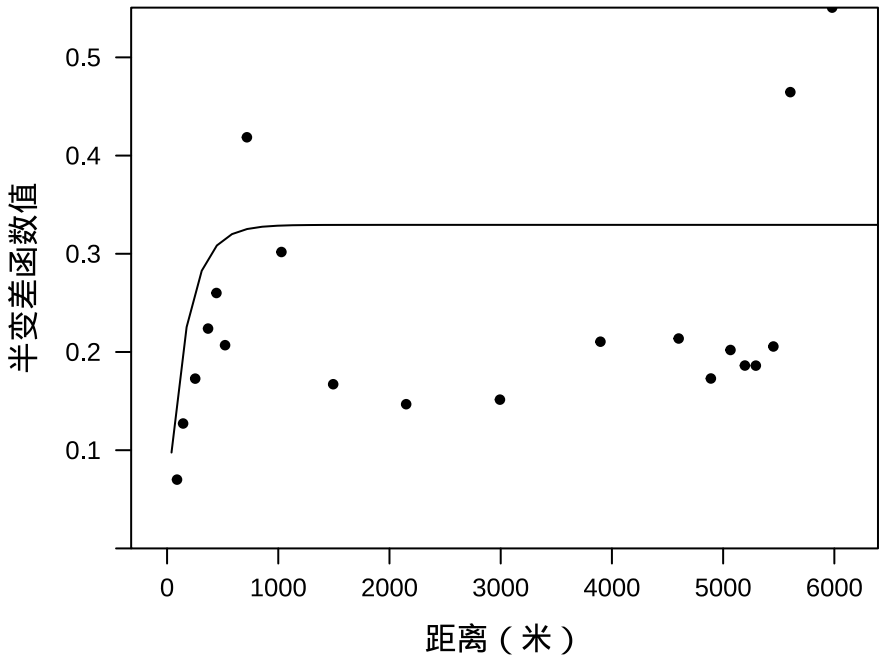


图 26.9: 残差的经验半变差图

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 26.10 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。

26.3.3 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。

$$\begin{aligned} \log\{\lambda(x_i)\} &= \beta + S(x_i) + Z_i \\ y(x_i) &\sim \text{Poisson}\big(t_i\lambda(x_i)\big) \end{aligned} \tag{26.5}$$

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。

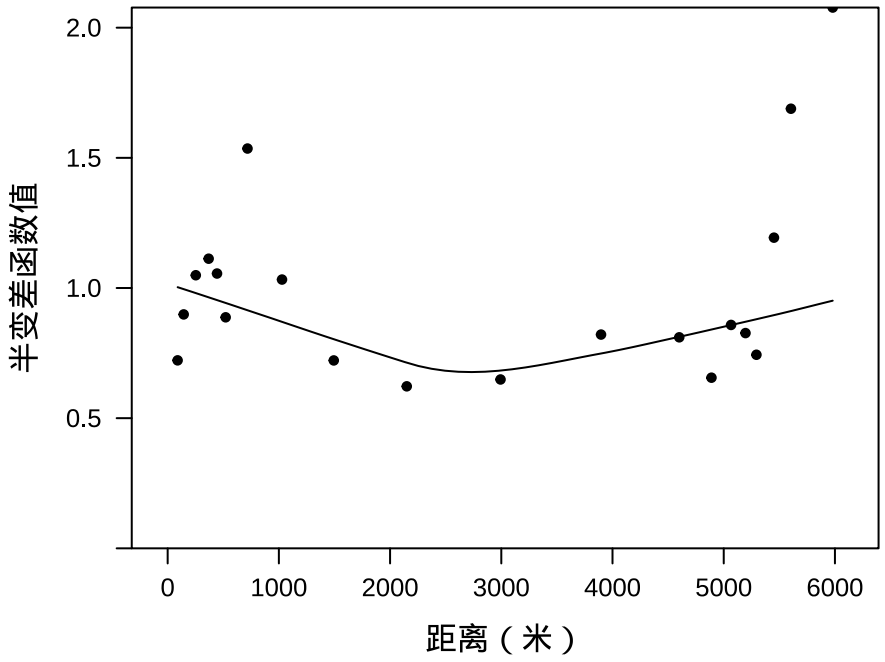


图 26.10: 标准化残差的经验半变差图

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,
  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 0.009211764
#>            --- 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 26.2 ，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。

26.4 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。

26.4.1 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。

```
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 26.11 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()
```

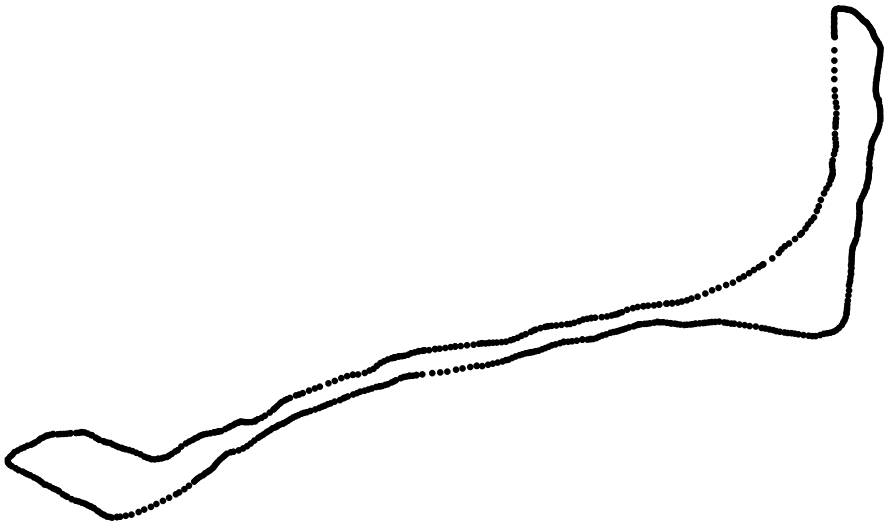
26.4.2 边界处理

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 26.12 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") +
  geom_sf(data = rongelap_coastline_buffer, fill = NA, color = "black") +
  theme_void()
```



(a) 点数据

(b) 多边形数据

图 26.11: 朗格拉普岛海岸线的表示

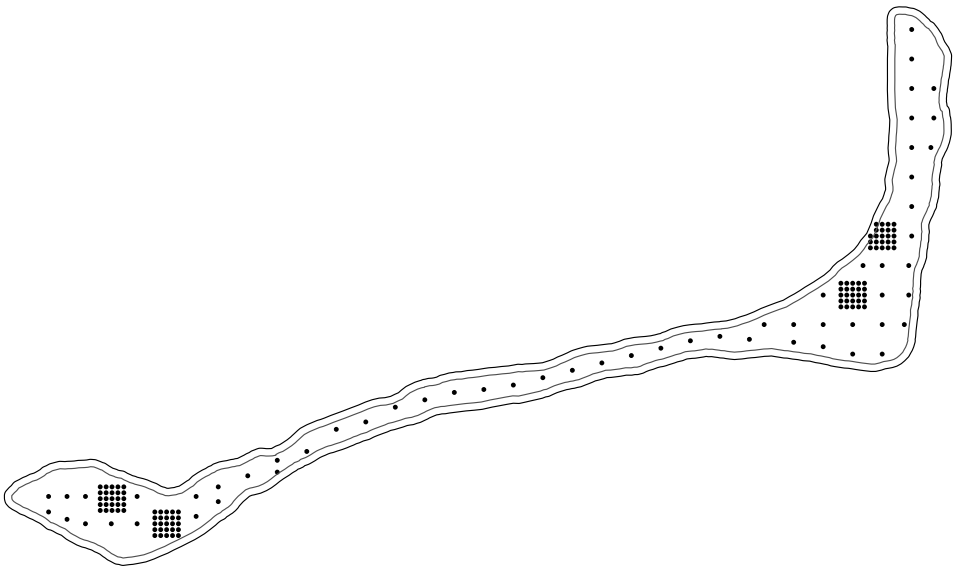


图 26.12: 朗格拉普岛海岸线及其缓冲区

26.4.3 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 33.7 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()
```

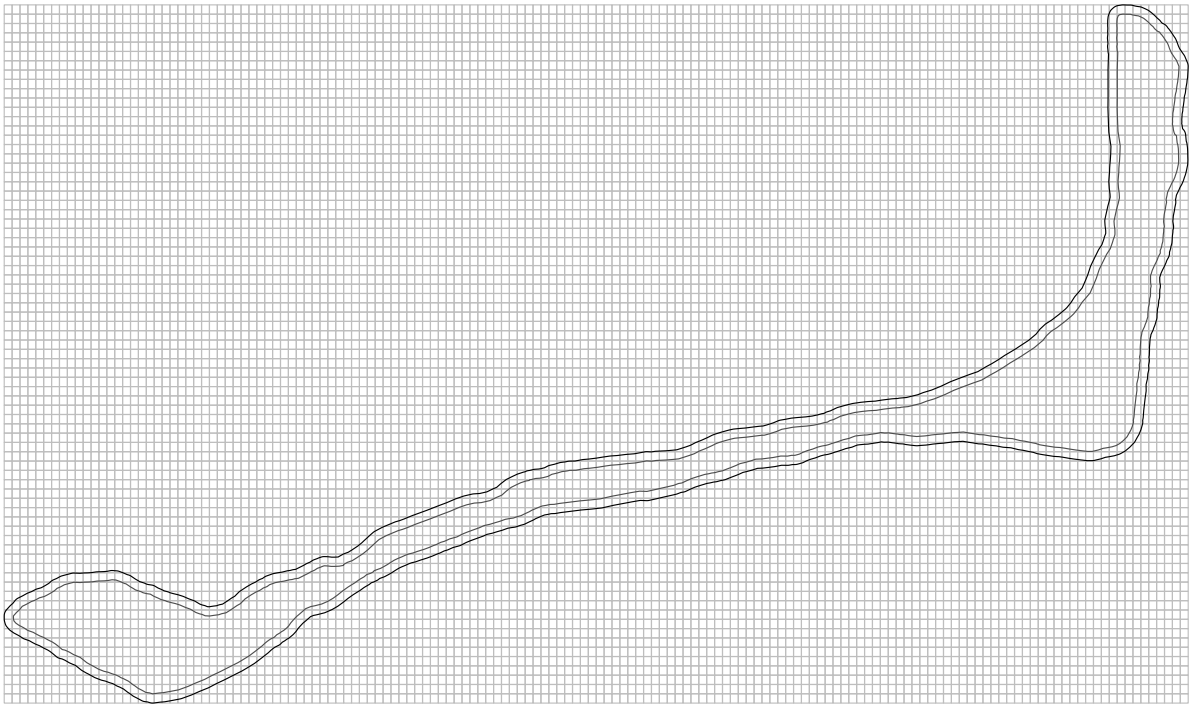


图 26.13: 朗格拉普岛规则化网格操作

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 33.7 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()
```

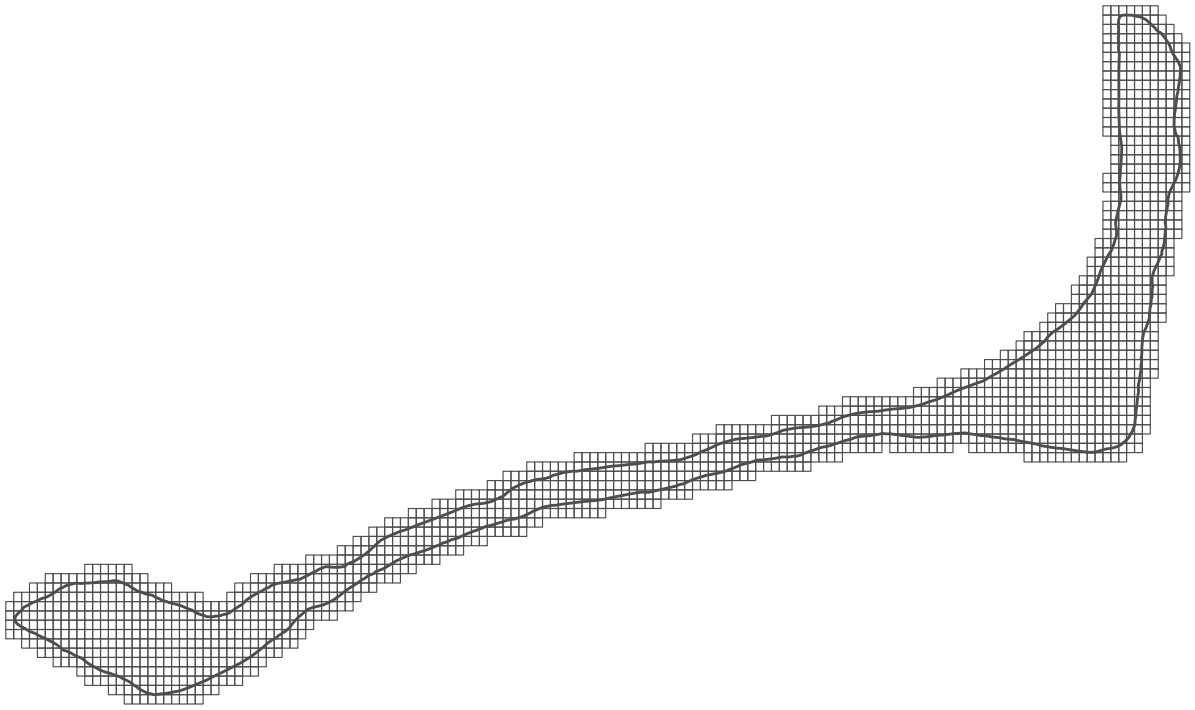


图 26.14: 朗格拉普岛规则网格划分结果

26.4.4 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"
)
```

```
#> 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))
```

26.4.5 展示结果

将预测结果以散点图的形式呈现到图上，见下图 26.15 ，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。

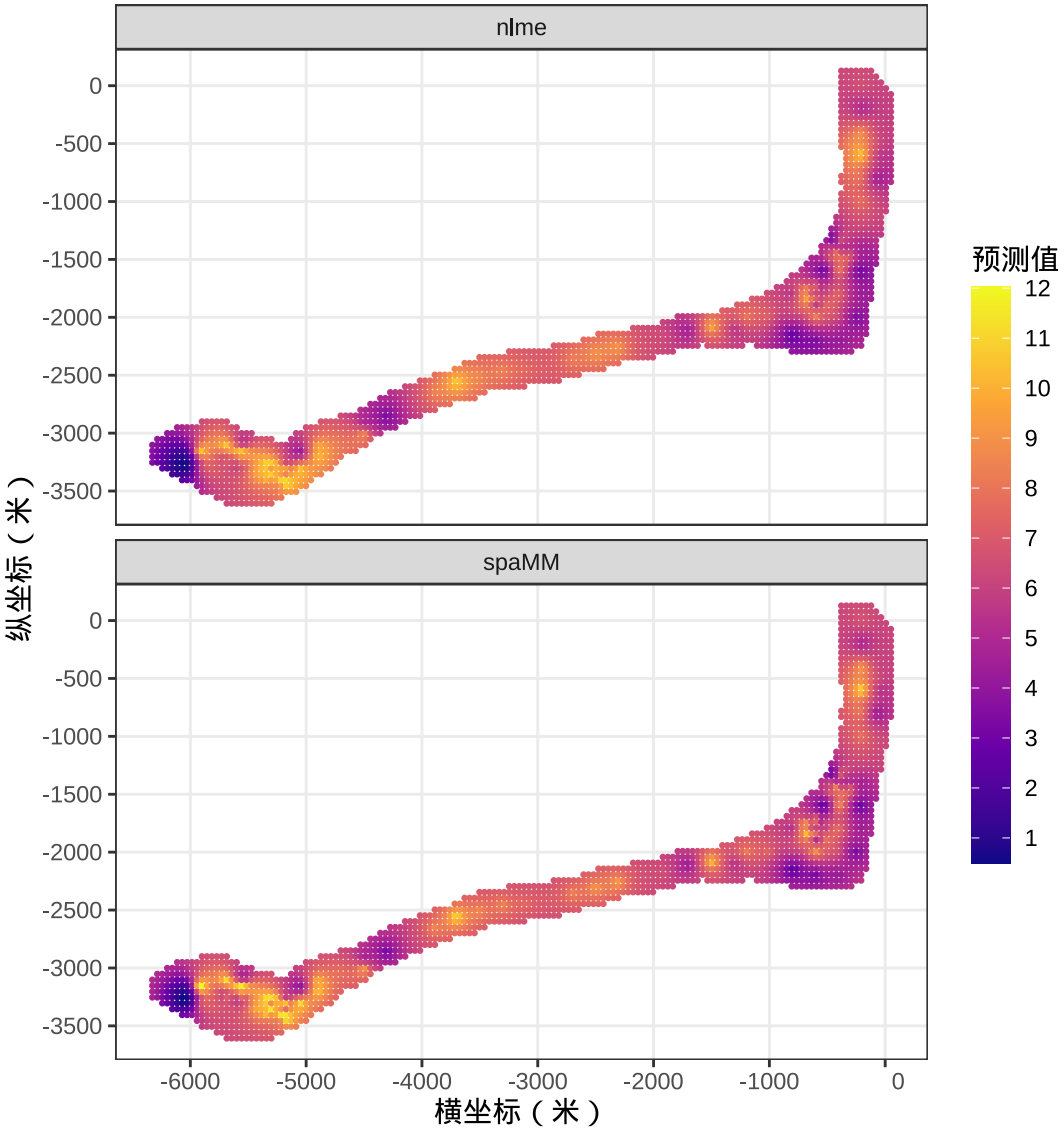


图 26.15: 朗格拉普岛核辐射强度的分布

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 26.16 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，捕捉到更多局部细节，因而，预测值的分布越曲折。

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。

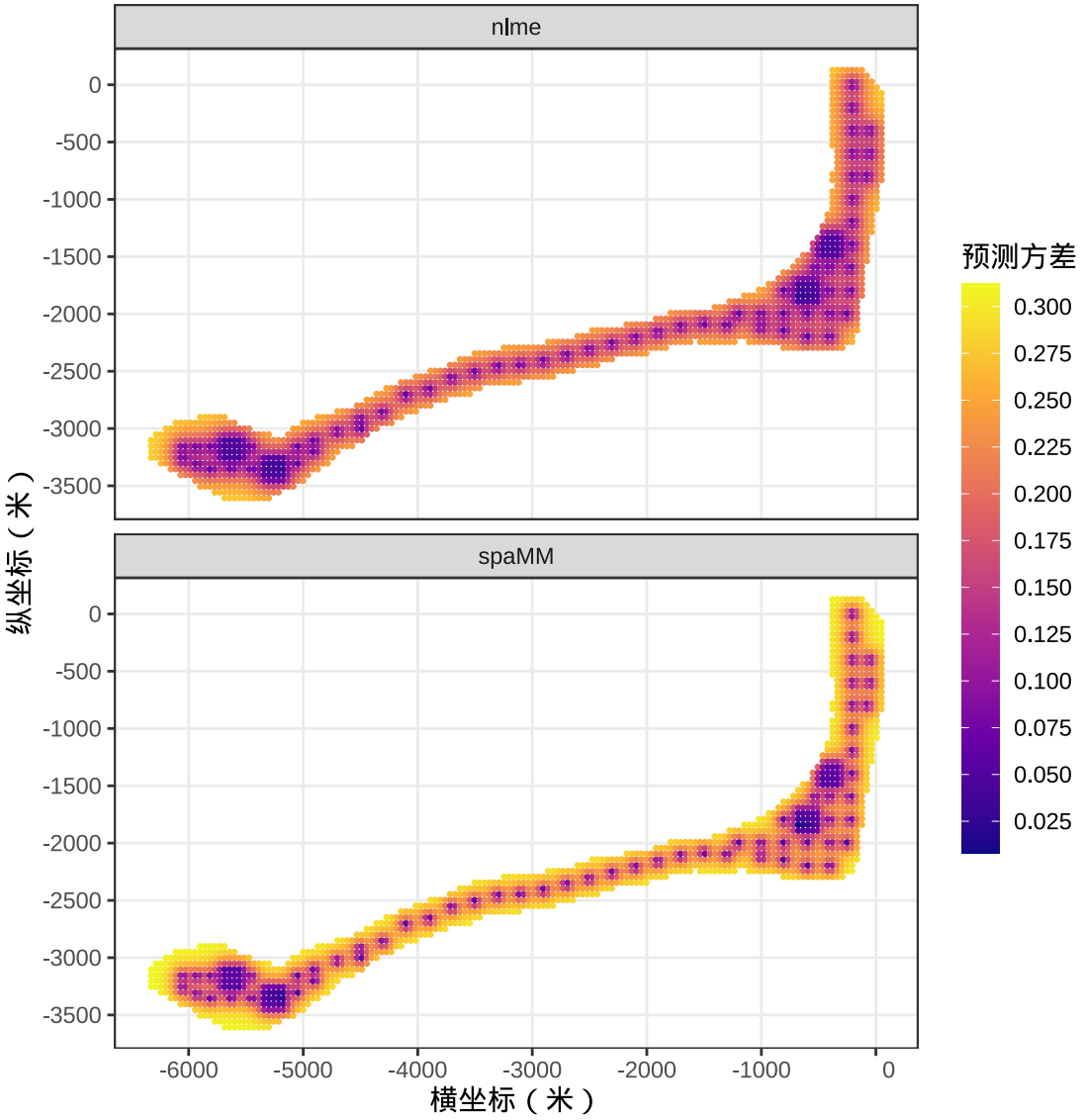


图 26.16: 核辐射强度预测方差的分布

```
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 `geom_stars()`，这可以和 **ggplot2** 内置的图层搭配使用。下图 26.17 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。

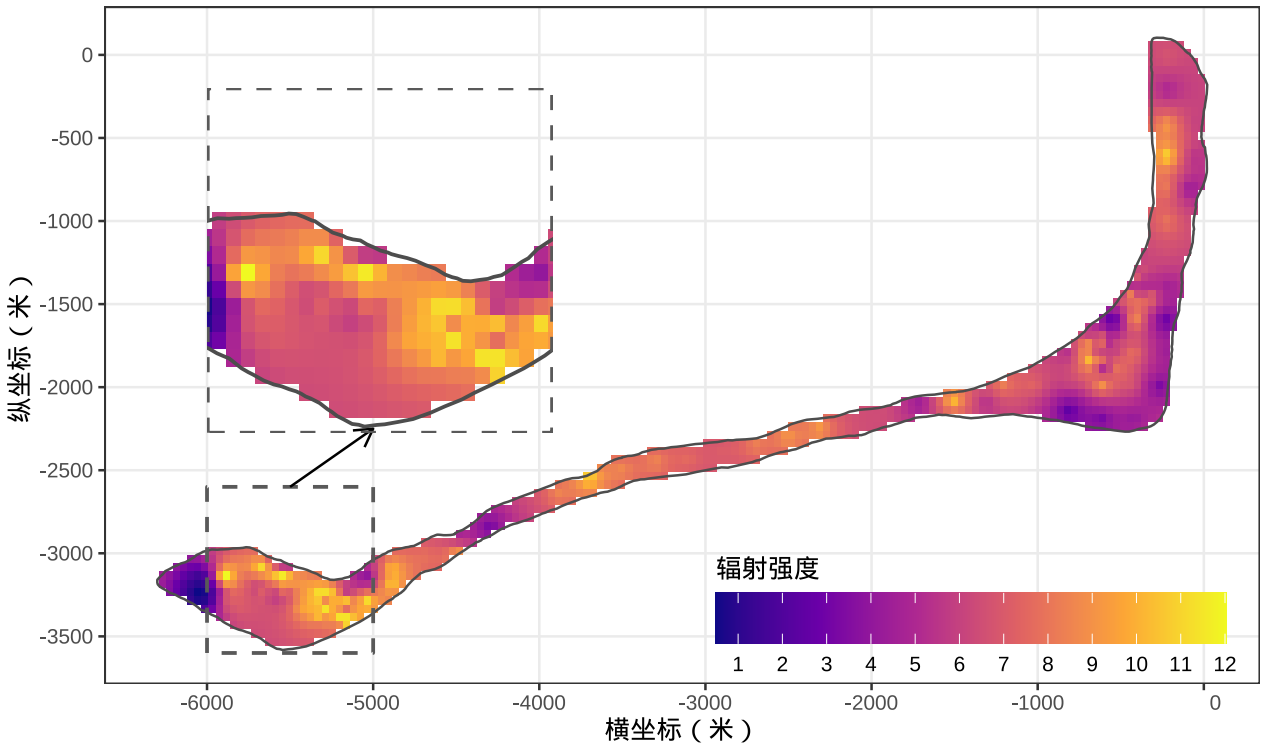


图 26.17: 朗格拉普岛核辐射强度的分布

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。

第六部分

优化建模

第二十七章　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。

27.1　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。

27.2　对数似然与损失函数

27.2.1　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

$$F(x)=\frac{1}{1+\exp(-\frac{x-m}{s})}$$

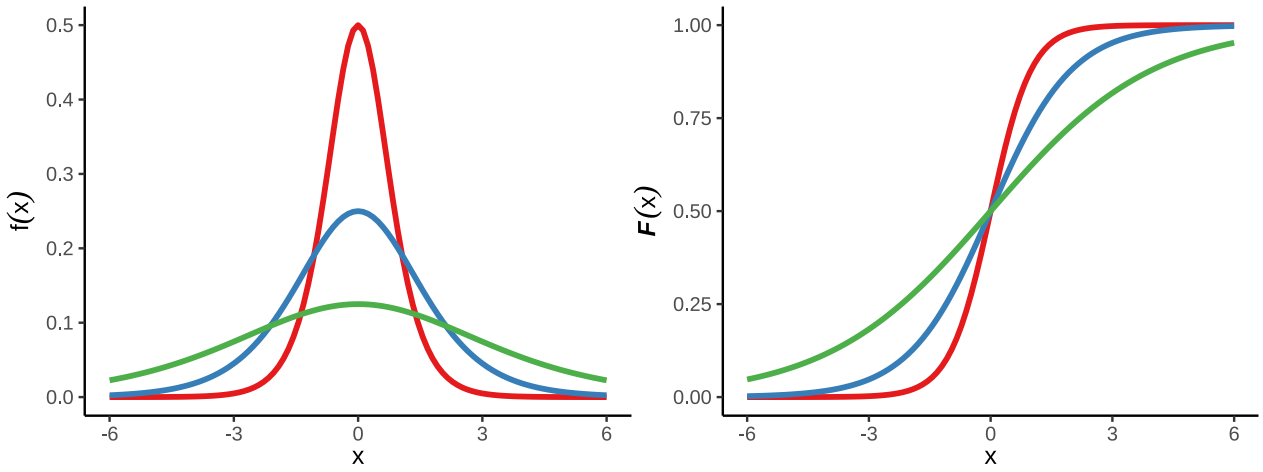
密度函数的形式为

$$f(x)=\frac{\exp(-\frac{x-m}{s})}{s(1+\exp(-\frac{x-m}{s}))^2}=\frac{\exp(\frac{x-m}{s})}{s(1+\exp(\frac{x-m}{s}))^2}$$

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 27.1: 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。

27.2.2 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

$$\boldsymbol{p}=\mathsf{E}Y=\text{Logistic}(X\boldsymbol{\beta})=\frac{1}{1+e^{-(\alpha+X\boldsymbol{\beta})}}=\frac{e^{\alpha+X\boldsymbol{\beta}}}{1+e^{\alpha+X\boldsymbol{\beta}}}$$

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：

$$\begin{aligned} \mathcal{L}(\alpha, \boldsymbol{\beta}) &= \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i} \\ &= \prod_{i=1}^{n} \Big(\frac{e^{\alpha+\boldsymbol{x}_i^\top\boldsymbol{\beta}}}{1+e^{\alpha+\boldsymbol{x}_i^\top\boldsymbol{\beta}}}\Big)^{y_i} \Big(\frac{1}{e^{\alpha+\boldsymbol{x}_i^\top\boldsymbol{\beta}}}\Big)^{1-y_i} \end{aligned} \tag{27.1}$$

关于参数 $\alpha, \boldsymbol{\beta}$ 的对数似然函数如下：

$$\begin{aligned} \ell(\alpha, \boldsymbol{\beta}) &= \log \mathcal{L}(\alpha, \boldsymbol{\beta}) \\ &= \sum_{i=1}^{n} \Big[y_i \log(p_i) + (1-y_i)\log(1-p_i)\Big] \\ &= \sum_{i=1}^{n} \Big[y_i \log\Big(\frac{e^{\alpha+\boldsymbol{x}_i\boldsymbol{\beta}}}{1+e^{\alpha+\boldsymbol{x}_i\boldsymbol{\beta}}}\Big) + (1-y_i)\log\Big(\frac{1}{e^{\alpha+\boldsymbol{x}_i\boldsymbol{\beta}}}\Big)\Big] \end{aligned} \tag{27.2}$$

对数似然函数 $\ell(\alpha, \boldsymbol{\beta})$ 关于参数 $\alpha, \boldsymbol{\beta}$ 的偏导数如下：

$$\begin{aligned} \frac{\partial \ell(\alpha, \boldsymbol{\beta})}{\partial \alpha} &= \sum_{i=1}^{n} \Big[\big(\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}\big)\frac{\partial p_i}{\partial \alpha}\Big] \\ \frac{\partial \ell(\alpha, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^{n} \Big[\big(\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}\big)\frac{\partial p_i}{\partial \beta}\Big] \\ &= \sum_{i=1}^{n} \Big[\big(\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}\big)p_i(1-p_i)\boldsymbol{x}_i^\top\Big] \end{aligned} \tag{27.3}$$

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。

27.3 数值优化问题求解器

27.3.1 `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 27.2 ，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 27.2 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",
```

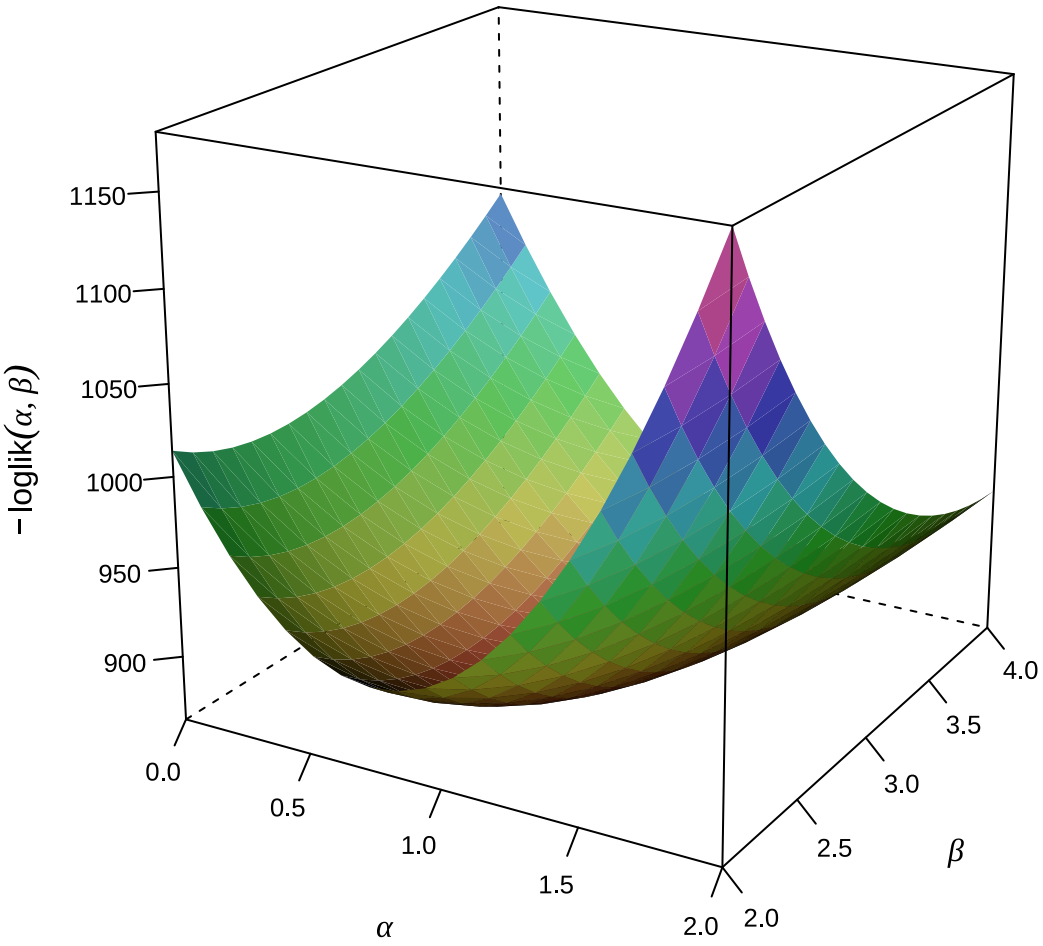


图 27.2: 二维情形下的逻辑回归模型的负对数似然函数曲面

```
  control = list(maxit = 10000)
)
```

```
#> $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
#> 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
#> 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。

27.3.2 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()`。

27.3.3 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")
```

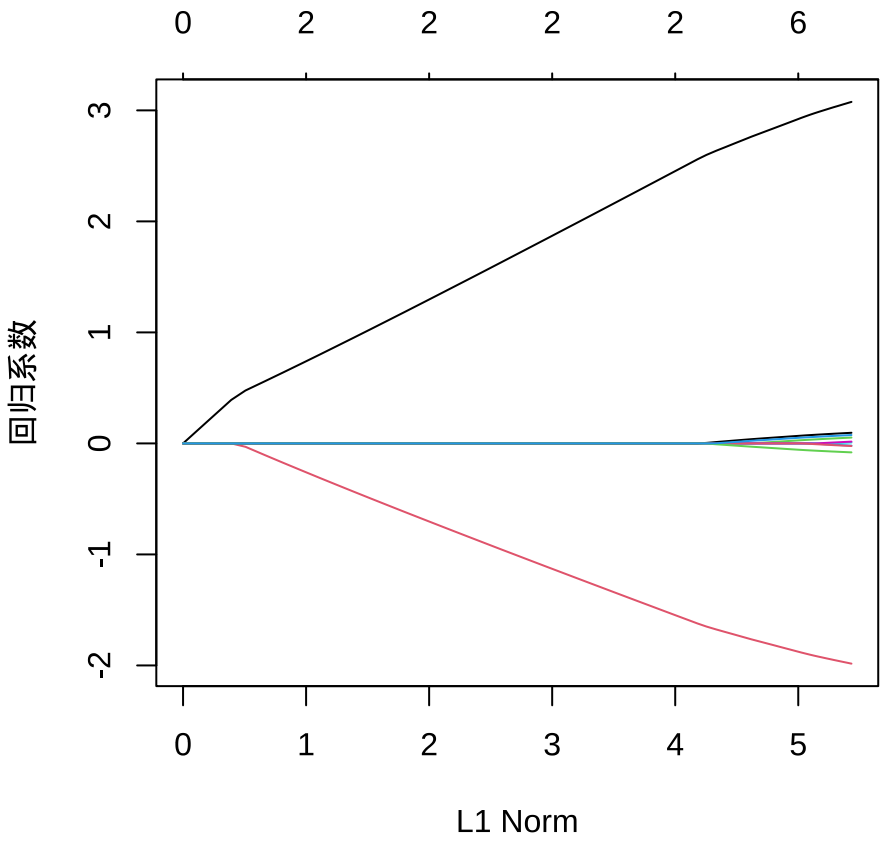


图 27.3: 回归系数的迭代路径

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"
)
```

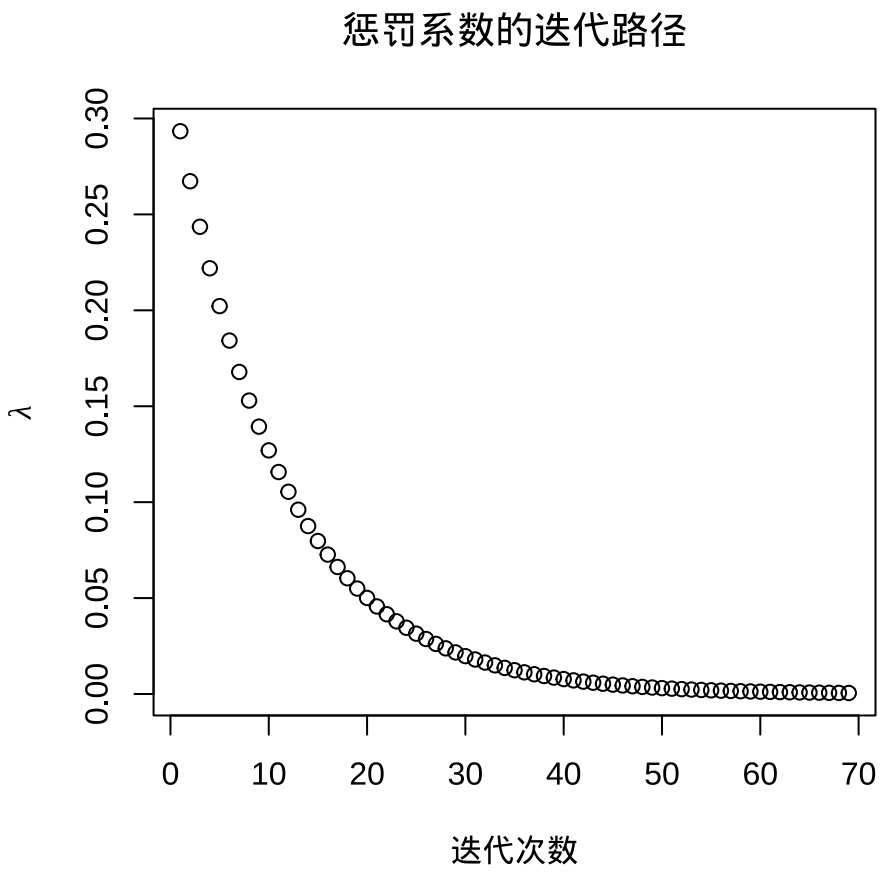


图 27.4: 惩罚系数的迭代路径

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
#> 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。

27.4 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。

27.4.1 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",
  xlab = "FPR", ylab = "TPR", main = "预测 ROC 曲线"
)
```

```
#> Setting levels: control = 0, case = 1

#> Setting direction: controls < cases
```

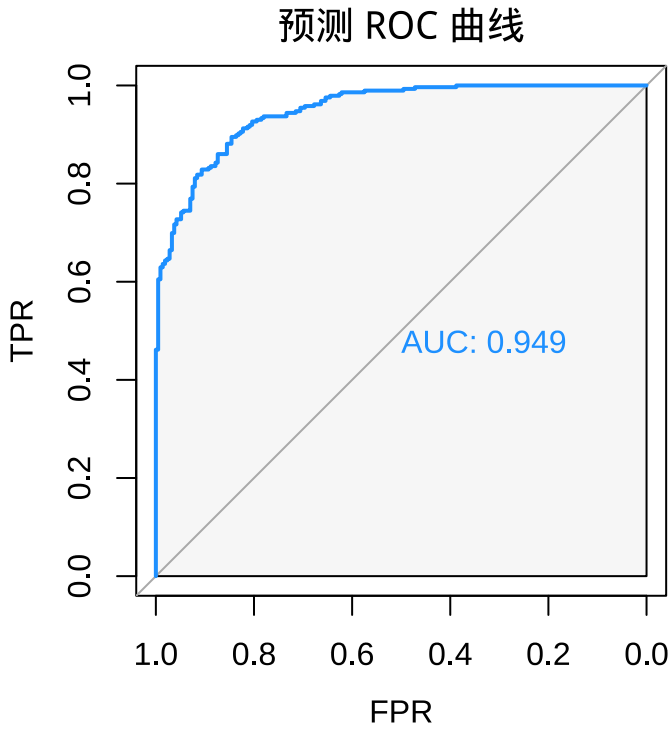


图 27.5: ROC 曲线

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。

27.4.2 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。

```r
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
```

第二十八章　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。

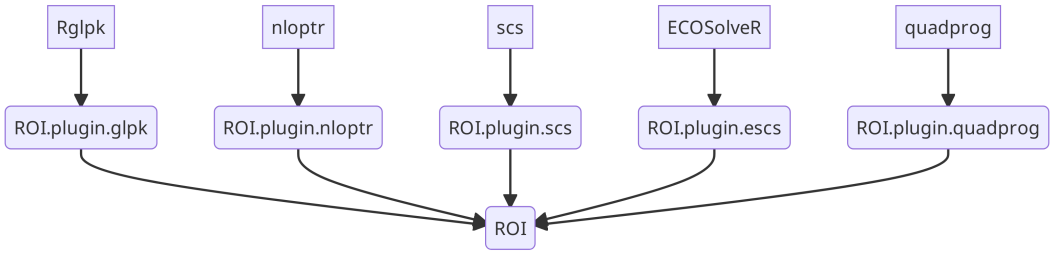


图 28.1: 数值优化扩展包、插件包和 ROI 包的关系图

```
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
  )
}
```

28.1 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：

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & -6x_1 - 5x_2 \\ \text{s.t.} \quad & \begin{cases} x_1 + 4x_2 \leq 16 \\ 6x_1 + 4x_2 \leq 28 \\ 2x_1 - 5x_2 \leq 6 \end{cases} \end{aligned}$$

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：

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \begin{bmatrix} -6 \\ -5 \end{bmatrix}^{\top} \boldsymbol{x} \\ \text{s.t.} \quad & \left\{ \begin{bmatrix} 1 & 4 \\ 6 & 4 \\ 2 & -5 \end{bmatrix} \boldsymbol{x} \leq \begin{bmatrix} 16 \\ 28 \\ 6 \end{bmatrix} \right. \end{aligned}$$

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.
```

```
# 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 28.1: **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()`

28.2 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：

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \frac{1}{2}\boldsymbol{x}^\top D\boldsymbol{x} + \boldsymbol{d}^\top \boldsymbol{x} \\ \text{s.t.} \quad & A\boldsymbol{x} \leq \boldsymbol{b} \end{aligned}$$

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

$$Q(x_1, x_2) = x_1^2 + x_2^2 - x_1 x_2 + 3x_1 - 2x_2$$

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 28.2 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。

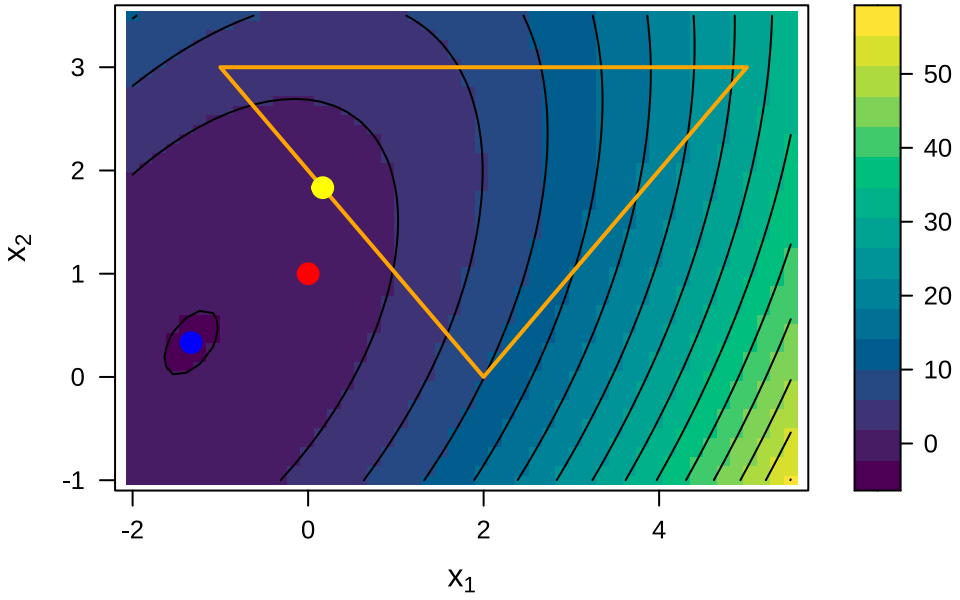


图 28.2: 对比无约束和有约束条件下的解

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。

28.3 凸锥优化

28.3.1 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 28.3 所示。

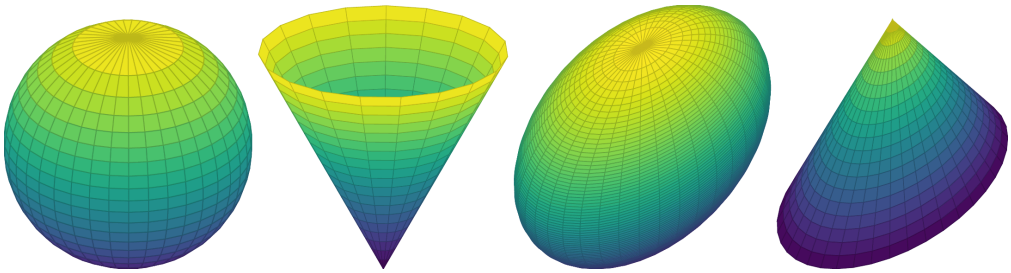


图 28.3: 常见的三维凸锥

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：

$$A(m_1,m_2,m_3)=\begin{bmatrix} 1 & m_1 & m_2 \\ m_1 & 1 & m_3 \\ m_2 & m_3 & 1 \end{bmatrix}.$$

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 28.3 所示，边界是一个三维曲面，曲面及其内部构成一个凸锥。

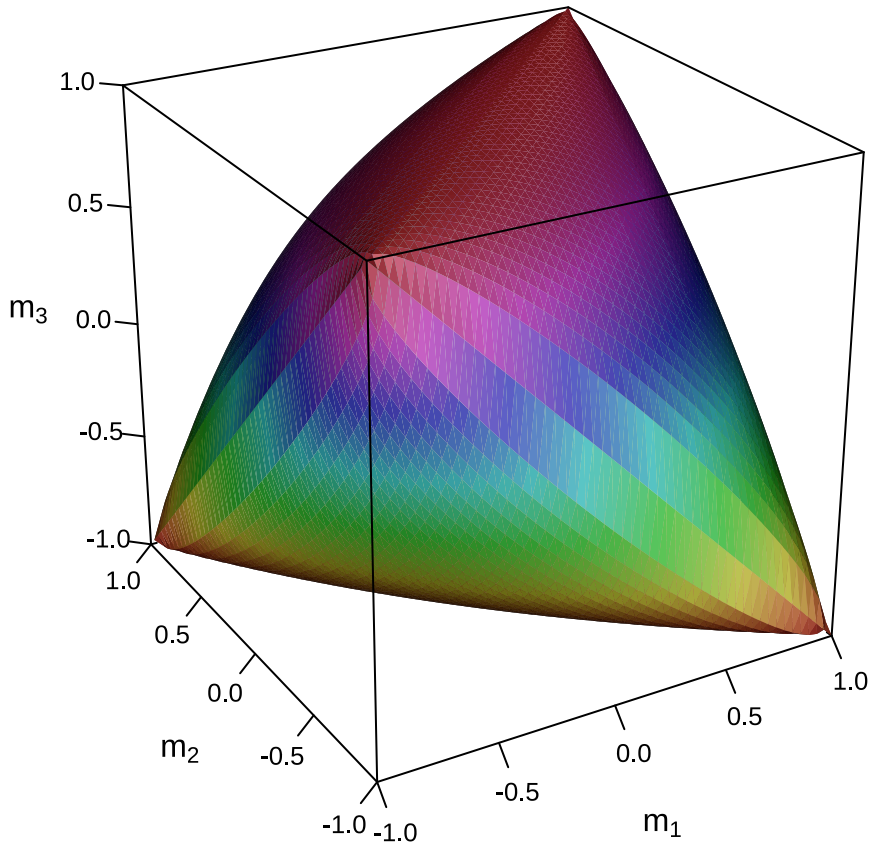


图 28.4: 锥

28.3.2 零锥

零锥的定义如下：

$$\mathcal{K}_{zero} = \{0\}$$

常用于表示线性等式约束。

28.3.3 线性锥

线性锥（Linear Cone）的定义如下：

$$\mathcal{K}_{lin} = \{x \in \mathbb{R} | x \geq 0\}$$

常用于表示线性不等式约束。

28.3.4 二阶锥

二阶锥（Second-order Cone）的定义如下：

$$\mathcal{K}_{soc}^{n} = \{(t, x) \in \mathbb{R}^n | x \in \mathbb{R}^{n-1}, t \in \mathbb{R}, \|x\|_2 \leq t\}$$

常用于凸二次优化问题。考虑如下二阶锥优化 SOCP 问题：

$$\begin{aligned} \max_{(\boldsymbol{y},t)} \quad & y_1 + y_2 \\ \text{s.t.} \quad & \sqrt{(2+3y_1)^2 + (4+5y_2)^2} \leq 6 + 7t \\ & y_1, y_2 \in \mathbb{R}, \quad t \in (-\infty, 9]. \end{aligned}$$

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

$$\sqrt{(b_2 - \boldsymbol{a_2}^\top \boldsymbol{x})^2 + (b_3 - \boldsymbol{a_3}^\top \boldsymbol{x})^2} \leq b_1 - \boldsymbol{a_1}^\top \boldsymbol{x}$$

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
```

28.3.5 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
```

28.3.6 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

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & 3x_1 + 5x_2 \\ \text{s.t.} \quad & 5 + x_1 \leq (2 + x_2)^4 \\ & x_1 \geq 0, \ x_2 \geq 2 \end{aligned}$$

约束条件 $5 + x_1 \leq (2 + x_2)^4$ 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
```

```
#> [1] 250.998234   2.000352
```

```
# 目标函数值
sol$objval
```

```
#> [1] 762.9965
```

28.3.7 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))
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
```

28.4 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。

表格 28.2: 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。

28.4.1 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 28.5 ，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)
```

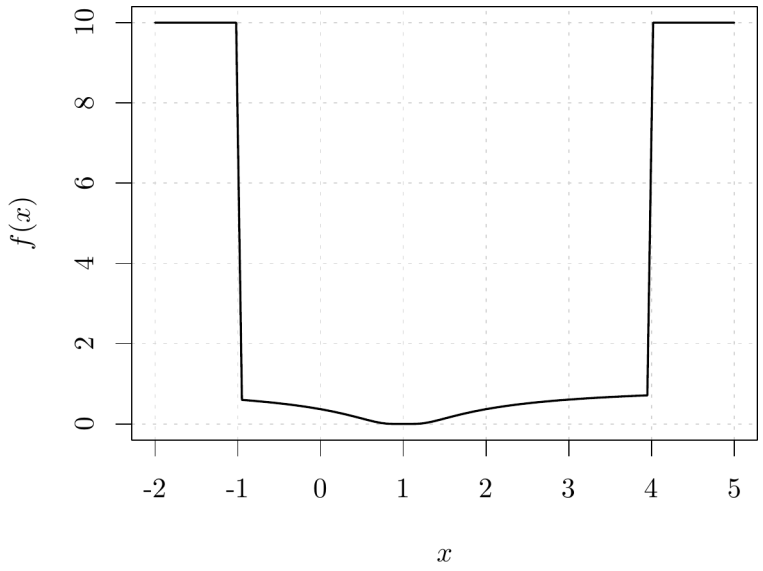


图 28.5: 一维函数图像

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。

28.4.2 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。

$$\begin{aligned}\min_{\boldsymbol{x}} y = & \sin\left((yx_1-0.5)^2+2x_1x_2^2-\frac{y}{10}\right)\cdot \\ & \exp\left(-\left(\left(x_1-0.5-\exp(-x_2+y)\right)^2+x_2^2-\frac{y}{5}+3\right)\right)\end{aligned}$$

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) {
    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 28.6 所示，可以获得关于隐函数的大致情况。

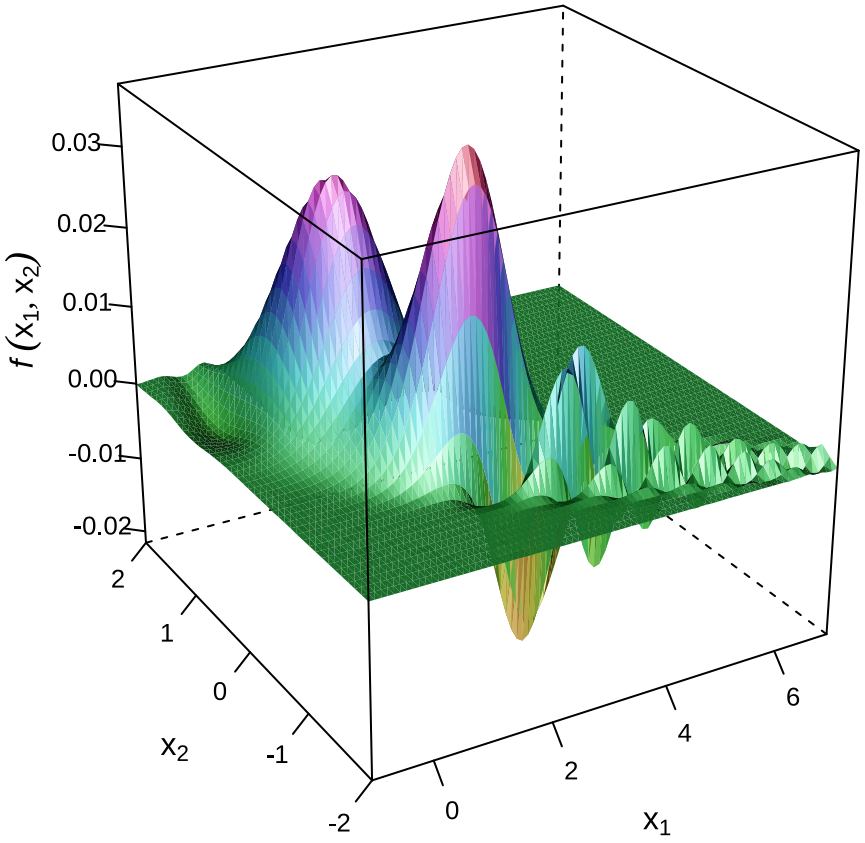


图 28.6: 隐函数图像

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)
}
# 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)
)
# 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。

28.4.3 多元无约束优化

28.4.3.1 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 28.7 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。

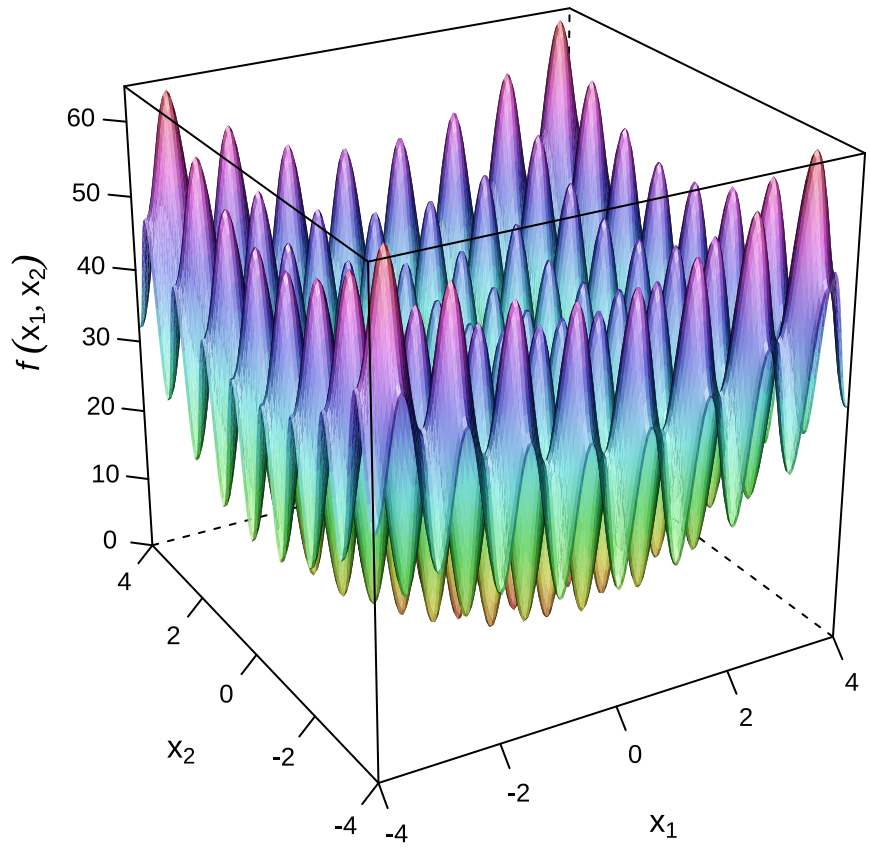


图 28.7: 二维 Rastrigin 函数图像

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
```

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

28.4.3.2 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 28.8a 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 28.8b 所示，可见，有不少局部陷阱，且分布在 $x_2 = x_1$ 的直线上。

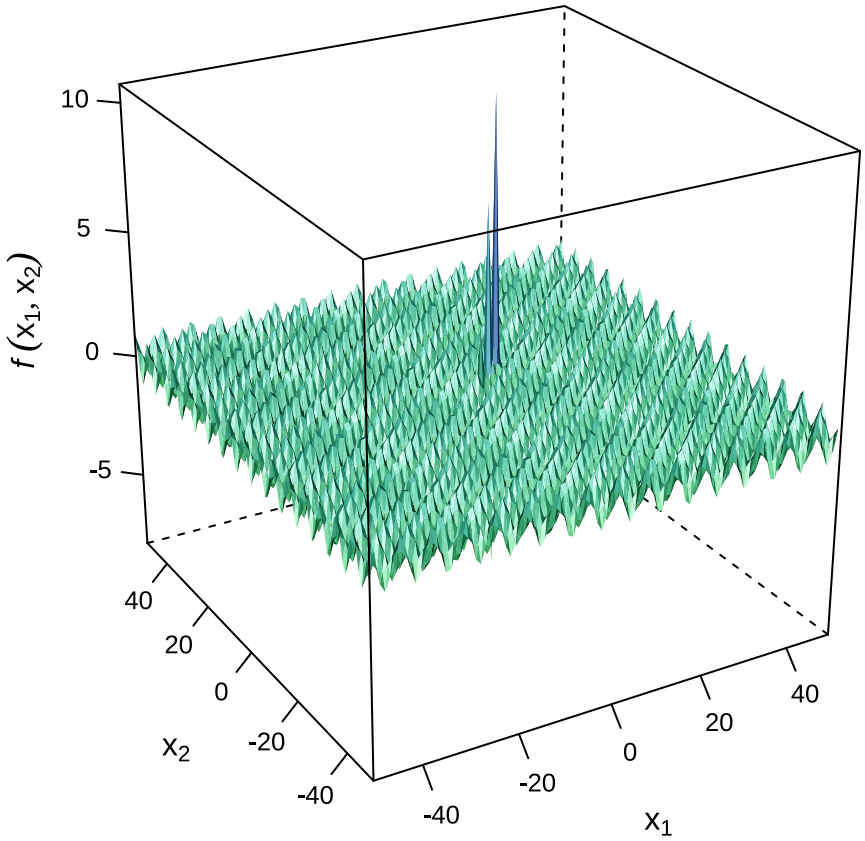
不失一般性，下面考虑 $x_1, x_2 \in [-50, 50]$ ，面对如此复杂的函数，调用全局优化器 `nloptr.directL` 寻优。

```
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
```

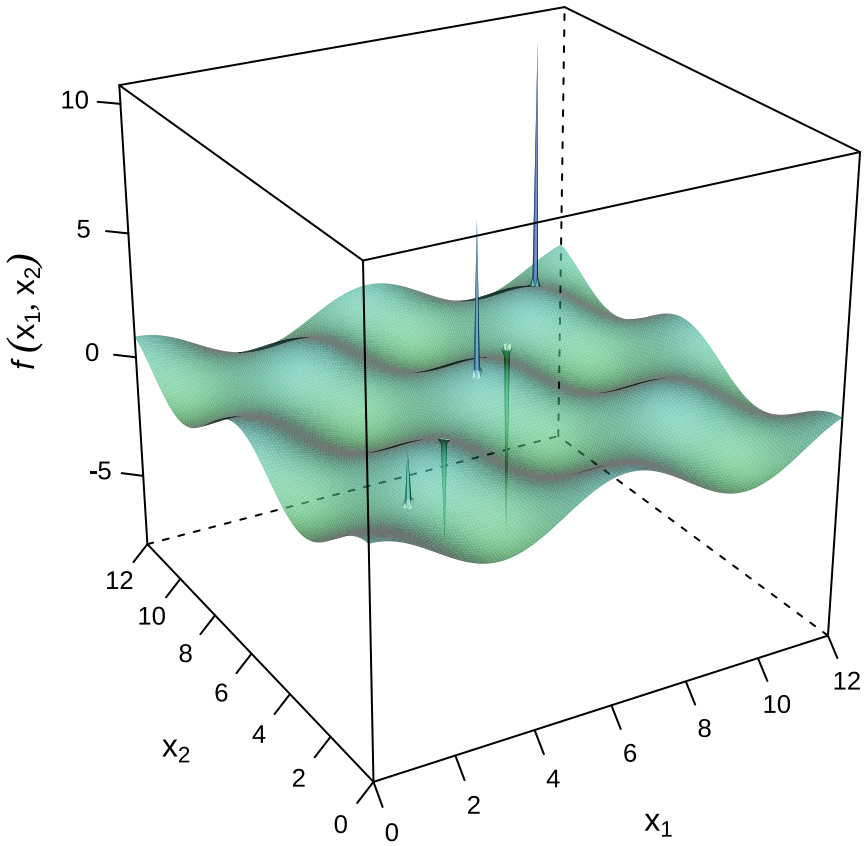
```
#> [1]  0.00000 22.22222
```

```
nlp$objval
```

```
#> [1] -0.9734211
```



(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 28.8: 局部放大前后的函数图像

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
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。

```
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
```

```
# 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。

28.4.4 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。

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & (x_1-1)^2+4\sum_{i=1}^{n-1}(x_{i+1}-x_i^2)^2 \\ \text{s.t.} \quad & 2\le x_1,x_2,\cdots,x_n\le 4 \end{aligned}$$

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 28.9 ，函数曲面和香蕉函数有些相似。

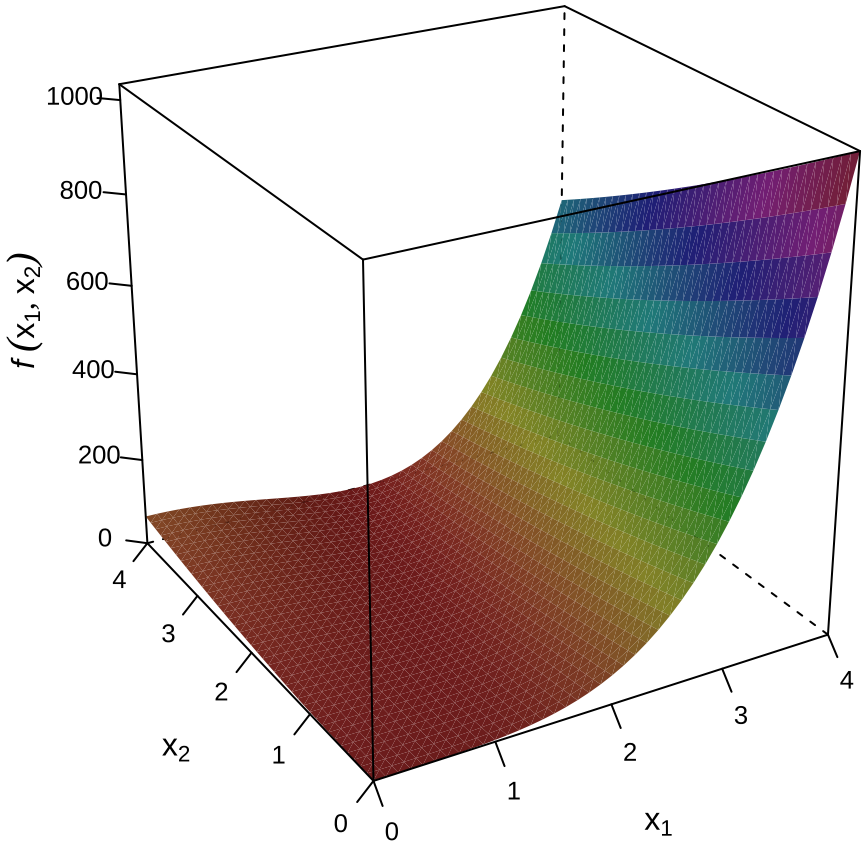


图 28.9: 类香蕉函数的曲面图

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，

28.4.4.1 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。

```
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。

28.4.4.2 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：

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ 2 \\ -4 \\ -4 \end{bmatrix}$$

```
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。

```
constrOptim(
  theta = rep(3, 25), # 初始值
  f = fn, # 目标函数
  method = "Nelder-Mead",
  control = list(maxit = 1000),
  ui = rbind(diag(rep(1, 25)), diag(rep(-1, 25))),
  ci = c(rep(2, 25), rep(-4, 25))
)
```

```
#> $par
#>  [1] 2.000081 2.000142 2.001919 2.000584 2.000007 2.000003 2.001097 2.001600
#>  [9] 2.000207 2.000042 2.000250 2.000295 2.000580 2.002165 2.000453 2.000932
#> [17] 2.000456 2.000363 2.000418 2.000474 2.009483 2.001156 2.003173 2.241046
#> [25] 3.990754
#>
#> $value
#> [1] 370.8601
#>
#> $counts
#> function gradient
#>    18036       NA
#>
#> $convergence
#> [1] 1
#>
#> $message
#> NULL
#>
#> $outer.iterations
#> [1] 19
#>
#> $barrier.value
#> [1] -0.003366467
```

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(
  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。

28.4.4.3 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",
  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。

28.4.4.4 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
```

28.4.5 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。

28.4.6 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。

表格 28.3: 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

28.4.6.1 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.
```

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。

28.4.6.2 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
```

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] 1.060300 4.584670 4.023580 1.289502
```

```
nlp$objval
```

```
#> [1] 17.24299
```

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。

28.5 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。

28.5.1 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
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。

28.5.2 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：

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \\
\text{s.t.} \quad & \begin{cases} \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, 2, \ldots, n \\ \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2, \ldots, n \\ x_{ij} = 0 \ \text{ 或 } \ 1 \end{cases}
\end{aligned}
$$

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
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。

28.5.3 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
```

28.5.4 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。

$$\min_{\boldsymbol{x}} \quad x_1^2 + x_2^2 - x_1 x_2 + 3x_1 - 2x_2$$

$$\text{s.t.} \quad \begin{cases} -x_1 - x_2 <= -2 \\ x_1 - x_2 <= 2 \\ x_2 <= 3. \\ x_1 \in \mathbb{Z} \end{cases}$$

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。

$$
\begin{aligned}
&\min_{(t,\boldsymbol{x})} \quad t \\
&\text{s.t.} \quad \begin{cases} x_1^2 + x_2^2 - x_1 x_2 + 3x_1 - 2x_2 \leq t \\ -x_1 - x_2 <= -2 \\ x_1 - x_2 <= 2 \\ x_2 <= 3. \\ x_1 \in \mathbb{Z} \end{cases}
\end{aligned}
$$

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。

$$\begin{aligned}\min_{(t,\boldsymbol{x})} \quad & t \\ \text{s.t.} \quad & \begin{cases} \boldsymbol{x}^\top D\boldsymbol{x} + 2\boldsymbol{d}^\top \boldsymbol{x} \leq t \\ A\boldsymbol{x} \leq \boldsymbol{b} \\ x_1 \in \mathbb{Z} \end{cases}\end{aligned}$$

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：

$$\begin{aligned}\min_{(t^\star,\boldsymbol{x})} \quad & t^\star \\ \text{s.t.} \quad & \begin{cases} \|D^{1/2}\boldsymbol{x} + D^{-1/2}\boldsymbol{d}\|_2 \leq t^\star \\ A\boldsymbol{x} \leq \boldsymbol{b} \\ x_1 \in \mathbb{Z} \end{cases}\end{aligned}$$

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")
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。

28.5.5 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：

```
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。

```
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。

```
# 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。

```
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
```

28.6 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。

28.7 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。

$$\begin{aligned}
\min_{\boldsymbol{x}} \quad & \exp(\sin(50x_1)) + \sin(60\exp(x_2)) + \sin(70\sin(x_1)) \\
& + \sin(\sin(80x_2)) - \sin(10(x_1 + x_2)) + \frac{(x_1^2 + x_2^2)^{\sin(x_2)}}{4} \\
\text{s.t.} \quad & \begin{cases} x_1 - \big((\cos(x_2))^{x_1} - x_1\big)^{x_2} \le 0 \\ -50 \le x_1, x_2 \le 50 \end{cases}
\end{aligned}$$

目标函数是不连续的，其函数图像如图 28.10 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。）

7. 求解如下非线性约束优化问题。

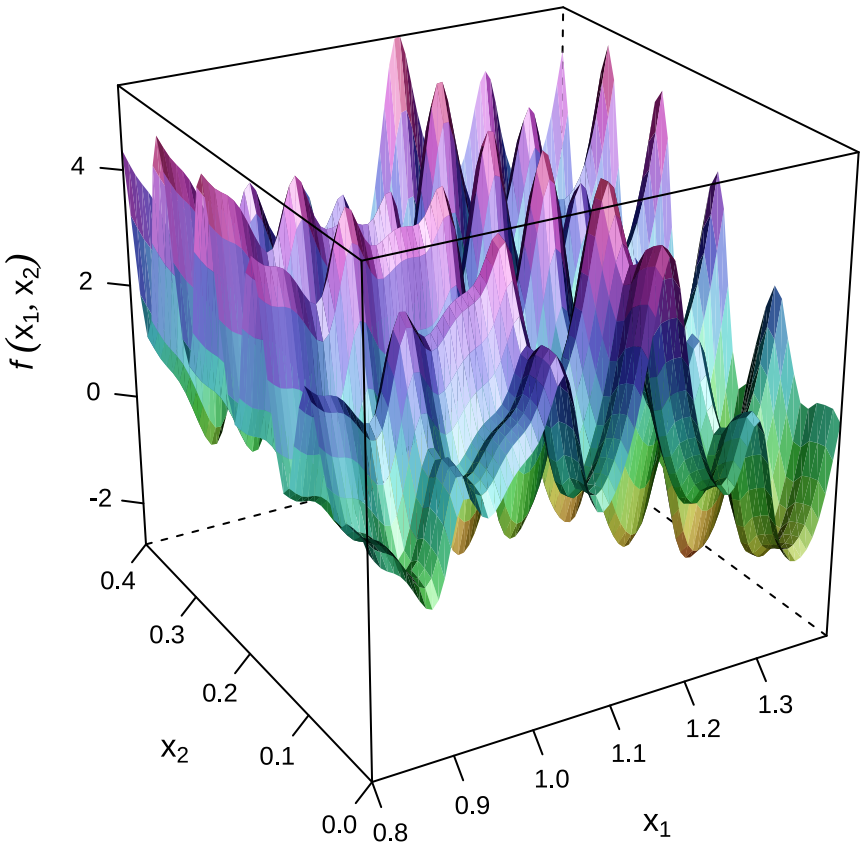


图 28.10: 目标函数的曲面图

$$\begin{aligned}\min_{\boldsymbol{x}} \quad & x_1^2\sin(x_2)+x_2^2\cos(x_1)\\ \text{s.t.} \quad & \begin{cases} 1\le 3x_1-x_2\le 3\\ x_1+x_2\ge 2\\ x_1x_2=2\\ \sin(x_1)\cos(x_2)\le 0.6\\ x_1,x_2\in(-100,100). \end{cases}\end{aligned}$$

第二十九章　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
  )
}
```

29.1　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。

$$
\begin{aligned}
\min \ & \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \\
\text{s.t.} \ & \sum_{i=1}^{n} x_{ij} = 1, \ j = 1, 2, \ldots, n, \\
& \sum_{j=1}^{n} x_{ij} = 1, \ i = 1, 2, \ldots, n, \\
& x_{ij} = 0 \text{ or } 1
\end{aligned}
$$

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 29.1 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？

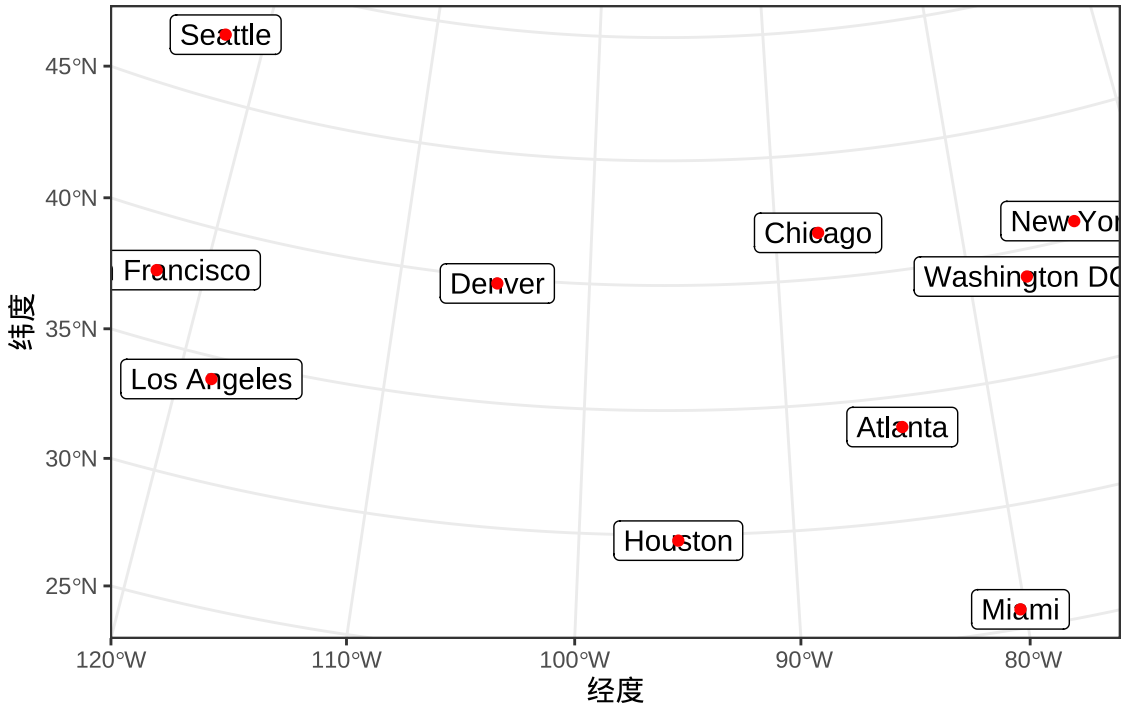


图 29.1: 10 个城市的分布图

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
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 29.2 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。

29.2 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。

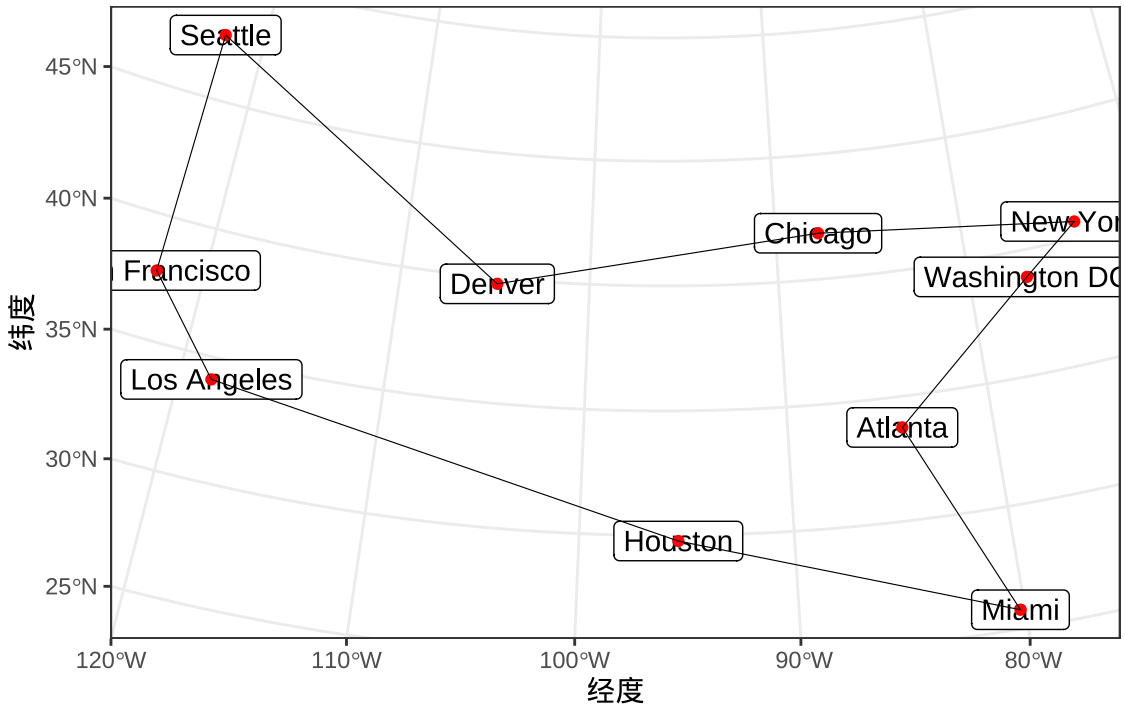


图 29.2: 10 个城市的路线图

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。

```
# 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,
#>
#> 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：

$$\begin{aligned} \max_{\boldsymbol{w}} \quad & \boldsymbol{w}^\top \hat{\boldsymbol{\mu}} \\ \text{s.t.} \quad & A\boldsymbol{w} \le \boldsymbol{b} \\ & \boldsymbol{w}^\top \hat{\Sigma} \boldsymbol{w} \le \sigma \end{aligned}$$

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
```

```
# 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。

```
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 $\frac{1}{12}$ 。

```
# 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
#> [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)
  ),
  # 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
```

29.3 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，β 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：

$$\mathsf{Cov}\{S(x_i), S(x_j)\} = \sigma^2 \exp(-\|x_i - x_j\|/\phi)$$

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 $\text{MVN}(\mathbf{0}, V_{\boldsymbol{\theta}})$，则 $\boldsymbol{y}(x)$ 也服从多元正态分布 $\text{MVN}(D\boldsymbol{\beta}, V_{\boldsymbol{\theta}})$。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] 63.931220  6.120712
```

```
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)
```

似然函数关于参数 (σ, ϕ) 的三维曲面见图 29.3 。

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)
```

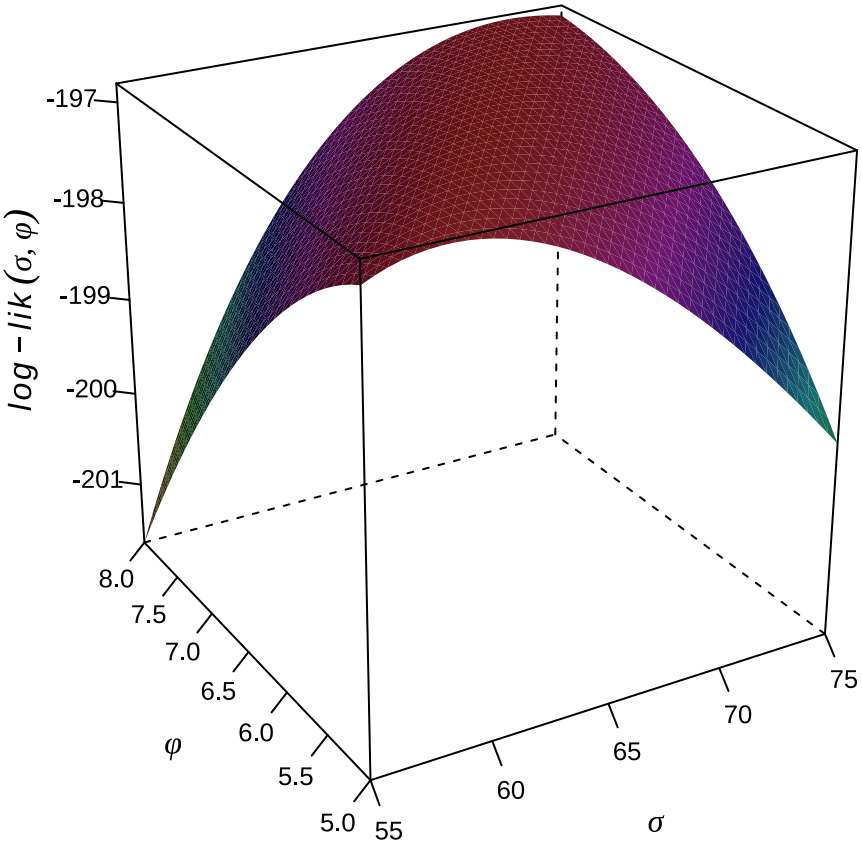


图 29.3: 对数似然函数的曲面图

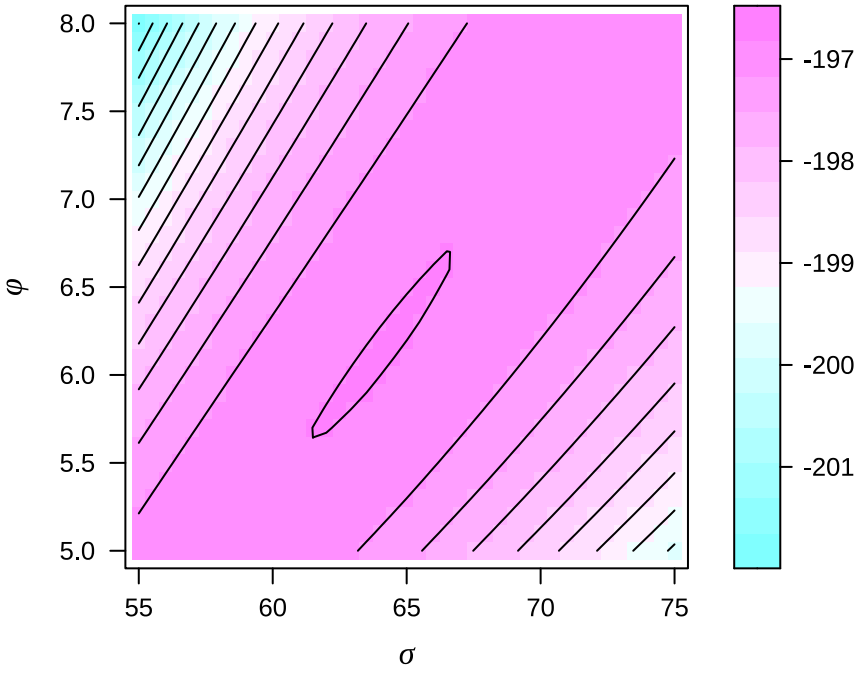


图 29.4: 对数似然函数的等高线图

```
)
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
```

29.4 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，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 Poisson(λ_1)，而以概率 $1-p$ 服从泊松分布 Poisson(λ_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 29.1 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 29.1: 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)
}
# 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:  -2139.707  pgrad:  3.908006
#> iter:  10  f-value:  -1990.003  pgrad:  1.495287
#> iter:  20  f-value:  -1989.946  pgrad:  0.6404415
#> iter:  30  f-value:  -1989.946  pgrad:  0.001675744
#> iter:  40  f-value:  -1989.946  pgrad:  0.0009390533
#> iter:  50  f-value:  -1989.946  pgrad:  0.0008685674
#> iter:  60  f-value:  -1990.011  pgrad:  2.632387
#> iter:  70  f-value:  -1989.947  pgrad:  0.2946376
#> iter:  80  f-value:  -1989.946  pgrad:  0.001391527
#>   Successful convergence.
```

```
ans
```

```
#> $par
#> [1] 0.3598838 1.2560924 2.6634024
#>
#> $value
#> [1] -1989.946
#>
#> $gradient
#> [1] 9.094947e-06
#>
#> $fn.reduction
#> [1] -149.761
#>
#> $iter
#> [1] 83
#>
#> $feval
#> [1] 91
#>
#> $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,] -907.1092  270.22879  341.25374
#> [2,]  270.2288 -113.47969  -61.68199
#> [3,]  341.2537  -61.68199 -192.78165
```

```
# 标准差
se <- sqrt(diag(solve(-hess)))
se
```

```
#> [1] 0.1946836 0.3500297 0.2504771
```

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)
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,] -1989.946      0.6401      2.6634      1.2561
#> [2,] -1996.408      0.3556      1.7075      2.3971
```

```
#> [3,] -1989.946      0.3599      1.2561      2.6634
```

29.5 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 29.5 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 $f(x;9,1)$ 。

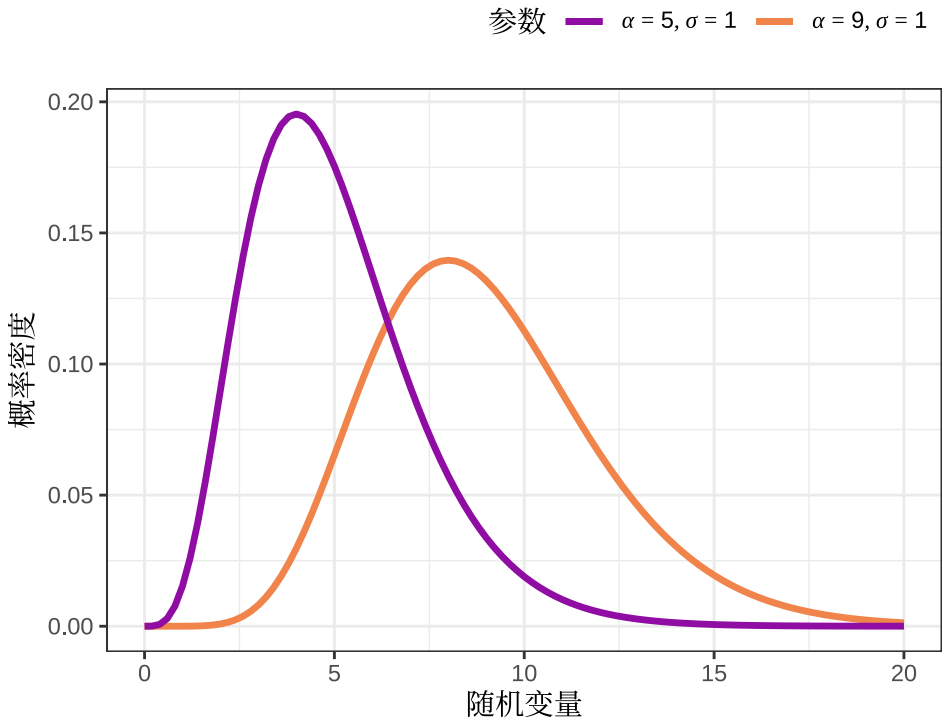


图 29.5: 伽马分布的概率密度函数

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：

$$\frac{\partial\ell(\alpha,\sigma)}{\partial\alpha} = -n\Big(\log(\sigma) + \big(\log\Gamma(\alpha)\big)'\Big) + \sum_{i=1}^{n}\log(x_i) = 0$$
$$\frac{\partial\ell(\alpha,\sigma)}{\partial\sigma} = -\frac{n\alpha}{\sigma} + \frac{\sum_{i=1}^{n}x_i}{\sigma^2} = 0$$

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

$$\log(\alpha) - \big(\log\Gamma(\alpha)\big)' = \log\big(\frac{1}{n}\sum_{i=1}^{n}x_i\big) - \frac{1}{n}\sum_{i=1}^{n}\log(x_i)$$

```r
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
```

```r
# 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
#>
#> $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
```

29.6 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` 。
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 29.2: 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

第七部分

贝叶斯建模

第三十章　广义线性模型

Stan 是一个贝叶斯统计建模和计算的概率推理框架，广泛应用于社会、生物、物理、工程、商业等领域，提供统计建模、数据分析和预测能力 (Gelman, Lee, 和 Guo 2015)，它提供一套成熟的概率编程语言 (Carpenter 等 2017)。Stan 还提供自动微分变分推断（ADVI）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（L-BFGS）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，在学术界和工业界的影响力都很广泛，下图 30.1 是 Stan 软件生态的一角。

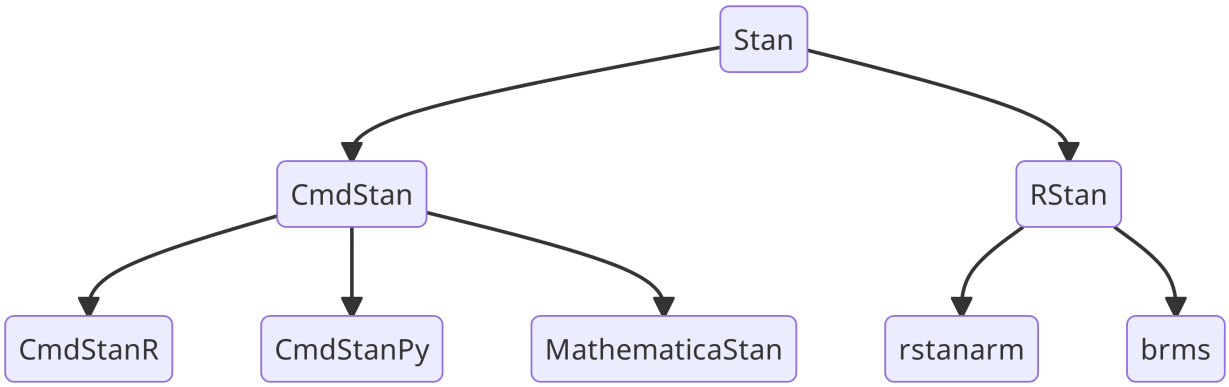


图 30.1: Stan、CmdStan 和 CmdStanR 的关系

CmdStan 是 Stan 的命令行接口，**CmdStanR** 是 CmdStan 的 R 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** 就可以使用上，**CmdStanR** 包与 Stan 是相互独立的更新机制。实际上，**CmdStanR** 只是负责处理 CmdStan 运行的结果。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。

rstan 是 Stan 的 R 语言接口，该接口依赖 **Rcpp**、**RcppEigen**、**BH**、**RcppParallel** 和 **StanHeaders** 等 R 包，由于存在众多上游 R 包依赖，**RStan** 的更新通常滞后于 Stan 的更新，而且滞后很多，不利于及时地使用最新的学术研究成果。因此，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，**cmdstanr** 和 CmdStan 是分离的，方便用户升级和维护。

rstanarm 和 **brms** 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，因此，安装、更新都比较麻烦。

30.1 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)
```

30.2 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)
```

```
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);
  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

```
# 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] 1.172526 1.114983 1.010774 1.125754
```

```
# 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>        <num>     <num>   <num>   <num>     <num>    <num> <num>     <num>
1 alpha        0.489     0.489 0.00948 0.00953     0.473  5.04e-1  1.00    4448.
```

```
2 beta[1]      0.290     0.290 0.0143  0.0144      0.267  3.14e-1  1.00    3144.
3 beta[2]      0.214     0.214 0.0145  0.0147      0.191  2.38e-1  1.00    3133.
4 lp__     -5388.    -5388.    1.20    1.02    -5390.    -5.39e+3  1.00    3232.
# i 1 more variable: ess_tail <num>
```

30.3 参数后验分布

加载 **bayesplot** 包，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]"))) +
  theme_classic() +
  labs(x = expression(beta[1]), y = expression(beta[2]))
```

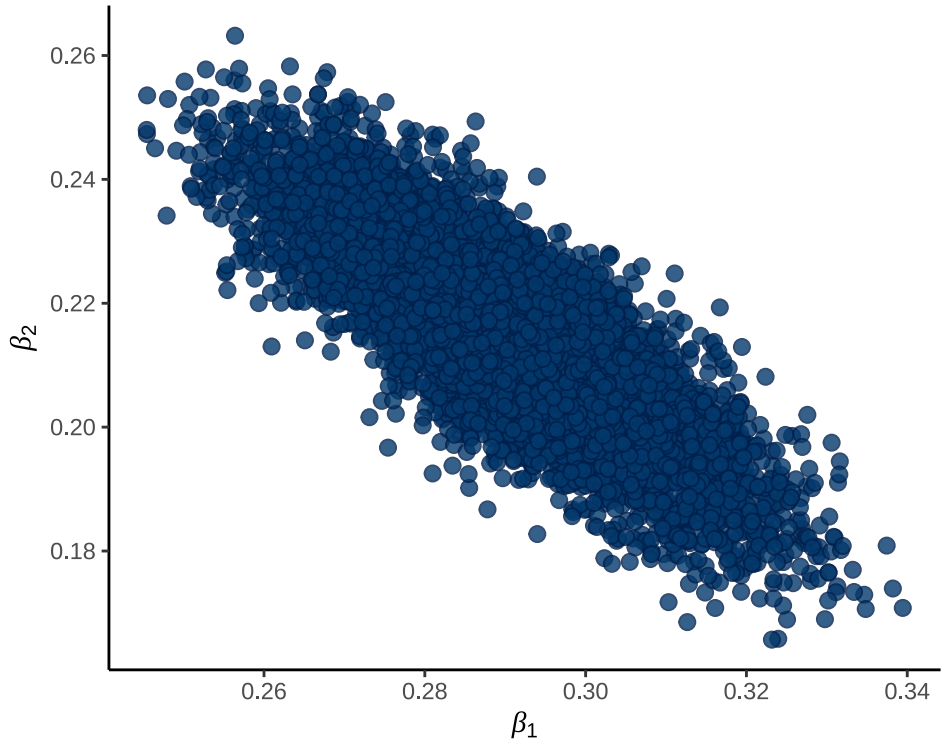


图 30.2: β_1 和 β_2 的联合分布

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]))
```

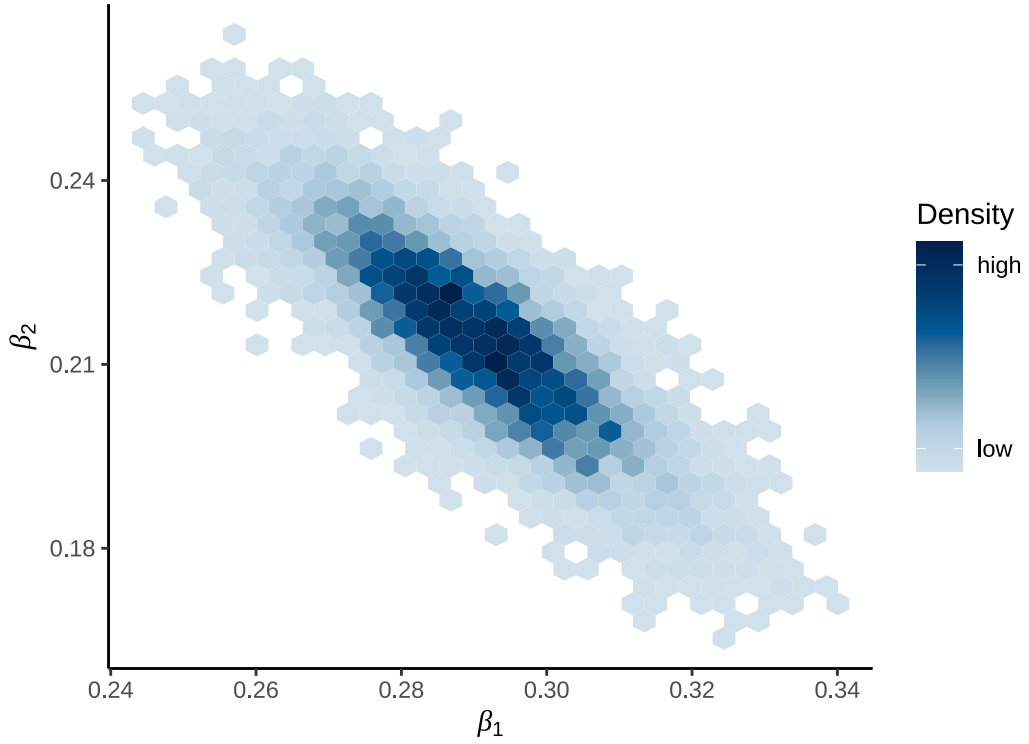


图 30.3: β_1 和 β_2 的热力图

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()
```

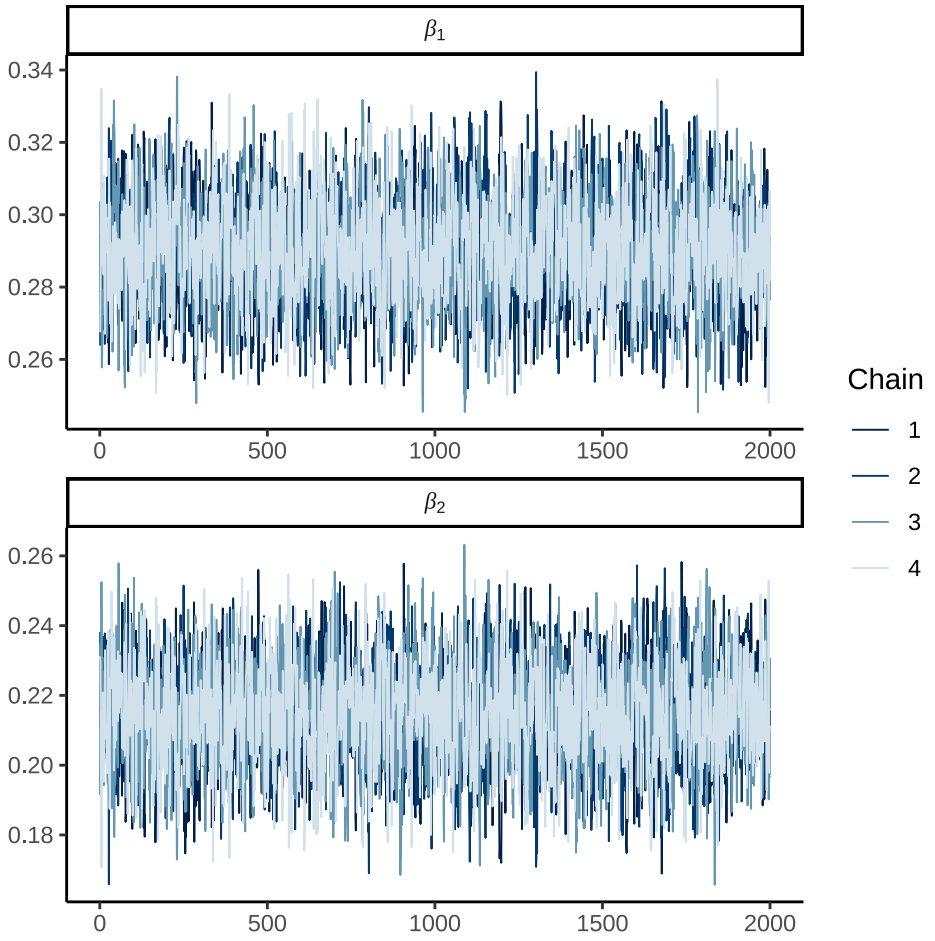


图 30.4: 各个参数的轨迹图

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()
```

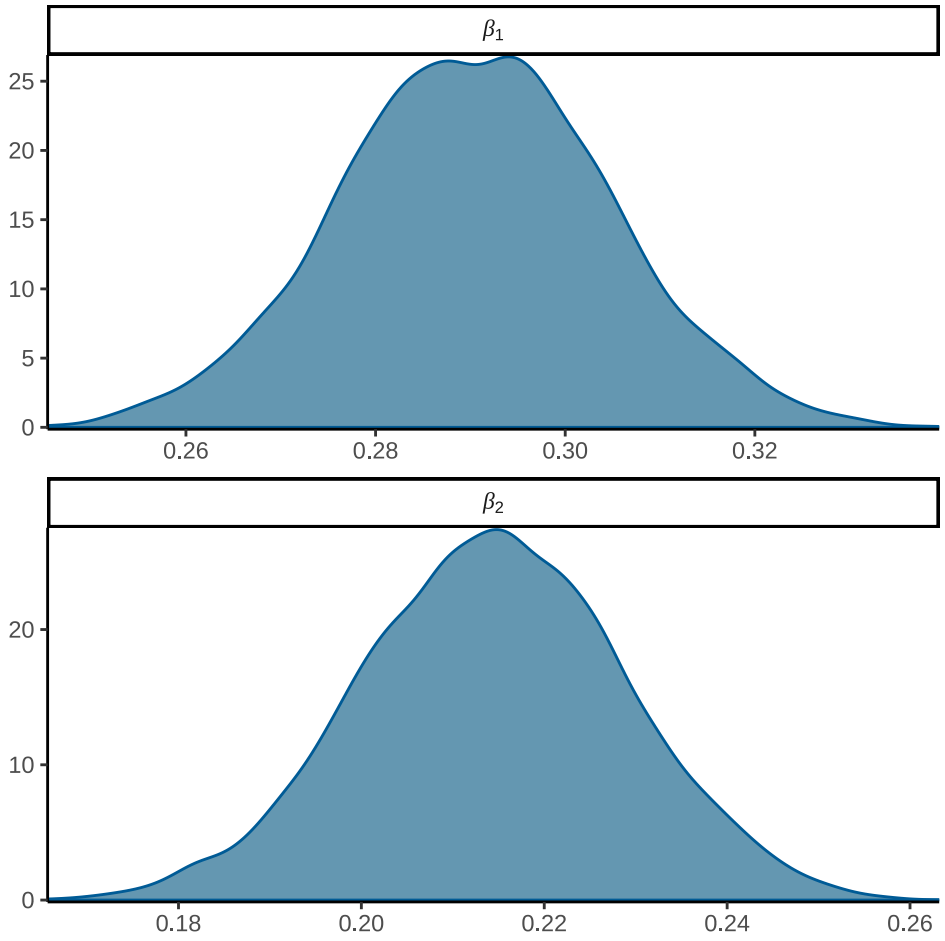


图 30.5: 各个参数的分布图（密度图）

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()
```

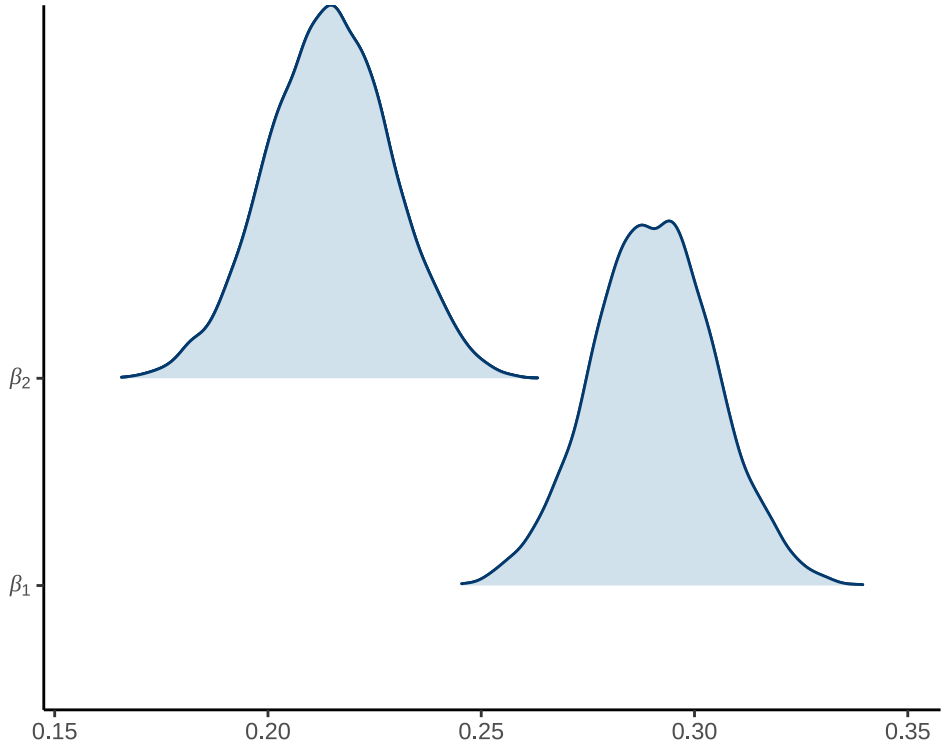


图 30.6: 各个参数的分布图（岭线图）

参数的 $\hat{R}$ 潜在尺度收缩因子

```
bayesplot::rhat(fit_poisson_stan, pars = "alpha")
```

```
   alpha
1.000897
```

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。

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()
```

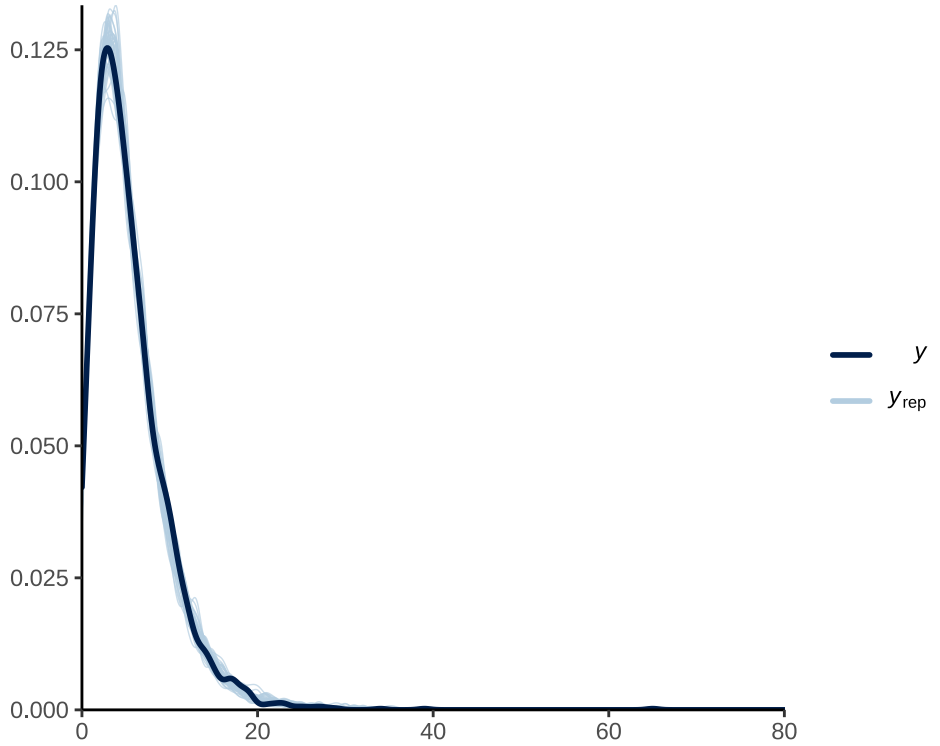


图 30.7: 后验预测诊断图（密度图）

30.4 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         2.9  0.1
waic       10772.8 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
```

```
          Estimate   SE
elpd_loo  -5386.4 37.7
p_loo         2.9  0.1
looic     10772.8 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.
```

30.5 brms 包

对于常见的统计模型，**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).
```

计算 LOO-CV

```
loo::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.
```

looic 指标的作用类似 AIC 指标，所以也几乎相同的。

```
brms::pp_check(fit_poisson_brm)
```

30.6 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 30.8 展示航天飞机 O 型环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。

2. 根据章节 二十六 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。

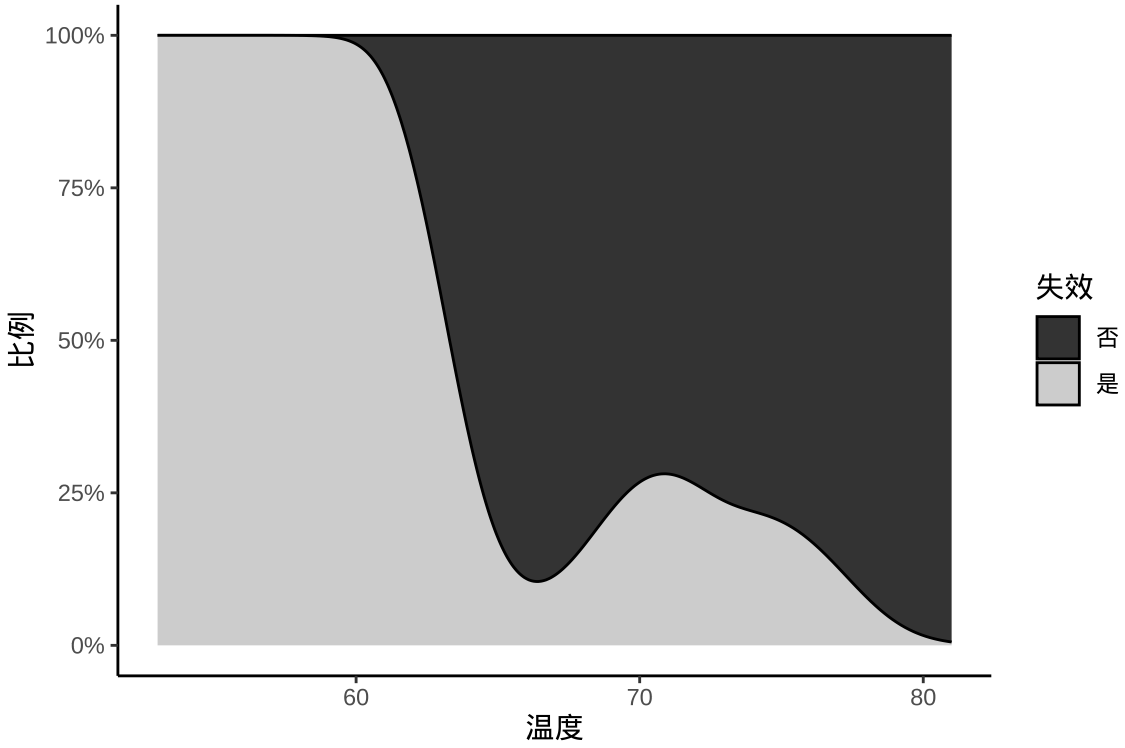


图 30.8: 航天飞机 O 型环在不同温度下失效的条件密度图

第三十一章　广义可加模型

广义可加模型是一种非线性模型

31.1　频率派

MASS 包的 mcycle 数据集

```
data(mcycle, package = "MASS")
library(ggplot2)
ggplot(data = mcycle, aes(x = times, y = accel)) +
  geom_point() +
  theme_bw()
```

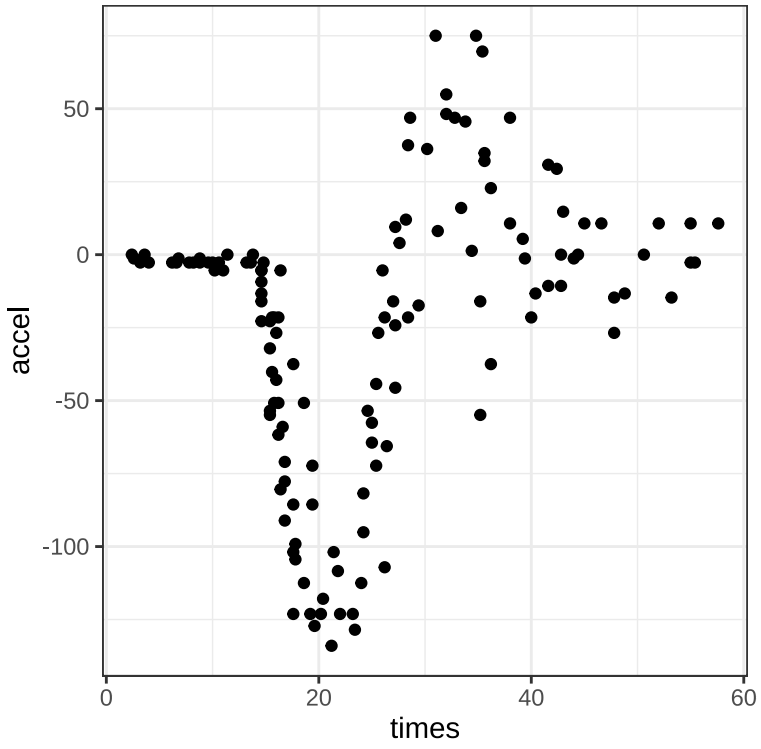


图 31.1: mcycle 数据集

样条回归

```
library(mgcv)
fgam <- gam(accel ~ s(times), data = mcycle, method = "REML")
summary(fgam)
```

```
#>
#> 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(fgam, 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(fgam)
```

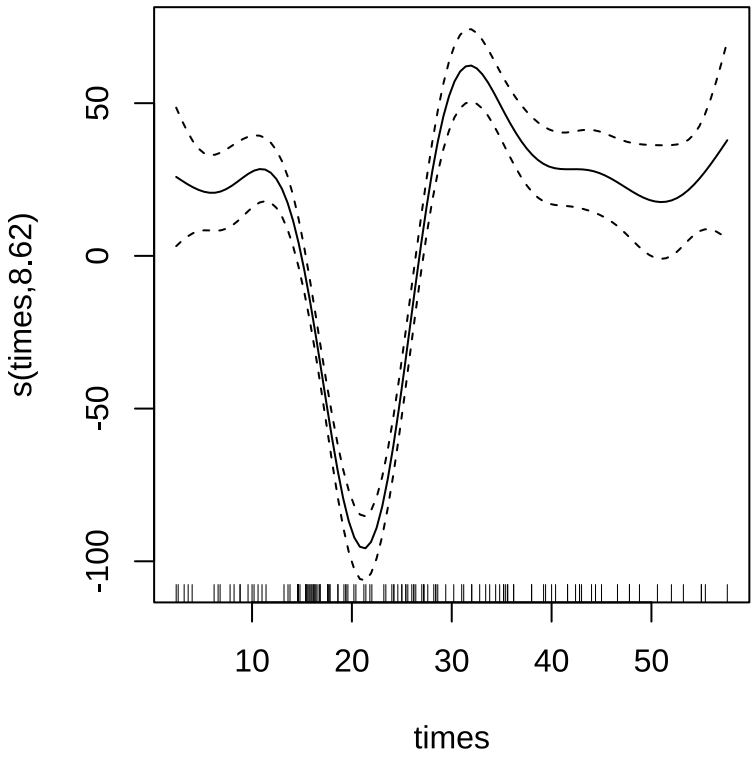


图 31.2: mcycle 数据集

31.2 贝叶斯派

```
library(cmdstanr)
```

```
library(brms)
bgam <- brm(bf(accel ~ s(times)),
  data = mcycle, family = gaussian(), cores = 2, seed = 20232023,
  iter = 4000, warmup = 1000, thin = 10, refresh = 0,
  control = list(adapt_delta = 0.99)
)
summary(bgam)
```

```
ms_bgam <- marginal_smooths(bgam)
plot(ms_bgam)
```

第三十二章　混合效应模型

```
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(splines)   # 样条
library(cmdstanr)  # 编译采样
library(ggplot2)   # 作图
library(bayesplot) # 后验分布
library(loo)       # LOO-CV
```

最好找 3 个真实数据集，其中数据集 `sleepstudy` 和 `cbpp` 均来自 **lme4** 包。

32.1　线性混合效应模型

32.1.1　频率派

32.1.1.1　nlme

```
data(sleepstudy, package = "lme4")
library(nlme)
fm1 <- lme(Reaction ~ Days, random = ~ Days | Subject, data = sleepstudy)
summary(fm1)
```

```
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
```

32.1.1.2 lme4

或使用 lme4 包，可以得到同样的结果

```r
fm2 <- lme4::lmer(Reaction ~ Days + (Days|Subject), data = sleepstudy)
summary(fm2)
```

```
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

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
```

32.1.2 贝叶斯派

32.1.2.1 cmdstanr

```
library(cmdstanr)
```

32.1.2.2 brms

```
bm <- brms::brm(Reaction ~ Days + (Days | Subject), data = sleepstudy)
summary(bm)
```

```
 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)          27.01      6.91    15.27    42.82 1.00     1655     2080
sd(Days)                6.54      1.51     4.15     9.93 1.00     1359     1917
```

```
cor(Intercept,Days)     0.08      0.30    -0.49     0.67 1.00      972     1465

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept   251.17      7.65   235.76   266.45 1.00     1958     2029
Days         10.37      1.72     7.01    13.81 1.00     1351     1999

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma    25.93      1.59    23.07    29.29 1.00     3005     2780

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).
```

```
brms::loo(bm)
```

```
Computed from 4000 by 180 log-likelihood matrix

         Estimate   SE
elpd_loo   -861.7 22.6
p_loo        34.6  8.7
looic      1723.4 45.2
------
Monte Carlo SE of elpd_loo is NA.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     170   94.4%   592
 (0.5, 0.7]   (ok)         7    3.9%   571
   (0.7, 1]   (bad)        1    0.6%   46
   (1, Inf)   (very bad)   2    1.1%   7
See help('pareto-k-diagnostic') for details.
Warning message:
Found 3 observations with a pareto_k > 0.7 in model 'bm'. It is recommended to set 'moment_match = TRUE' in orc
```

```
# Display Conditional Effects of Predictors
brms::conditional_effects(bm, effects = "Days")
# Non-Linear Hypothesis Testing
brms::hypothesis(bm, hypothesis = "Days > 10")
```

32.2 广义线性混合效应模型

二项分布

32.2.1 频率派

32.2.1.1 GLMMadaptive

```
data(cbpp, package = "lme4")
library(GLMMadaptive)
fgm1 <- mixed_model(
  fixed = cbind(incidence, size - incidence) ~ period,
  random = ~ 1 | herd, data = cbpp, family = binomial(link = "logit")
)
summary(fgm1)
```

```
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
 -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
```

32.2.1.2 lme4

或使用 lme4 包，可以得到同样的结果

```
fgm2 <- lme4::glmer(
  cbind(incidence, size - incidence) ~ period + (1 | herd),
  family = binomial("logit"), data = cbpp
)
summary(fgm2)
```

```
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|)
(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
```

32.2.1.3 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)
fgm3 <- gam(
  cbind(incidence, size - incidence) ~ period + s(herd, bs = "re"),
  data = cbpp, family = binomial(link = "logit"), method = "REML"
)
summary(fgm3)
```

```
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:
         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(fgm3)
```

```
Standard deviations and 0.95 confidence intervals:

          std.dev     lower    upper
s(herd) 0.6818673 0.3953145 1.176135

Rank: 1/1
```

32.2.2 贝叶斯派

32.2.2.1 cmdstanr

```
library(cmdstanr)
```

32.2.2.2 brms

```
bgm <- brms::brm(
  incidence | trials(size) ~ period + (1 | herd),
  family = binomial("logit"), data = cbpp
)
```

32.3 广义可加混合效应模型

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。

32.3.1 频率派

32.3.1.1 mgcv (gam)

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

近似高斯过程、高斯过程的核函数，**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|)
(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
```

```
# 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")
)
```

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。

```
library(sf)
```

```
Linking to GEOS 3.11.1, GDAL 3.6.4, PROJ 9.1.1; 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))
# 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")

# 预测
```

```
rongelap_grid_df$ypred <- as.vector(predict(fit_rongelap_gam, newdata = rongelap_grid_df, type =
# 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)
```

核辐射强度的空间分布

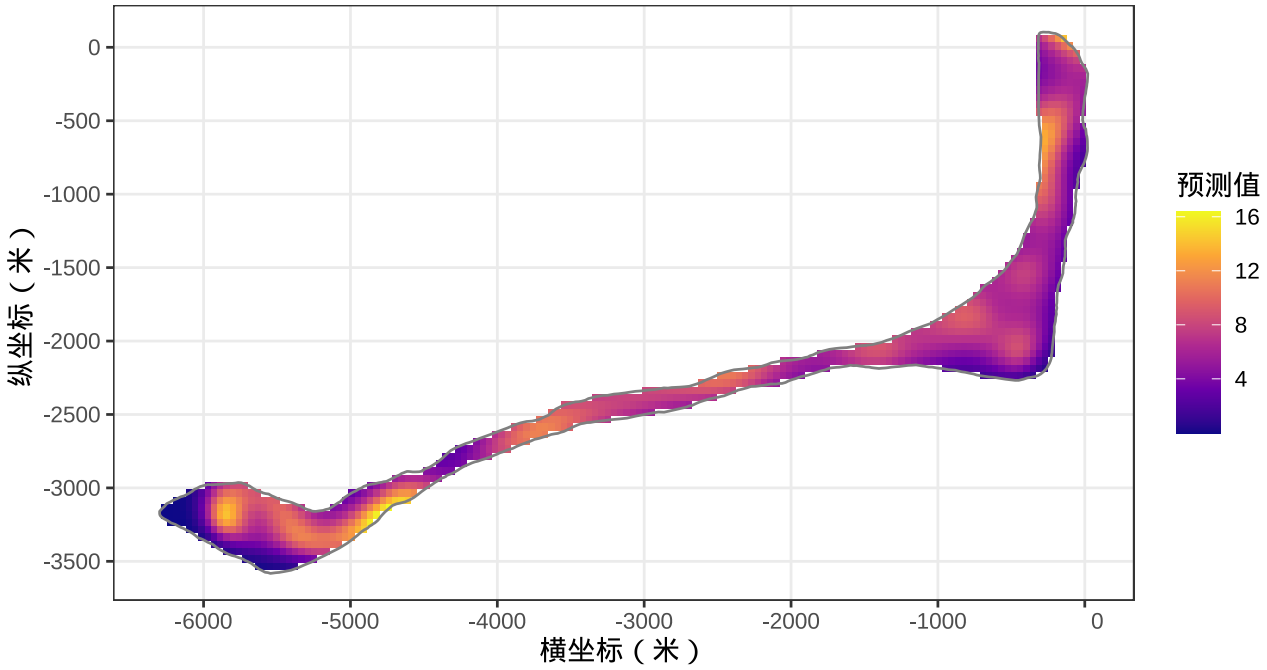


图 32.1: 核辐射强度的预测分布

32.3.1.2 mgcv (INLA)

mgcv 包的函数 `ginla()` 实现简化版 INLA

```
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
fit_rongelap_ginla <- ginla(G = rongelap_gam)
```

其中，$k = 50$ 表示 50 个参数

```
plot(
  fit_rongelap_ginla$beta[1, ], fit_rongelap_ginla$density[1, ],
  type = "l", xlab = "intercept", ylab = "density"
)
```

32.3.2 贝叶斯派

32.3.2.1 brms

参考 **brms** 包的函数 `gp()` / `s()` 和 **mgcv** 包的函数 `gamm()` 的帮助文档，先用模拟数据检测

```
library(cmdstanr)
```

mgcv 包的函数 `gamSim()` 专门用于模拟数据。

```
sim_dat <- mgcv::gamSim(eg = 4, dist = "normal", n = 90, scale = 2)
```

```
Factor `by' variable example
```

参数 $n = 90$ 设置样本量，参数 `dist = "normal"` 设置变量分布，参数 `eg` 设置模拟数据的生成模型，取值如下

1. Gu and Wahba 4 univariate term example.
2. A smooth function of 2 variables.
3. Example with continuous by variable.
4. Example with factor by variable.
5. An additive example plus a factor variable.
6. Additive + random effect.
7. As 1 but with correlated covariates.

```
str(sim_dat)
```

```
'data.frame':   90 obs. of  8 variables:
 $ y  : num  -0.15 8.146 0.102 3.358 2.74 ...
 $ x0 : num  0.0109 0.1811 0.8612 0.2848 0.6463 ...
 $ x1 : num  0.625 0.866 0.944 0.897 0.318 ...
 $ x2 : num  0.502 0.364 0.346 0.579 0.437 ...
 $ fac: Factor w/ 3 levels "1","2","3": 3 3 1 1 1 2 1 3 3 1 ...
 $ f1 : num  2 1.82 1.77 1.94 1.96 ...
 $ f2 : num  -1.031 -1.686 -1.763 -0.577 -1.36 ...
 $ f3 : num  2.74 5.41 6.08 3.07 3.37 ...
```

模拟分组的高斯过程

```
# 按变量 fac 的不同水平
fit_sim_dat <- brms::brm(y ~ gp(x2, by = fac), data = sim_dat, chains = 2)
summary(fit_sim_dat)
plot(brms::conditional_effects(fit_sim_dat), points = TRUE)
```

32.3.2.2 INLA

INLA 近似贝叶斯计算

```
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)
)
# 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
)
spde <- inla.spde2.matern(mesh = mesh, alpha = 3/2, constr = TRUE)
indexs <- inla.spde.make.index(name = "s", n.spde = spde$n.spde)
lengths(indexs)

A <- inla.spde.make.A(mesh = mesh, loc = as.matrix(rongelap[, c("cX", "cY")]) )
coop <- as.matrix(rongelap_grid_df[, c("cX", "cY")])
Ap <- inla.spde.make.A(mesh = mesh, loc = coop)
dim(Ap)

# 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 = A,
  effects = list(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 = Ap,
  effects = list(s = indexs)
)

# 合并数据 stk.full has stk.e and stk.p
stk.full <- inla.stack(stk.e, stk.p)

formula <- y ~ f(s, model = spde)

res <- inla(formula,
  data = inla.stack.data(stk.full),
  E = E, # E 已知漂移项
  control.family = list(link = "log"),
  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"
)

summary(res)

res$summary.fixed
res$summary.hyperpar
```

```
index <- inla.stack.index(stk.full, tag = "pred")$data

pred_mean <- res$summary.fitted.values[index, "mean"]
pred_ll <- res$summary.fitted.values[index, "0.025quant"]
pred_ul <- res$summary.fitted.values[index, "0.975quant"]

dpm <- rbind(
  data.frame(
    cX = rongelap_grid_df[, 1], cY = rongelap_grid_df[, 2],
    value = pred_mean, variable = "pred_mean"
  ),
  data.frame(
    cX = rongelap_grid_df[, 1], cY = rongelap_grid_df[, 2],
    value = pred_ll, variable = "pred_ll"
  ),
  data.frame(
    cX = rongelap_grid_df[, 1], cY = rongelap_grid_df[, 2],
    value = pred_ul, variable = "pred_ul"
  )
)

dpm_sf <- st_as_sf(dpm, coords = c("cX", "cY"), dim = "XY")

ggplot(dpm_sf[dpm_sf$variable =="pred_mean", ]) +
  geom_sf(aes(color = value)) +
  scale_color_viridis_c(option = "C", name = expression(lambda)) +
  theme_bw()
```

32.4 总结

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。

32.5 习题

1. **MASS** 包的地形数据集 topo 为例，高斯过程回归模型

```
data(topo, package = "MASS")
bgamm1 <- brms::brm(
  z ~ gp(x, y, cov = "exp_quad"),
  # family = Gamma(link = "inverse"),
  data = topo, chains = 2, seed = 20232023,
  warmup = 1000, iter = 2000, thin = 1, refresh = 0,
  control = list(adapt_delta = 0.99)
)
summary(bgamm1)
# 高斯过程近似计算
bgamm2 <- brms::brm(
  z ~ gp(x, y, cov = "exp_quad", c = 5 / 4, k = 50),
  data = topo, chains = 2, seed = 20232023,
  warmup = 1000, iter = 2000, thin = 1, refresh = 0,
  control = list(adapt_delta = 0.99)
)
summary(bgamm2)
```

brms 包的函数 `gp()` 的参数 k 表示近似高斯过程 GP 所用的基函数的数目

```
me3 <- brms::conditional_effects(bgamm1, ndraws = 200, spaghetti = TRUE)
plot(me3, ask = FALSE, points = TRUE)
```

第三十三章　高斯过程回归

33.1　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 $\mathrm{MVN}(\boldsymbol{0},I)$，则 $\boldsymbol{X}=\boldsymbol{\mu}+C\boldsymbol{Z}$ 服从多元正态分布 $\mathrm{MVN}(\boldsymbol{\mu},\Sigma)$。

33.1.1　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;
}
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
  output_dir = "data-raw/",
```

```
    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>     <dec:4> <dec:4> <dec:4> <dec:> <dec:4> <dec> <dec>   <dec:4>  <dec:4>
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
```

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)
```

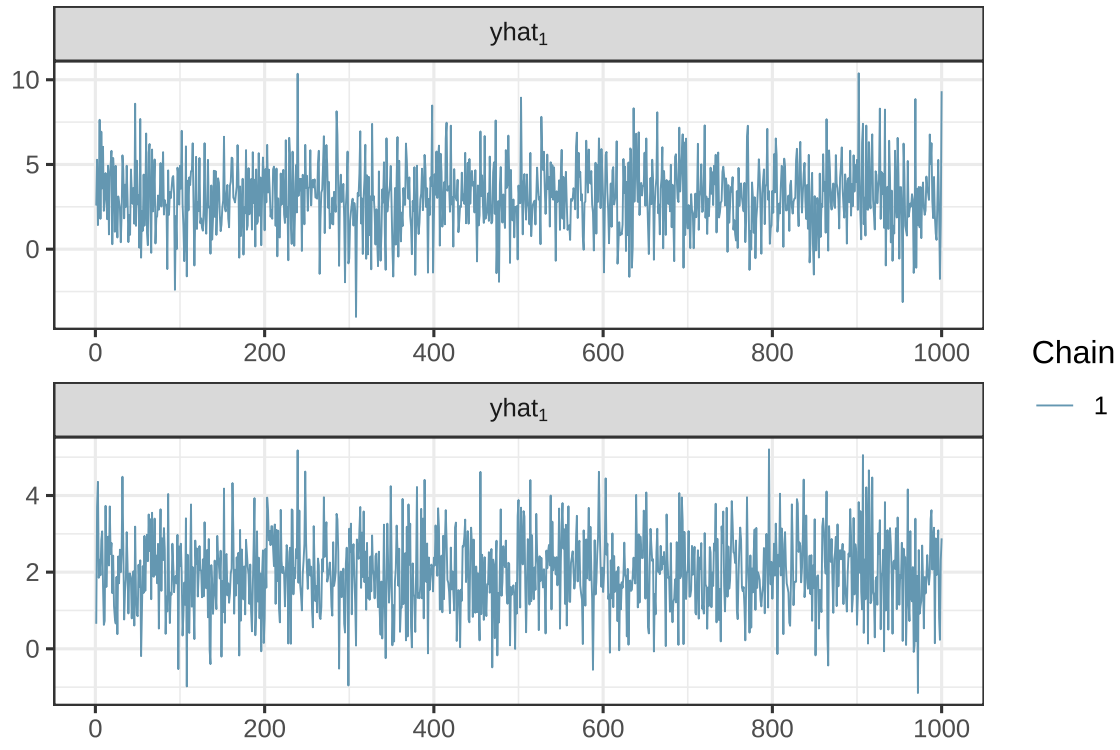


图 33.1: 采样序列的轨迹和分布

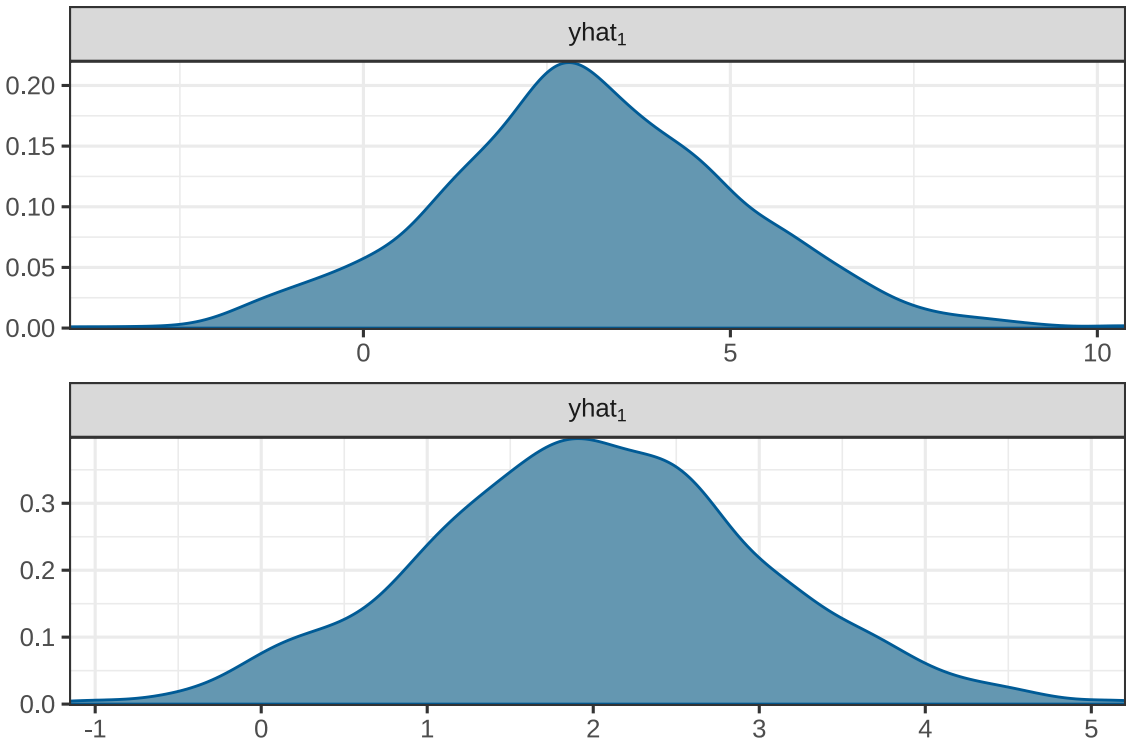


图 33.2: 采样序列的轨迹和分布

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]))
```

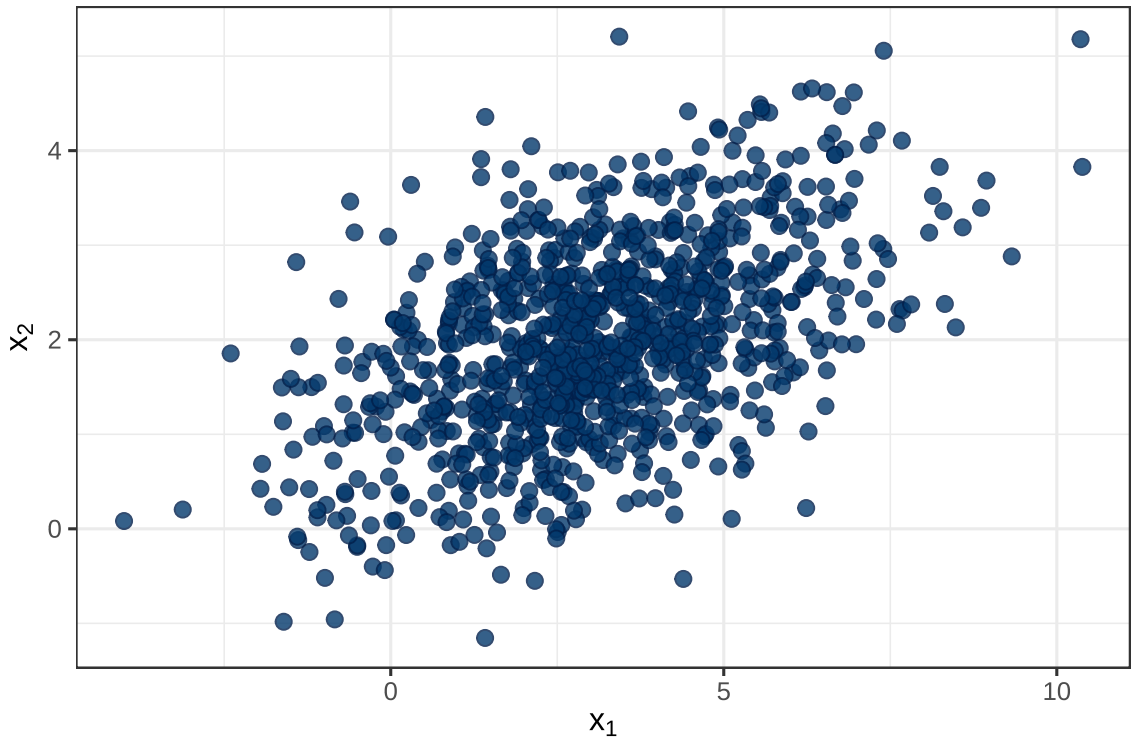


图 33.3: 生成二元正态分布的随机数

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
```

33.1.2 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
- 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。

```
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。

```
# 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
)
# 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
  output_dir = "data-raw/",
  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>      <dec:3> <dec:> <dec:> <dec:> <dec:> <dec:> <dec>   <dec:3>  <dec:3>
 1 mu[1]         1.02   1.02  0.0162 0.0154  0.999  1.05   1.00    1605.    1250.
 2 mu[2]         2.00   2.00  0.0385 0.0386  1.93   2.06   1.00    1300.    1138.
 3 mu[3]        -4.86  -4.86  0.0703 0.0724 -4.98  -4.75   1.00    1737.    1549.
 4 Sigma[1,1]    0.250  0.249 0.0110 0.0108  0.231  0.268  1.00    1847.    1336.
 5 Sigma[2,1]    0.427  0.426 0.0235 0.0234  0.388  0.466  1.00    1507.    1261.
 6 Sigma[3,1]    0.210  0.209 0.0361 0.0363  0.151  0.271  1.00    1472.    1202.
 7 Sigma[1,2]    0.427  0.426 0.0235 0.0234  0.388  0.466  1.00    1507.    1261.
 8 Sigma[2,2]    1.47   1.47  0.0649 0.0635  1.37   1.58   1.01    1506.    1265.
```

```
 9 Sigma[3,2]  -1.41  -1.41  0.0955 0.0957 -1.57  -1.26   1.00    1643.    1572.
10 Sigma[1,3]   0.210  0.209 0.0361 0.0363  0.151  0.271  1.00    1472.    1202.
11 Sigma[2,3]  -1.41  -1.41  0.0955 0.0957 -1.57  -1.26   1.00    1643.    1572.
12 Sigma[3,3]   5.28   5.27  0.225  0.221   4.92   5.67   1.00    1605.    1603.
```

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]")),
  diag_fun = "dens", off_diag_fun = "hex"
)
```

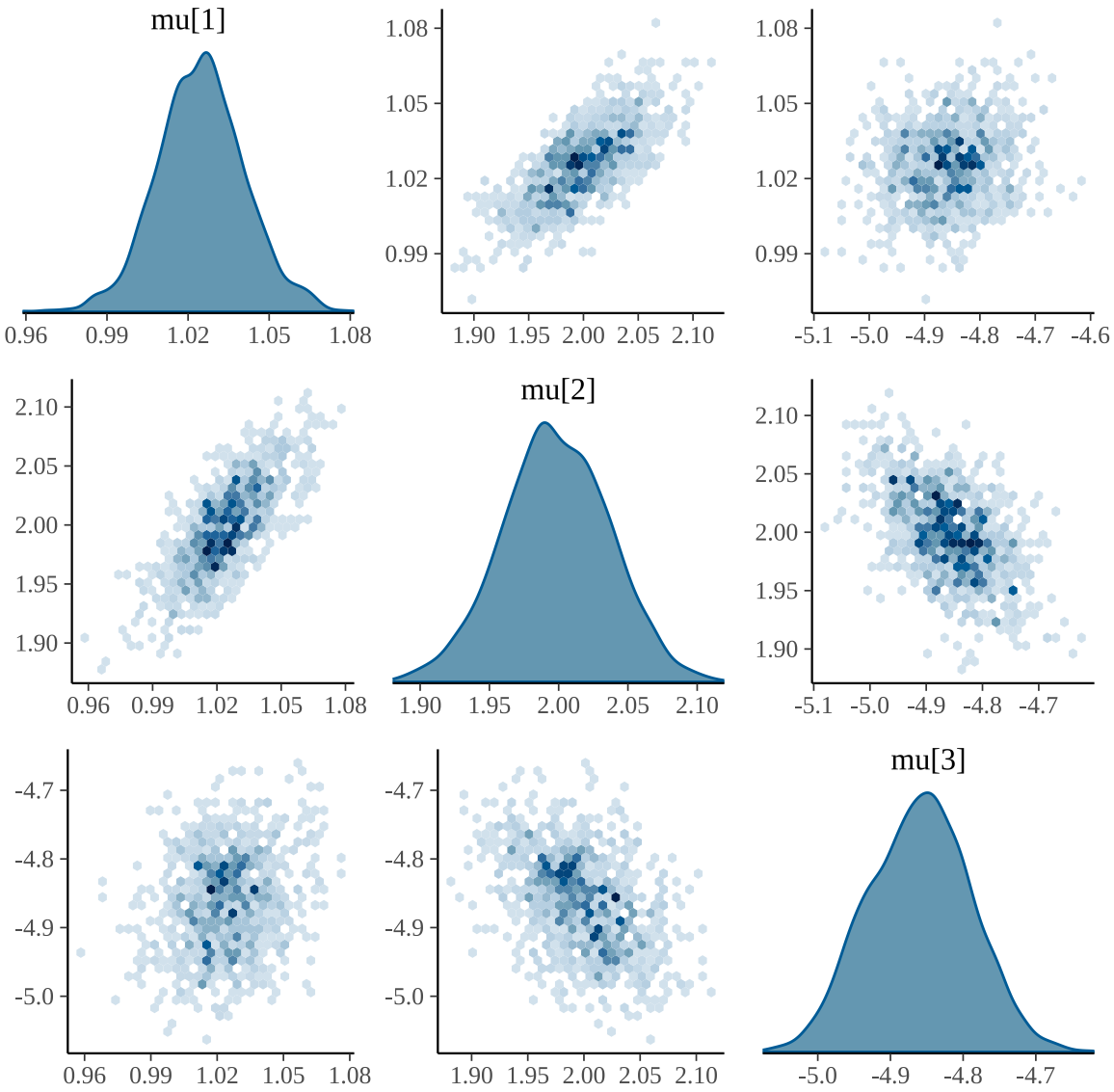


图 33.4: 三元正态分布

33.2 二维高斯过程

定义

33.2.1 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，

$$\sigma = 10, \phi = 1$$

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
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
  output_dir = "data-raw/",
  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)
```

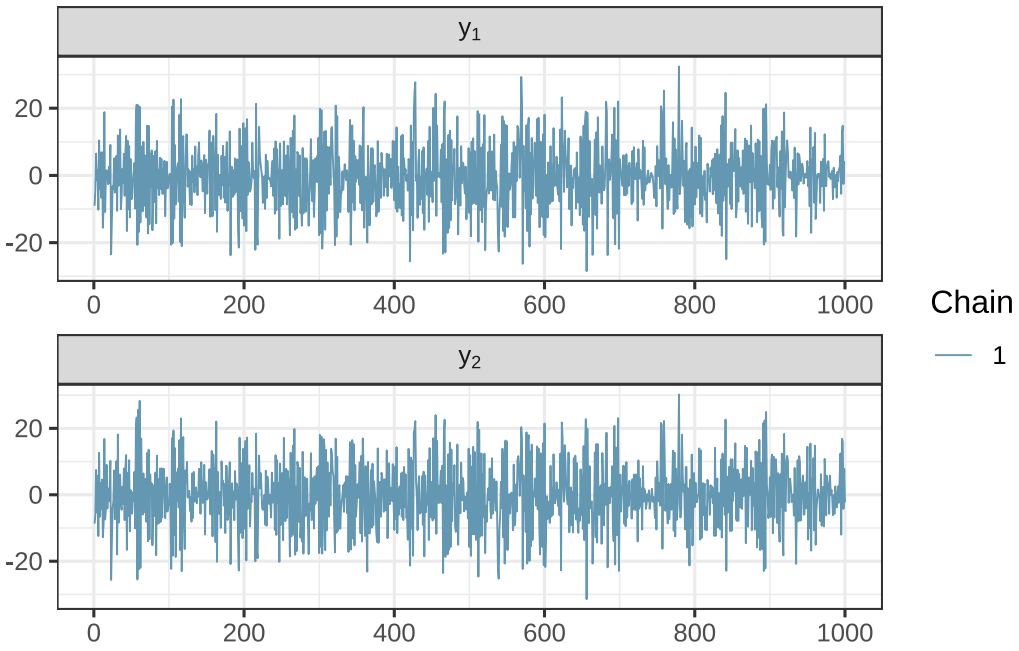


图 33.5: 位置 1 和 2 处的迭代轨迹

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] 100.5436
```

```
# y[1] 的标准差
sd(y1_mean)
```

```
[1] 10.02714
```

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, ])
```

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]))
```

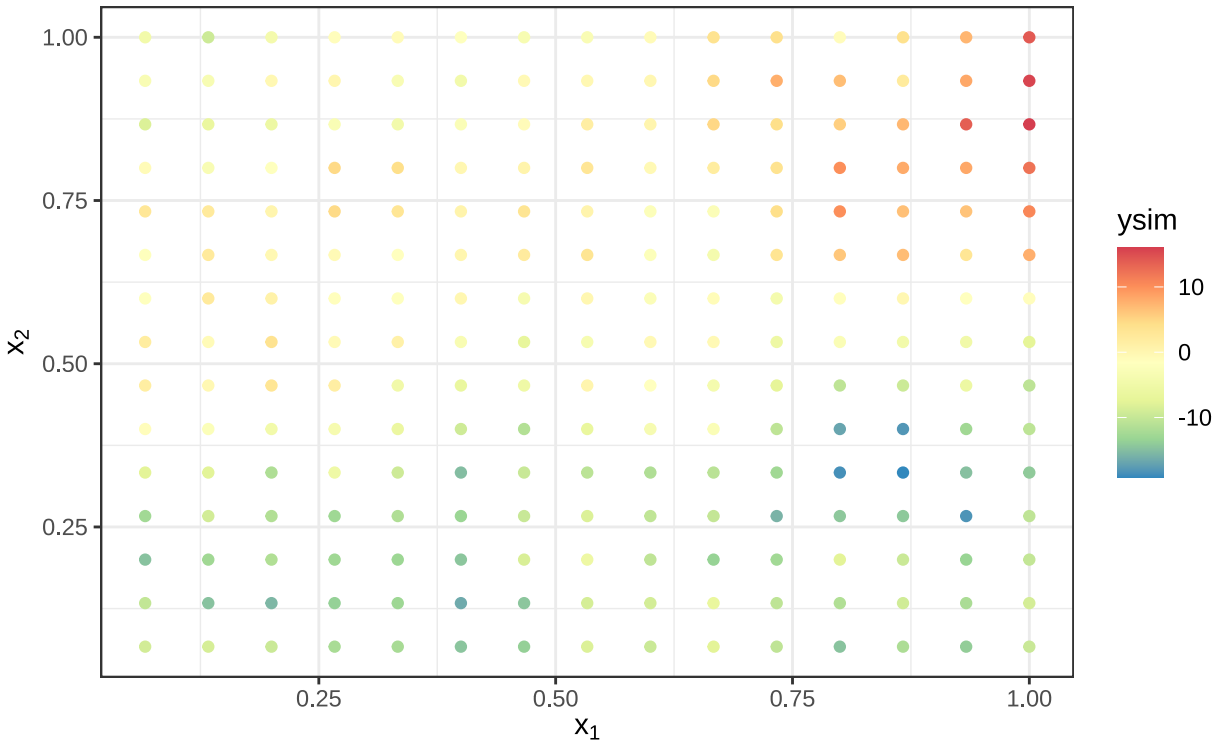


图 33.6: 二维高斯过程

33.2.2 二维高斯过程拟合

模拟二维高斯过程的数据，按照 Stan 官网给出的拟合代码，效率极低。

```
# 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)
)

# 拟合二维高斯过程
fit_gaussian_process <- mod_gaussian_process$sample(
  data = gaussian_process_d,     # 观测数据
  init = inits_gaussian_process, # 迭代初值
  iter_warmup = 1000,   # 每条链预处理迭代次数
  iter_sampling = 2000, # 每条链总迭代次数
  chains = nchains,     # 马尔科夫链的数目
  parallel_chains = 2,  # 指定 CPU 核心数，可以给每条链分配一个
  threads_per_chain = 2, # 每条链设置一个线程
  show_messages = FALSE, # 不显示迭代的中间过程
  refresh = 0,           # 不显示采样的进度
  output_dir = "data-raw/",
  seed = 20232023        # 设置随机数种子，不要使用 set.seed() 函数
)
# 诊断
fit_gaussian_process$diagnostic_summary()
```

```
$num_divergent
[1] 0 0
```

```
$num_max_treedepth
[1] 0 0

$ebfmi
[1] 1.097944 1.083674
```

输出结果

```
fit_gaussian_process$summary()
```

```
# A tibble: 3 x 10
  variable     mean   median     sd    mad       q5      q95  rhat ess_bulk
  <chr>       <num>    <num>  <num>  <num>    <num>    <num> <num>    <num>
1 lp__     -354.    -354.    0.964  0.703  -356.    -353.      1.00    1395.
2 phi         0.308    0.302 0.0503 0.0486    0.235    0.400  1.00    1196.
3 sigma       5.25     5.24  0.389  0.375     4.65     5.94   1.00    1246.
# i 1 more variable: ess_tail <num>
```

33.3 高斯过程回归

33.3.1 模型介绍

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)。

$$\log\{\lambda(x_i)\} = \beta + S(x_i)$$
$$y(x_i) \sim \text{Poisson}\big(t_i\lambda(x_i)\big)$$

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：

$$\text{Cov}\{S(x_i), S(x_j)\} = \sigma^2 \exp\big(-\frac{\|x_i - x_j\|_2}{\phi}\big)$$
$$\text{Cov}\{S(x_i), S(x_j)\} = \sigma^2 \exp\big(-\frac{\|x_i - x_j\|_2^2}{2\phi^2}\big)$$
$$\text{Cov}\{S(x_i), S(x_j)\} = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{\|x_i - x_j\|_2}{\phi}\right)^{\nu} K_\nu \left(\sqrt{2\nu}\frac{\|x_i - x_j\|_2}{\phi}\right)$$
$$K_\nu(x) = \int_0^\infty \exp(-x\cosh t)\cosh(\nu t)\mathrm{d}t$$

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。

33.3.2 观测数据

```
# 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
)
# 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)
  )
})
```

33.3.3 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）")
```

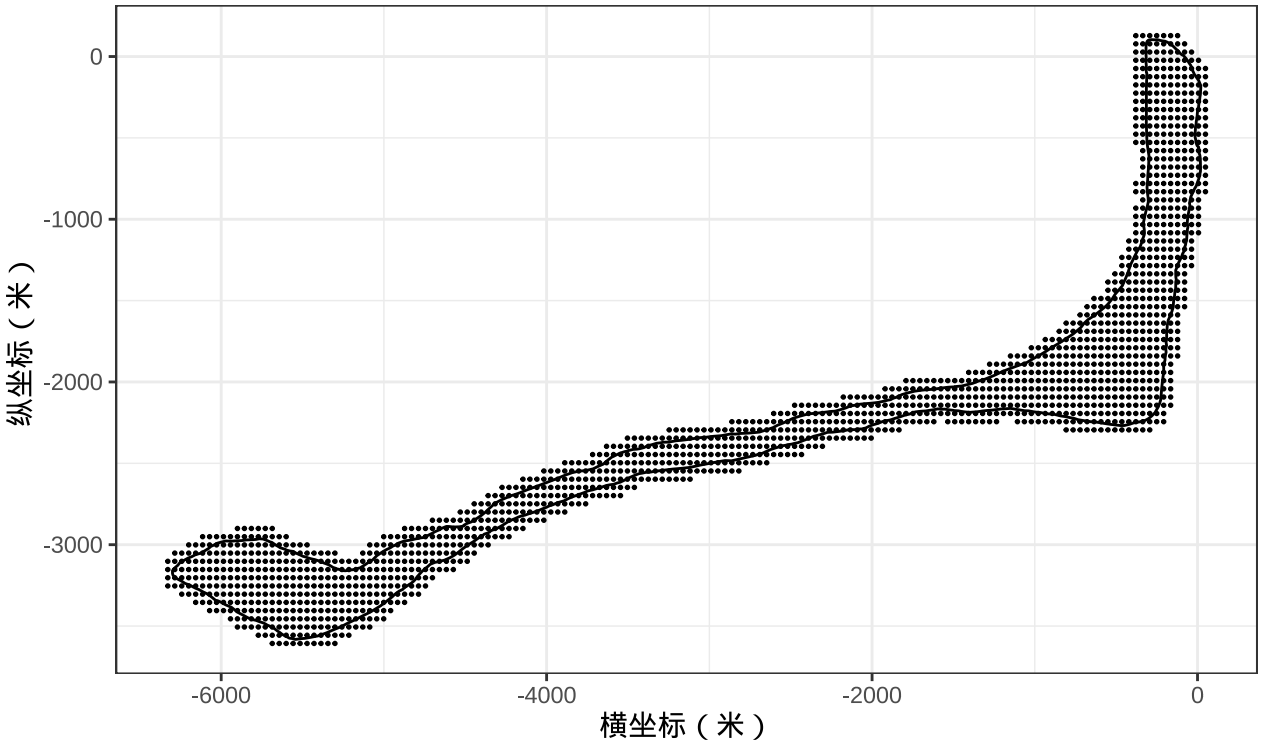


图 33.7: 朗格拉普岛

33.3.4 模型编码

指定各个参数 β, σ, ϕ 的先验分布

$$
\begin{aligned}
\beta &\sim \text{std\_normal}(0, 1) \\
\sigma &\sim \text{inv\_gamma}(5, 5) \\
\phi &\sim \text{half\_std\_normal}(0, 1) \\
\boldsymbol{\lambda} &\sim \text{multivariate\_normal}(\boldsymbol{\beta}, \sigma^2\Sigma + \delta^2 I) \\
\boldsymbol{y} &\sim \text{poisson\_log}\big(\log(\text{offsets}) + \boldsymbol{\lambda}\big)
\end{aligned}
$$

其中，$\beta, \sigma, \phi, \delta, \Sigma, I$ 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);
  phi ~ std_normal();

  lambda ~ multi_normal_cholesky(mu, L_K);
  y ~ poisson_log(log_offsets + lambda);
}
```

```
# 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
  output_dir = "data-raw/",
  seed = 20232023
)
# 诊断
fit_rongelap_poisson$diagnostic_summary()
```

```
$num_divergent
[1] 0 0

$num_max_treedepth
[1] 0 0

$ebfmi
[1] 1.078629 1.048734
```

```
# 泊松对数模型
fit_rongelap_poisson$summary(
  variables = c("lp__", "beta", "sigma", "phi"),
  .num_args = list(sigfig = 3, notation = "dec")
)
```

```
# A tibble: 4 x 10
  variable         mean       median      sd      mad           q5          q95
  <chr>          <dec:3>      <dec:3> <dec:3>  <dec:3>      <dec:3>      <dec:3>
1 lp__      3402193.     3402190        9.44   14.8     3402180      3402210
2 beta            1.78         1.79    0.139   0.116         1.56         1.97
3 sigma           0.633        0.611   0.112   0.0749        0.516        0.817
4 phi             0.0267       0.0233  0.0158  0.00780       0.0144       0.0468
# i 3 more variables: rhat <dec:3>, ess_bulk <dec:3>, ess_tail <dec:3>
```

```
# 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)
```

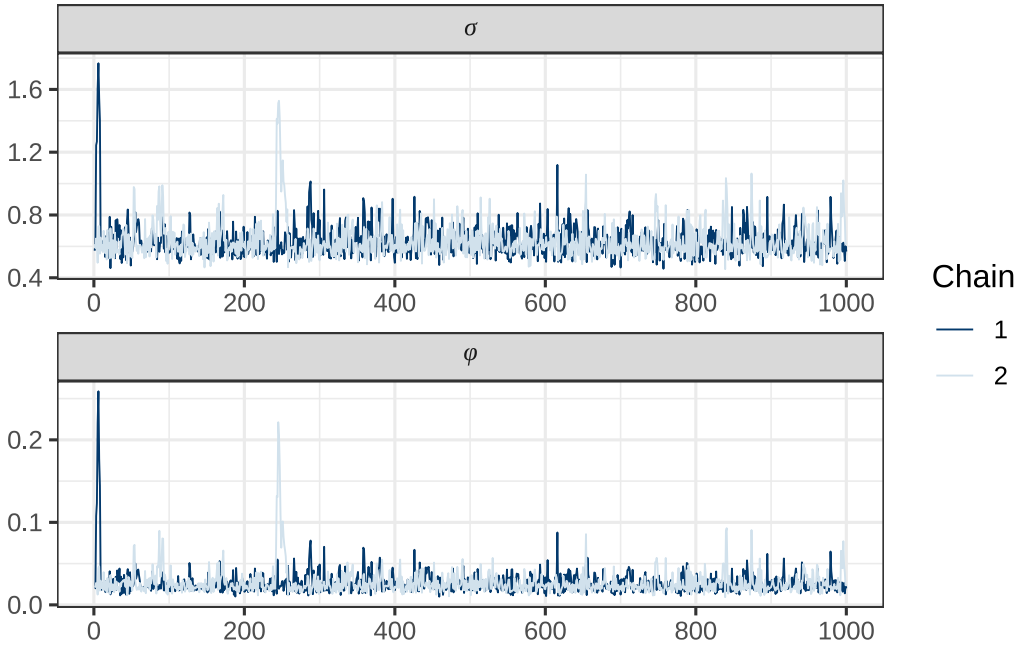


图 33.8: σ 和 ϕ 的迭代轨迹

```
# 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)
```

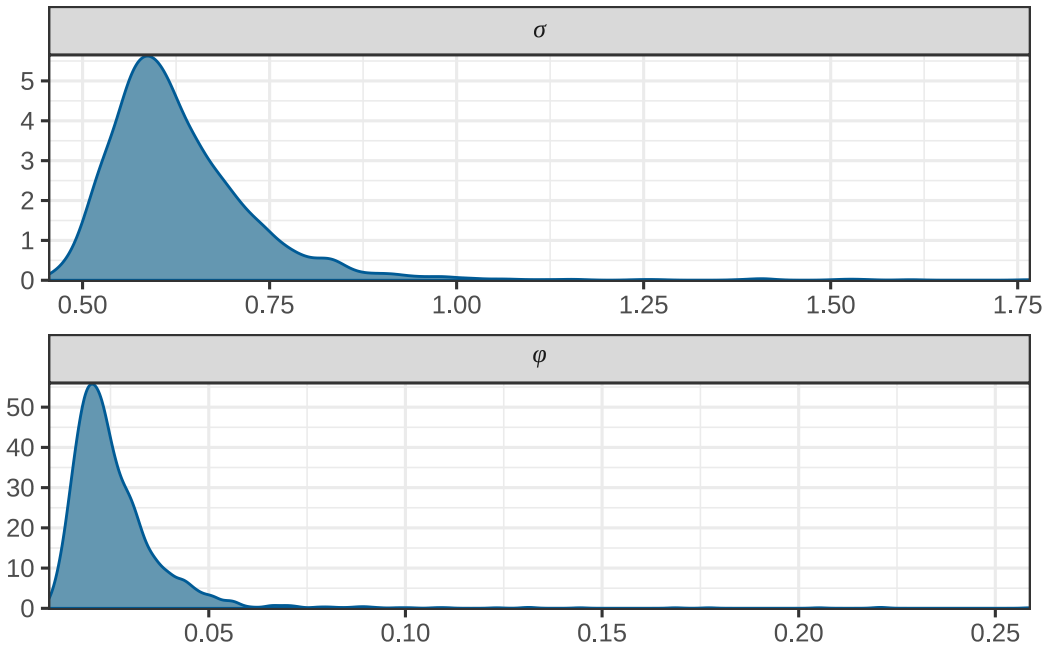


图 33.9: σ 和 ϕ 的后验分布

33.3.5 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;
      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];
  }
}
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) {
    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$sample(
  data = rongelap_poisson_pred_d,   # 观测数据
  init = inits_data_poisson_pred,   # 迭代初值
  iter_warmup = 500,            # 每条链预处理迭代次数
  iter_sampling = 1000,         # 每条链总迭代次数
  chains = nchains,             # 马尔科夫链的数目
  parallel_chains = 2,      # 指定 CPU 核心数，可以给每条链分配一个
  threads_per_chain = 2,    # 每条链设置一个线程
  show_messages = FALSE,    # 不显示迭代的中间过程
  refresh = 0,              # 不显示采样的进度
  output_dir = "data-raw/",
  seed = 20232023           # 设置随机数种子，不要使用 set.seed() 函数
)
# 诊断信息
fit_rongelap_poisson_pred$diagnostic_summary()
```

```
$num_divergent
[1] 0 0

$num_max_treedepth
[1] 0 0

$ebfmi
[1] 0.9731337 1.0948685
```

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>     <num>  <num>  <num>    <num>  <num>  <num> <num>    <num>     <num>
1 beta     1.77   1.79   0.161  0.116    1.54   1.98    1.00     862.      422.
2 sigma    0.640  0.614  0.125  0.0801   0.515  0.838   1.00     698.      357.
3 phi      0.0277 0.0240 0.0183 0.00822 0.0142 0.0506  1.01     648.      378.
```

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   -937.7 4.2
p_loo       123.0 2.5
looic      1875.3 8.4
------
Monte Carlo SE of elpd_loo is NA.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)       1    0.6%   285
 (0.5, 0.7]   (ok)        20   12.7%   53
   (0.7, 1]   (bad)      107   68.2%   9
   (1, Inf)   (very bad)  29   18.5%   3
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()
```

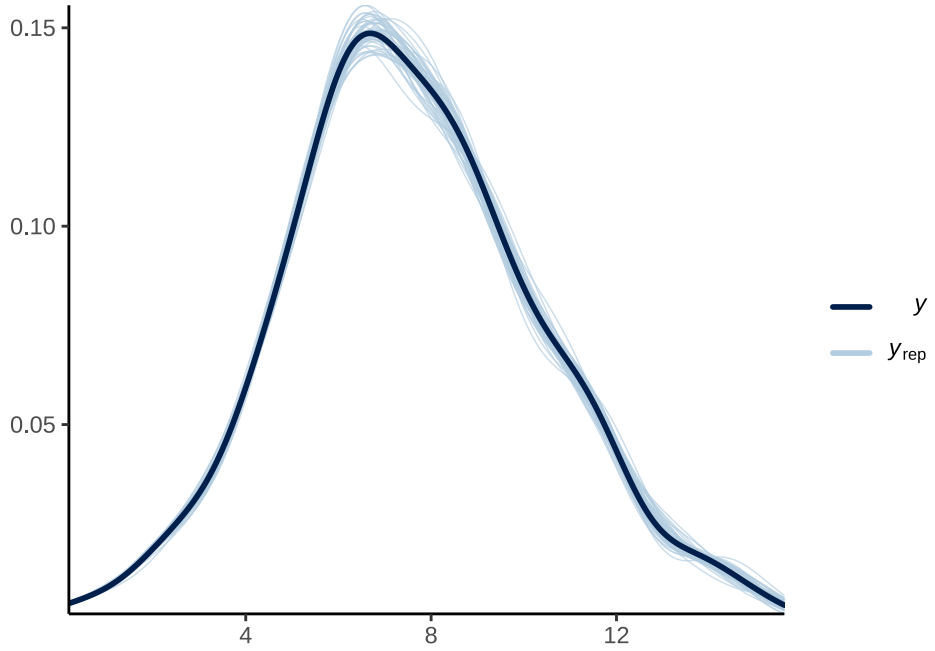


图 33.10: 后验预测诊断图（密度图）

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

```
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) -1.225810   86
2     371  300 POINT (-6050 -3165)  0.158503  350
3    1931  300 POINT (-5925 -3320)  1.917280 2042
4    4357  300 POINT (-5925 -3165)  2.695390 4559
5    2114  300 POINT (-5800 -3350)  1.940120 2118
```

```
6    2318  300 POINT (-5800 -3165)  2.054870 2382
7    1975  300 POINT (-5625 -3350)  1.902300 1958
8    1912  300 POINT (-5700 -3260)  1.893520 1953
9    1902  300 POINT (-5700 -3220)  1.842530 1956
10   1882  300 POINT (-5700 -3180)  1.854380 1920
```

观测值和预测值的情况

```
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
```

```
summary(rongelap_sf$yhat / rongelap_sf$time)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.2867  5.9050  7.4867  7.6162  9.5000 15.6233
```

展示采样点处的预测值

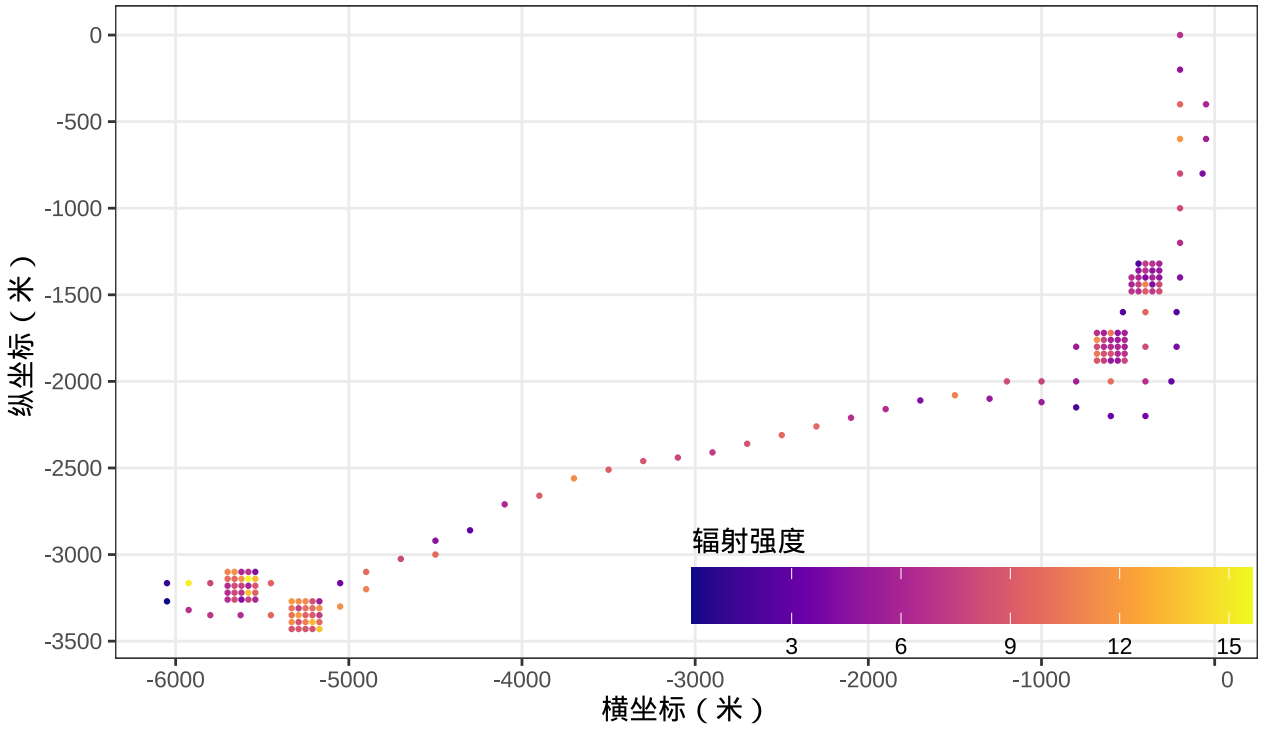


图 33.11: 朗格拉普岛核辐射强度的分布

未采样点的预测

```
# 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  100 752.3305
2 -5643.145 -3606.997  100 754.3100
3 -5600.347 -3606.997  100 760.3005
4 -5557.549 -3606.997  100 779.2050
5 -5514.751 -3606.997  100 786.0715
6 -5471.953 -3606.997  100 797.1550
```

```
# 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.
 0.4941  6.2172  7.4148  7.3111  8.5660 12.8870
```

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)
```

33.4 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,
  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"
```

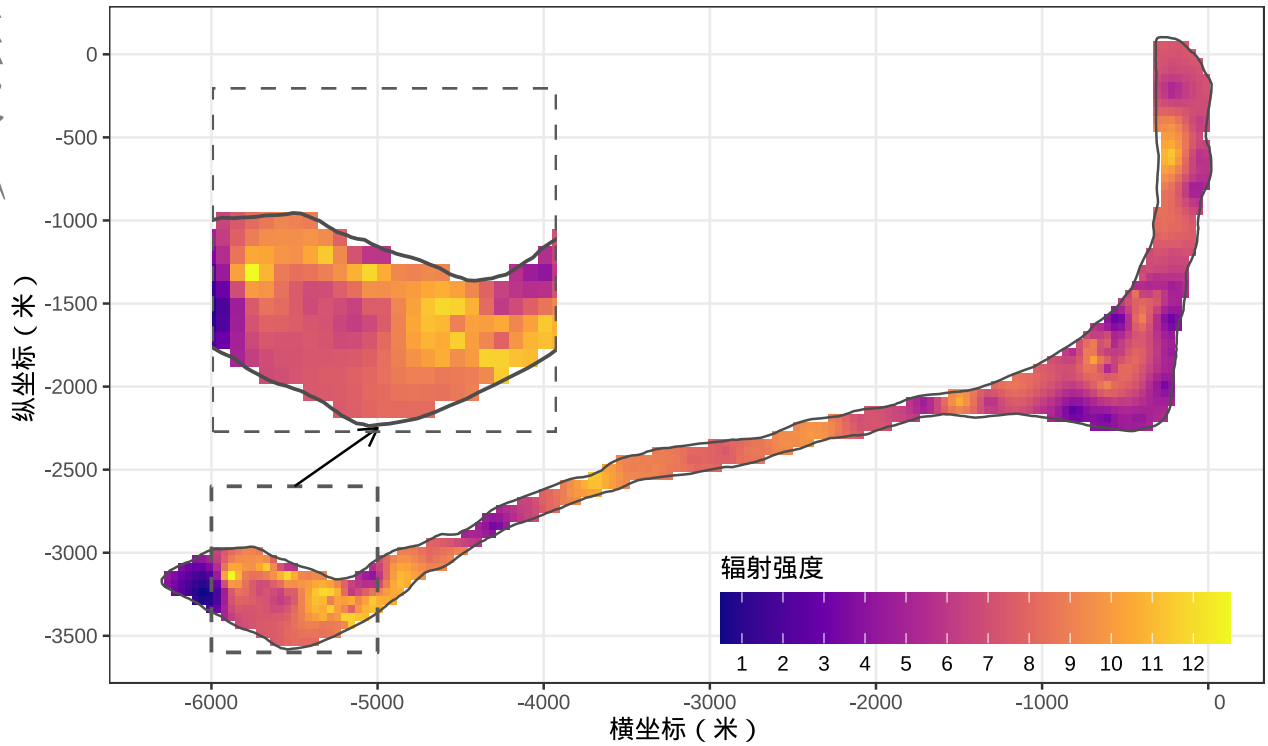


图 33.12: 朗格拉普岛核辐射强度的分布

```
)
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"
)
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 33.1: 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。

33.5 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);
    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。

第八部分

机器学习

第三十四章　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。

- 线性分类器
  - 多项回归模型
  - 线性判别分析
- 非线性分类器
  - 二次判别分析
  - 朴素贝叶斯
  - 支持向量机
  - K 最近邻
  - 神经网络
  - 决策树
  - 随机森林
  - 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。

34.1 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")
```

```
plot(iris_glmnet)
```

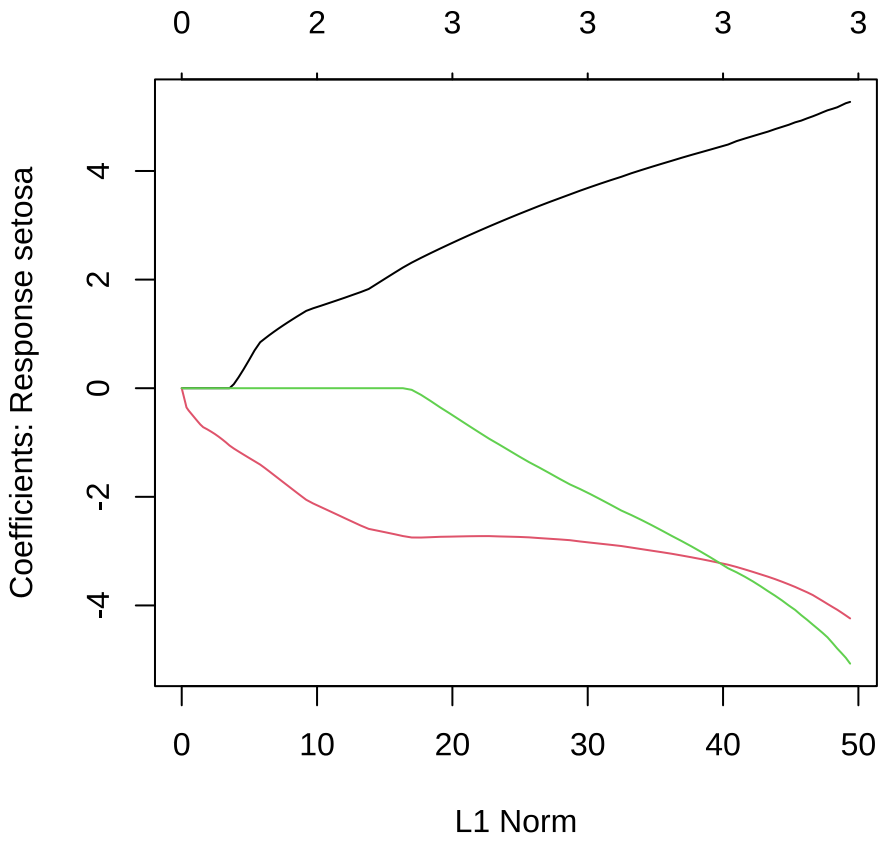


图 34.1: 回归系数的迭代路径

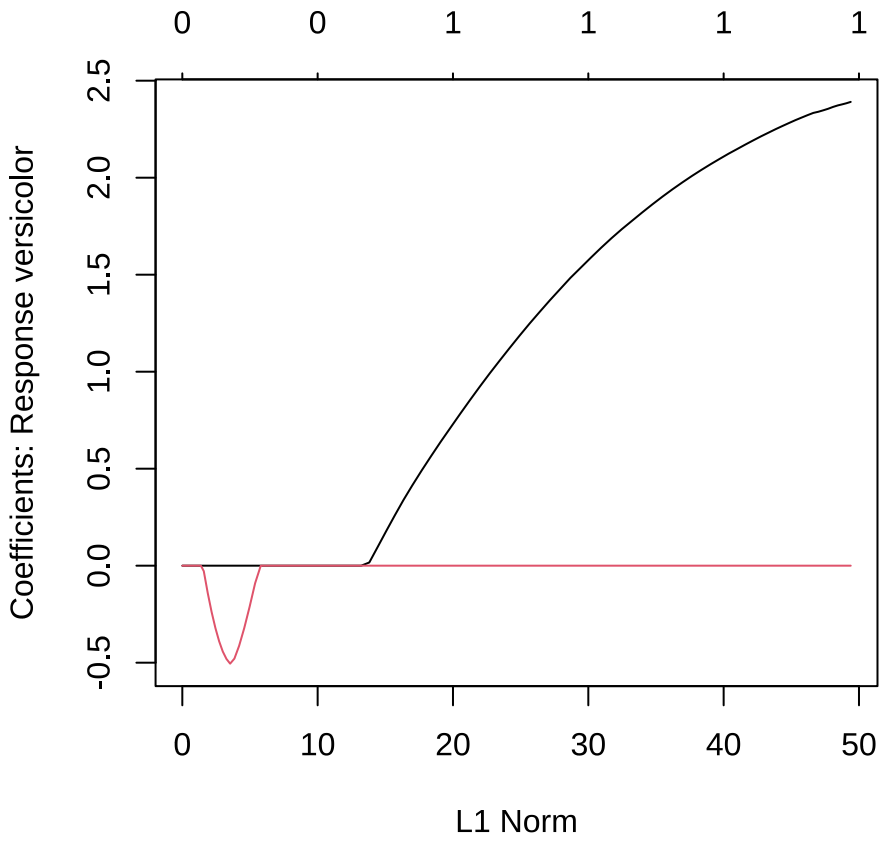


图 34.2: 回归系数的迭代路径

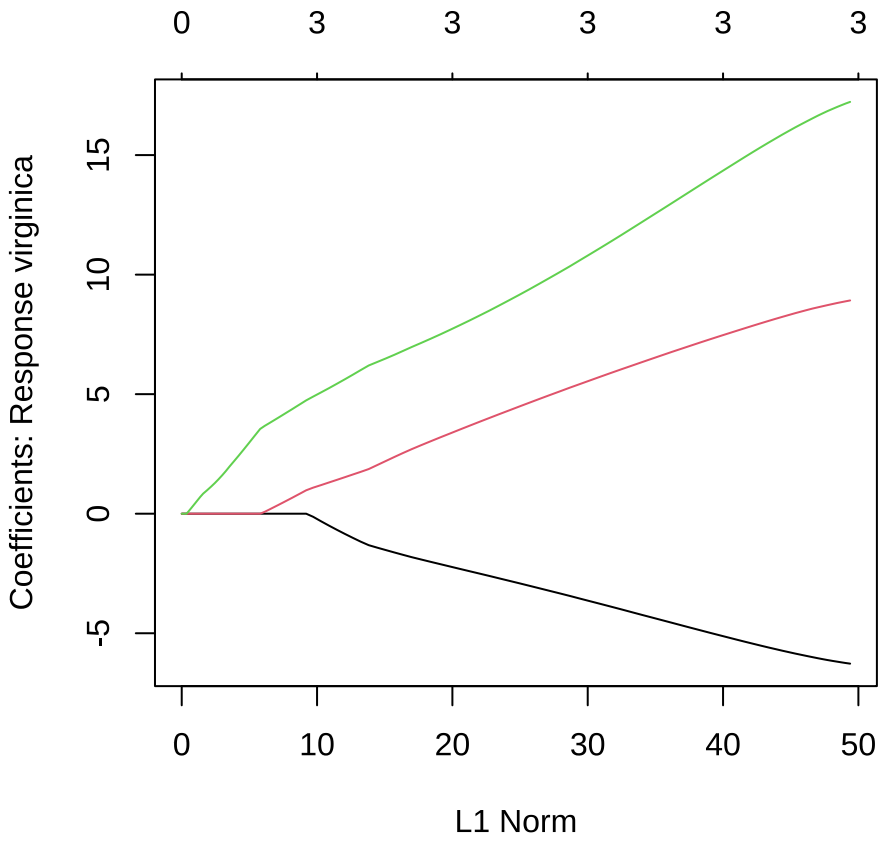


图 34.3: 回归系数的迭代路径

```
plot(iris_glmnet$lambda,
  ylab = expression(lambda), xlab = "迭代次数",
  main = "惩罚系数的迭代路径"
)
```

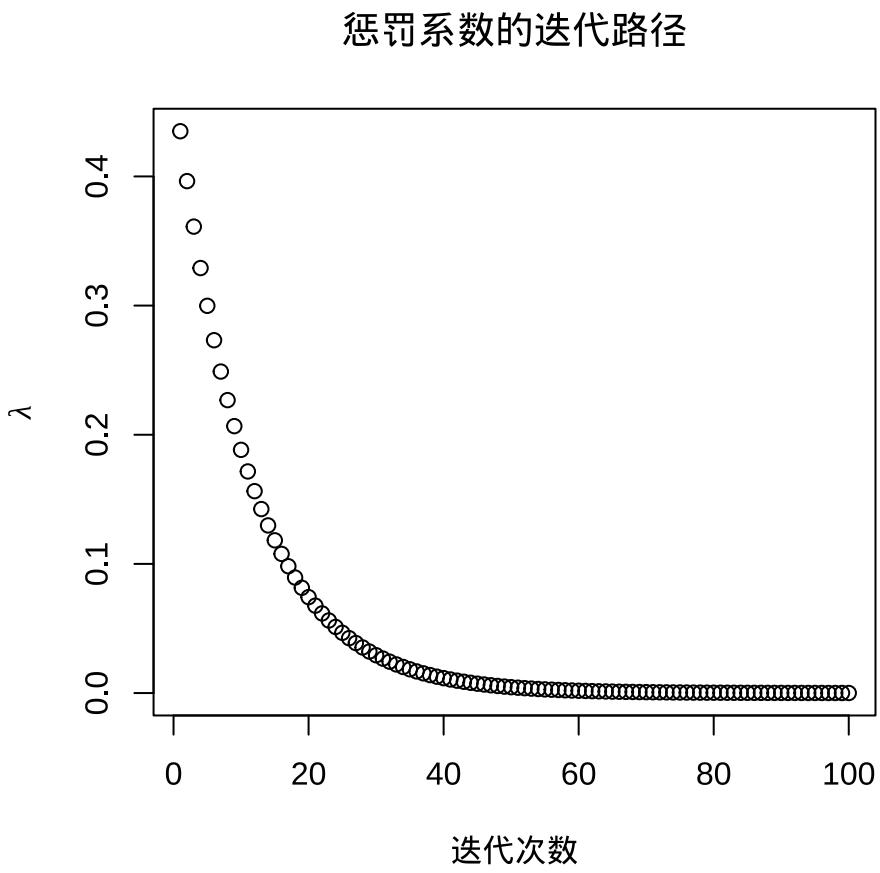


图 34.4: 惩罚系数的迭代路径

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])
```

```
iris_pred_glmnet setosa versicolor virginica
      setosa         50          0         0
      versicolor      0         49         1
      virginica       0          1        49
```

34.2 线性判别分析

```
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
```

```
# 预测
iris_lda_pred <- predict(iris_lda, iris[, -5])$class
```

```
# 预测结果
table(iris_lda_pred, iris[, 5])
```

```
iris_lda_pred setosa versicolor virginica
   setosa         50          0         0
   versicolor      0         48         1
   virginica       0          2        49
```

34.3 二次判别分析

```
# 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
```

```
# 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
```

34.4 朴素贝叶斯

```
library(e1071) # 朴素贝叶斯
iris_nb <- naiveBayes(Species ~ ., data = iris)
iris_nb
```

```
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
```

```
# 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
  setosa         50          0         0
  versicolor      0         47         3
  virginica       0          3        47
```

34.5 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

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  1.64941903013327

Number of Support Vectors : 71

Objective Function Value : -6.8974 -7.8187 -19.3951
Training error : 0.02
```

kernlab 包 (Karatzoglou 等 2004) 的绘图函数 `plot()` 仅支持二分类模型。

```
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         49         2
   virginica       0          1        48
```

34.6 K 最近邻

```
# 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))
```

```
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         3
  virginica       0          2        22
```

34.7 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
 -2.74 -12.40  -8.49  -4.05  -0.80
 b->h2 i1->h2 i2->h2 i3->h2 i4->h2
```

```
-1.41  -0.79  -4.19  10.47   6.76
b->h3 i1->h3 i2->h3 i3->h3 i4->h3
-0.46  -6.36  -1.43  -5.88  -2.63
b->h4 i1->h4 i2->h4 i3->h4 i4->h4
-0.79  -1.00  -5.86  12.23   5.91
b->o1 h1->o1 h2->o1 h3->o1 h4->o1
 9.91  12.16   9.78  -2.72 -29.18
b->o2 h1->o2 h2->o2 h3->o2 h4->o2
-3.85  -1.45  -9.99   3.12  18.74
b->o3 h1->o3 h2->o3 h3->o3 h4->o3
-5.43 -10.16  -0.18  -1.20  10.51
```

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         24        24
    virginica       0         26        26
```

34.8 决策树

```
library(rpart)
iris_rpart <- rpart(Species ~ ., data = iris)
iris_rpart
```

```
n= 150

node), split, n, loss, yval, (yprob)
      * 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) *
```

```
library(rpart.plot)
rpart.plot(iris_rpart)
```

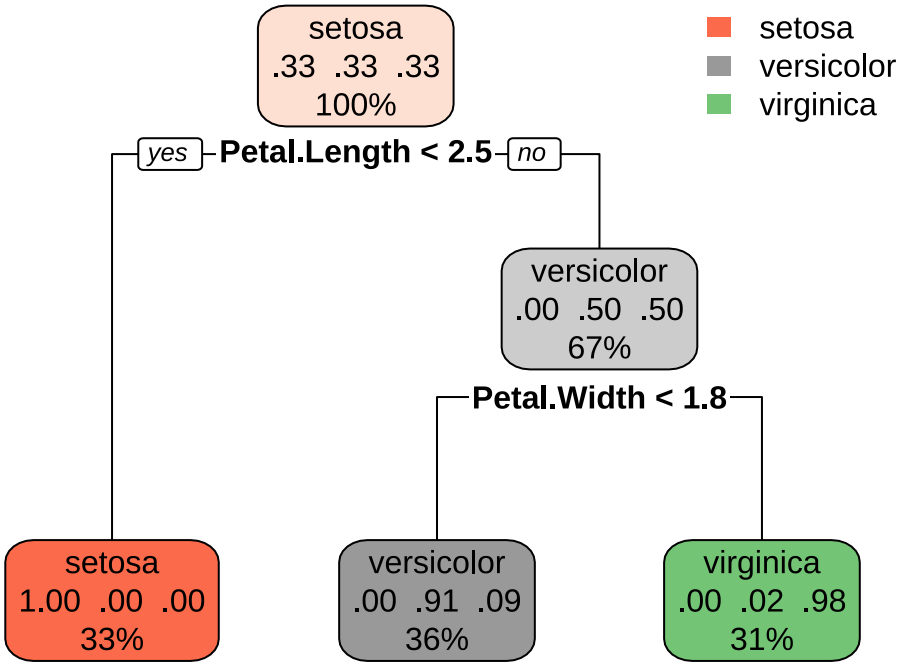


图 34.5: 分类回归树

预测结果，训练误差

```
# 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。

34.9 随机森林

```
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: 4.67%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         50          0         0        0.00
versicolor      0         47         3        0.06
virginica       0          4        46        0.08
```

```
varImpPlot(iris_rf, main = "变量重要性")
```

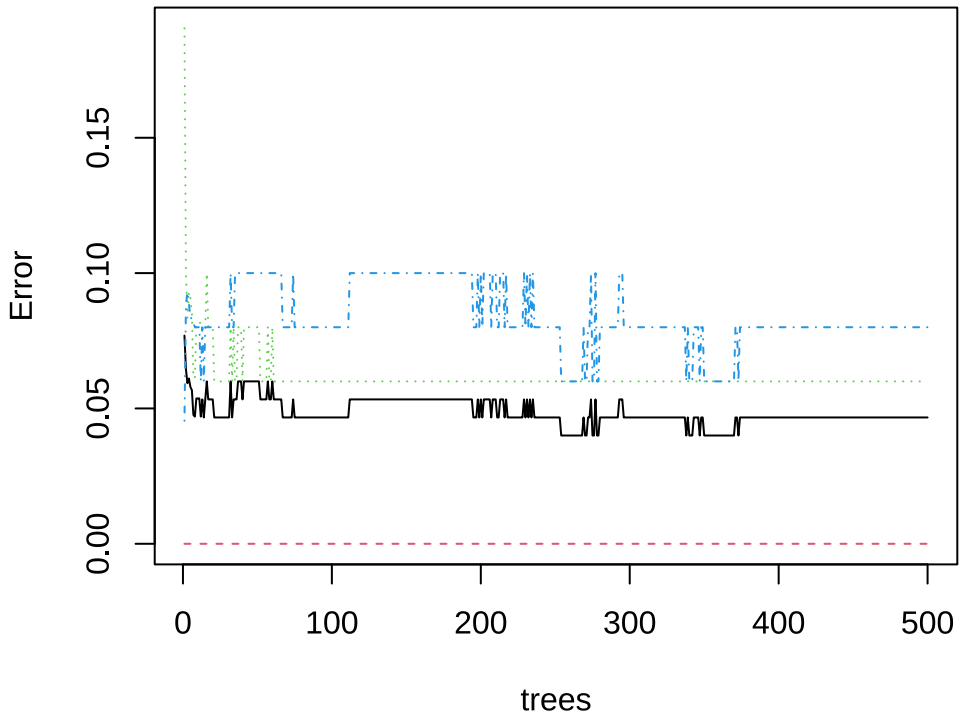


图 34.6: 随机森林

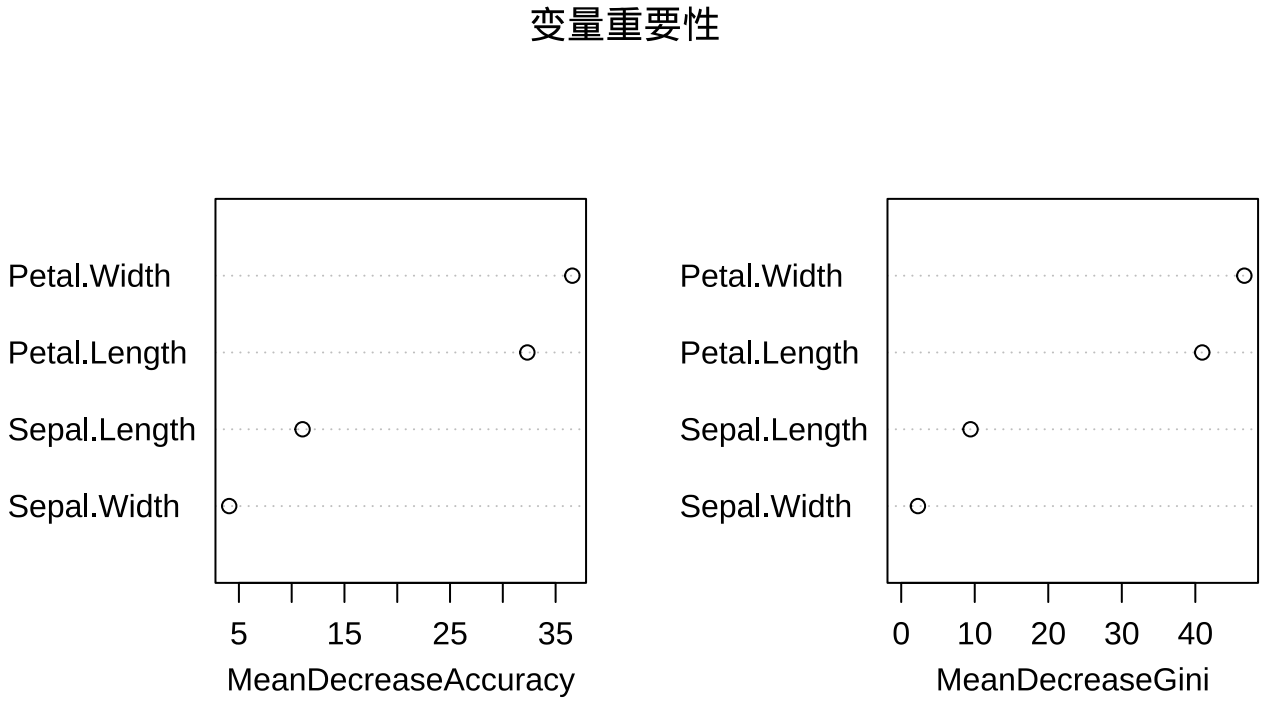


图 34.7: 变量重要性

```
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
```

34.10 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")
  )
}
```

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
  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
```

34.11 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")`

34.12 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。

第三十五章　聚类问题

第三十六章　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 ...
#>  $ 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。

36.1 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。

36.1.1 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
```

36.1.2 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
```

36.1.3 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    8.028    7.887    7.629    6.533    5.914    5.753
#> adjCV        9.206    8.027    7.888    7.627    6.533    5.909    5.749
#>
#> 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")
```

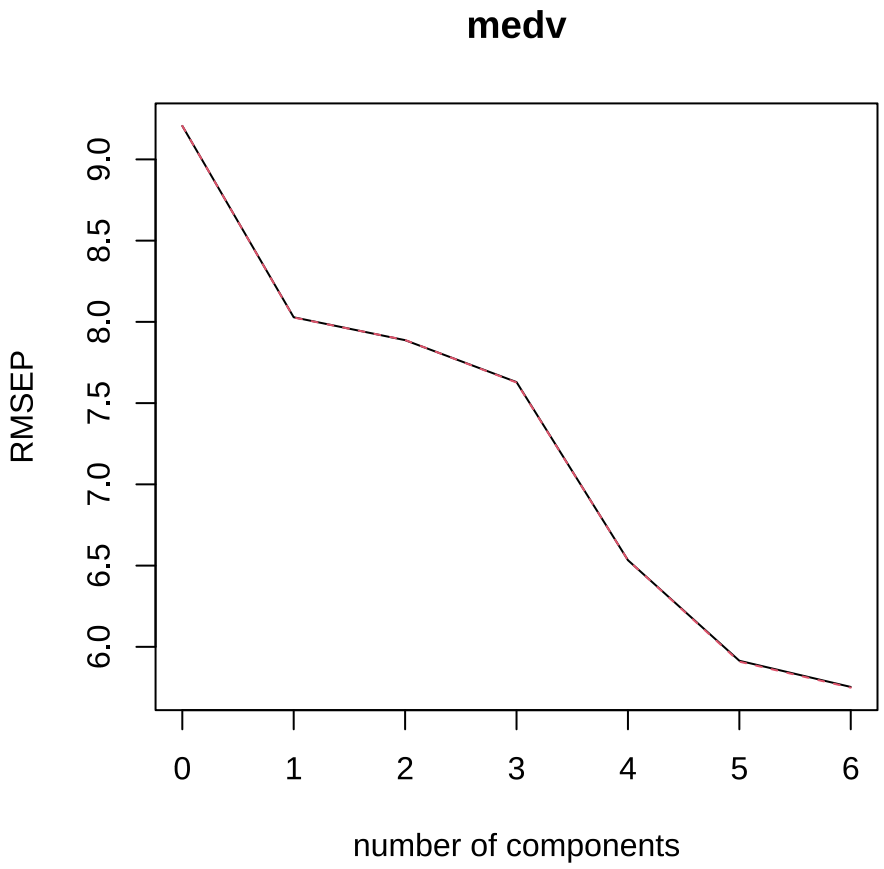


图 36.1: RMSE 随成分数量的变化

36.1.4 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    8.052    8.029    7.794    7.767    7.612    6.060
```

```
#> adjCV        9.206    8.051    8.028    7.792    7.765    7.618    6.053
#>
#> 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
```

36.2 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 36.1: 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	松弛 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。

36.2.1 岭回归

岭回归

$$\mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2 + \lambda \|\boldsymbol{\beta}\|_2^2$$

```
library(glmnet)
fit_ridge <- glmnet(x = Boston[, -14], y = Boston[, "medv"], family = "gaussian", alpha = 0)
```

```
plot(fit_ridge)
```

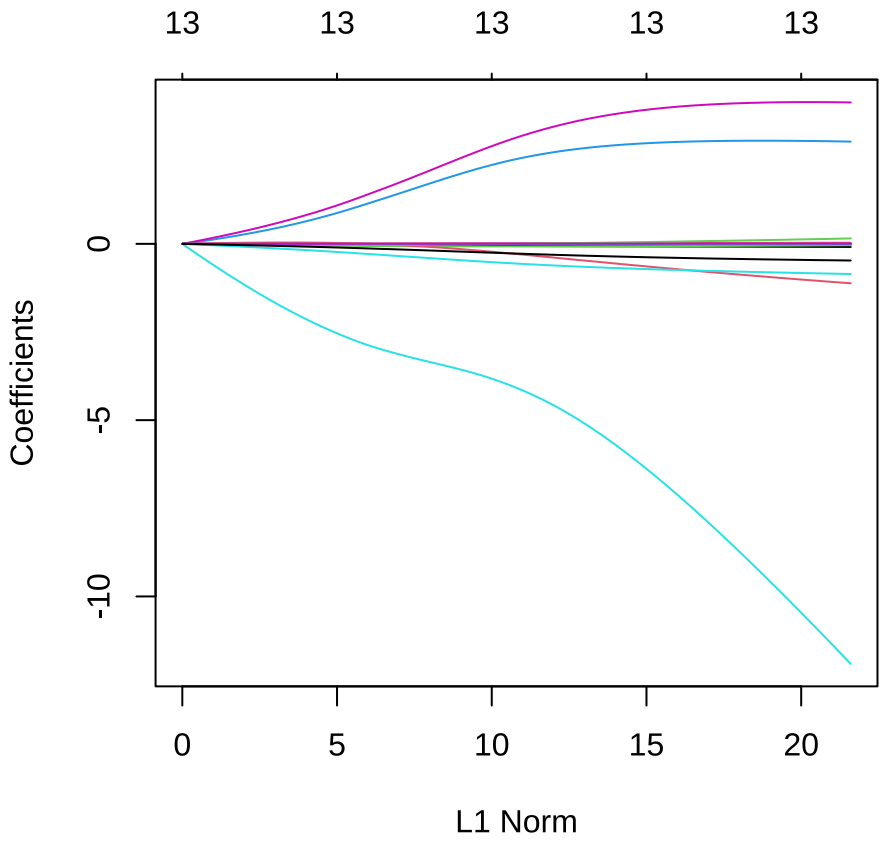


图 36.2: 回归系数的迭代路径

```
plot(fit_ridge$lambda,
  ylab = expression(lambda), xlab = "迭代次数",
  main = "惩罚系数的迭代路径"
)
```

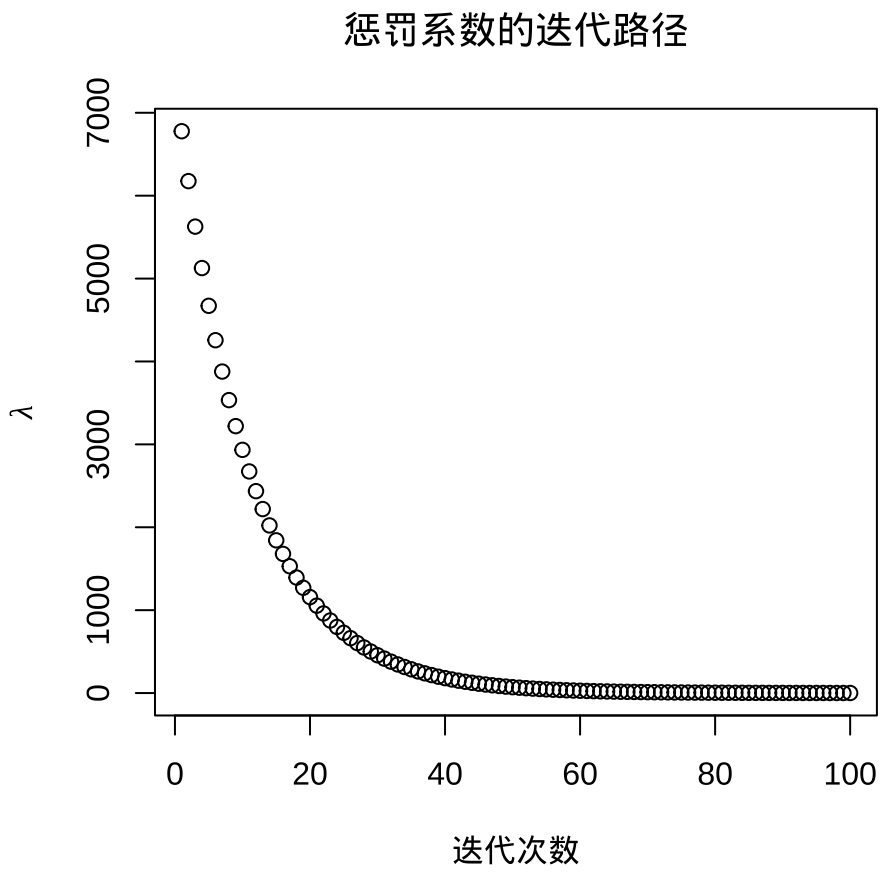


图 36.3: 惩罚系数的迭代路径

```
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
```

36.2.2 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)
```

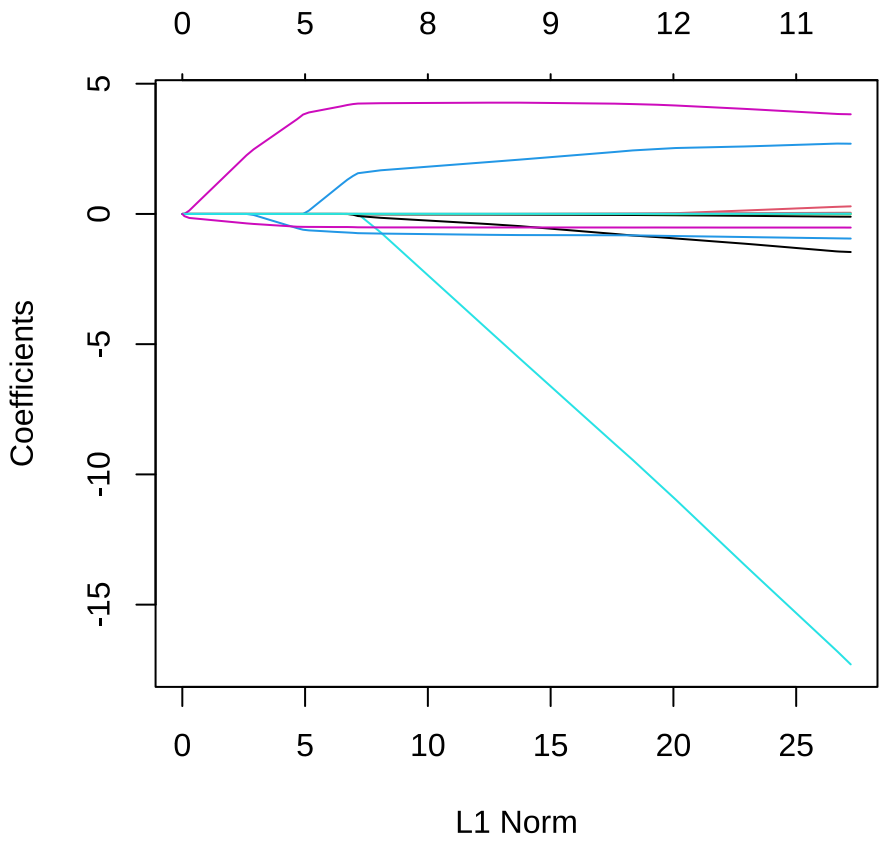


图 36.4: 回归系数的迭代路径

```
plot(fit_lasso$lambda,
  ylab = expression(lambda), xlab = "迭代次数",
  main = "惩罚系数的迭代路径"
)
```

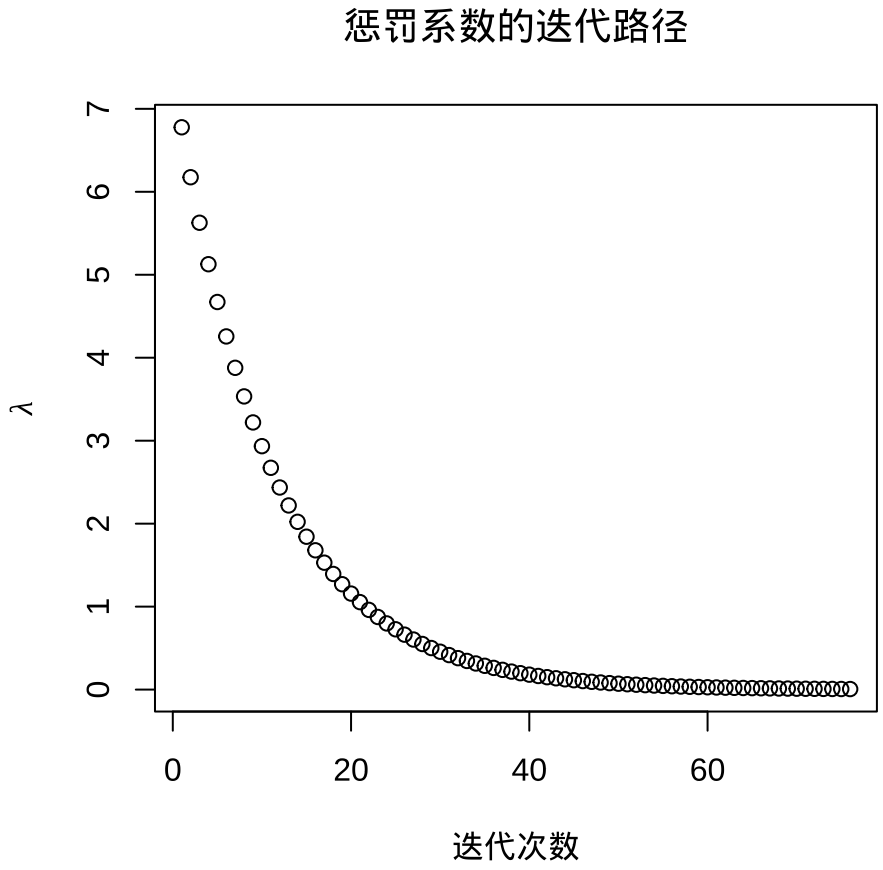


图 36.5: 惩罚系数的迭代路径

```
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
#> 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
```

36.2.3 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)
```

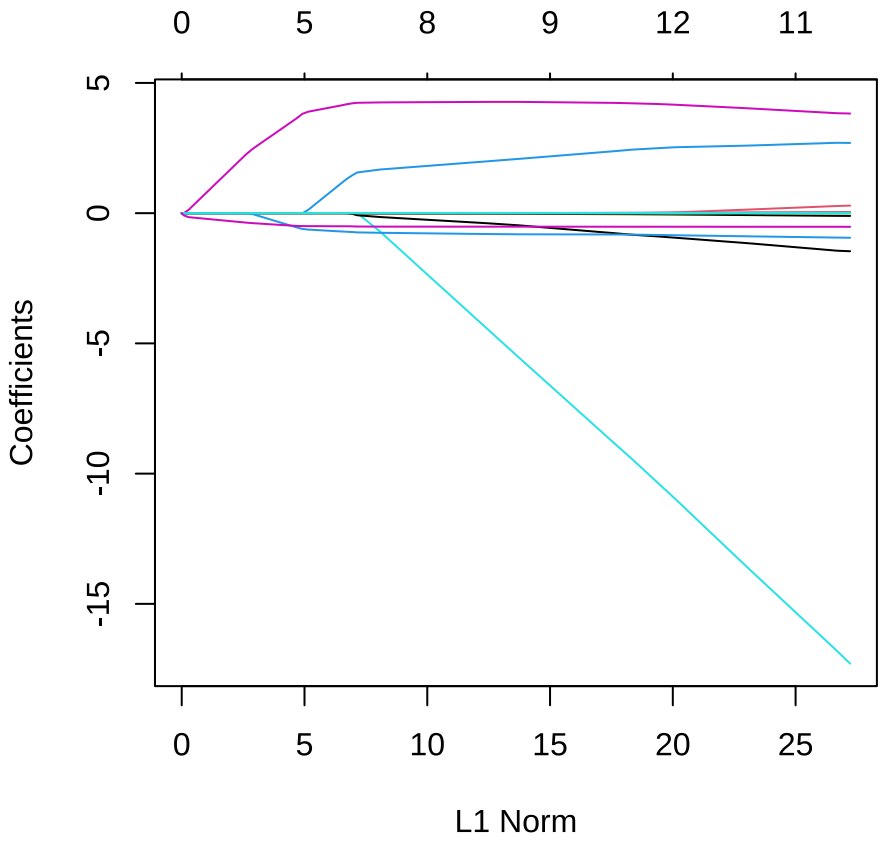


图 36.6: 回归系数的迭代路径

```
plot(fit_elasticnet$lambda,
  ylab = expression(lambda), xlab = "迭代次数",
  main = "惩罚系数的迭代路径"
)
```

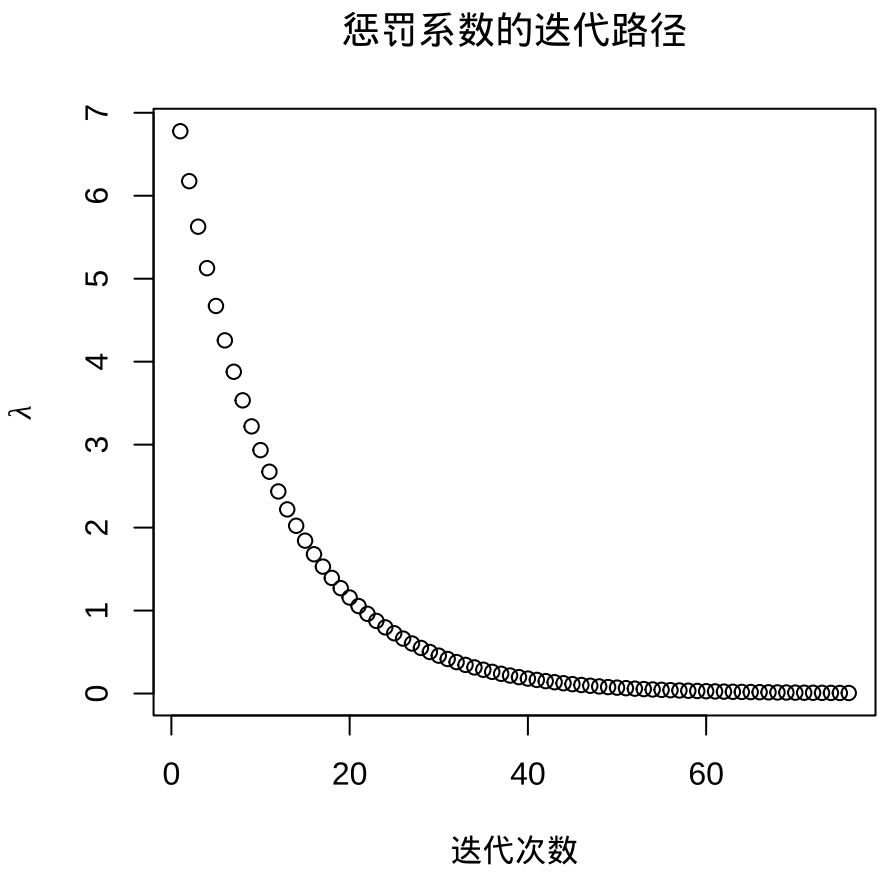


图 36.7: 惩罚系数的迭代路径

```
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
#> rad           0.252686499
```

```
#> tax          -0.009826799
#> ptratio      -0.929988657
#> black         0.009025875
#> lstat        -0.522499839
```

36.2.4 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 $w_j = (|\hat{\beta}_{ridge_j}|)^\gamma$，$\gamma$ 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。

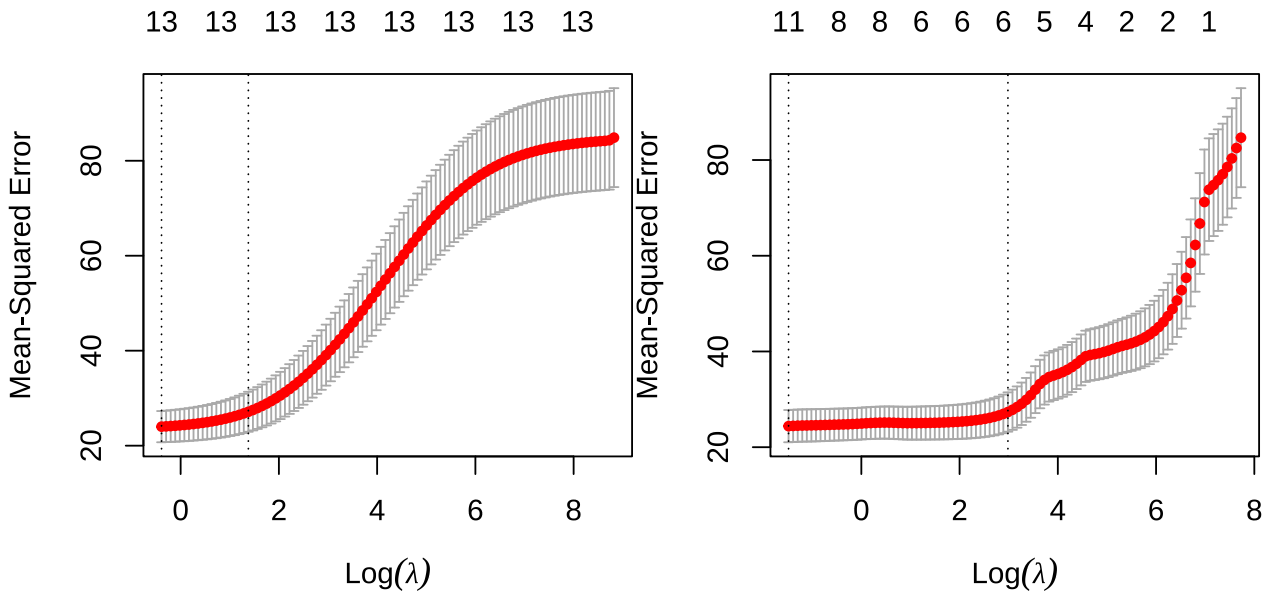
```
# 岭权重 gamma = 1
g <- 1
set.seed(20232023)
## 岭回归
ridge_model <- cv.glmnet(
  x = as.matrix(Boston[, -14]),
  y = Boston[, 14], alpha = 0
)
ridge_coef <- as.matrix(coef(ridge_model, s = ridge_model$lambda.min))
ridge_weight <- 1 / (abs(ridge_coef[-1, ]))^g

## 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)
```

λ 超参数



(a) 岭回归 (b) 自适应 Lasso 回归

图 36.8: 自适应 Lasso 回归模型的超参数选择

```
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
```

36.2.5 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)
```

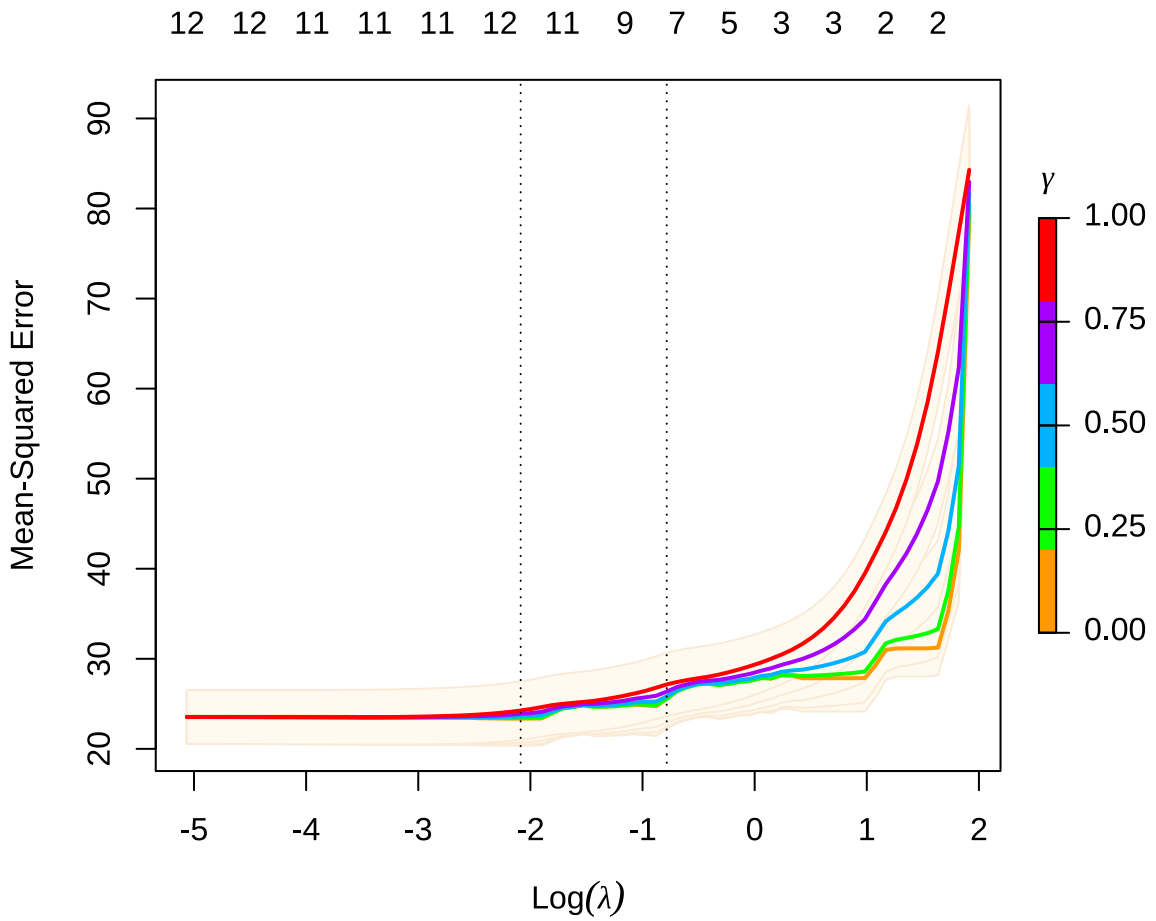


图 36.9: 回归系数的迭代路径

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
```

36.2.6 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)
```

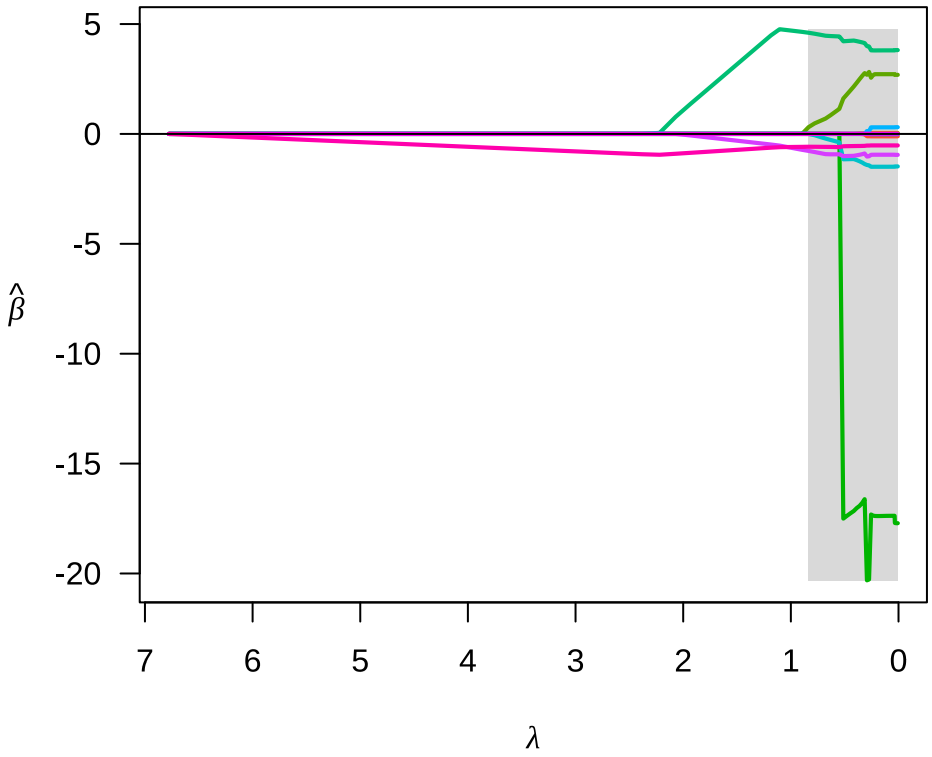


图 36.10: 回归系数的迭代路径

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
#> 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        *
```

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)
```

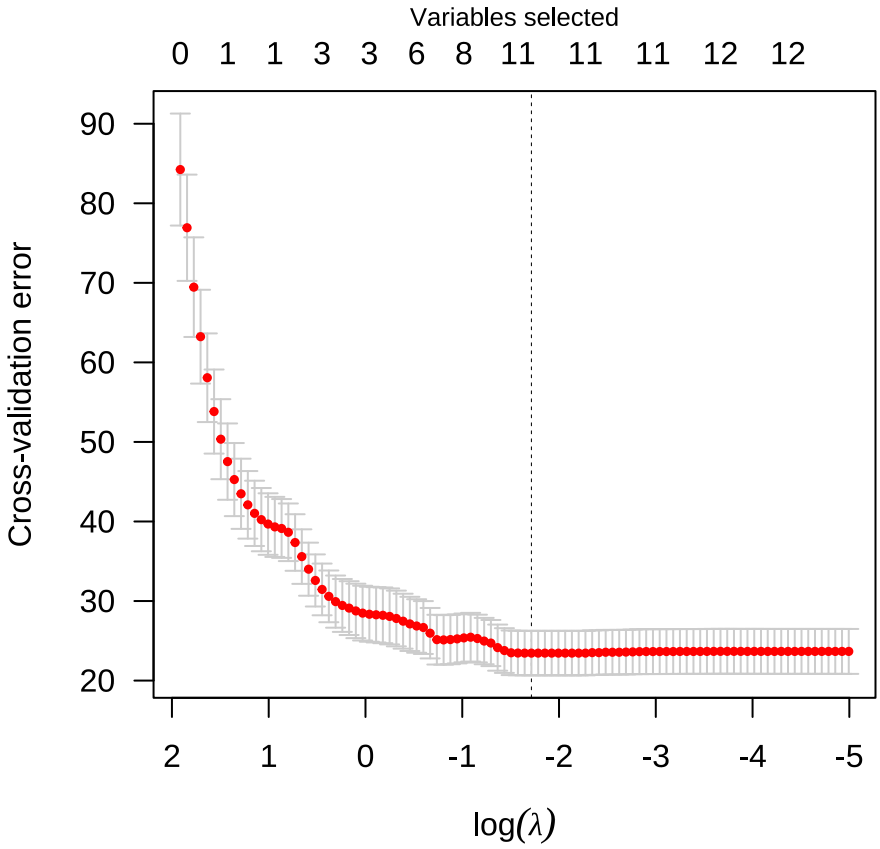


图 36.11: 惩罚系数的迭代路径

36.2.7 SCAD

```
fit_scad <- ncvreg(X = Boston[, -14], y = Boston[, "medv"], penalty = "SCAD")
```

```
plot(fit_scad)
```

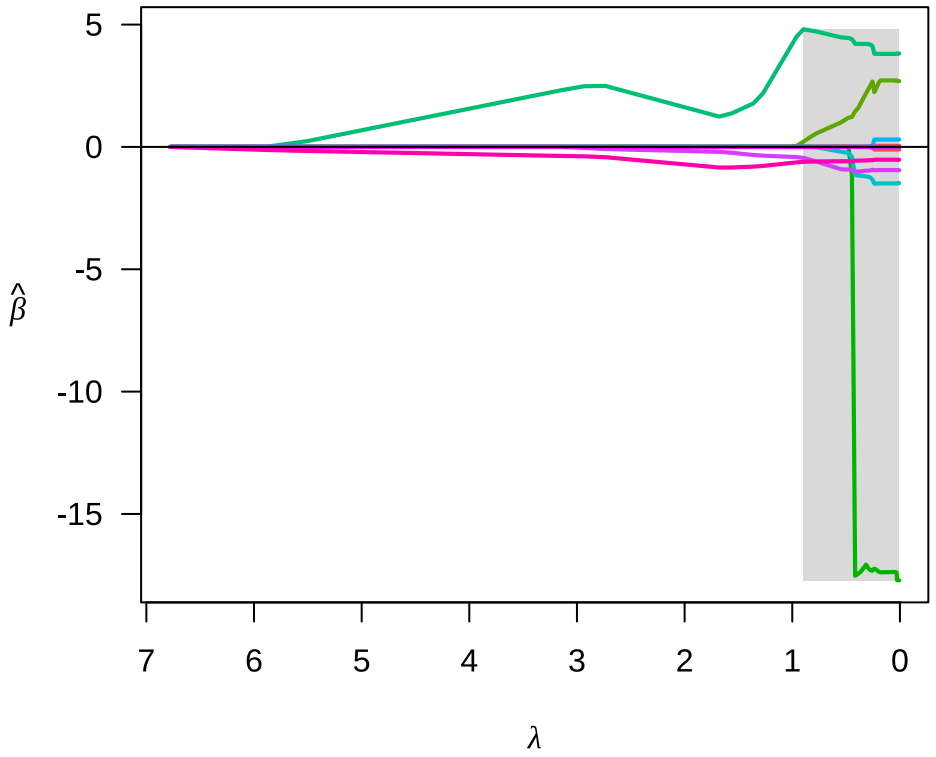


图 36.12: 回归系数的迭代路径

```
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)
```

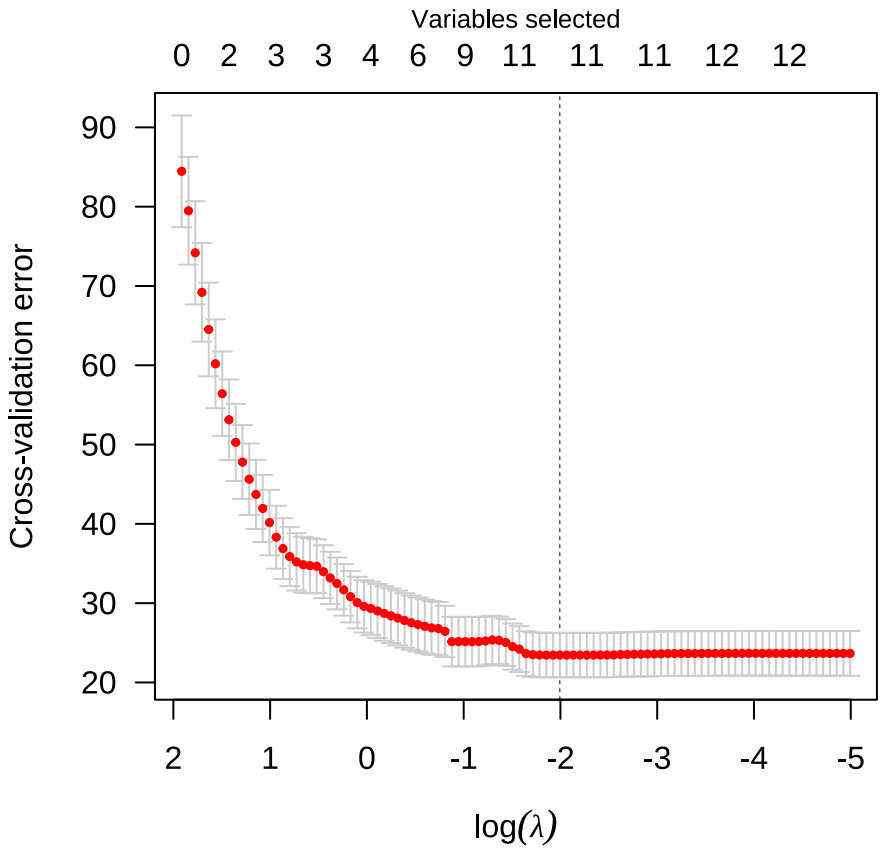


图 36.13: 惩罚系数的迭代路径

36.2.8 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)。

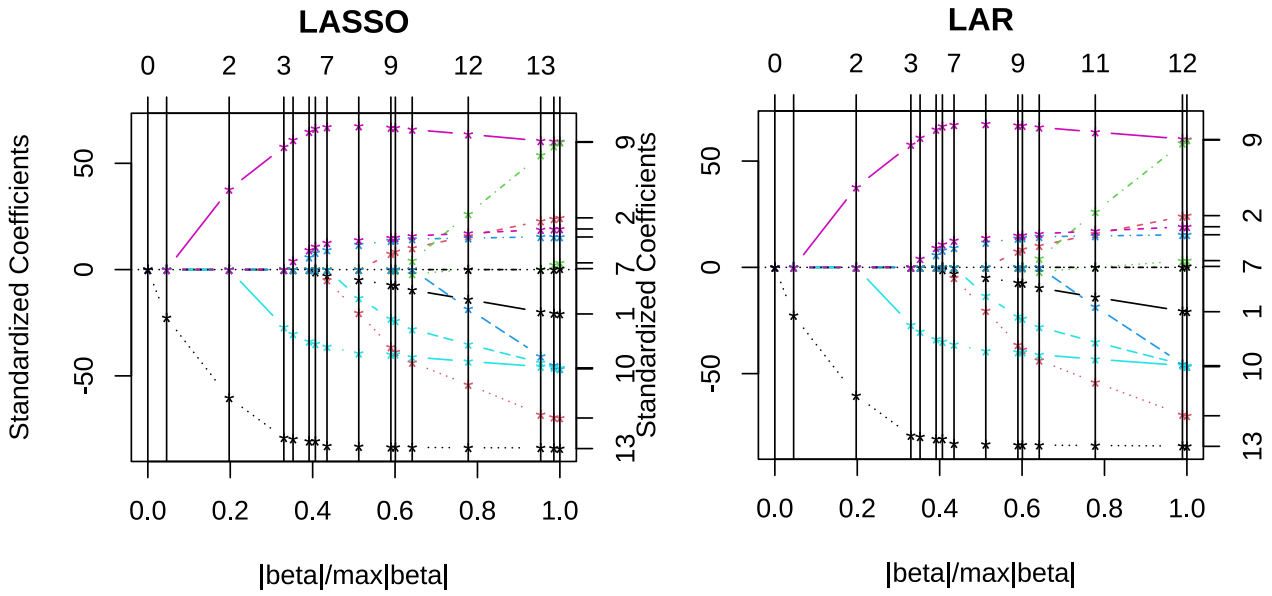
```
library(lars)
# Lasso 回归
fit_lars_lasso <- lars(
  x = as.matrix(Boston[, -14]), y = as.matrix(Boston[, "medv"]),
  type = "lasso", trace = FALSE, normalize = TRUE, intercept = TRUE
)
# LAR 回归
fit_lars_lar <- lars(
  x = as.matrix(Boston[, -14]), y = as.matrix(Boston[, "medv"]),
```

```
  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

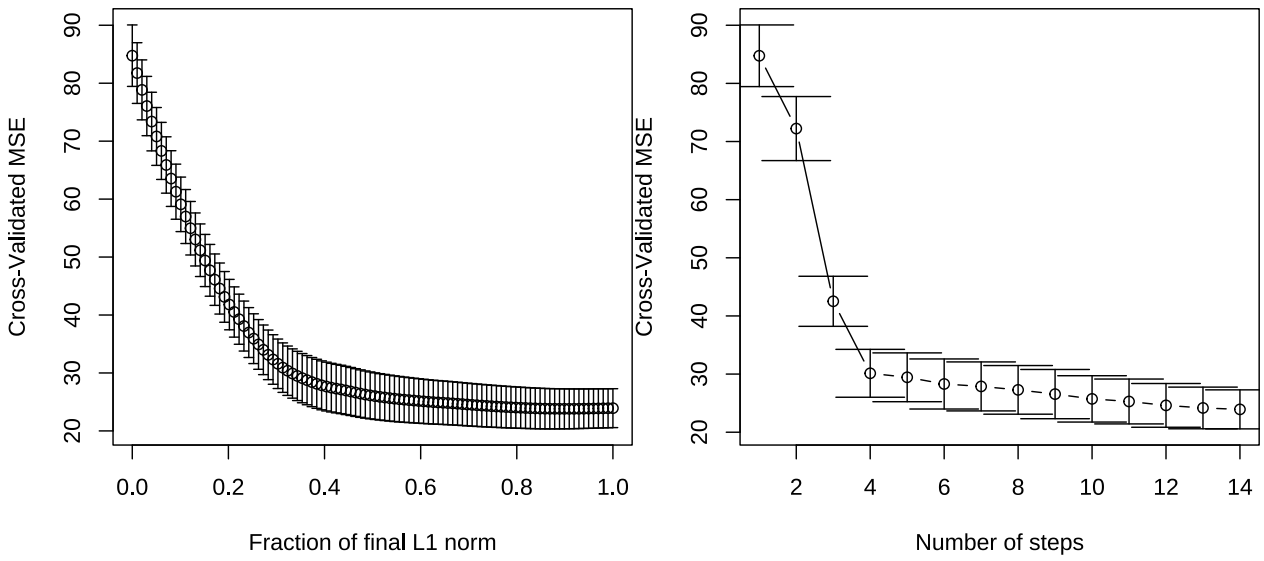
(b) 最小角回归

图 36.14: 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
```

)



(a) Lasso 回归 (b) 最小角回归

图 36.15: 交叉验证均方误差的变化

36.2.9 最优子集回归

$$\mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2 + \lambda\|\boldsymbol{\beta}\|_0$$

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图。

```
plot(fit_abess, label = TRUE, main = "惩罚系数的迭代路径")
```

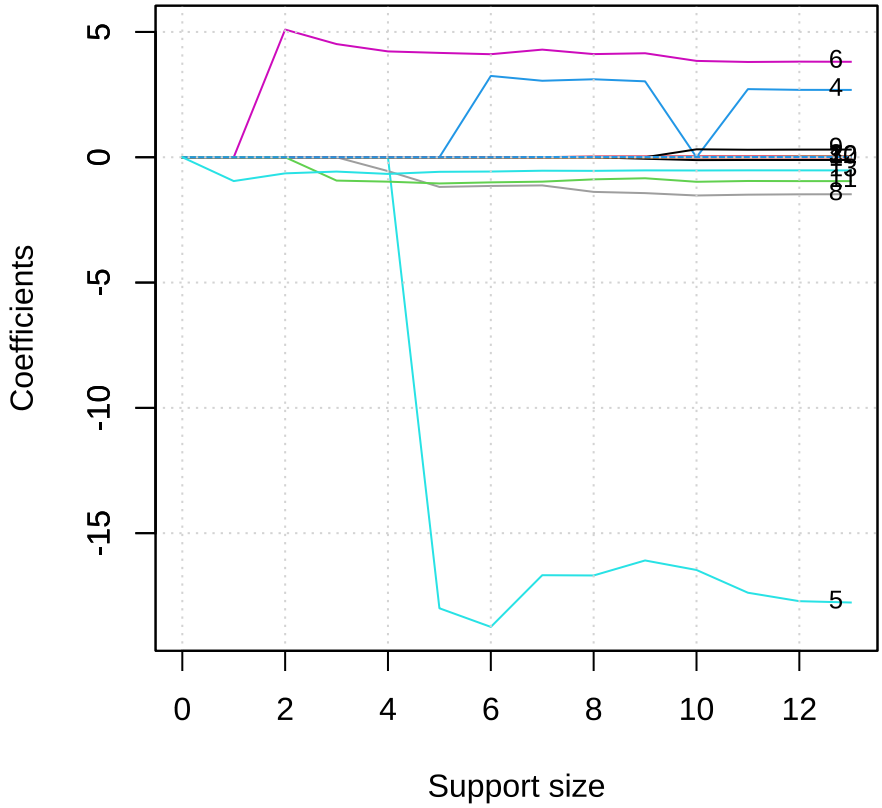


图 36.16: 惩罚系数的迭代路径

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, type = "tune", main = "交叉验证筛选变量个数")
```

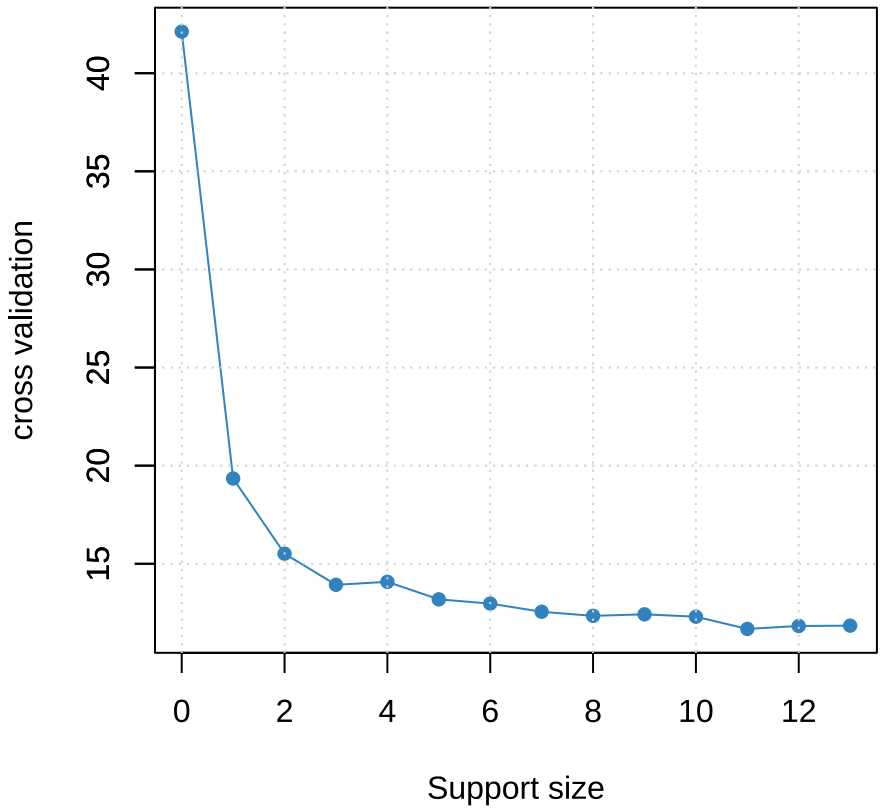


图 36.17: 交叉验证筛选变量个数

从上图可以看出，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
#>   .. .. ..$ : 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 13
```

36.3 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
#>
#> Gaussian Radial Basis kernel function.
#>  Hyperparameter : sigma =  0.10736655837088
#>
#> Number of Support Vectors : 339
#>
#> Objective Function Value : -79.4033
#> Training error : 0.095675
```

```
# 预测
pred_medv_svm <- predict(fit_ksvm, newdata = Boston)
# RMSE
rmse(Boston$medv, pred_medv_svm)
```

```
#> [1] 2.844798
```

36.4 神经网络

单隐藏层的神经网络

```
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] 3.363021
```

36.5 决策树

```
library(rpart)
fit_rpart <- rpart(medv ~ .,
  data = Boston, control = rpart.control(minsplit = 5)
)

pred_medv_rpart <- predict(fit_rpart, newdata = Boston[, -14])

rmse(Boston$medv, pred_medv_rpart)
```

```
#> [1] 3.565888
```

```
library(rpart.plot)
rpart.plot(fit_rpart)
```

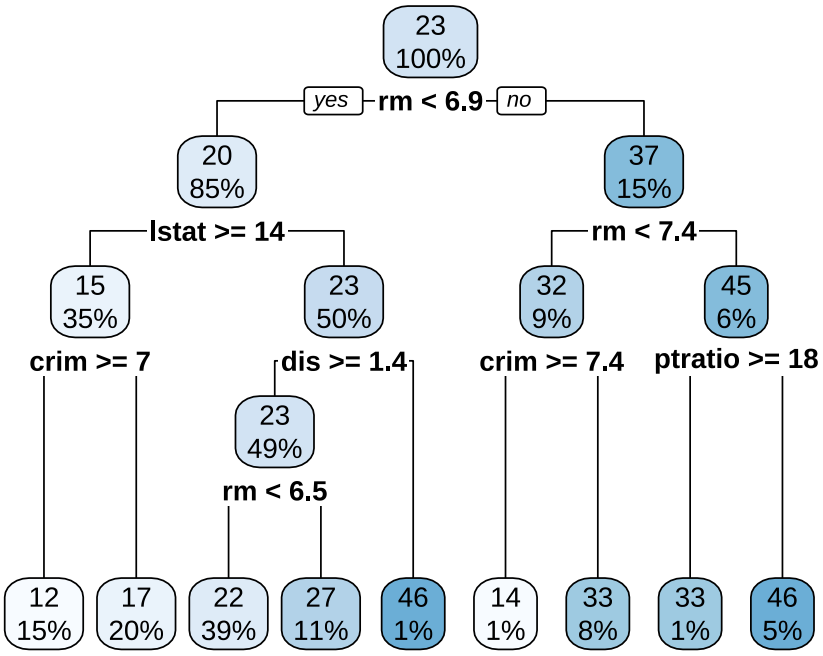


图 36.18: 分类回归树

36.6 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: 9.858359
#>                     % Var explained: 88.32
```

```
pred_medv_rf <- predict(fit_rf, newdata = Boston[, -14])
rmse(Boston$medv, pred_medv_rf)
```

```
#> [1] 1.380899
```

36.7 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
)
```

```
#> [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
```

参考文献

Agresti, Alan. 2007. *An Introduction to Categorical Data Analysis*. 2nd 本. Hoboken, New Jersey: John Wiley & Sons, Inc.

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.

Berkelaar, Michel 等. 2023. *lpSolve: Interface to Lp_solve v. 5.5 to Solve Linear/Integer Programs*. https://CRAN.R-project.org/package=lpSolve.

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.

Bion, Ricardo. 2018. *ggtech: ggplot2 tech themes and scales*.

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.

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.

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.

Christensen, O. F., 和 P. J. Ribeiro Jr. 2002. 《geoRglm: A package for generalised linear spatial models》. *R News* 2 (2): 26–28.

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.

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.

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.

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.

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.

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.

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.

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.

Gross, Calli, 和 Philipp Ottolinger. 2016. *ggThemeAssist: Add-in to Customize ggplot2 Themes*.

https://CRAN.R-project.org/package=ggThemeAssist.

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.

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.

Johnson, Steven G. 2023. *The NLopt nonlinear optimization package*. https://CRAN.R-project.org/package=nloptr.

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.

Kassambara, Alboukadel. 2022. *ggpubr: ggplot2 Based Publication Ready Plots*. https://CRAN.R-project.org/package=ggpubr.

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.

Kothari, Aditya. 2022. *ggTimeSeries: Time Series Visualisations Using the Grammar of Graphics*. https://CRAN.R-project.org/package=ggTimeSeries.

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.

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, 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.

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.

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. 2022. *ggdensity: Interpretable Bivariate Density Visualization with ggplot2*. https://CRAN.R-project.org/package=ggdensity.

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 Colour-Maps*. https://CRAN.R-project.org/package=scico.

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.

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.

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.

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.

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.

"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.

Theussl, Stefan, 和 Kurt Hornik. 2023. *Rglpk: R/GNU Linear Programming Kit Interface*. https://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*.

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/.

Warnes, J. J., 和 B. D. Ripley. 1987. 《Problems with likelihood estimation of covariance functions of spatial gaussian processes》. *Biometrika* 74 (3). 640-642.

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*. Springer-Verlag New York.

https://ggplot2.tidyverse.org.

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, 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.

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 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.

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

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。

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
```

[1] https://stat.ethz.ch/pipermail/r-help/2006-March/101596.html

```
[2,]  1.2  3.0
```

```
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
```

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
```

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
[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
```

```
         [,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
```

```
     [,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,] -6.661338e-16  3.330669e-16
[2,]  3.837386e-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

$$\mathbf{H}_u\mathbf{y} = \mathbf{y} - 2\mathbf{u}\mathbf{u}^\top\mathbf{y}$$

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 (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
[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
```

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,] 1.000000e+00 8.052102e-18
[2,] 8.052102e-18 1.000000e+00
```

```
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
[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
```

```
     [,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 -3.613656e-16
[2,] 2.145922e-16  4.379501e-01
```

```
# I = VV'
res$v %*% t(res$v)
```

```
              [,1]          [,2]
[1,]  1.000000e+00 -1.647255e-17
[2,] -1.647255e-17  1.000000e+00
```

```
# I = UU'
res$u %*% t(res$u)
```

```
              [,1]          [,2]
[1,]  1.000000e+00 -8.246538e-17
[2,] -8.246538e-17  1.000000e+00
```

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。

$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}$$
$$\boldsymbol{\epsilon} \sim \mathrm{MVN}(\mathbf{0}, \sigma^2 I)$$

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 $\hat{\boldsymbol{\beta}}, \hat{\sigma^2}$ 。

$$\hat{\boldsymbol{\beta}} = (X^\top X)^{-1} X^\top \boldsymbol{y}$$
$$\hat{\sigma^2} = \frac{\boldsymbol{y}^\top (I - X(X^\top X)^{-1} X^\top)\boldsymbol{y}}{n - \mathrm{rank}(X)}$$

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。

$$
\begin{aligned}
\hat{\boldsymbol{y}} &= X(X^\top X)^{-1}X^\top \boldsymbol{y} \\
&= X\big(L^\top L\big)^{-1}X^\top \boldsymbol{y} \\
&= XL^{-1}(L^\top)^{-1}X^\top \boldsymbol{y}
\end{aligned}
$$

```
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。

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。

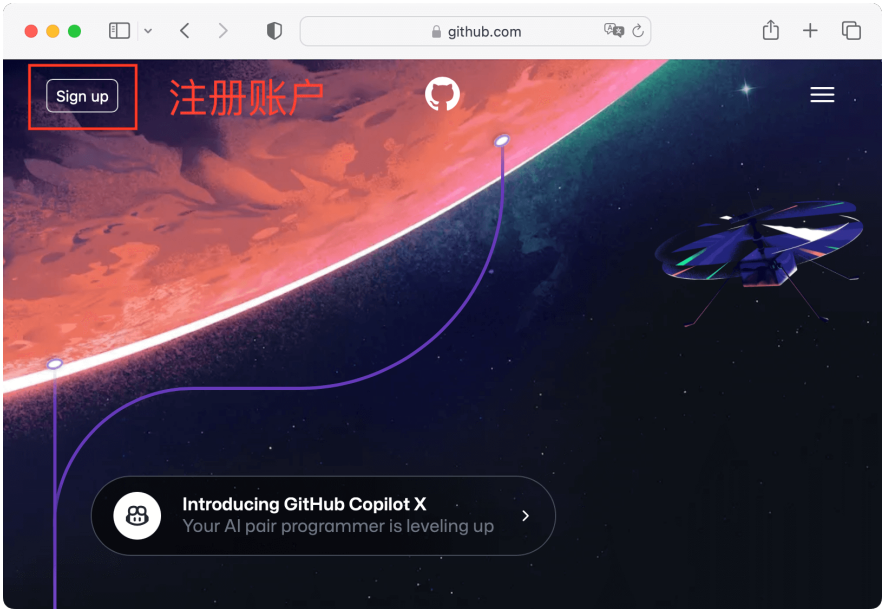


图 C.1: 点击注册

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。

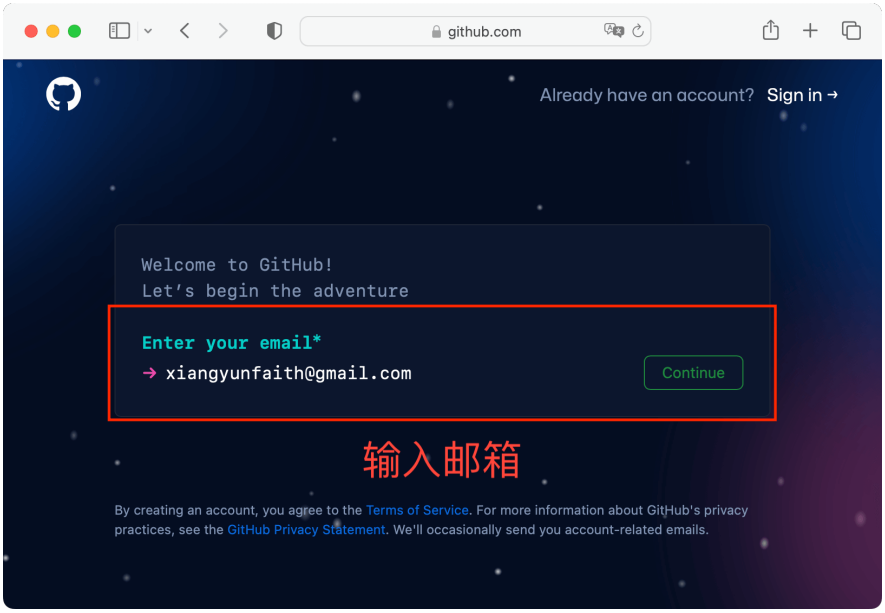


图 C.2: 输入邮箱

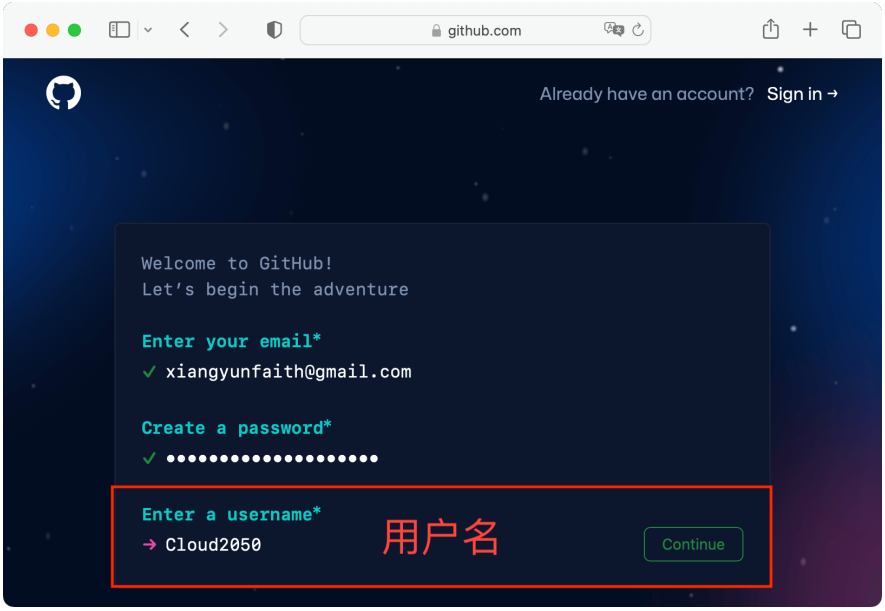


图 C.3: 输入用户名

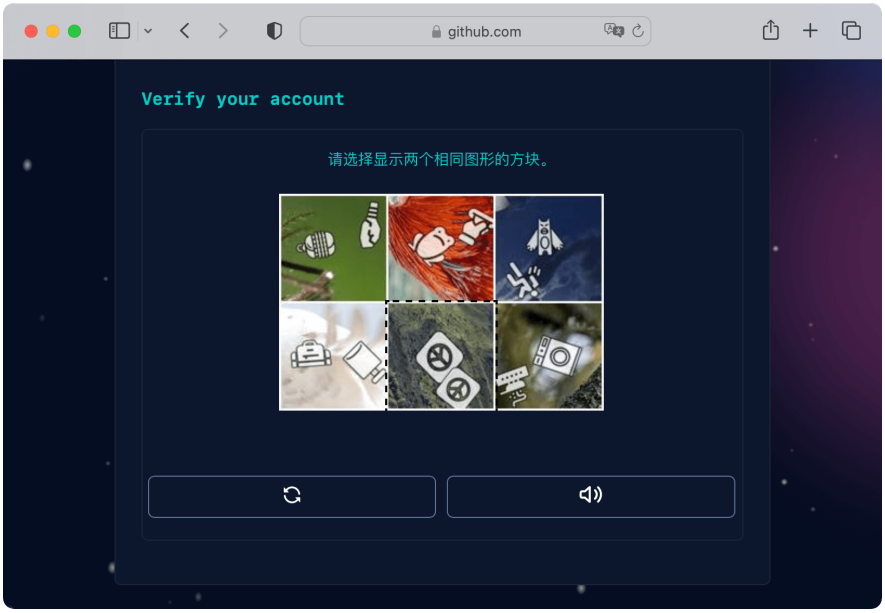


图 C.4: 回答问题

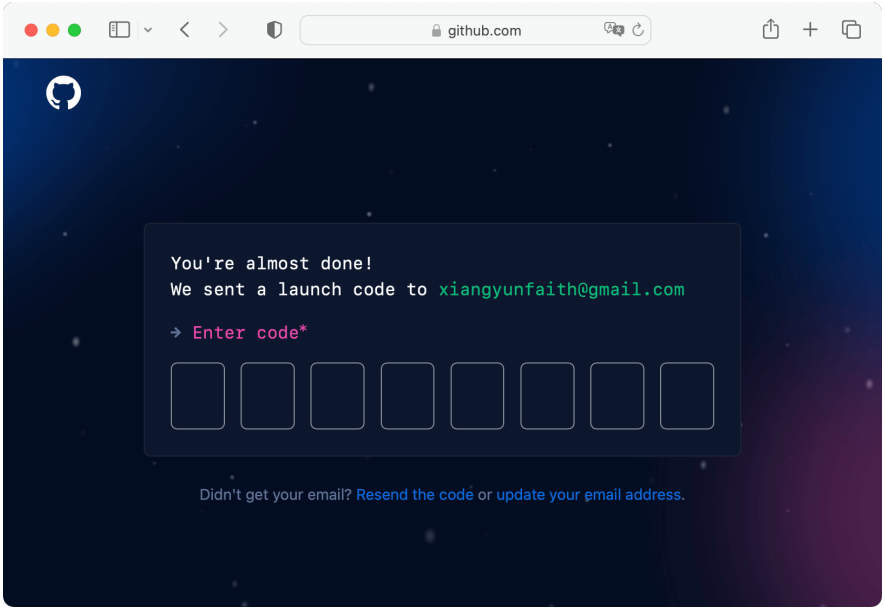


图 C.5: 输入验证码

将收到的验证码输入进去，完成账户验证。

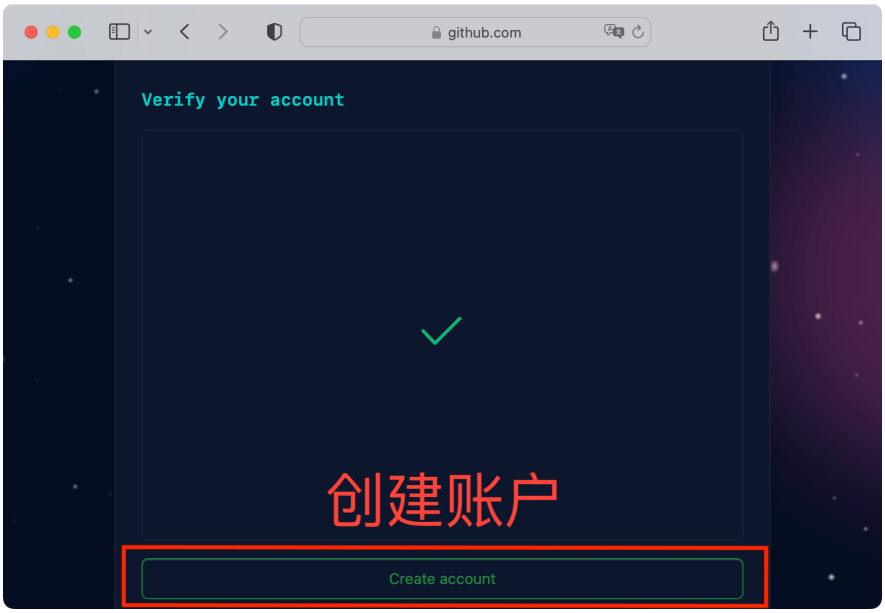


图 C.6: 验证账户

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"
```

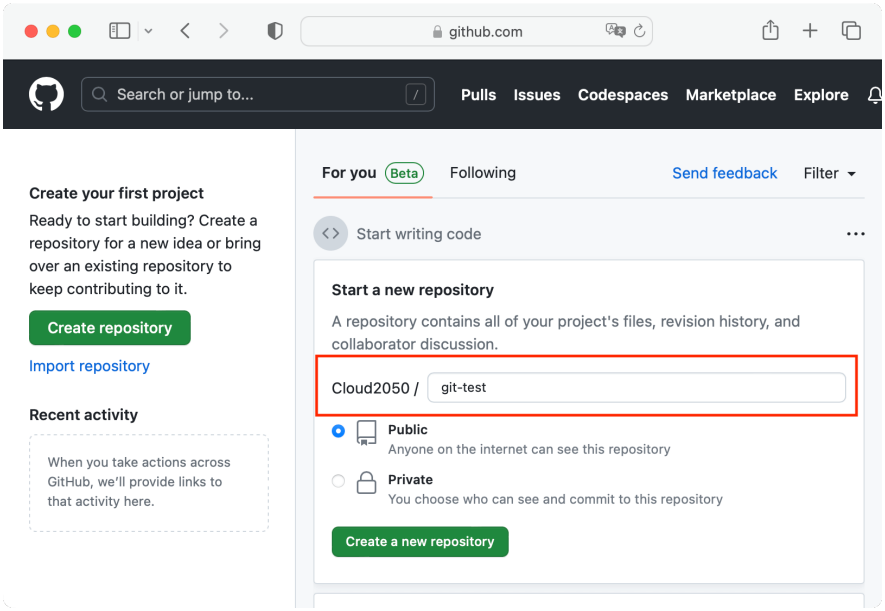


图 C.7: 创建代码仓库

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"
```

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。

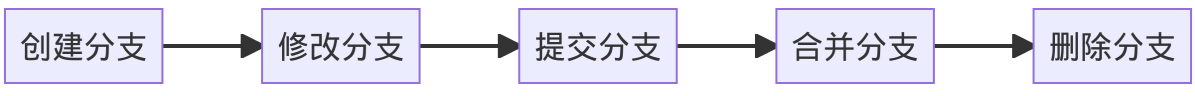


图 C.8: Git 分支操作

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
# 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

```
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
```

索引