R 语言数据分析实战

黄湘云

2024 年 7 月 6 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** (Wickham 2016) 和 **lattice** (Sarkar 2008) 等众多 R 包。考虑到要同时支持 DOCX、EPUB、PDF 和 HTML 四种书籍格式，书中使用 **knitr** 包和 **gt** 包制作静态的表格。

为方便测试贡献者提供的 PR，本书托管在 Github 上，同时启用 Github Action 服务，为书籍自定义了一个可复现全书内容的运行环境，包括 R 软件、扩展包和系统软件依赖，详见仓库中的 DESCRIPTION 文件。你现在看到的是在线编译版本，使用 Quarto 1.4.550，最新一次编译时间是 2024-07-06 16:12:30。

```
xfun::session_info(packages = c(
  "ggplot2", "gganimate", "ggrepel", "ggridges",
  "patchwork", "shiny", "plotly", "igraph",
  "tidygraph", "ggraph", "dplyr", "purrr", "tidyr", "httr",
  "data.table", "rsconnect", "knitr", "rmarkdown", "gt", "DT",
  "showtext", "gifski", "tinytex", "magick"
), dependencies = FALSE)
```

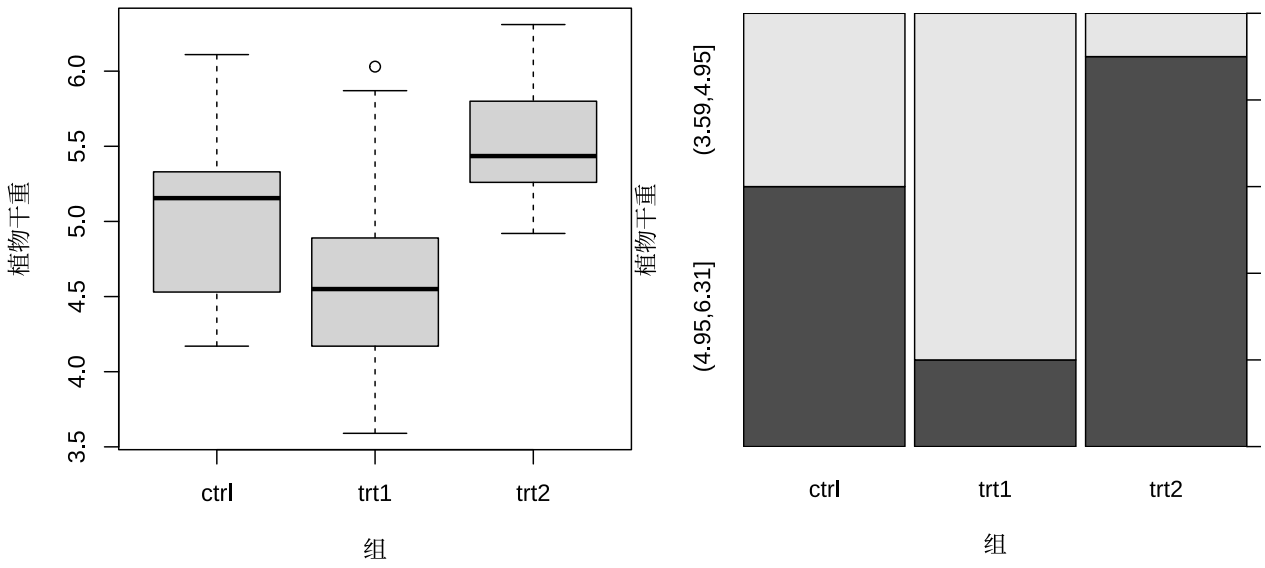
```
#> R version 4.4.0 (2024-04-24)
#> Platform: aarch64-apple-darwin20
#> Running under: macOS Sonoma 14.5
```

```
#>
#> Locale: en_US.UTF-8 / en_US.UTF-8 / en_US.UTF-8 / C / en_US.UTF-8 / en_US.UTF-8
#>
#> Package version:
#>   data.table_1.15.4 dplyr_1.1.4       DT_0.33           gganimate_1.0.9
#>   ggplot2_3.5.1     ggraph_2.2.1      ggrepel_0.9.5     ggridges_0.5.6
#>   gifski_1.12.0.2   gt_0.10.1         httr_1.4.7        igraph_2.0.3
#>   knitr_1.47        magick_2.8.3      patchwork_1.2.0   plotly_4.10.4
#>   purrr_1.0.2       rmarkdown_2.27    rsconnect_1.3.1   shiny_1.8.1.1
#>   showtext_0.9.7    tidygraph_1.3.1   tidyr_1.3.1       tinytex_0.51
#>
#> Pandoc version: 3.1.11
#>
#> LaTeX version used:
#>   TeX Live 2024 (TinyTeX) with tlmgr 2024-05-24
```

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 12.7 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》(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. 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。
4. 统计分析：统计检验、相关分析、分类数据、功效分析
5. 数据建模：网络数据、文本数据、时序数据
6. 优化建模：统计计算、数值优化、优化问题

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。

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 https://www.gamlss.com/ 试图将一系列统计模型，如线性模型 LM、广义线性模型 GLM、广义可加模型 GAM、线性混合效应模型 LMM、广义线性混合效应模型 GLMM、广义可加混合效应模型 GAMM 纳入统一的广义可加模型框架下 (Rigby 和 Stasinopoulos 2005)。2009 年 Håvard Rue 开发**INLA** 包，类似 **gamlss** 包，同样是整合一系列统计模型，纳入一个新的基于贝叶斯视角的集成嵌套拉普拉斯框架下，主要应用于空间统计领域，更多详见 https://www.r-inla.org/。

数据探索和分析

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

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

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

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

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

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

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

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

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

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

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

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

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

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

表格 2: anscombe 数据集

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

5	5.68	5	4.74	5	5.73	8	6.89

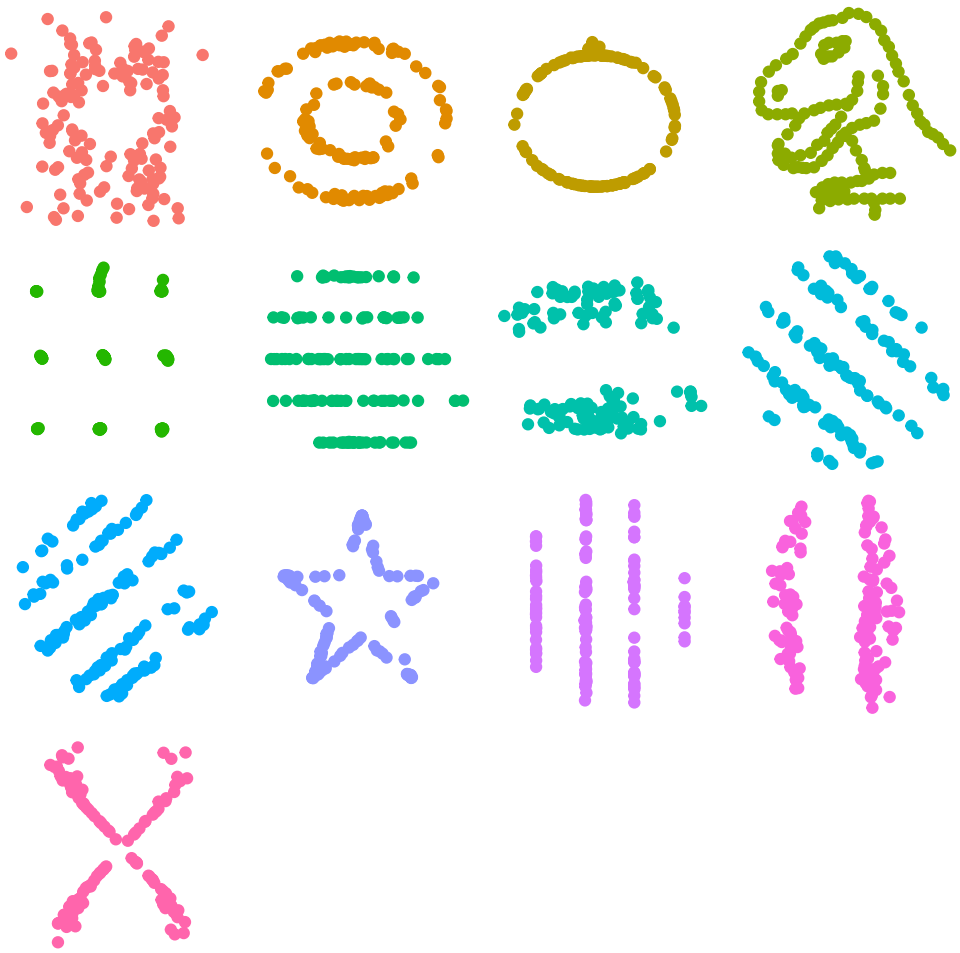
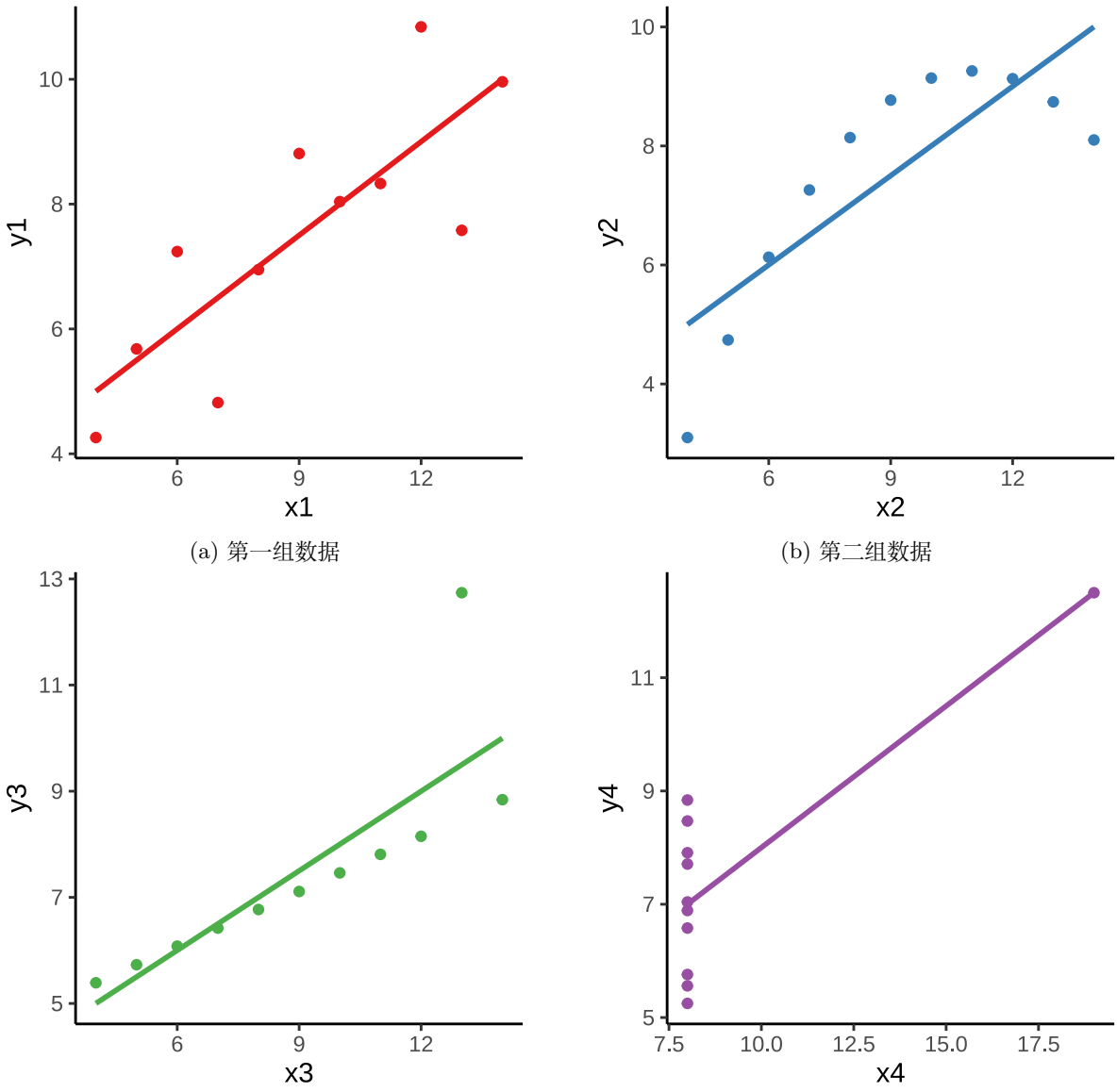


图 2: 数据可视化为何如此重要

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 3 分别描述各组数据的关系时，却发现四组数据之间有极大的差异，且只有第一组数据看起来符合线性模型的条件 (Anscombe 1973)。



(a) 第一组数据

(b) 第二组数据

(c) 第三组数据

(d) 第四组数据

图 3: 数据可视化为何如此重要

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

数据展示和交流

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

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

章节 六 首先介绍 **plotly** 包绘图的基础语法以及与 **ggplot2** 包绘图的关系，其次介绍制作常用的交互图形，如条形图、直方图、箱线图、曲线图等，最后介绍一些常用的技巧，如导出静态图片、添加水印徽标等。

章节 七 首先介绍 **DT** 包制作交互表格的基础语法，其次介绍常用的功能，如列分层分组、按列配色、列格式化、搜索排序、数据导出等，最后介绍一些基础的 CSS 和 JavaScript 知识，支持一些中高级的表格定制功能。

章节 八 首先介绍 **shiny** 包制作交互应用的整体概览，如前端布局、后端计算、筛选器、模块交互等，其次从易到难介绍一个完整的数据应用，最后介绍生产级的 Shiny 应用开发的技术栈。

章节 九 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

第一章　数据对象

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

1.1　数据类型

1.1.1　整型

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

```
[1] 1 2
```

1.1.2　逻辑型

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

```
[1]  TRUE FALSE
```

1.1.3　字符型

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

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

1.1.4　日期型

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

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

1.1.5 数值型

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

```
[1] 1.0 1.2
```

1.2 数据结构

1.2.1 向量

所有元素都是同一类型

1.2.2 矩阵

所有元素都是同一类型

1.2.3 数组

所有元素都是同一类型

1.2.4 列表

元素可以属于不同类型

1.2.5 因子

1.2.6 数据框

同列的元素类型必须一致，不同列的元素类型可以不同。

1.2.7 ts

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

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

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

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

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

```
# 数据类型
class(x)
```

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

```
# 存储类型
mode(x)
```

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

```
# 数据结构
str(x)
```

```
 Time-Series [1:100] from 2017 to 2017: 0.431 0.822 1.149 -2.479 0.544 ...
```

函数 start() 和 end() 查看开始和结束的时间点。

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

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

函数 time() 可以查看在以上时间区间的划分。

```
time(x)
```

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

函数 tsp() 可以查看其期初、期末和周期。

```
tsp(x)
```

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

第二章　数据获取

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

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

2.1　从本地文件读取

利用 Base R 提供的基础函数从各类文件导入数据

2.1.1　csv 文件

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

2.1.2　xlsx 文件

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

openxlsx 读/写 xlsx 文件

2.1.3 arrow 文件

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

2.2 从数据库中导入

从各类数据库导入数据，比如 RSQLite 等

2.2.1 RSQLite

2.2.2 odbc

2.2.3 RJDBC

很多数据库都有 Java 接口驱动

2.3 从各类网页中抓取

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

2.3.1 豆瓣排行榜

2.3.2 链家二手房

2.4 从数据接口中获取

2.4.1 Github

从 Github API 接口中获取托管在 Github 上的 R 包的信息，比如点赞、关注和转发的数量。首先从 CRAN 上获得 R 包元数据信息，接着筛选出托管在 Github 上的 R 包，清理出 R 包在 Github 上的网址。

```
pdb <- readRDS(file = "data/cran-package-db-20231231.rds")
# 过滤出 Github
pdb <- subset(
  x = pdb, subset = !duplicated(Package) & grepl(pattern = "github", x = BugReports),
  select = c("Package", "Maintainer", "Title", "BugReports")
```

```
)
# 掐头去尾
pdb$repo <- sub(x =  pdb$BugReports, pattern = "(http|https)://(www\\.){0,1}github\\.com/", replace
pdb$repo <- sub(x = pdb$repo, pattern = "/{1,}(issues|blob).*", replacement = "")
pdb$repo <- sub(x = pdb$repo, pattern = "/{1,}(discussions|wiki)", replacement = "")
pdb$repo <- sub(x = pdb$repo, pattern = "/$", replacement = "")
```

获取某代码仓库信息的 Github API 是 https://api.github.com/repos ，为了批量地访问 API ，收集想要的数据，将数据请求、结果整理的过程打包成一个函数。

```
github_stats <- function(repo) {
  url <- paste("https://api.github.com/repos", repo, sep = "/")
  # 最多允许失败 5 次，每失败一次休息 5s
  req <- xfun::retry(curl::curl_fetch_memory, url = url, .times = 5, .pause = 5)
  x <- jsonlite::fromJSON(rawToChar(req$content))
  # 爬失败的标记一下
  if(is.null(x$stargazers_count)) x$stargazers_count <- x$subscribers_count <- x$forks_count <- -1
  # 爬一个休息 1s
  Sys.sleep(1)
  data.frame(
    repo = repo,
    # 点赞 仓库上 star 的人数
    stargazers_count = x$stargazers_count,
    # 关注 仓库上 watch 的人数
    subscribers_count = x$subscribers_count,
    # 转发 仓库上 fork 的人数
    forks_count = x$forks_count
  )
}
```

下面测试一下这段代码，获取代码仓库 yihui/knitr 的点赞、关注和转发的人数。

```
# 测试代码
github_stats(repo = "yihui/knitr")
```

```
         repo stargazers_count subscribers_count forks_count
1 yihui/knitr             2359               116         873
```

理论上，使用函数 `lapply()` 遍历所有 R 包可得所需数据，将数据收集函数应用到每一个 R 包上再合并结果，即如下操作。

```
# 合并数据
gh_repo_db <- data.table::rbindlist(lapply(pdb$repo, github_stats))
```

实际上，在没有访问令牌的情况下，Github API 的访问次数是有限制的，只有 60 次（一段时间内）。首

先在 Github 开发者设置中申请一个应用，获得应用名称 (appname)、客户端 ID (key) 和密钥 (secret)，下面借助 **httr** 包配置 OAuth 凭证。

```
library(httr)
# Github API Oauth2
oauth_endpoints("github")
# 应用名称 (appname) 、客户端 ID (key) 和密钥 (secret)
myapp <- oauth_app(
  appname = "Application Name", key = "Client ID",
  secret = "Client Secrets"
)
# 获取 OAuth 凭证
github_token <- oauth2.0_token(oauth_endpoints("github"), myapp)
# 使用 API
gtoken <- config(token = github_token)
```

修改函数 `github_stats()` 中请求 Github API 的一行代码，发送带密钥的 GET 请求。

```
req <- xfun::retry(GET, url = url, config = gtoken, .times = 5, .pause = 5)
```

此外，请求难免出现意外，按照上面的方式，一旦报错，数据都将丢失。因此，要预先准备存储空间，每获取一条数据就存进去，如果报错了，就打个标记。

```
# 准备存储数据
gh_repo_db <- data.frame(
  repo = pdb$repo, stargazers_count = rep(-1, length(pdb$repo)),
  subscribers_count = rep(-1, length(pdb$repo)),
  forks_count = rep(-1, length(pdb$repo))
)
# 不断更新数据
while (any(gh_repo_db$stargazers_count == -1)) {
  tmp <- gh_repo_db[gh_repo_db$stargazers_count == -1, ]
  for (repo in tmp$repo) {
    gh_repo_db[gh_repo_db$repo == repo, ] <- github_stats(repo = repo)
  }
  if(repo == tmp$repo[length(tmp$repo)]) break
}
```

最后，将收集整理好的数据保存到磁盘上，下面按点赞数量给 R 包排序，篇幅所限，仅展示前 20。

```
gh_repo_db <- readRDS(file = "data/gh-repo-db-2023.rds")
gh_repo_db <- gh_repo_db[!duplicated(gh_repo_db$repo),]
gh_repo_db <- gh_repo_db[order(gh_repo_db$stargazers_count, decreasing = T),]
head(gh_repo_db, 20)
```

	repo	stargazers_count	subscribers_count	forks_count
8434	dmlc/xgboost	25266	909	8707
5553	facebook/prophet	17415	425	4474
4307	mlflow/mlflow	16365	292	3793
3807	Microsoft/LightGBM	15821	437	3798
265	apache/arrow	13080	356	3220
3080	h2oai/h2o-3	6624	384	2016
2790	tidyverse/ggplot2	6210	308	2028
3430	interpretml/interpret	5894	141	706
6921	rstudio/shiny	5180	339	1818
4317	mlpack/mlpack	4668	185	1577
1754	tidyverse/dplyr	4612	246	2131
640	rstudio/bookdown	3565	122	1263
1430	Rdatatable/data.table	3437	170	977
6316	rstudio/rmarkdown	2758	146	977
2269	wesm/feather	2708	97	174
5324	plotly/plotly.R	2467	117	628
5084	thomasp85/patchwork	2344	49	159
1389	r-lib/devtools	2340	120	760
3658	yihui/knitr	2326	115	877
6868	satijalab/seurat	2034	75	867

将发布在 Github 上的受欢迎的 R 包列出来了，方便读者选用，也看到一些有意思的结果。

1. 机器学习相关的 R 包靠在最前面，实际上，它们（占十之七八）多是对应软件的 R 语言接口，点赞的数目应当算上其它语言接口的贡献。
2. 在机器学习之后，依次是数据可视化（ggplot2、shiny、plotly.R、patchwork）、数据操作（dplyr、data.table、feather）和可重复性计算（bookdown、rmarkdown、knitr）、R 包开发（devtools）和生物信息（seurat）。

最后，简要说明数据的情况：以上观察结果是基于 CRAN 在 2023-12-31 发布的 R 包元数据，8475 个 R 包在 Github 托管源代码，这些 R 包的点赞、关注和转发数据是在 2024-01-30 爬取的。其中，共有 29 个 R 包不按规矩填写、改名字、换地方、甚至删库了，这些 R 包是可忽略的。当然，也存在一些 R 包并未托管在 Github 上，但质量不错，比如 glmnet 包、colorspace 包、fGarch 包等，应当是少量的。

2.4.2 中国地震台网

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

2.4.3 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。

2.4.4 美国人口调查局

美国人口调查局

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

2.4.5 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

第三章　数据清洗

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

3.1　正则表达式

3.1.1　量词

3.1.2　级联

3.1.3　断言

正向查找 / 反向查找

3.1.4　反向引用

3.1.5　命名捕捉

3.2　字符串操作

3.2.1　查找

`grep()` / `grepl()` 返回是否匹配的结果

3.2.2　替换

`sub()` / `gsub()` 替换一次和多次

3.2.3 提取

`regexpr()` / `gregexpr()`

`regexec()` / `gregexec()`

第四章　数据操作

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

4.1　操作工具

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

4.1.1　Base R

在 data.frame 的基础上，提供一系列辅助函数实现各类数据操作。

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

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

4.1.2　data.table

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

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

```
      Species   cnt
       <fctr> <int>
1:     setosa    50
2: versicolor    50
3:  virginica    50
```

4.1.3 dplyr

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

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

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

4.1.4 SQL

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

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

按 Species 分组统计数据条数，SQL 查询语句如下：

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

SQL 代码执行的结果如下：

```
iris_preview
```

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

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

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

```
# Source:   SQL [3 x 2]
# Database: sqlite 3.46.0 [/Users/runner/work/_temp/Library/RSQLite/db/datasets.sqlite]
# Groups:   Species
  Species        n
  <chr>      <int>
1 setosa        50
2 versicolor    50
3 virginica     50
```

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

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

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

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

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

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

用完后，关闭连接通道。

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

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

4.2 Base R 操作

介绍最核心的 Base R 数据操作，如筛选、排序、变换、聚合、重塑等

4.2.1 筛选

筛选操作可以用函数 subset() 或 [实现

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

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

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

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

4.2.2 变换

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

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

```
'data.frame':   150 obs. of  6 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

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

4.2.3 排序

排序操作可以用函数 order() 实现

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

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

4.2.4 聚合

聚合操作可以用函数 aggregate() 实现

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

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

4.2.5 合并

两个数据框的合并操作可以用函数 merge() 实现

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

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

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

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

```
3  D NA  4
```

```
# 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
```

4.2.6 重塑

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

```
# 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
```

4.3 data.table 操作

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

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

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

4.3.1 筛选

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

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

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

4.3.2 变换

变换操作可以用函数 :=

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

```
Classes 'data.table' and 'data.frame':  150 obs. of  6 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 ...
 $ Species_N   : int  1 1 1 1 1 1 1 1 1 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

4.3.3 排序

排序操作可以用函数 order()

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

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

4.3.4 聚合

聚合操作用函数 .() 和 by 组合

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

```
      Species  mean
       <fctr> <num>
1:     setosa 5.006
2: versicolor 5.936
3:  virginica 6.588
```

4.3.5 合并

合并操作也是用函数 merge() 来实现。

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

```
Key: <a2>
       a2    a1    b1
   <char> <num> <num>
1:      A     1     2
2:      B     2     3
3:      C     3    NA
```

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

```
Key: <a2>
       a2    a1    b1
   <char> <num> <num>
1:      A     1     2
2:      B     2     3
3:      D    NA     4
```

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

```
Key: <a2>
       a2    a1    b1
   <char> <num> <num>
1:      A     1     2
```

```
2:      B     2     3
```

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

```
Key: <a2>
       a2    a1    b1
   <char> <num> <num>
1:      A     1     2
2:      B     2     3
3:      C     3    NA
4:      D    NA     4
```

4.3.6 重塑

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

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

```
Key: <id>
      id     A     B
   <num> <num> <num>
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
       <fctr>       <num>       <num>
1:     setosa    2.639001   0.6904897
2: versicolor    3.539735   0.8650777
3:  virginica    3.906836   0.9015345
```

第五章　数据处理

5.1　缺失值处理

缺失是一种非常常见的数据问题。

5.1.1　查找

缺失值在数据框中的位置

5.1.2　汇总

缺失值的占比、分布情况，可视化获得缺失的结构 VIM

5.1.3　替换

替换数据框中的缺失值

5.1.4　插补

mice Multivariate Imputation by Chained Equations 缺失值插补

5.2　异常值处理

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

5.2.1 检测

5.2.2 识别

5.2.3 处理

5.3 离群值处理

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

5.3.1 检测

5.3.2 识别

5.3.3 处理

第二部分

数据交流

第六章　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。

6.1　基础元素

6.1.1　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 6.1 。

表格 6.1: **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 6.1 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 6.1 是一样的。

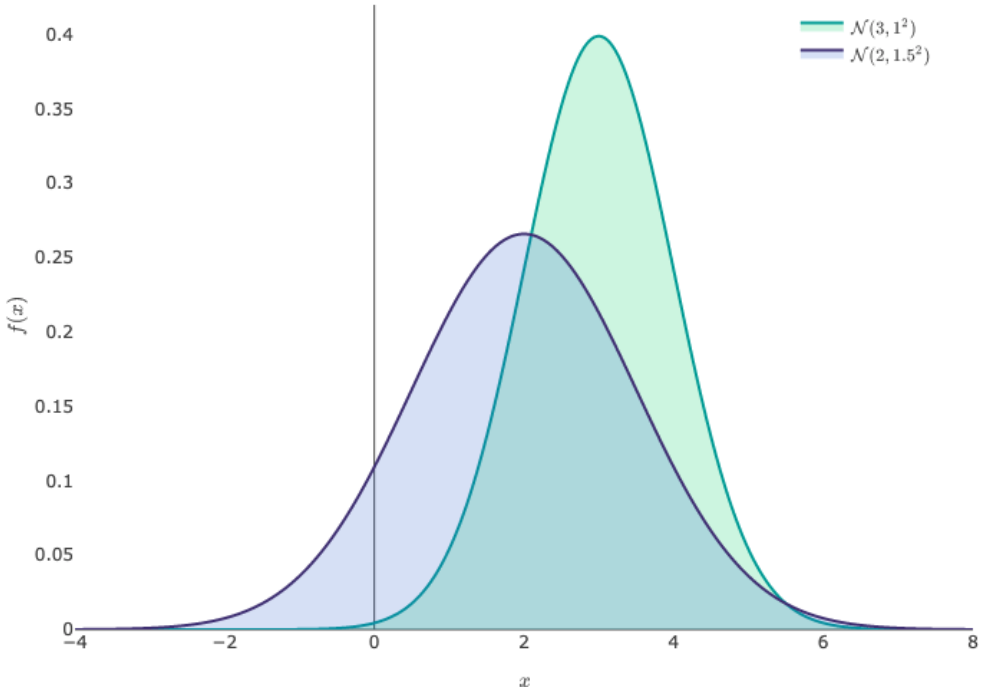


图 6.1: 默认风格的简单散点图

```
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。

6.1.2 配色

在图 6.1 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)
```

6.1.3 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')
  )
```

6.1.4 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"
  )
```

6.1.5 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"
  )
```

6.1.6 字体

6.1.7 图例

6.2 常用图形

6.2.1 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 6.2: 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 6.2 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")
  )
```

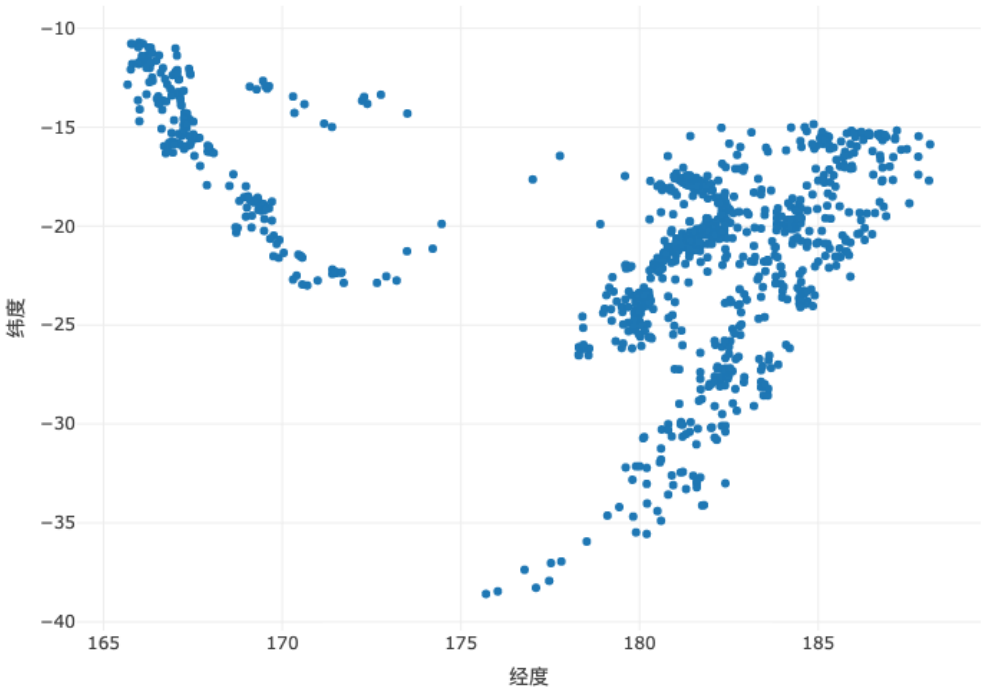


图 6.2: 普通散点图

6.2.2 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")
  )
```

6.2.3 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")
  )
```

6.2.4 直方图

地震次数随震级的分布变化，下图 6.3 为频数分布图

```
# https://plotly.com/r/reference/histogram/
plotly::plot_ly(quakes, x = ~mag, type = "histogram") |>
  plotly::layout(
    xaxis = list(title = "震级"),
    yaxis = list(title = "次数")
  )
```

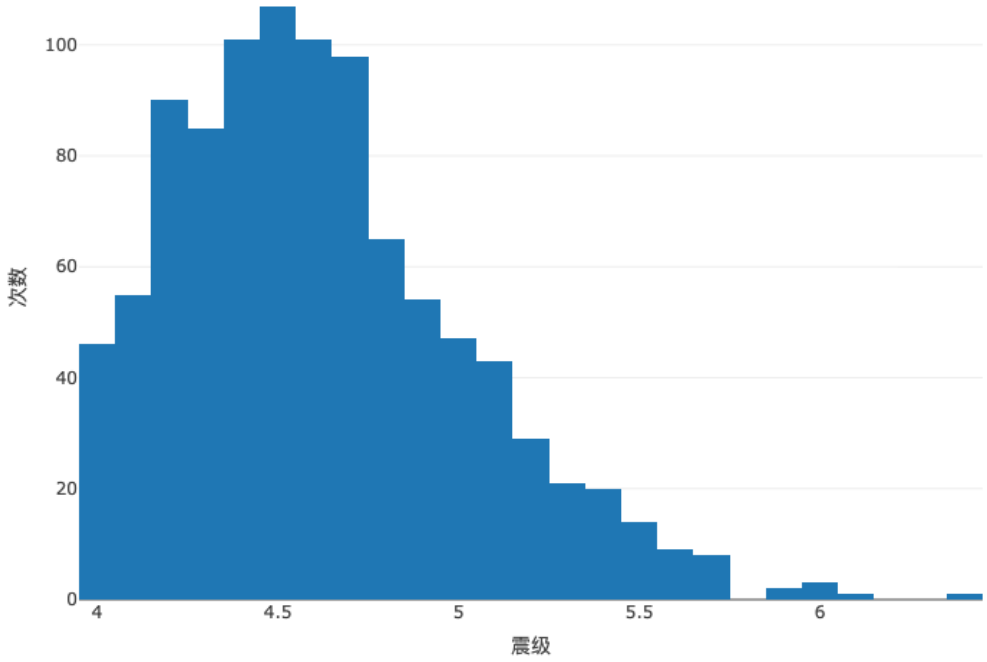


图 6.3: 地震震级的频数分布图

地震震级的概率分布，下图 6.4 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")
  )
```

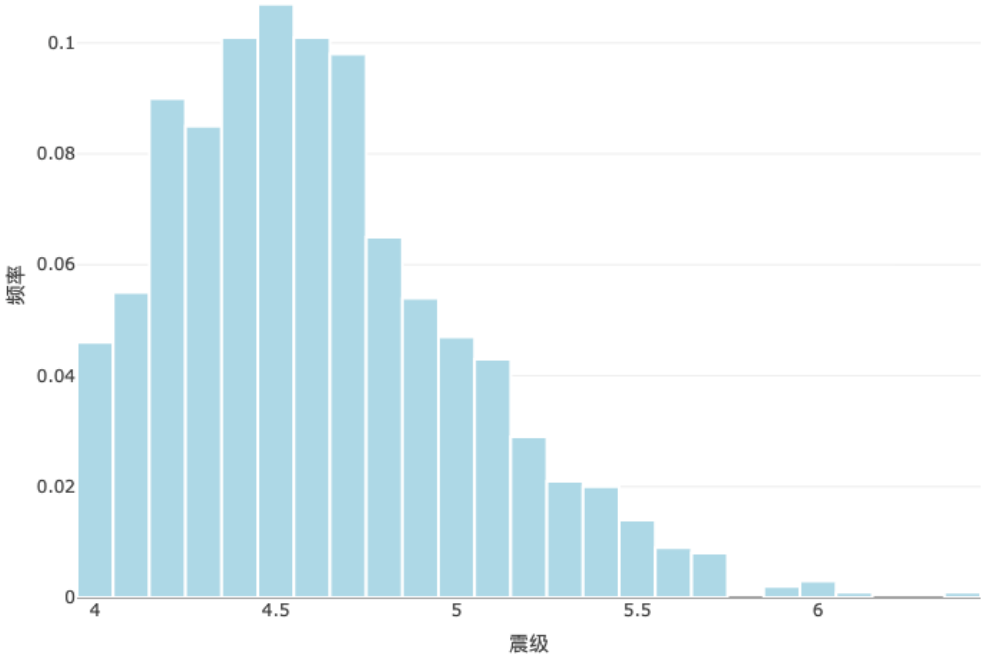


图 6.4: 地震震级的频率分布图

`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 6.5 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")
```

```
  )
```

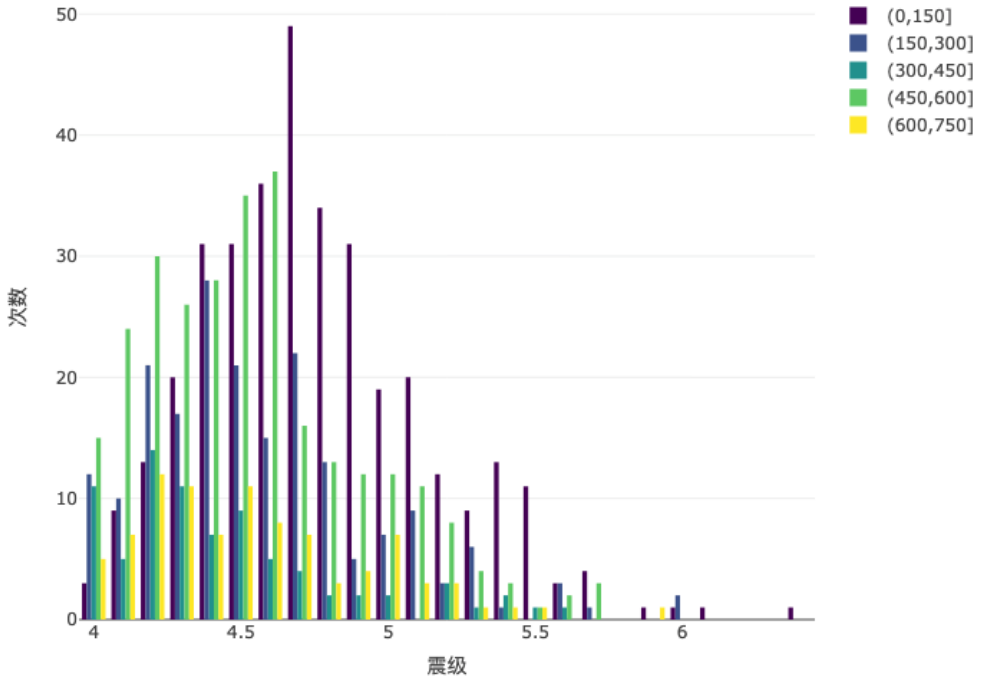


图 6.5: 地震震级的频率分布图

6.2.5 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")
  )
```

6.2.6 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 6.6 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")
  )
```

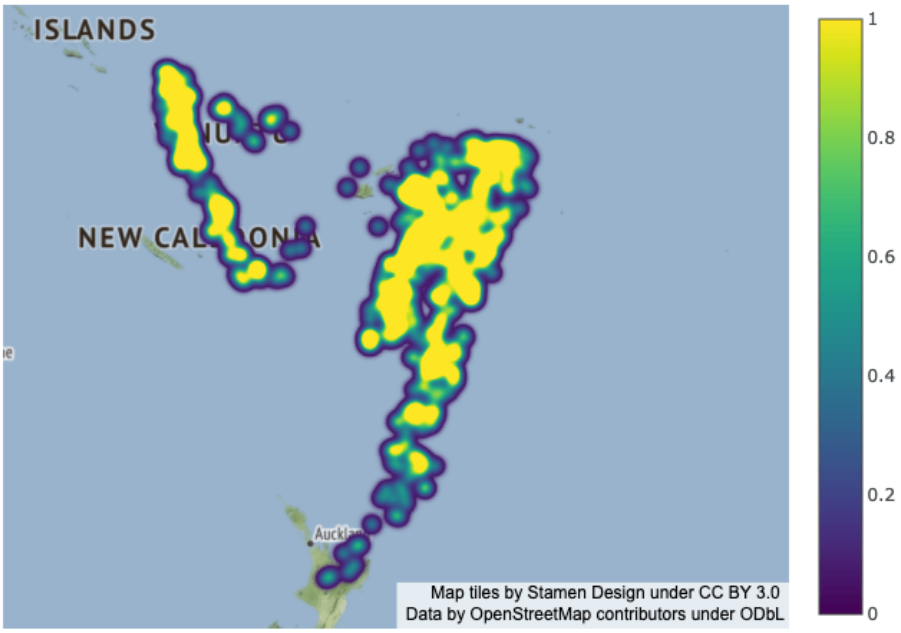


图 6.6: 空间点数据的核密度估计

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。

6.2.7 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 6.7 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"
  )
```

6.2.8 动态图

本节参考 plotly 包的官方示例渐变动画，数据来自 SVN 代码提交日志，统计 Martin Maechler 和 Brian Ripley 的年度代码提交量，他们是 R Core Team 非常重要的两位成员，长期参与维护 R 软件及社区。下图展示 1999-2022 年 Martin Maechler 和 Brian Ripley 的代码提交量变化。

```
# https://plotly.com/r/animations/
trunk_year_author <- aggregate(data = svn_trunk_log, revision ~ year + author, FUN = length)
# https://plotly.com/r/cumulative-animations/
accumulate_by <- function(dat, var) {
  var <- lazyeval::f_eval(f = var, data = dat)
```

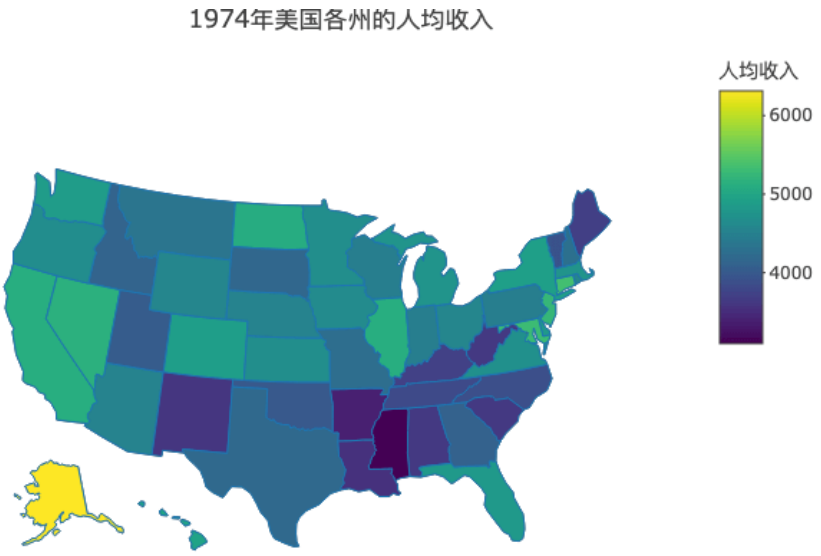


图 6.7: 1974 年美国各州的人均收入

```
  lvls <- plotly:::getLevels(var)
  dats <- lapply(seq_along(lvls), function(x) {
    cbind(dat[var %in% lvls[seq(1, x)], ], frame = lvls[[x]])
  })
  dplyr::bind_rows(dats)
}

subset(trunk_year_author, year >= 1999 & author %in% c("ripley", "maechler")) |>
  accumulate_by(~year) |>
  plotly::plot_ly(
    x = ~year, y = ~revision, split = ~author,
    frame = ~frame, type = "scatter", mode = "lines",
    line = list(simplyfy = F)
  ) |>
  plotly::layout(
    xaxis = list(title = "年份"),
    yaxis = list(title = "代码提交量")
  ) |>
  plotly::animation_opts(
    frame = 100, transition = 0, redraw = FALSE
```

```
  ) |>
  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。

6.3 常用技巧

6.3.1 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)
```

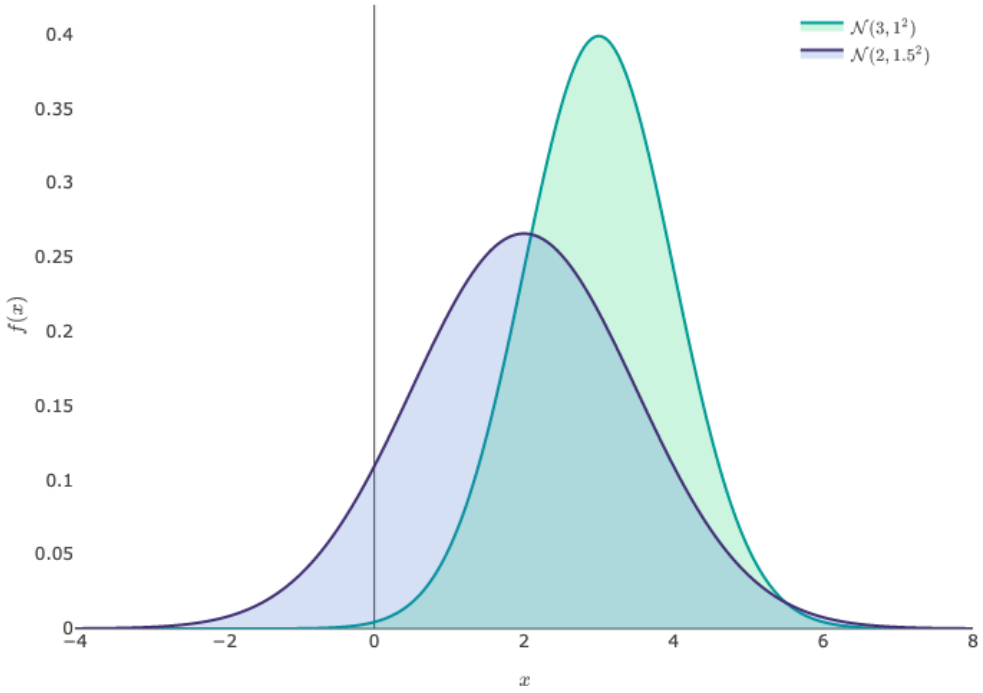


图 6.8: 设置数学公式

6.3.2 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
```

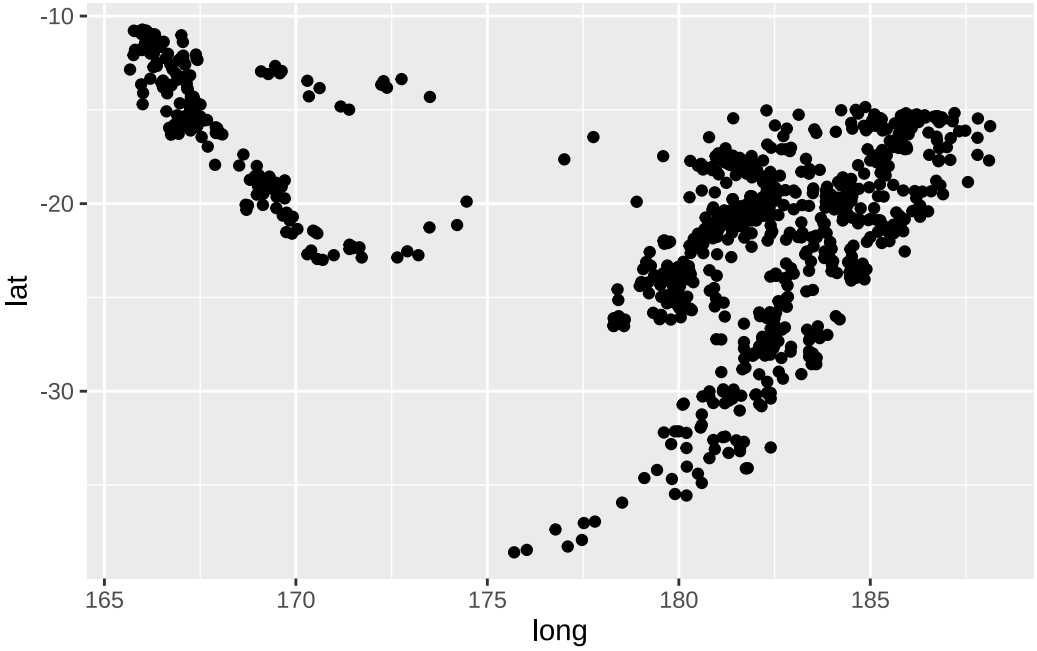


图 6.9: ggplot2 绘制的静态图形

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")
```

6.3.3 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 6.10 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
    )
  ))
```

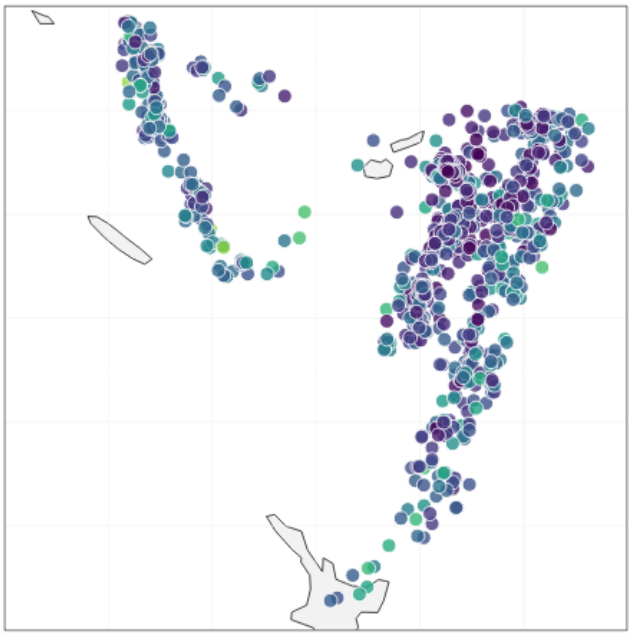


图 6.10: 空间点数据图

6.3.4 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
    )
  )
```

6.3.5 多图布局

将两个图形做上下排列

```
p1 <- plotly::plot_ly(
  data = trunk_year, x = ~year, y = ~revision, type = "bar"
) |>
  plotly::layout(
    xaxis = list(title = "年份"),
    yaxis = list(title = "代码提交量")
  )

p2 <- plotly::plot_ly(
  data = trunk_year, x = ~year, y = ~revision, type = "scatter",
  mode = "markers+lines", line = list(shape = "spline")
) |>
  plotly::layout(
    xaxis = list(title = "年份"),
    yaxis = list(title = "代码提交量")
  )

htmltools::tagList(p1, p2)
```

plotly 包提供的函数 subplot() 专门用于布局排列，下图的上下子图共享 x 轴。

```
plotly::subplot(plotly::style(p1, showlegend = FALSE),
                plotly::style(p2, showlegend = FALSE),
                nrows = 2, margin = 0.05, shareX = TRUE, titleY = TRUE)
```

下图展示更加灵活的布局形式，嵌套使用布局函数 subplot() 实现。

```
p11 <- plotly::subplot(plotly::style(p1, showlegend = FALSE),
  plotly::style(p2, showlegend = FALSE),
  nrows = 1, margin = 0.05, shareY = TRUE, titleX = TRUE
)

plotly::subplot(p11,
  plotly::style(p2, showlegend = FALSE),
  nrows = 2, margin = 0.05, shareY = FALSE, titleX = FALSE
)
```

6.3.6 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))
```

第七章　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**。

7.1　基础功能

7.1.1　创建表格

7.1.2　添加标题

7.1.3　添加注释

7.1.4　水平滚动

7.1.5　垂直滚动

7.1.6　数据分页

7.1.7　适应宽度

7.1.8　行列分组

7.1.9　列格式化

7.1.10　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",
  as.character(htmltools::em("强调")), '<a href="#" onclick="alert(\'Hello World\');">Hello</a>',
  as.character(htmltools::span(style = "color:red", "正常")), "正常"
)
```

根据数据的大小配上颜色

```
colorize_num <- function(x) {
  ifelse(x > 0,
    sprintf("<span style='color:%s'>%s</span>", "green", x),
    sprintf("<span style='color:%s'>%s</span>", "red", x)
  )
}
colorize_pct <- function(x) {
  ifelse(x > 0,
    sprintf("<span style='color:%s'>%s</span>", "green", scales::percent(x, accuracy = 0.01)),
    sprintf("<span style='color:%s'>%s</span>", "red", scales::percent(x, accuracy = 0.01))
  )
}

colorize_pp <- function(x) {
  ifelse(x > 0,
    sprintf("<span style='color:%s'>%s</span>", "green", paste0(round(100*x, digits = 2), "PP")),
    sprintf("<span style='color:%s'>%s</span>", "red", paste0(round(100*x, digits = 2), "PP"))
  )
}

colorize_text <- function(x, color = "red") {
    sprintf("<span style='color:%s'>%s</span>", color, x )
}

library(DT)
datatable(
  data = dat, escape = FALSE,
  colnames = c(colorize_text("第1列", "red"),
               as.character(htmltools::em("第2列"))),
  options = list(
    pageLength = 5, # 每页显示5行
    dom = "t"
```

表格 7.1: 数据配色

	第1列	第2列
1	加粗	超链
2	强调	Hello
3	正常	正常

```
  )
)
```

Base R 内置的 R 包含有丰富的数据集，非常适合演示图形和阐述统计理论，后面技术和理论部分的介绍大多围绕内置的数据集展开，数据集及其描述如下表所示：

```
# 抽取 R 包信息
Pkgs <- sapply(list.files(R.home("library")), function(x) {
  packageDescription(pkg = x, fields = "Priority")
})
# 抽取内置 R 包列表
CorePkgs <- names(Pkgs[Pkgs %in% c("base", "recommended") & !is.na(Pkgs)])
# 抽取 R 包的数据集
BaseDataSets <- data(package = CorePkgs)$results[, c("Package", "Item", "Title")]

library(DT)
datatable(BaseDataSets,
  rownames = FALSE, # 不显示行名
  extensions = c("Buttons", "RowGroup"),
  options = list(
    pageLength = 10, # 每页显示的行数
    language = list(url = "//cdn.datatables.net/plug-ins/1.10.11/i18n/Chinese.json"), # 汉化
    dom = "Bfrtp", # 去掉显示行数 i、过滤 f 的能力，翻页用 p 表示
    ordering = F, # 去掉列排序
    buttons = c("copy", "csv", "excel", "print"), # 提供打印按钮
    rowGroup = list(dataSrc = 0), # 按 Package 列分组
    columnDefs = list(
      list(className = "dt-center", targets = 0), # 不显示行名，则 targets 从 0 开始，否则从 1 开始
      list(visible = FALSE, targets = 0) # 不显示 Package 列
    )
  )
```

表格 7.2: Base R 包内置的数据集

Copy CSV Excel Print Search:

Item	Title
datasets	
AirPassengers	Monthly Airline Passenger Numbers 1949-1960
BJsales	Sales Data with Leading Indicator
BJsales.lead (BJsales)	Sales Data with Leading Indicator
BOD	Biochemical Oxygen Demand
CO2	Carbon Dioxide Uptake in Grass Plants
ChickWeight	Weight versus age of chicks on different diets
DNase	Elisa assay of DNase
EuStockMarkets	Daily Closing Prices of Major European Stock Indices, 1991-1998
Formaldehyde	Determination of Formaldehyde
HairEyeColor	Hair and Eye Color of Statistics Students

Previous 1 2 3 4 5 … 37 Next

)

7.2 扩展功能

7.2.1 汉化表格

7.2.2 下载数据

7.3 其它工具

第八章　交互应用

```
library(shiny)
```

一个简单示例，介绍一个 Shiny 应用的各个常见组成部分。一个快速改变风格的主题包。介绍交互表格、交互图形与 Shiny 集成，如 **DT**、**plotly**、**leaflet** 等。介绍 Shiny 工业化应用的开发过程。

8.1　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)
```

8.1.1 UI 前端

8.1.2 Server 后端

8.2 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。

8.2.1 筛选器

单个筛选器、独立筛选器、筛选器联动

8.2.2 输入框

数值型、文本型

8.2.3 动作按钮

提交按钮、响应按钮

8.2.4 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)
```

8.3 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

8.3.1 页面布局

8.3.2 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。

8.3.3 交互图形

ggiraph 包

8.4 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

8.4.1 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)
```

8.4.2 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) { }
)
```

8.4.3 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)
```

8.4.4 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)
```

8.5 Shiny 主题

8.5.1 bslib 包

- bslib

8.5.2 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)
```

8.6 Shiny 优化

提升 shiny 仪表盘访问性能的 4 个建议

8.7 Shiny 部署

8.7.1 promises 并发

shiny 异步编程实现并发访问，多人同时访问 Shiny 应用的情况下，解决必须等另一个人完成访问的情况下才能继续访问的问题。

```
library(shiny)
library(future)
library(promises)

plan(multiprocess)

ui <- fluidPage(
  h2("测试异步下载"),
  tags$ol(
    tags$li("Verify that plot appears below"),
    tags$li("Verify that pressing Download results in 5 second delay, then rock.csv being downloade
    tags$li("Check 'Throw on download?' checkbox and verify that pressing Download results in 5 sec
  ),
  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)
```

8.8 Shiny 替代品

R Markdown + Shiny 文档

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

Quarto Dashboard 文档

8.9 Shiny 案例

- radiant 探索性数据分析解决方案

8.10 总结

事实上，作为 BI 工程师，相当一部分工作是与数据开发结合的。从 Kafka 接入埋点上报的原始日志（ODS 层）、清洗抽取特定业务/领域内的数据（Fact 事实层）、面向某一类任务的主题数据（topic 主题

层)、面向特定数据产品的应用数据（app 应用层)。

- 数据仓库 Hive 数据开发：事实、主题和应用层
- 数据计算 Spark 数据开发工具 Spark SQL / Hive SQL 任务调度
- 数据报表 MySQL / Doris 数据同步工具 Hive2MySQL 同步应用层数据
- 数据展示 Dashboard 应用开发工具 Shiny RStudio Shiny Server

报表开发从数据仓库的 DWD 层开始，可能一些业务原因，我们需要从 ODS 层甚至从点击流的日志数据开始，经过数据清洗、提取、聚合成为支撑 BI 报表最底层的基础表，存储在 Hive 中，然后对这一系列的基础表根据 BI 展示的需要进行第二层聚合形成中间表，这两层数据根据业务情况做增量更新或者全量更新，并将中间表同步到 MySQL 仓库中，全量更新的情况，往往更新数据比较大，建议用 sqoop 做数据的同步。创建第二层的中间表稍有些灵活性，原则是在中间表之上对应的数据操作和可视化是容易实现且效率较高的，否则应该构造第三层的中间表，绝不能将大规模的数据集直接导入 R 中进行分析和可视化，拖慢前端展示的速度，占用过多的服务器资源。

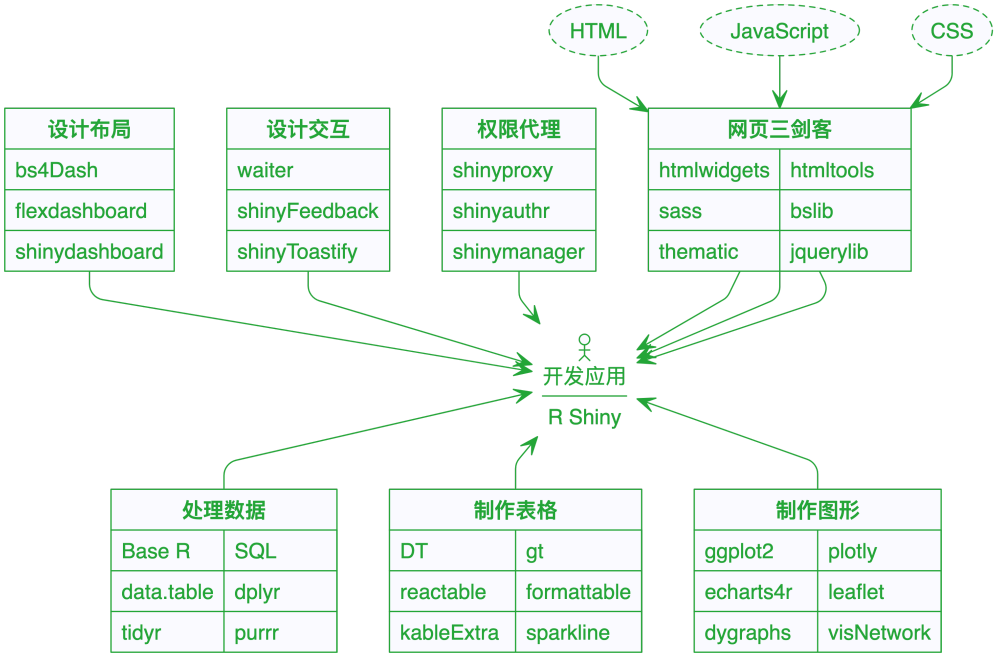


图 8.1: Shiny 生态系统

- 连接数据库。根据数据库的情况选择相应的 R 接口包，比如连接 MySQL 数据库可以用 RMySQL 包，值得一提，odbc 包支持连接相当多的数据库。
- 数据操作。根据需要处理的数据规模，可以选择 Base R、data.table 或者 dplyr 做数据操作，推荐和管道操作一起使用，增加代码可读性。
- 交互表格。推荐 reactable 和 DT 包做数据呈现。
- 交互图形。推荐功能强大的 plotly 包，可以先用 ggplot2 绘制，然后调用 plotly 包的 `ggplotly()` 函数将静态图转化为交互图。

- 针对特定应用场景的其它交互可视化工具包，比如 leaflet 可以将地图嵌入 Shiny 应用，dygraphs 可以将时间序列塞进去。
- Shiny 组件。shinyFeedback 提供用户输入的反馈。shinyWidgets 提供自定义 widget 的功能。miniUI 专为小屏设计，shinyMobile 在 IOS 和安卓手机上访问 shiny 应用。
- Shiny 主题。比如 shinythemes 包可以统一配色，dashboardthemes 提供更加深度的主题，shinytableau 提供仿 Tableau 的 dashboard 框架。sass 在 CSS 样式层面重定义风格。bslib 通过 Bootstrap 3/4/5 定制 Shiny 和 R Markdown 主题。
- Shiny 权限。shinymanager / shinyauthr 支持单个 shiny 应用的权限管理，firebase 提供访问权限设置 https://firebase.john-coene.com/。
- Shiny 框架。ShinyStudio 打造基于容器架构的协作开发环境的开源解决方案，golem 构建企业级 shiny 应用的框架，RinteRface 开发的系列 R 包也试图打造一套完整的解决方案，并配有速查小抄 cheatsheets。
- Shiny 部署。shiny-server 以网络服务的方式支持 shiny 应用，shinyproxy 提供企业级部署 shiny 应用的开源解决方案。

自 RStudio 推出 Shiny 系列产品以来，一些公司进一步根据所需扩展和定制，比如 Appsilon、RinteRface、ThinkR-open、dreamRs 和datastorm-open 等。经过商业公司和个人开发者的努力，Shiny 生态非常庞大，资源非常丰富。

- Shiny 入门 https://shiny.posit.co/r/getstarted/。
- Shiny 扩展包 https://github.com/nanxstats/awesome-shiny-extensions。
- Shiny 常用技巧和提示 https://github.com/daattali/advanced-shiny。
- Shiny 各类资源列表 https://github.com/grabear/awesome-rshiny。

特别值得一提，Shiny 方面的三本专著。

- Hadley Wickham 的书 Mastering Shiny。
- Colin Fay, Sébastien Rochette, Vincent Guyader, Cervan Girard 的书 Engineering Production-Grade Shiny Apps。
- David Granjon 的书 Outstanding User Interfaces with Shiny。

第九章　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。

9.1　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。

9.1.1　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`

9.1.2　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 9.1: 三种鸢尾花

```
flowchart LR
  A[Hard edge] --> B(Round edge)
  B --> C{Decision}
  C --> D[Result one]
  C --> E[Result two]
```

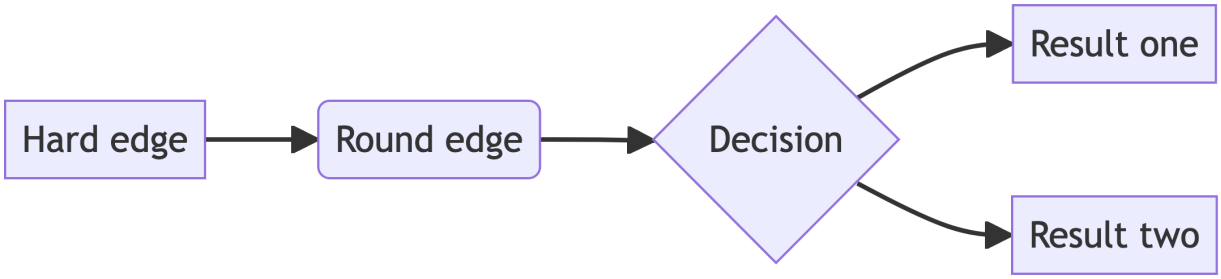


图 9.2: 流程图

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()
```

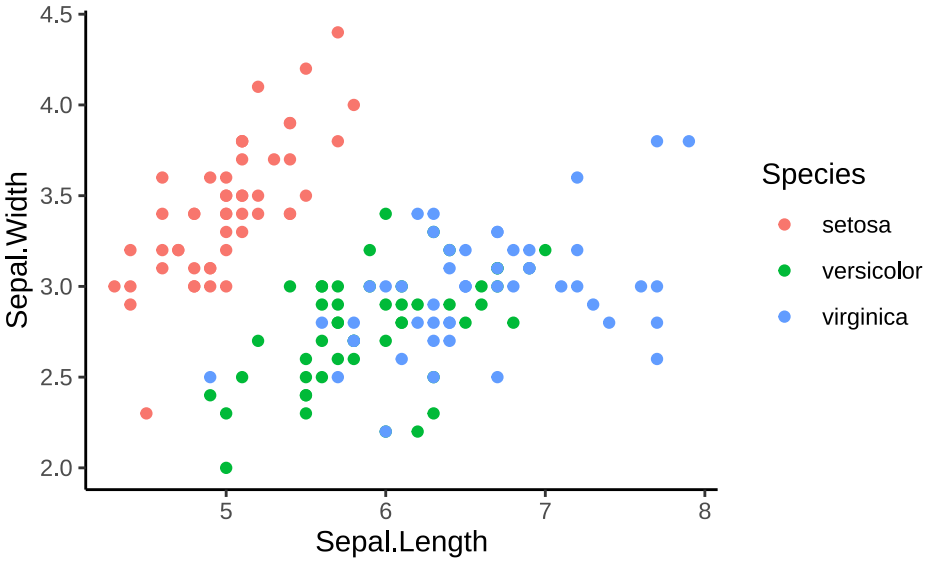


图 9.3: 一幅简单的 ggplot2 图形

9.1.3 表格

Markdown 原生支持的表格和 **knitr** 包制作的表格。

表格 9.2: 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 9.3: 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

9.1.4 列表

常见的列表有无序列表、有序列表及其嵌套。

表格 9.4: 几种列表

Markdown 语法	输出

`* 无序列表`
`  + 子条目 1`
`  + 子条目 2`
`  - 子子条目 1`
`*  条目 2`
 	
`  继续（缩进 4 格）`
	• 无序列表
	– 子条目 1
	– 子条目 2
	∗ 子子条目 1
	• 条目 2
	继续 (缩进 4 格)

`1. 有序列表`
`2. 条目 2`
`  i) 子条目 1`
`  A. 子子条目 1`
	1. 有序列表
	2. 条目 2
	i) 子条目 1
	A. 子子条目 1

`(@)  第一个人是好的`
 	
`第二个人是坏的`
 	
`(@)  第三个人是丑陋的`
	(1) 第一个人是好的
	第二个人是坏的
	(2) 第三个人是丑陋的

`::: {}`
`1. 一个列表`
`:::`
 	
`::: {}`
`1. 又一个列表`
`:::`
	1. 一个列表
	1. 又一个列表

`术语` `: 定义`	**术语** 定义

在 `(@)` 中添加标识符，如 `(@good)` 就可以引用列表中的条目 (1)。

9.1.5 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。

9.1.6 脚注

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

— Paul Murrell

9.1.7 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 9.1 。

$$\begin{aligned} \mathbf{y} &= X\boldsymbol{\beta} + \boldsymbol{\epsilon} \\ y_i &= \mathbf{x}_i\boldsymbol{\beta} + \epsilon_i \end{aligned} \tag{9.1}$$

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 9.2 。

$$\begin{aligned} \mathsf{Var}\{\hat{\boldsymbol{\beta}}\} &= \mathsf{Var}\{(X^\top X)^{-1}X^\top \mathbf{y}\} \\ &= (X^\top X)^{-1}X^\top \mathsf{Var}\{\mathbf{y}\}\big((X^\top X)^{-1}X^\top\big)^\top \\ &= (X^\top X)^{-1}X^\top \mathsf{Var}\{\mathbf{y}\}X(X^\top X)^{-1} \\ &= (X^\top X)^{-1}X^\top \sigma^2 I X(X^\top X)^{-1} \\ &= (X^\top X)^{-1}\sigma^2 \end{aligned} \tag{9.2}$$

值得注意，

1. LaTeX 命令 `\mathbf` 只对英文字母 a, b, c, A, B, C 加粗，对希腊字母 $\theta, \alpha, \beta, \ldots, \gamma$ 加粗应该使用命令 `\boldsymbol`。

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

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。

```
$$
\def\bm#1{{\boldsymbol #1}}
$$
```

9.2 制作报告

Quarto Report 文档

9.2.1 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)
```

9.3 制作演示

Quarto Presentation

9.4 编写书籍

Quarto Book 网页格式

第十章　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。

10.1　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。

10.1.1 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
```

10.1.2 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 10.1: LaTeX 公式排版环境

可编号	无编号	作用
align	align*	多行公式对齐
equation	equation*	可与 split / cases 等环境嵌套使用
multline	multline*	长公式折行
gather	gather*	多行公式居中

不可编号的排版环境，行内公式排版，用一对美元符号 `$ $` 或一对小括号 `\( \)`。行间公式排版，用一对双美元符号 `$$ $$` 或一对中括号 `\[ \]` 。

LaTeX 支持丰富的数学符号大、小写英文字母，大、小写希腊字母，字母可以加粗、倾斜，字母也可以设置为等宽字体或衬线字体，还可以设置花体、空心体等。一些常用的数学符号样式见下表。

大写	小写	加粗	无衬线
X	x	$\boldsymbol{X}$	X
衬线	花体	空心体	花体
X	$\mathcal{X}$	$\mathbb{X}$	$\mathscr{X}$
大写	小写	加粗	无衬线
Γ	γ	$\boldsymbol{\gamma}$	$\mathsf{\Gamma}$

```
\[
\Bigg(\sqrt{\frac{M}{1 - \big(\frac{r}{\widetilde{x_1 + \cdots + u_N}} \big)^2}
\big(\sum_{\beta =1}^{N} \sum_{i=1}^{n}\frac{\partial u_{\beta}}{\partial x_i} + 1 \big) }
+ \sqrt{XY} \Bigg)^3
\]
```

amsmath 宏包渲染效果如下：

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

newtxtext 和 **newtxmath** 宏包常组合在一起，提供一套 New Times 字体风格的文本和数学公式，一种介于。

```
\documentclass[b5paper]{article}
\usepackage{amsmath}
\usepackage{newtxtext,newtxmath}
\begin{document}
\[
\Bigg(\sqrt{\frac{M}{1 - \big(\frac{r}{\widetilde{x_1 + \cdots + u_N}} \big)^2}
\big(\sum_{\beta =1}^{N} \sum_{i=1}^{n}\frac{\partial u_{\beta}}{\partial x_i} + 1 \big) }
+ \sqrt{XY} \Bigg)^3
\]
\end{document}
```

newtxmath 宏包渲染效果如下：

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

图 10.1: newtxmath 包渲染的公式效果

10.1.3 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 10.2: 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：

```
1 library(stats)    % 提供 lowess, rpois, rnorm 等函数
2 library(graphics) % 提供 plot 方法
3 plot(cars)
4 lines(lowess(cars))
```

图 10.3: lstlisting 抄录环境

10.1.4 插入图表

首先在导言区加载 graphicx 宏包，然后可以使用 `\includegraphics` 命令插入图片，该命令有一些选项，`[width=.65\textwidth]` 表示插入的图片占页面宽度的 65%。

```
\usepackage{graphicx}
```

在正文中 figure 环境是专门用来处理的图片，选项 `[h]` 表示将图片就插入此处，不要浮动。center 环境将图片居中，`\caption` 和 `\label` 命令分别用来指定图片的标题和标签。

```
\begin{figure}[h]
  \begin{center}
    \includegraphics[width=.65\textwidth]{images/peaks.png}
    \caption{图片的标题}
    \label{fig:figure}
  \end{center}
\end{figure}
```

渲染效果如下

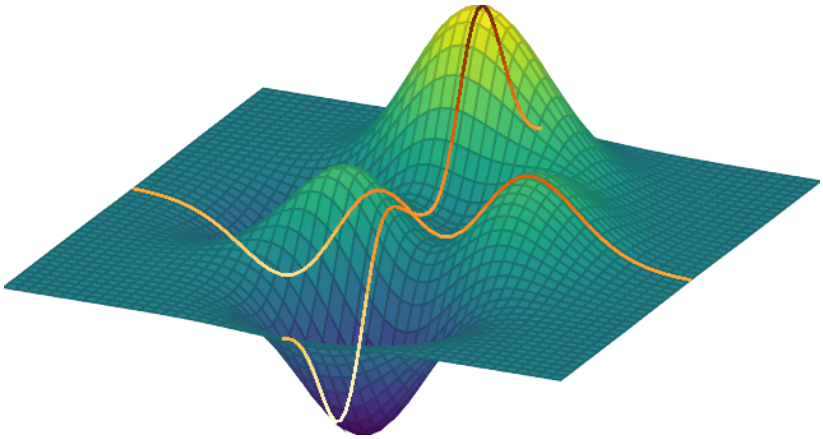


图 10.4: 图片的标题

table 环境用于制作控制表格位置，tabular 用于制作表格，控制表头、每个列和每个格子。

```
\begin{table}[h!]
  \begin{center}
    \begin{tabular}{|c c c|}
     \hline
      列1 & 列2 & 列3 \\
      \hline
      1 & 6 & 77 \\
      2 & 7 & 15 \\
      3 & 8 & 44 \\
```

```
    \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 10.3: 表格的标题

列 1	列 2	列 3
1	6	77
2	7	15
3	8	44

10.1.5 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。

10.1.6 PDF-A/X

启用 PDF/X 或 PDF/A 标准，导言区加载 pdfx 宏包

```
% PDF/A-1b 标准
\usepackage[a-1b]{pdfx}
```

示例文档内容如下：

```
\documentclass[b5paper]{article}
\usepackage{amsmath} % boldsymbol
\usepackage[a-1b]{pdfx}
\title{Math in LaTeX}
\author{Zhang San}
\begin{document}
\maketitle
```

```
\tableofcontents
\section{Math}
\begin{align}
\boldsymbol{x},\mathbf{x},\mathsf{x},x
\end{align}
\begin{verbatim}
require(stats) # for lowess, rpois, rnorm
require(graphics) # for plot methods
plot(cars)
lines(lowess(cars))
\end{verbatim}
\end{document}
```

编译 LaTeX 文档的命令如下：

```
xelatex --shell-escape -output-driver="xdvipdfmx -z 0" <filename>.tex
```

10.2 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}
```
```

10.2.1 中英字体

10.2.2 数学公式

10.2.3 代码抄录

10.2.4 插入图表

10.2.5 交叉引用

10.2.6 布局排版

```
---
title: "R Markdown 双栏排版"
subtitle: "副标题"
author: "张三"
date: "`r Sys.Date()`"
mathspec: yes
fontsize: 10pt
graphics: yes
lof: yes
geometry: margin=1.18in
output:
  bookdown::pdf_book:
    number_sections: yes
    toc: yes
    fig_crop: no
    latex_engine: xelatex
    base_format: rmarkdown::pdf_document
```

```
    citation_package: natbib
    template: null
    extra_dependencies:
      sourcecodepro:
       - scale=0.85
      ctex:
       - heading=true
       - fontset=fandol
      caption:
       - labelfont=bf
       - singlelinecheck=off
       - textfont=it
       - justification=centering
      Alegreya: null
keywords:
  - 动态文档
  - 双栏排版
subject: "可重复研究与动态文档"
abstract: |
  这里是摘要内容
bibliography:
 - packages.bib
biblio-style: plainnat
natbiboptions: "authoryear,round"
link-citations: true
colorlinks: true
classoption: "UTF8,a4paper,twocolumn"
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

R Markdown 文档混合了代码、图形和文字内容[@rmarkdown]。

代码 {#sec:code}

```{r cars}
```
```

```
summary(cars)
```

插图 {#sec:plot}

```{r}
#| fig-iris,
#| fig.cap="鸢尾花数据集",
#| fig.width=5,
#| fig.height=4,
#| fig.showtext=TRUE,
#| out.width="95%",
#| echo=FALSE

library(ggplot2)
ggplot(iris, aes(Sepal.Length, Sepal.Width)) +
  geom_point(aes(colour = Species)) +
  scale_colour_brewer(palette = "Set1") +
  labs(
    title = "鸢尾花数据的散点图",
    x = "萼片长度", y = "萼片宽度", colour = "鸢尾花类别",
    caption = "鸢尾花数据集最早见于 Edgar Anderson (1935) "
  )
```

参考文献 {#chap:refer}

10.3 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}
```

10.3.1 中英字体

10.3.2 数学公式

10.3.3 代码抄录

10.3.4 插入图表

插入图片的语法是 `![]()`{}，中括号内是插图标题，小括号内是插图存放路径，大括号内是插图的标识符和属性，比如 `width="65%"` 设置图片的宽度为页面宽度的 65%。

```
![An Elephant](images/peaks.png){#fig-quarto-figure width="65%"}
```

渲染效果如下：

表格默认左对齐，冒号加虚线、虚线加冒号、虚线两侧加冒号分别对应左对齐、右对齐和居中对齐。

```
| Default | Left | Right | Center |
|---------|:-----|------:|:------:|
| 12      | 12   |    12 |   12   |
| 123     | 123  |   123 |  123   |
| 1       | 1    |     1 |   1    |

: 制作表格的管道语法 {#tbl-quarto-table}
```

渲染效果如下：

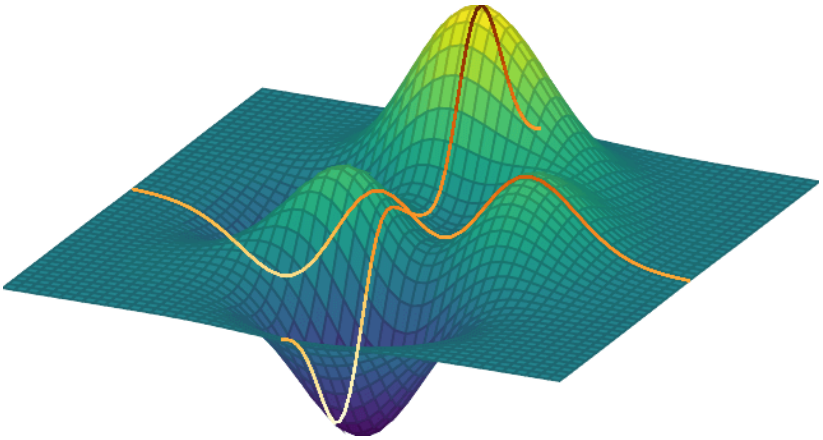


图 10.5: Peaks 函数图像

表格 10.4: 制作表格的管道语法

Default	Left	Right	Center
12	12	12	12
123	123	123	123
1	1	1	1

10.3.5 交叉引用

`@tbl-quarto-table` 插入表格 10.4 ，`@fig-quarto-figure` 插入图 10.5 。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。

10.4 Quarto beamer

Quarto 制作 beamer 幻灯片

```
---
title: "Quarto 幻灯片模版"
author:
  - 张三
  - 李四
institute:
  - XX 大学
  - XX 学院
```

```yaml
date: today
date-format: long
documentclass: beamer
classoption:
  - 11pt
  - compress
  - xcolor=x11names
  - UTF8
lang: zh
format:
  beamer:
    theme: Singapore
    fonttheme: structurebold
    pdf-engine: lualatex
    include-in-header:
      text: |
        \usecolortheme[named=SpringGreen4]{structure}
        \usepackage[fontset=fandol]{ctex}
    keep-tex: false
    mathspec: true
    toc: true
    navigation: horizontal
    latex-min-runs: 2
    latex-auto-install: false
link-citations: true
---

# In the morning

## Getting up

-   Turn off alarm
-   Get out of bed

## Breakfast

-   Eat eggs
-   Drink coffee

# In the evening
```

```
## Dinner

-   Eat spaghetti
-   Drink wine

## Going to sleep

-   Get in bed
-   Count sheep
```

Pandoc's Markdown 制作 beamer 幻灯片

```
---
title: "Quarto 幻灯片模版"
author:
  - 张三
  - 李四
institute:
  - XX 大学
  - XX 学院
mathspec: true
toc: true
toc-title: "目录"
---

# In the morning

## Getting up

-   Turn off alarm
-   Get out of bed

## Breakfast

-   Eat eggs
-   Drink coffee

# In the evening

## Dinner
```

```
-   Eat spaghetti
-   Drink wine

## Going to sleep

-   Get in bed
-   Count sheep
```

将 Markdown 文档转化为 beamer 幻灯片的命令

```
pandoc --pdf-engine lualatex -t beamer \
  --variable theme="Singapore" --variable fonttheme="structurebold" \
  --variable classoption="xcolor=x11names" \
  --variable header-includes="\usecolortheme[named=SpringGreen4]{structure}" \
  --variable header-includes="\usepackage[fontset=fandol]{ctex}" \
  -f markdown pandoc-beamer.md -o pandoc-beamer.pdf
```

第十一章　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。

11.1　Word 文档

11.1.1　Markdown 制作 Word 文档

本节探索 (R) Markdown + Pandoc 以 Word 格式作为最终交付的可能性。

11.1.2　R Markdown 制作 Word 文档

docxtools、officer 和 officedown 大大扩展了 rmarkdown 包在制作 Word/PPT 方面的功能。

11.1.3　自定义 Word 模版

R Markdown 借助 Pandoc 将 Markdown 转化为 Word 文档，继承自 Pandoc 的扩展性，R Markdown 也支持自定义 Word 模版，那如何自定义呢？首先，我们需要知道 Pandoc 内建的 Word 模版长什么样子，然后我们依样画葫芦，制作适合实际需要的模版。获取 Pandoc 自带的 Word 和 PPT 模版，只需在命令行中执行

```
# DOCX 模版
pandoc -o custom-reference.docx --print-default-data-file reference.docx
# PPTX 模版
pandoc -o custom-reference.pptx --print-default-data-file reference.pptx
```

这里其实是将 Pandoc 自带的 docx 文档 reference.docx 拷贝一份到 custom-reference.docx，而后将 custom-reference.docx 文档自定义一番，但仅限于借助 MS Word 去自定义样式。

- Word 文档的 YAML 元数据定义
- 如何深度自定义文档模版

bookdown 提供的函数 `word_document2()` 相比于 **rmarkdown** 提供的 `word_document()` 支持图表的交叉引用，更多细节详见帮助 `?bookdown::word_document2`。

11.2 PowerPoint 演示

11.3 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。

11.3.1 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。

11.3.2 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")
)
```

第三部分

统计分析

第十二章　常见的统计检验

本章亮点

1. 比较全面地展示各类统计检验问题的 R 语言实现，其覆盖面之广，远超市面上同类 R 语言书籍。从连续数据到离散数据，从单样本到两样本，再到一般的多样本，触及最前沿的热门话题。
2. 对每类统计检验问题都给出示例及 R 语言实现，涉及近 40 个统计检验方法。在组织结构上，本章按照数据情况对检验方法分类，方便读者根据手头的数据情况，快速从众多的方法中定位最合适的检验方法，符合从数据出发进行分析实战的要求。

> The Earth is Round ($p < 0.05$)
>
> — Jacob Cohen (Cohen 1994)

Jacob Cohen 实际谈的是更加深刻的问题。开篇介绍为什么需要假设检验，做检验和不做检验有什么区别？R. A. Fisher 将抽样分布、参数估计和假设检验列为统计推断的三个中心内容，可见假设检验的重要地位。经过近百余年的发展，假设检验具有丰富的内容，可以从不同的角度对检验方法进行归类。

- 检验方法归类：参数与非参数检验方法。
- 检验计算方式：近似 Approximate、精确 Extract、模拟 Simulation 和重抽样 Bootstrap 等方式。
- 检验对象归类：位置参数（均值）和尺度参数（方差）的检验。
- 检验总体数量归类：单总体、两个总体和多个总体。
- 检验总体分布归类：正态、二项、泊松、多项分布等。
- 检验总体维度归类：分一维、二维和多维的情形。
- 检验样本的数量：小样本 $n < 30$ 和大样本 $n \geq 30$。

χ^2 分布、t 分布和 F 分布作为最基础的三大抽样分布，分别是由 K. Pearson 卡尔 · 皮尔逊、W. S. Gosset 哥塞特、R. A. Fisher 费希尔提出，并以他们的名字命名的。在假设检验中，也有许多检验方法是以提出者的名字命名的。本来名字具有突出效果，由检验方法联系人物名称，可以帮助记忆，但如此之多，以至于很难一一记住。因此，本文也不按检验方法罗列，但是，推荐读者了解这些统计大师的工作和故事，相信会加深对这些检验方法的理解。关于检验方法，如有不明白的地方，可以查看维基百科词条。对每个检验问题，本章给出原假设和备择假设，检验统计量及其服从的分布，R 语言实现（自编

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
```

12.1 单样本检验

12.1.1 正态总体均值检验

12.1.1.1 方差已知

$$
\begin{aligned}
\text{I} \quad & H_0 : \mu - \mu_0 \le 0 \quad vs. \quad H_1 : \mu - \mu_0 > 0 \\
\text{II} \quad & H_0 : \mu - \mu_0 \ge 0 \quad vs. \quad H_1 : \mu - \mu_0 < 0 \\
\text{III} \quad & H_0 : \mu - \mu_0 = 0 \quad vs. \quad H_1 : \mu - \mu_0 \ne 0
\end{aligned}
$$

设 $x_1, \cdots, x_n$ 是来自总体 $\mathcal{N}(\mu, \sigma^2)$ 的样本，样本均值和方差分别

$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$ ， $s^2 = \frac{1}{n-1}\sum_{i=1}^n (x_i - \bar{x})^2$

考虑到 $\bar{x} \sim \mathcal{N}(\mu, \sigma^2/n)$ ，则检验统计量服从正态分布

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

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

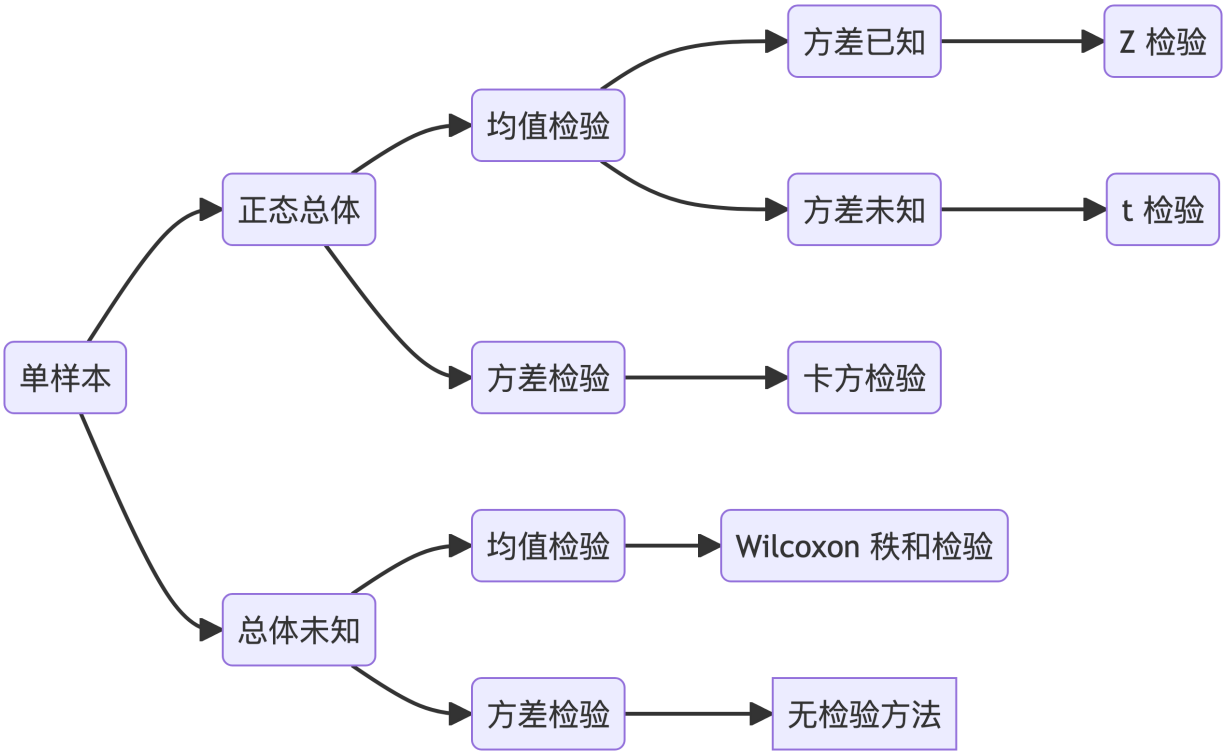


图 12.1: 单样本检验

```
set.seed(20232023)
n <- 20
# 样本
x <- rnorm(n, mean = 1.8, sd = 2)
# 检验统计量
u <- (mean(x) - 1) / (2 / sqrt(n))
# 临界值
qnorm(p = 1 - 0.05, mean = 0, sd = 1)
```

```
#> [1] 1.644854
```

```
# P 值
1 - pnorm(q = u)
```

```
#> [1] 0.005082465
```

重要

随机变量 X 服从标准正态分布，它的概率分布函数如下：

$$P(X \leq u) = \phi(u) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{u} \mathrm{e}^{-t^2/2} \mathrm{dt}$$

若已知概率 $p = 0.95$ ，则对应的下分位点可用函数 `qnorm()` 计算。

```
qnorm(p = 0.95, mean = 0, sd = 1)
#> [1] 1.644854
```

12.1.1.2 方差未知

$$
\begin{aligned}
\text{I} \quad & H_0 : \mu - \mu_0 \leq 0 \quad vs. \quad H_1 : \mu - \mu_0 > 0 \\
\text{II} \quad & H_0 : \mu - \mu_0 \geq 0 \quad vs. \quad H_1 : \mu - \mu_0 < 0 \\
\text{III} \quad & H_0 : \mu - \mu_0 = 0 \quad vs. \quad H_1 : \mu - \mu_0 \neq 0
\end{aligned}
$$

考虑到

$$
\begin{aligned}
&\frac{\bar{x} - \mu}{\sigma / \sqrt{n}} \sim \mathcal{N}(0, 1) \\
&\frac{(n-1)s^2}{\sigma^2} \sim \chi^2(n-1) \\
&\mathsf{E}\{s^2\} = \sigma^2 \quad \mathsf{Var}\{s^2\} = \frac{2\sigma^4}{n-1}
\end{aligned}
$$

根据 t 分布的定义，检验统计量服从 t 分布，即 $t \sim t(n-1)$

$$
t = \frac{\bar{x} - \mu_0}{s / \sqrt{n}}
$$

假定 $\mu_0 = 1$ 对于检验问题 I ，拒绝域 $\{t \geq t_{1-\alpha}(n-1)\}$

```
# 检验统计量
t0 <- (mean(x) - 1) / sqrt(var(x) / n)
# 临界值
qt(p = 1 - 0.05, df = n - 1)
```

```
#> [1] 1.729133
```

```
# P 值
1 - pt(q = t0, df = n - 1)
```

```
#> [1] 0.01569596
```

> **注释**
>
> 英国统计学家 William Sealy Gosset (1876-1937) 于 1908 年在杂志《Biometrics》上以笔名 Student 发表论文《The Probable Error of a Mean》(”Student” 1908)，论文中展示了独立同正态分布的样本 $x_1, \ldots, x_n \overset{i.i.d}{\sim} \mathcal{N}(\mu, \sigma^2)$ 的样本方差 s^2 和样本标准差 s 的抽样分布，根据均值和标准差不相关的性质导出 t 分布，宣告 t 分布的诞生，因其在小样本领域的突出贡献，W. S. Gosset 进入世纪

名人录 (Heyde 等 2001)。

表格 12.1: t 分布的分位数表

	0.75	0.8	0.9	0.95	0.975	0.99	0.995	0.999
1	1.0000	1.3764	3.0777	6.3138	12.7062	31.8205	63.6567	318.3088
2	0.8165	1.0607	1.8856	2.9200	4.3027	6.9646	9.9248	22.3271
3	0.7649	0.9785	1.6377	2.3534	3.1824	4.5407	5.8409	10.2145
4	0.7407	0.9410	1.5332	2.1318	2.7764	3.7469	4.6041	7.1732
5	0.7267	0.9195	1.4759	2.0150	2.5706	3.3649	4.0321	5.8934
6	0.7176	0.9057	1.4398	1.9432	2.4469	3.1427	3.7074	5.2076
7	0.7111	0.8960	1.4149	1.8946	2.3646	2.9980	3.4995	4.7853
8	0.7064	0.8889	1.3968	1.8595	2.3060	2.8965	3.3554	4.5008
9	0.7027	0.8834	1.3830	1.8331	2.2622	2.8214	3.2498	4.2968
10	0.6998	0.8791	1.3722	1.8125	2.2281	2.7638	3.1693	4.1437

12.1.2 正态总体方差检验

卡方检验 χ^2 检验统计量服从卡方分布。

$$\begin{aligned} \text{I} \quad & H_0 : \sigma^2 - \sigma_0^2 \leq 0 \quad vs. \quad H_1 : \sigma^2 - \sigma_0^2 > 0 \\ \text{II} \quad & H_0 : \sigma^2 - \sigma_0^2 \geq 0 \quad vs. \quad H_1 : \sigma^2 - \sigma_0^2 < 0 \\ \text{III} \quad & H_0 : \sigma^2 - \sigma_0^2 = 0 \quad vs. \quad H_1 : \sigma^2 - \sigma_0^2 \neq 0 \end{aligned}$$

一般假定均值 μ 是未知的。检验统计量服从卡方分布 $\chi^2(n-1)$

$$\chi^2 = \frac{(n-1)s^2}{\sigma_0^2}$$

设 $\sigma_0^2 = 1.5^2$ ，考虑检验问题 I

```
# 检验统计量
chi <- (n - 1) * var(x) / 1.5^2
# 临界值
qchisq(p = 1 - 0.05, df = n -1)
```

```
#> [1] 30.14353
```

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

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

R 软件提供很多统计分布的计算，因此，不再需要查分位数表，现算即可。计算自由度为 n 概率为 p 的 χ^2 分布的分位数 $\chi^2_p(n)$ ，即

$$P(\chi^2(n) \leq \chi^2_p(n)) = p$$

若已知自由度为 1 ，概率为 0.05，则可借助分位数函数 `qchisq()` 计算对应的（下）分位点。

```
qchisq(p = 0.05, df = 1)
```

```
#> [1] 0.00393214
```

同理，也可以获得 χ^2 分布的分位数表格 12.2 ，计算出来的分位数保留 4 位小数。

表格 12.2: χ^2 分布的分位数表

	0.005	0.01	0.025	0.05	0.1	0.9	0.95	0.975	0.99	0.995
1	0.0000	0.0002	0.0010	0.0039	0.0158	2.7055	3.8415	5.0239	6.6349	7.8794
2	0.0100	0.0201	0.0506	0.1026	0.2107	4.6052	5.9915	7.3778	9.2103	10.5966
3	0.0717	0.1148	0.2158	0.3518	0.5844	6.2514	7.8147	9.3484	11.3449	12.8382
4	0.2070	0.2971	0.4844	0.7107	1.0636	7.7794	9.4877	11.1433	13.2767	14.8603
5	0.4117	0.5543	0.8312	1.1455	1.6103	9.2364	11.0705	12.8325	15.0863	16.7496
6	0.6757	0.8721	1.2373	1.6354	2.2041	10.6446	12.5916	14.4494	16.8119	18.5476
7	0.9893	1.2390	1.6899	2.1673	2.8331	12.0170	14.0671	16.0128	18.4753	20.2777
8	1.3444	1.6465	2.1797	2.7326	3.4895	13.3616	15.5073	17.5345	20.0902	21.9550
9	1.7349	2.0879	2.7004	3.3251	4.1682	14.6837	16.9190	19.0228	21.6660	23.5894
10	2.1559	2.5582	3.2470	3.9403	4.8652	15.9872	18.3070	20.4832	23.2093	25.1882

12.1.3 总体未知均值检验

有了均值和方差，为什么还要位置参数和尺度参数？为了更一般地描述问题，扩展范围。特别是在总体分布未知或知之甚少的情况下做检验，不再仅限于均值和方差这样的特征量。

考虑前面正态总体均值检验中的假设 I 的形式，若总体的分布形式未知，则需要 Wilcoxon （威尔科克森）秩和检验 `wilcox.test()` 来做均值的比较。

```
wilcox.test(x = x, mu = 1, alternative = "greater")
```

```
#>
#>  Wilcoxon signed rank exact test
#>
#> data:  x
#> V = 163, p-value = 0.01479
#> alternative hypothesis: true location is greater than 1
```

相比于 t 检验，P 值更小。

12.1.4 总体未知方差检验

12.2 两样本检验

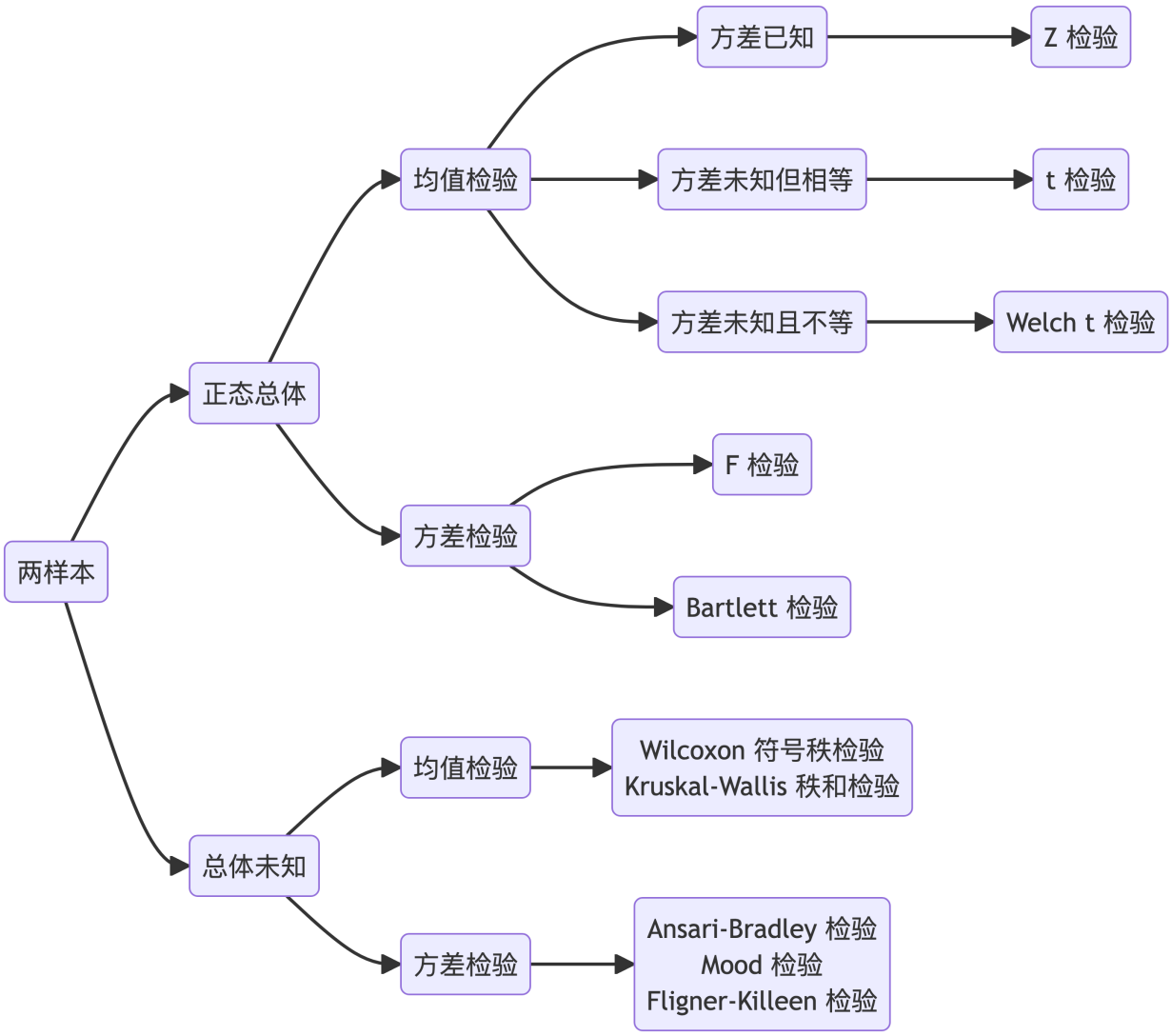


图 12.2: 两样本检验

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。

12.2.1 正态总体均值检验

两样本均值之差的检验

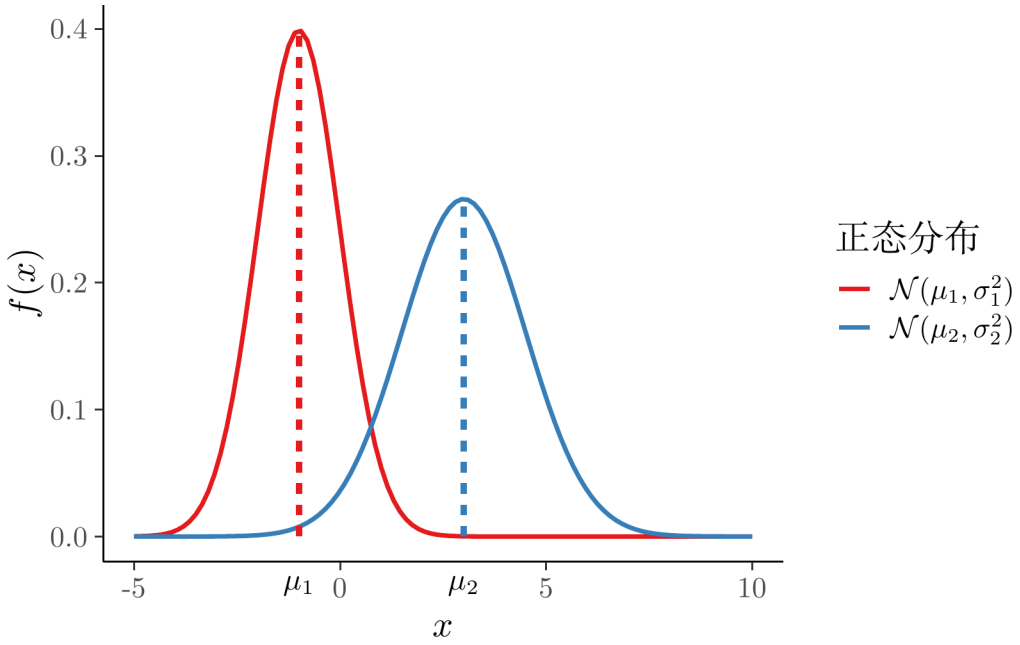


图 12.3: 两样本均值之差的检验

常见检验问题

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

12.2.1.1 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
```

```r
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 $\mu_1 - \mu_2 > 0$。

```r
# 计算拒绝域
qnorm(1 - 0.05)
```

```
#> [1] 1.644854
```

```r
# 计算 P 值
1 - pnorm(u0)
```

```
#> [1] 6.048939e-12
```

12.2.1.2 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，

$$\bar{x} = \sum_{i=1}^{n_1} x_i \quad \bar{y} = \sum_{i=1}^{n_2} y_i$$

$$s_0^2 = \frac{1}{n_1 + n_2 - 2} \left(\sum_{i=1}^{n_1} (x_i - \bar{x})^2 + \sum_{i=1}^{n_2} (y_i - \bar{y})^2 \right)$$

```r
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
```

12.2.1.3 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

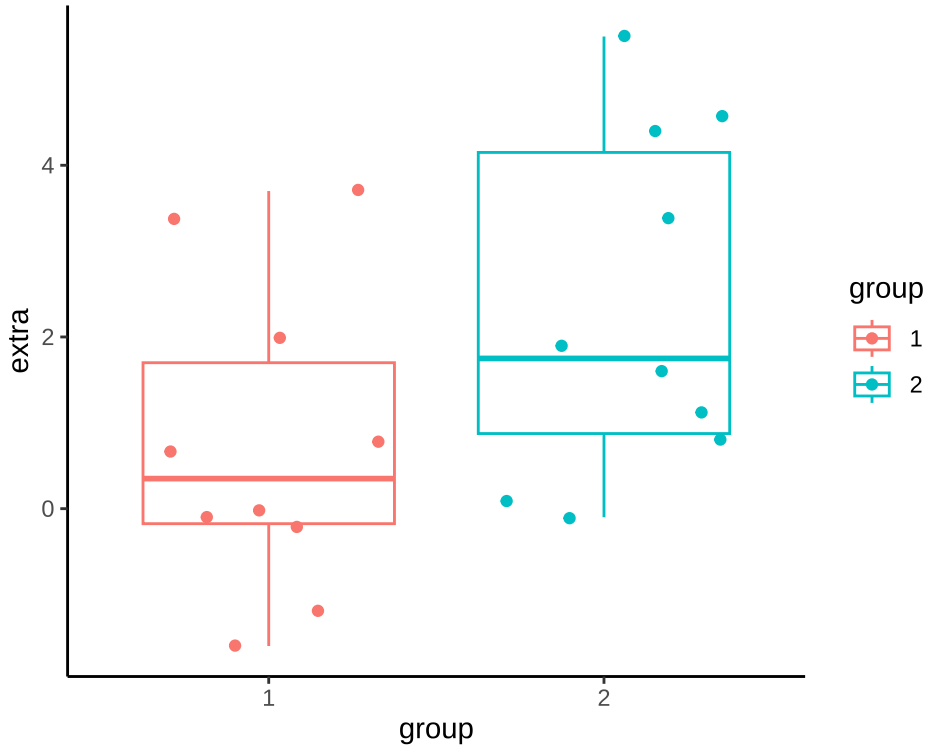


图 12.4: 学生睡眠数据的分布

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

```
#>
```

```
#>  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。许宝𫚉（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𫚉的生平事迹和学术成就，见《许宝𫚉先生的生平和学术成就》。钟开莱（Kai-Lai Chung）将许宝𫚉的论文集整理出版 (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 12.5 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。

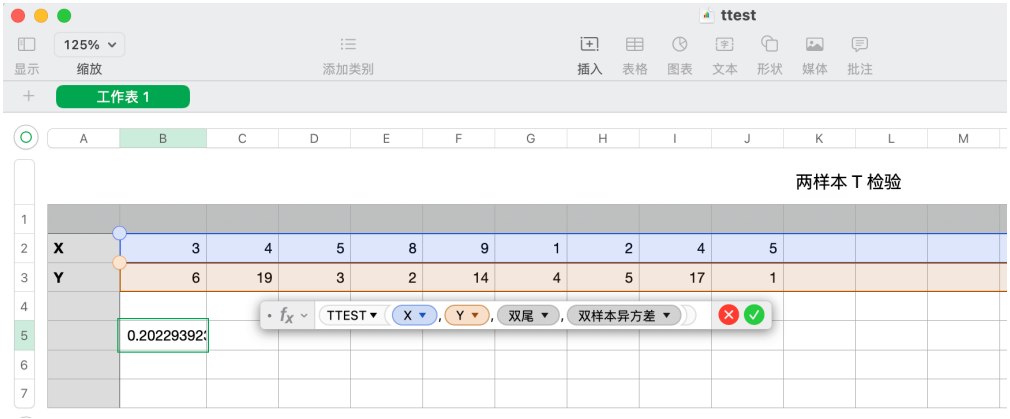


图 12.5: 办公软件 Numbers 的两样本 t 检验

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
```

12.2.2 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。

12.2.3 总体未知均值检验

在总体分布未知的情况下，比较均值是否相等的检验。

- `wilcox.test()` 适用于单样本和两样本的均值检验，单样本 Wilcoxon 秩和检验，两样本 Wilcoxon 符号秩和检验，后者也叫 Mann-Whitney 检验。
- `kruskal.test()` 适用于两样本和多样本，比较多个均值是否相等的检验，Kruskal-Wallis 秩和检验。

单样本和两样本 `wilcox.test()`。

```
wilcox.test(extra ~ group, data = sleep)
```

```
#> Warning in wilcox.test.default(x = DATA[[1L]], y = DATA[[2L]], ...): cannot
#> compute exact p-value with ties
```

```
#>
#>  Wilcoxon rank sum test with continuity correction
#>
#> data:  extra by group
#> W = 25.5, p-value = 0.06933
#> alternative hypothesis: true location shift is not equal to 0
```

coin 包提供渐进 Wilcoxon-Mann-Whitney 检验

```
# Asymptotic Wilcoxon-Mann-Whitney Test
wilcox_test(extra ~ group, data = sleep, conf.int = TRUE)
```

```
#>
#>  Asymptotic Wilcoxon-Mann-Whitney Test
#>
#> data:  extra by group (1, 2)
#> Z = -1.8541, p-value = 0.06372
#> alternative hypothesis: true mu is not equal to 0
#> 95 percent confidence interval:
#>  -3.500000e+00  1.270214e-10
#> sample estimates:
#> difference in location
#>              -1.347344
```

```
# Exact Wilcoxon-Mann-Whitney Test
wilcox_test(
  extra ~ group, data = sleep,
  distribution = "exact", conf.int = TRUE
)
```

```
#>
#>  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。

12.2.4 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 12.3 。

表格 12.3: 检验方法分类

两个样本	多个样本
• Ansari-Bradley 检验 `ansari.test()` • Mood 检验 `mood.test()`	• Fligner-Killeen 检验 `fligner.test()`

以 A. R. Ansari 和 R. A. Bradley 命名的 Ansari-Bradley 检验 (Ansari 和 Bradley 1960)，对应的 R 函数是 `ansari.test()` ，以 A. M. Mood 命名的 Mood 检验 (Mood 1954)，对应的 R 函数是 `mood.test()` ，这两者都属于两样本的非参数检验，检验尺度参数是否相同 (齐性)。以 M. A. Fligner 和 T. J. Killeen 命名的 Fligner-Killeen 检验 (Fligner 和 Killeen 1976)，对应的 R 函数是 `fligner.test()` ，也属于非参数检验，适用于两样本和多样本的情况。非参数检验常涉及位置参数和尺度参数这一对概念，就正态分布而言，位置参数可以理解为均值 μ ，尺度参数可以理解为方差 σ^2 。

```
ansari.test(extra ~ group, data = sleep)
```

```
#> 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
```

12.3 多样本检验

本节考虑 Base R 内置的 PlantGrowth 数据集，它收集自 Annette J. Dobson 所著书籍《An Introduction to Statistical Modelling》(Dobson 1983) 第 2 章第 2 节的案例 — 研究植物在两种不同试验条件下的生长情况，植物通过光合作用吸收土壤的养分和空气中的二氧化碳，完成积累，故以植物的干重来刻画植物的生长情况，首先将几乎相同的种子随机地分配到实验组和对照组，基于完全随机实验设计 (completely randomized experimental design)，经过预定的时间后，将植物收割，干燥并称重。

```
str(PlantGrowth)
```

```
#> 'data.frame':    30 obs. of  2 variables:
#>  $ weight: num  4.17 5.58 5.18 6.11 4.5 4.61 5.17 4.53 5.33 5.14 ...
#>  $ group : Factor w/ 3 levels "ctrl","trt1",..: 1 1 1 1 1 1 1 1 1 1 ...
```

设立对照组（控制组）ctrl 和实验组 trt1 和 trt2，比较不同的处理方式对植物干重的影响

```
summary(PlantGrowth)
```

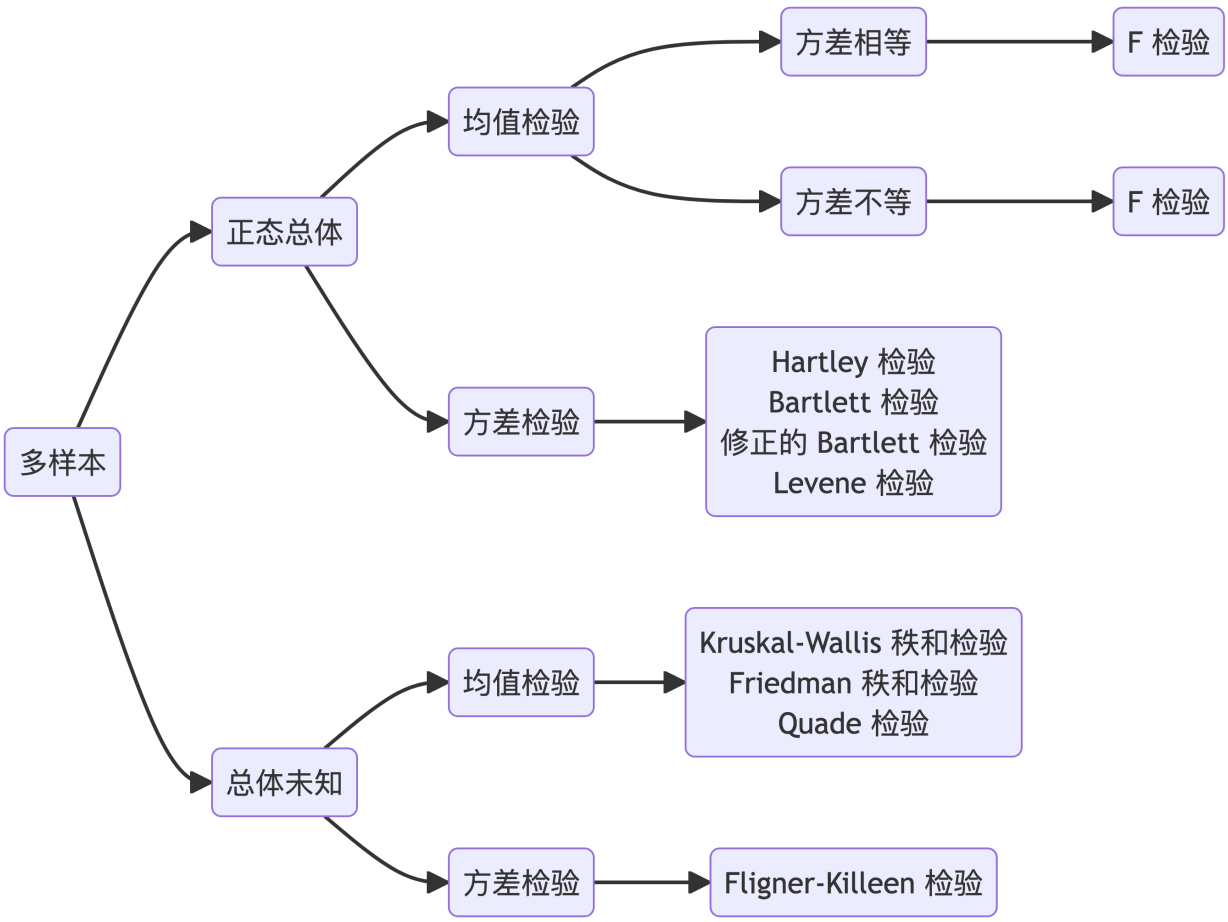


图 12.6: 多样本检验

```
#>      weight       group
#>  Min.   :3.590   ctrl:10
#>  1st Qu.:4.550   trt1:10
#>  Median :5.155   trt2:10
#>  Mean   :5.073
#>  3rd Qu.:5.530
#>  Max.   :6.310
```

每个组都有 10 颗植物，生长情况如图 12.7 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()
```

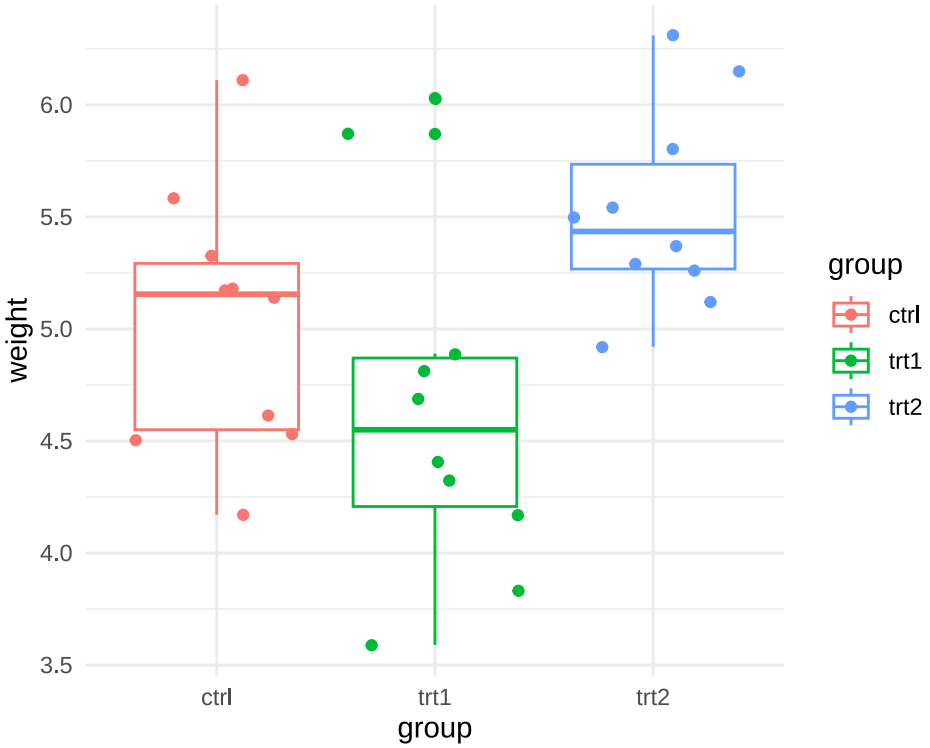


图 12.7: 植物干重

12.3.1 正态总体均值检验

12.3.1.1 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 12.4

表格 12.4: 线性回归的输出

	估计值	标准差	t 统计量	P 值
α	5.032	0.1971	25.5265	0.0000
β_1	-0.371	0.2788	-1.3308	0.1944
β_2	0.494	0.2788	1.7720	0.0877

12.3.1.2 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)
```

12.3.2 正态总体方差检验

总体服从正态分布，有四种常见的参数检验方法：

1. Hartley 检验：各组样本量必须相等。
2. Bartlett 检验：各组样本量可以相等或不等，但每个组的样本量必须不低于 5。
3. 修正的 Bartlett 检验：在样本量较大或较小、相等或不等场合都可使用。
4. Levene 检验：相当于单因素组间方差分析，相比于 Bartlett 检验，Levene 检验更加稳健。

> 提示
>
> 在总体分布未知的情况下，检验方差齐性的非参数方法也都可以用在这里。

设 $x_1,\cdots,x_{n_1}$ 是来自总体 $\mathcal{N}(\mu_1,\sigma_1^2)$ 的样本，设 $y_1,\cdots,y_{n_2}$ 是来自总体 $\mathcal{N}(\mu_2,\sigma_2^2)$ 的样本，设 $z_1,\cdots,z_{n_3}$ 是来自总体 $\mathcal{N}(\mu_3,\sigma_3^2)$ 的样本。

$$\sigma_1^2=\sigma_2^2=\sigma_3^2 \quad vs. \quad \sigma_1^2,\sigma_2^2,\sigma_3^2 \quad \text{不全相等}$$

Bartlett （巴特利特）检验 `bartlett.test()` 要求总体的分布为正态分布，检验各个组的方差是否有显著性差异，即方差齐性检验，属于参数检验，适用于多个样本的情况。

```
# 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
```

12.3.3 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 。1879 年迈克尔逊光速测量数据记录了五次实验，每次试验测量 20 次光速。数据集 morley 中光速 Speed 已经编码过了，为了展示方便，原始观测速度减去了 299000 (km/sec)。

```
# 光速实验
quade.test(Speed ~ Expt | Run, data = morley)
```

```
#>
#>  Quade test
#>
#> data:  Speed and Expt and Run
#> Quade F = 3.6494, num df = 4, denom df = 76, p-value = 0.008976
```

```
ggplot(data = morley, aes(x = Expt, y = Speed, group = Expt)) +
  geom_boxplot() +
  geom_jitter() +
  theme_minimal() +
  labs(x = "Expt", y = "Speed (km/sec)")
```

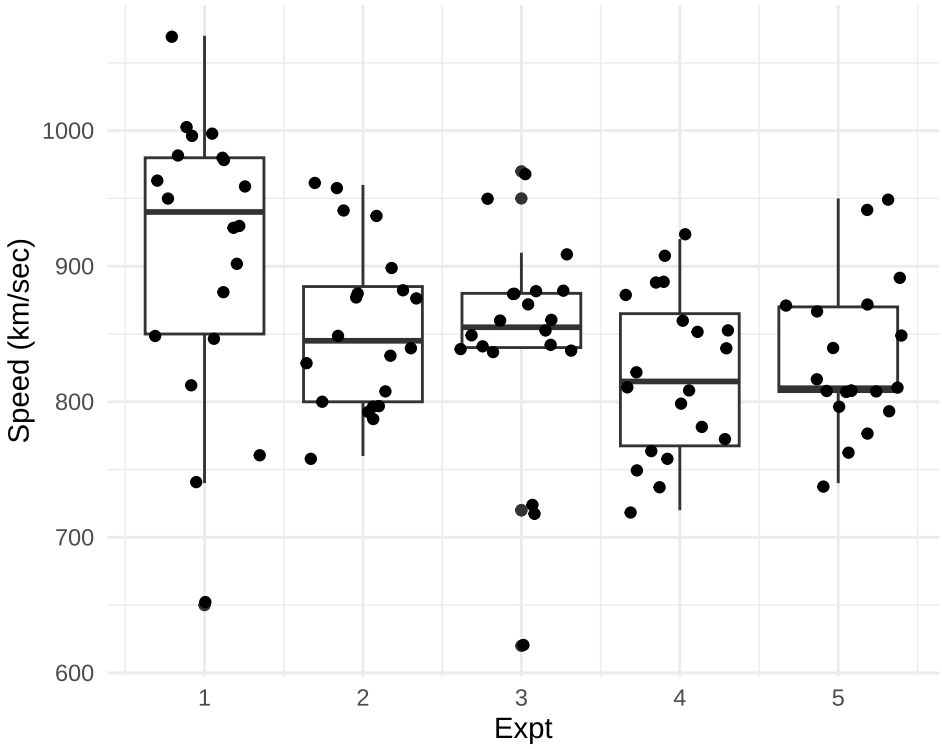


图 12.8: 1879 年迈克尔逊光速实验数据

12.3.4 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。

12.4 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。

表格 12.5: 配对样本检验

样本	R 函数
两样本	• t.test(paired = TRUE) 正态总体均值检验 • wilcox.test(paired = TRUE) 总体未知均值检验

12.4.1 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。

```
sleep2 <- reshape(sleep, direction = "wide",
                  idvar = "ID", timevar = "group")
t.test(Pair(extra.1, extra.2) ~ 1, data = sleep2)
```

```
#>
#>  Paired t-test
#>
#> data:  Pair(extra.1, extra.2)
#> 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
```

```
# R < 4.4.0
# t.test(extra ~ group, data = sleep, paired = TRUE)
```

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 $(0.7001140, 2.459886)$ 。

12.4.2 配对 Wilcoxon 检验

Wilcoxon 检验函数 `wilcox.test()` 设置 `paired = TRUE` 可以做配对检验，但是仅限于两个组。

```
# 不支持三组及以上
# wilcox.test(weight ~ group, data = PlantGrowth, paired = TRUE)
wilcox.test(Pair(extra.1, extra.2) ~ 1, data = sleep2)
```

```
#> Warning in wilcox.test.default(x = respVar[, 1L], y = respVar[, 2L], paired =
#> TRUE, : cannot compute exact p-value with ties
```

```
#> Warning in wilcox.test.default(x = respVar[, 1L], y = respVar[, 2L], paired =
#> TRUE, : cannot compute exact p-value with zeroes
```

```
#>
#>  Wilcoxon signed rank test with continuity correction
#>
#> data:  Pair(extra.1, extra.2)
#> V = 0, p-value = 0.009091
#> alternative hypothesis: true location shift is not equal to 0
```

```
# R < 4.4.0
# wilcox.test(extra ~ group, data = sleep, paired = TRUE)
```

12.5 多重假设检验

同时检验多个统计假设。

表格 12.6: 多重假设检验

样本	R 函数
多样本	• `pairwise.t.test()` 正态总体均值检验 • `pairwise.prop.test()` 二项总体比例检验 • `pairwise.wilcox.test()` 总体未知均值检验

12.5.1 多重 t 检验

数据集 sleep 仅有两个组，数据集 PlantGrowth 包含三个组，下面以数据集 PlantGrowth 为例，介绍做多个组同时进行两两比较的 t 检验。

```
# 样本成对的情况
with(PlantGrowth, pairwise.t.test(x = weight, g = group, paired = TRUE))
```

```
#>
#>  Pairwise comparisons using paired t tests
#>
#> data:  weight and group
#>
#>      ctrl  trt1
#> trt1 0.346 -
#> trt2 0.220 0.058
#>
#> P value adjustment method: holm
```

函数 pairwise.t.test() 以 P 值给出两两配对比较的结果，trt1 和 ctrl 配对比较，P 值为 0.346，trt2 和 ctrl 配对比较，P 值为 0.220，以此类推。

```
# 样本非成对的情况
with(PlantGrowth, pairwise.t.test(x = weight, g = group))
```

```
#>
#>  Pairwise comparisons using t tests with pooled SD
#>
#> data:  weight and group
#>
#>      ctrl  trt1
#> trt1 0.194 -
#> trt2 0.175 0.013
#>
#> P value adjustment method: holm
```

12.5.2 多重比例检验

对于离散数据，做两两比例检验，采用函数 pairwise.prop.test() ，如下示例含有 4 个组。

```
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
```

12.5.3 Wilcoxon 检验

Wilcoxon 检验的是两个总体的均值是否相等。

函数 `pairwise.wilcox.test()` 做两个及以上组的两两比较检验。

```
with(PlantGrowth, pairwise.wilcox.test(x = weight, g = group))
```

```
#> Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot compute
#> exact p-value with ties

#>
#>  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
#>
#> data:  weight and group
#>
#>      ctrl  trt1
#> trt1 0.199 -
#> trt2 0.126 0.027
#>
#> P value adjustment method: holm
```

12.5.4 Dunn 检验

dunn.test 包提供函数 dunn.test() 实现 Dunn 检验，将 Kruskal-Wallis 秩和检验用于两两比较。

```
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
```

12.6 总体分布的检验

前面介绍的检验方法都是对总体的某个特征数（均值、方差）进行检验，下面介绍的检验方法是针对分布的性质。比如样本是否来自正态分布，两个样本是否来自同一分布，样本点之间是否相互独立，样本点列是否平稳等。通过检验方法探索样本的分布性质。

12.6.1 正态性检验

什么样的数据是正态的，理论上是清楚的，对统计建模来说，更实际的问题是什么样的数据是够正态的！探索性数据分析是不断提出假设和验证假设的过程。

> Usually (but not always) doing tests of normality reflect a lack of understanding of the power of rank tests, and an assumption of high power for the tests (qq plots don't always help with

> that because of their subjectivity). When possible it's good to choose a robust method. Also, doing pre-testing for normality can affect the type I error of the overall analysis.
>
> — Frank Harrell [1]

检验：拒绝原假设和接受原假设的风险，数据本身和理论的正态分布的距离，抛开 P 值

Shapiro 和 Wilk 提出的 W 检验 (Shapiro 和 Wilk 1965) ，对应的 R 函数为 `shapiro.test()`

```
set.seed(20232023)
x <- rnorm(100, mean = 5, sd = 3)
shapiro.test(x)
```

```
#>
#>  Shapiro-Wilk normality test
#>
#> data:  x
#> W = 0.98635, p-value = 0.3954
```

> The issue really comes down to the fact that the questions: "exactly normal?", and "normal enough?" are 2 very different questions (with the difference becoming greater with increased sample size) and while the first is the easier to answer, the second is generally the more useful one.
>
> — Greg Snow [2]

EP 检验对多种备择假设有较高的效率，利用样本的特征函数和正态分布的特征函数的差的模的平方产生的一个加权积分得到 EP 检验统计量 (Epps 和 Pulley 1983)

> 提示
>
> 样本量 $n \geq 200$ EP 检验统计量 T_{EP} 非常接近 $n = \infty$ 时 T_{EP} 的分位数。

设 $x_1, \ldots, x_n$ 是来自正态总体 $\mathcal{N}(\mu, \sigma^2)$ 的样本，EP 检验统计量定义为

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

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

12.6.2 同分布检验

Lilliefors 检验 [3] 和单样本的 ks 检验的关系

[1] https://stat.ethz.ch/pipermail/r-help/2005-April/070508.html
[2] https://stat.ethz.ch/pipermail/r-help/2009-May/390164.html
[3] https://personal.utdallas.edu/~herve/Abdi-Lillie2007-pretty.pdf

> 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
```

12.6.3 相关性检验

样本的相关性检验 `cor.test()`：Pearson's 相关系数检验，Kendall's τ 检验或者 Spearman's ρ 检验。基于美国高等法院律师对州法官的评级数据集 USJudgeRatings 介绍各项评分之间的相关性。

```
# cor.test(method = "pearson")  # lm(y ~ 1 + x)
cor.test(~ CONT + INTG, data = USJudgeRatings)
```

```
#>
#>  Pearson's product-moment correlation
#>
#> data:  CONT and INTG
#> t = -0.8605, df = 41, p-value = 0.3945
#> alternative hypothesis: true correlation is not equal to 0
#> 95 percent confidence interval:
#>  -0.4168591  0.1741182
#> sample estimates:
#>        cor
#> -0.1331909
```

其中，变量 CONT 表示律师与法官的联系次数，INTG 表示司法公正。

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

[4] https://stat.ethz.ch/pipermail/r-help/2004-February/045597.html

12.6.4 独立性检验

时间序列独立性检验 `Box.test()` 计算 Box-Pierce 或 Ljung-Box 检验统计量来检查给定时间序列的独立性假设。

12.6.5 平稳性检验

时间序列单位根检验，检验时间序列平稳性 Phillips-Perron 的单位根检验 `PP.test()`

```
PP.test(x, lshort = TRUE)
```

12.7 多元分布情形

- Hotelling T^2 检验：总体服从多元正态分布，两样本均值之差的检验。
- Mauchly 球形检验：总体服从多元正态分布，单样本协方差矩阵的检验。

12.7.1 Hotelling T^2 检验

Hotelling T^2 检验是一维情形下两样本 t 检验的多维推广。

12.7.2 Mauchly 球形检验

Mauchly 球形检验 `mauchly.test()` 检验：Wishart 分布的协方差矩阵是否正比于给定的矩阵。一组样本来自多元正态分布，样本的协方差矩阵是关于样本的随机矩阵，随机矩阵的分布服从 Wishart 分布。

如果 $\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_m}$，$\boldsymbol{x_i} \in \mathbb{R}^p$，$\boldsymbol{x_i} \overset{i.i.d}{\sim} \mathrm{MVN}(0, \Sigma)$，即 m 个样本点都服从均值为 0，协方差矩阵为 Σ 的 p 维多元正态分布 $\mathrm{MVN}(0, \Sigma)$，且样本点之间相互独立。则 $X = \boldsymbol{x}^\top \boldsymbol{x}$ 服从参数为 Σ，自由度为 m 的 Wishart 分布 $W_p(\Sigma, m)$。概率密度函数如下

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

其中，Γ_p 是多元伽马函数，定义如下

$$\Gamma_p(\frac{m}{2}) = \pi^{p(p-1)/4} \prod_{j=1}^{p} \Gamma(\frac{m}{2} - \frac{j-1}{2})$$

R 语言内置了一个模拟数生成器，可以直接模拟出服从 Wishart 分布 $W_p(\Sigma, m)$ 的样本，$m = \mathrm{df}, \Sigma = \mathrm{Sigma}$。R 语言命令如下：

```
rWishart(n, df, Sigma)
```

其中，整型参数 n 指定样本量，数值参数 df 指定自由度，正定的 $p \times p$ 矩阵 Sigma 指定 Wishart 分布的矩阵参数。rWishart() 返回一个 $p \times p \times n$ 数组 R，其中 $R[,,i]$ 是正定矩阵，是服从 Wishart 分布 $W_p(\Sigma, m)$ 的一个样本点，其中 $m = \text{df}, \Sigma = \text{Sigma}$。

```
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
```

12.8 假设检验的一些注记

真实数据的情况是复杂多样的，本章按照数据情况对检验方法分类，方便读者根据手头的数据情况，快速从众多的方法中定位最合适的检验方法。依次是单样本检验、两样本检验、多样本检验、计数数据检验、配对样本检验。如果已知符合参数检验的条件，优先考虑参数检验。如果不确定是否符合参数检验的条件，对参数检验和非参数检验方法都适用，非参数检验方法的功效更大，方法更优。在总体分布未知的情况下，无论是对均值检验还是对方差检验，大部分情况下都需要非参数检验方法。

在假设检验理论方面作出贡献的人非常多，自 Karl Pearson 提出卡方统计量、卡方分布和卡方检验以来，陆续涌现出来一批人及载入史册的工作，见下表。不难看出，19 世纪后半叶至 20 世纪前半叶，假设检验理论经过一个世纪的发展趋于成熟。从假设检验这个细分领域也印证了世界的统计中心从英国逐渐转移到美国，相比而言，中国在这方面的贡献微乎其微。笔者同时也注意到很多检验方法都是以人名命名的，且已经被编写到各类统计软件中。R 语言中有十分丰富的统计检验函数，根据这些函数及其帮助文档可以追溯到检验方法的发明者，再从维基百科中找到学者及其提出的检验方法的详情页，最后，根据学者的出生日期排序整理成表格。

表格 12.7: 对假设检验理论有重要贡献的学者

姓名	国籍	出生	死亡	寿命	贡献
K. Pearson	英国	1857-03-27	1936-04-27	79	卡方分布、卡方检验
C. Spearman	英国	1863-09-10	1945-09-17	82	Spearman's ρ

姓名	国籍	出生	死亡	寿命	贡献
W. S. Gosset	英国	1876-06-13	1937-10-16	61	t 分布、t 检验
R. A. Fisher	英国	1890-02-17	1962-07-29	72	F 检验、Fisher 精确检验
F. Wilcoxon	美国	1892-09-02	1965-11-18	73	Wilcoxon 秩检验
H. Cramér	瑞士	1893-09-25	1985-10-05	92	Cramér's V
J. Neyman	波兰、美国	1894-04-16	1981-08-05	87	Neyman-Pearson 引理
E. S. Pearson	英国	1895-08-11	1980-06-12	84	Neyman-Pearson 引理
H. Hotelling	美国	1895-09-29	1973-12-26	78	Hotelling T^2 检验
E. J. G. Pitman	澳大利亚	1897-10-29	1993-07-21	95	Pitman 估计
J. Wishart	英国	1898-11-28	1956-07-14	57	Wishart 分布
Q. M. McNemar	美国	1900-02-20	1986-07-03	86	McNemar 检验
F. Yates	英国	1902-05-12	1994-06-17	92	Yates 矫正
A. Wald	匈牙利	1902-10-31	1950-12-13	48	Wald 检验
A. Kolmogorov	苏联	1903-04-25	1987-10-20	84	Kolmogorov-Smirnov 检验
S. S. Wilks	美国	1906-06-17	1964-03-07	57	Wilks 检验/似然比检验
J. W. Mauchly	美国	1907-08-30	1980-01-08	72	Mauchly 球形检验
M. Kendall	英国	1907-09-06	1983-03-29	76	Kendall's τ
W. G. Cochran	英国、美国	1909-07-15	1980-03-29	70	Cochran–Mantel–Haenszel 检验
M. S. Bartlett	英国	1910-06-18	2002-01-08	91	Bartlett 检验
W. M. Haenszel	美国	1910-06-19	1998-03-13	87	Cochran–Mantel–Haenszel 检验
B. L. Welch	英国	1911	1989-12-29	78	Welch t 检验
H. O. Hartley	德国	1912-04-13	1980-12-30	68	Hartley 检验
M. Friedman	美国	1912-07-31	2006-11-16	94	Friedman 秩和检验
W. A. Wallis	美国	1912-11-05	1998-10-12	85	Kruskal-Wallis 检验
H. Levene	美国	1914-01-17	2003-07-02	89	Levene 检验
J. W. Tukey	美国	1915-06-16	2000-07-26	85	Tukey's HSD 检验
O. J. Dunn	美国	1915-09-01	2008-01-12	92	Dunn 检验
E. L. Lehmann	法国、美国	1917-11-20	2009-09-12	91	Lehmann-Scheffé 定理
T. W. Anderson	美国	1918-06-05	2016-09-17	98	Anderson–Darling 检验
N. Mantel	美国	1919-02-16	2002-05-25	83	Cochran–Mantel–Haenszel 检验
W. Kruskal	美国	1919-10-10	2005-04-21	85	Kruskal-Wallis 检验
George E. P. Box	英国、美国	1919-10-18	2013-03-28	93	Box-Pierce 检验
C. R. Rao	印度、美国	1920-09-10	2023-08-22	102	Score 检验
M. Wilk	加拿大	1922-12-18	2013-02-19	90	Shapiro-Wilk 检验
J. Durbin	英国	1923-06-30	2012-06-23	88	Durbin 检验

姓名	国籍	出生	死亡	寿命	贡献
L. Le Cam	法国	1924-11-18	2000-04-25	75	渐近理论
H. Lilliefors	美国	1928-06-14	2008-02-23	80	Lilliefors 检验
S. S. Shapiro	美国	1930-07-13	-	93	Shapiro-Wilk 检验

> **注释**
>
> 笔者仅根据自己搜集了解的材料制作此表，受一定的局限，或有缺漏和主观。20 世纪 60 年代后，假设检验理论开始成熟起来了，所以仅考虑 1930 年以前出生的。此外，学者需具有一定的名气，至少收录在维基百科词条里。

其中，最重要的统计学家及其学术传承关系见下图 12.9 。

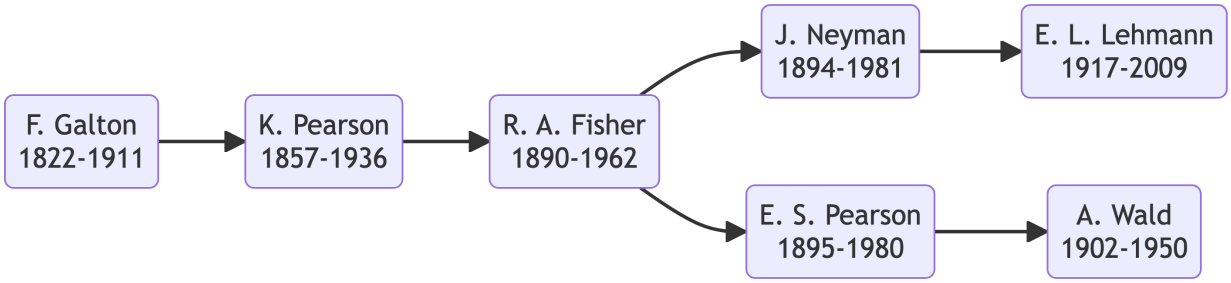


图 12.9: 最重要的统计学家及其学术传承关系

F. Galton 是 K. Pearson 的老师，E. S. Pearson 是 K. Pearson 的儿子。E. L. Lehmann 是 J. Neyman 的学生，J. Neyman 和 E. S. Pearson 一起提出 N-P 引理，是置信区间和假设检验理论的奠基人。假设检验和区间估计、决策理论是紧密相关的，A. Wald 是继 J. Neyman 和 E. S. Pearson 之后，继续开疆拓土的一位统计学家，不幸的是，在一场飞机事故中英年早逝。

12.8.1 假设检验和多重比较的关系

FDR 是 False Discovery Rate 的简称

12.8.2 假设检验和方差分析的关系

12.8.2.1 单因素一元方差分析

函数 `aov()` 可以做单、双因素一元方差分析

```
fit_aov <- aov(weight ~ group, data = PlantGrowth)
```

两两比较，多重比较

```
TukeyHSD(fit_aov)
```

```
#>   Tukey multiple comparisons of means
#>     95% family-wise confidence level
#>
#> Fit: aov(formula = weight ~ group, data = PlantGrowth)
#>
#> $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。

12.8.2.2 双因素一元方差分析

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

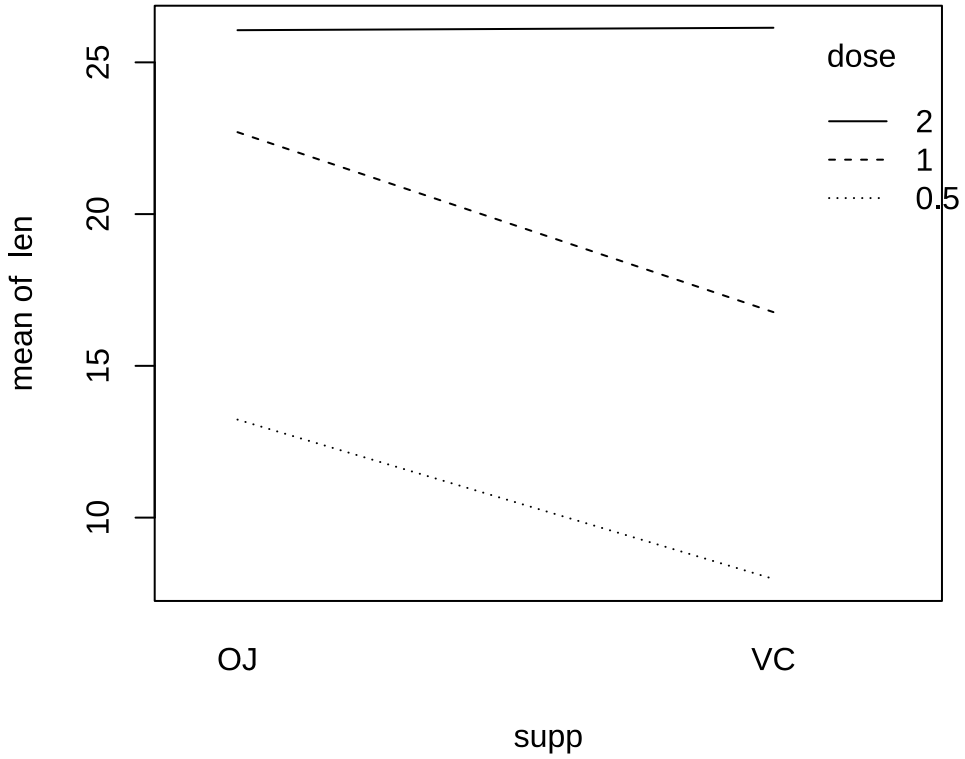


图 12.10: OJ 和 VC 的交互作用

如果 `dose = 2`，则 `len` 与提供的方式 `supp` 没有关系。

```
fit_aov <- aov(len ~ supp * dose, data = ToothGrowth)
fit_aov
```

```
#> Call:
#>    aov(formula = len ~ supp * dose, data = ToothGrowth)
#>
#> Terms:
#>                       supp      dose supp:dose Residuals
```

```
#> Sum of Squares   205.3500 2224.3043   88.9201  933.6349
#> Deg. of Freedom         1         1         1        56
#>
#> Residual standard error: 4.083142
#> Estimated effects may be unbalanced
```

12.8.2.3 单因素多元方差分析

PlantGrowth 属于一元方差分析，观测变量只有植物干重一个变量。如果推广到多个变量，就是多元方差分析 multivariate analysis of variance 。不同种类的鸢尾花的萼片长度的分布有所不同。

```
library(ggplot2)
library(ggridges)
ggplot(data = iris, aes(x = Sepal.Length, y = Species, fill = Species)) +
  scale_fill_brewer(palette = "Greys") +
  geom_density_ridges(bandwidth = 0.2) +
  theme_ridges(font_size = 12, font_family = "sans")
```

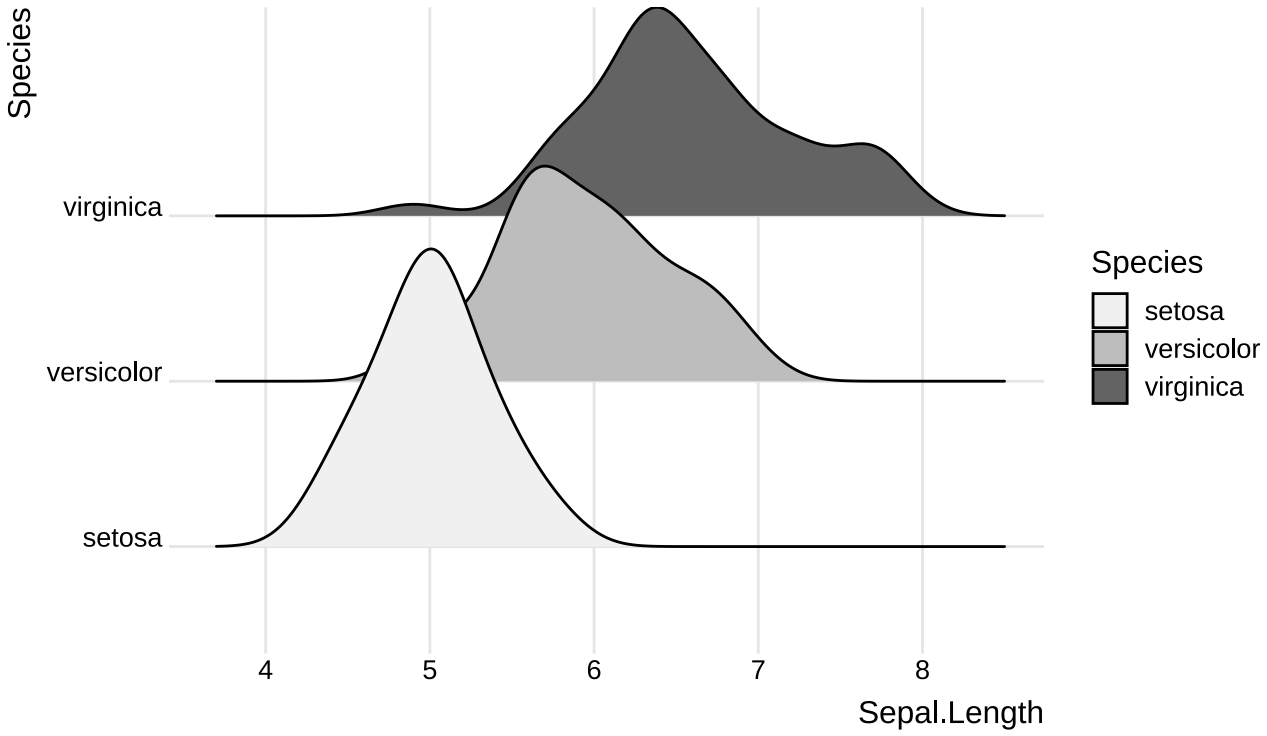


图 12.11: 鸢尾花萼片长度的分布

```
fit <- manova(cbind(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width) ~ Species, data = iris)
summary(fit, test = "Wilks")
```

```
#>            Df    Wilks approx F num Df den Df    Pr(>F)
```

```
#> Species     2 0.023439   199.15      8    288 < 2.2e-16 ***
#> Residuals 147
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

P 值小于 0.0.5，说明 iris 数据集三个组的均值向量有显著差异。关于均值向量的检验方法，请看 `?summary.manova` 。

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
```

12.8.3 假设检验与区间估计的关系

区间估计的意义是解决点估计可靠性问题，它用置信系数解决了对估计结果的信心问题，弥补了点估计的不足。置信系数是最大的置信水平。

Base R 提供的 `binom.test()` 函数可以精确计算置信区间，即所谓的 Clopper-Pearson 区间，而 `prop.test()` 函数可近似计算置信区间，即所谓的 Wilson 区间。以单样本的比例检验为例。

```
# 近似区间估计
prop.test(x = 2, n = 10, p = 0.95, conf.level = 0.95, correct = TRUE)
```

```
#> Warning in prop.test(x = 2, n = 10, p = 0.95, conf.level = 0.95, correct =
#> TRUE): Chi-squared approximation may be incorrect
```

```
#>
#>  1-sample proportions test with continuity correction
#>
#> data:  2 out of 10, null probability 0.95
#> X-squared = 103.16, df = 1, p-value < 2.2e-16
#> alternative hypothesis: true p is not equal to 0.95
#> 95 percent confidence interval:
#>  0.03542694 0.55781858
```

```
#> sample estimates:
#>   p
#> 0.2
```

```
# 精确区间估计
binom.test(x = 2, n = 10, p = 0.95, conf.level = 0.95)
```

```
#>
#>  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。

12.8.4 常见的统计检验是线性模型

两样本的均值检验：非参数检验方法

12.8.4.1 Wilcoxon 符号秩检验

与 `wilcox.test()` 等价的线性模型

```
signed_rank <- function(x) sign(x) * rank(abs(x))
fit <- lm(signed_rank(extra) ~ group, data = sleep)
summary(fit)
```

```
#>
#> Call:
#> lm(formula = signed_rank(extra) ~ group, data = sleep)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -14.55  -6.55   0.90   6.90  13.95
```

```
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)    3.050      2.872   1.062   0.3022
#> group2         8.300      4.061   2.044   0.0559 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 9.081 on 18 degrees of freedom
#> Multiple R-squared:  0.1884, Adjusted R-squared:  0.1433
#> F-statistic: 4.177 on 1 and 18 DF,  p-value: 0.05589
```

12.8.4.2 Kruskal-Wallis 秩和检验

与 kruskal.test() 等价的线性模型表示。

```
fit <- lm(rank(extra) ~ group, data = sleep)
summary(fit)
```

```
#>
#> Call:
#> lm(formula = rank(extra) ~ group, data = sleep)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -8.450 -3.925 -0.500  5.275  8.950
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)    8.050      1.738   4.633 0.000207 ***
#> group2         4.900      2.457   1.994 0.061520 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 5.495 on 18 degrees of freedom
#> Multiple R-squared:  0.1809, Adjusted R-squared:  0.1354
#> F-statistic: 3.976 on 1 and 18 DF,  p-value: 0.06152
```

12.8.5 假设检验的工业应用

传统的试验设计为什么不适用于互联网？因为 Fisher 的实验设计和方差分析，主要针对的是受控对象，比如测试武器、肥料配比、飞机制造等实体的东西。互联网是虚拟经济，实验的对象是人，对平台来说，人的行为是半知半解，更不受控，所以需要成千上万、乃至几十万的样本才能抵消样本内部的随机性。互联网数据的噪声太多、太大了，微小的变化就好像一粒小石子扔进大海里，要获得样本间显著的差异性，需要累积相当的样本量。另一方面，大型的互联网公司，搜索、推荐、广告等业务相对成熟，提升关键指标，拿到好的结果，往往比较困难。成熟的业务几乎不太可能一次实验拿到很好的结果，所以，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。

12.9 习题

1. 分析《红楼梦》的情景描写。参考 2009 年东南大学韦博成教授将两个独立二项总体的等价性检验应用于《红楼梦》前 80 回与后 40 回某些文风差异的统计分析 (韦博成 2009)。

2. 根据数据集 chickwts 分析不同喂食方式对小鸡体重的影响。(单因素方差分析)

```
ggplot(data = chickwts, aes(x = feed, y = weight)) +
  geom_boxplot() +
  geom_jitter() +
  theme_minimal()
```

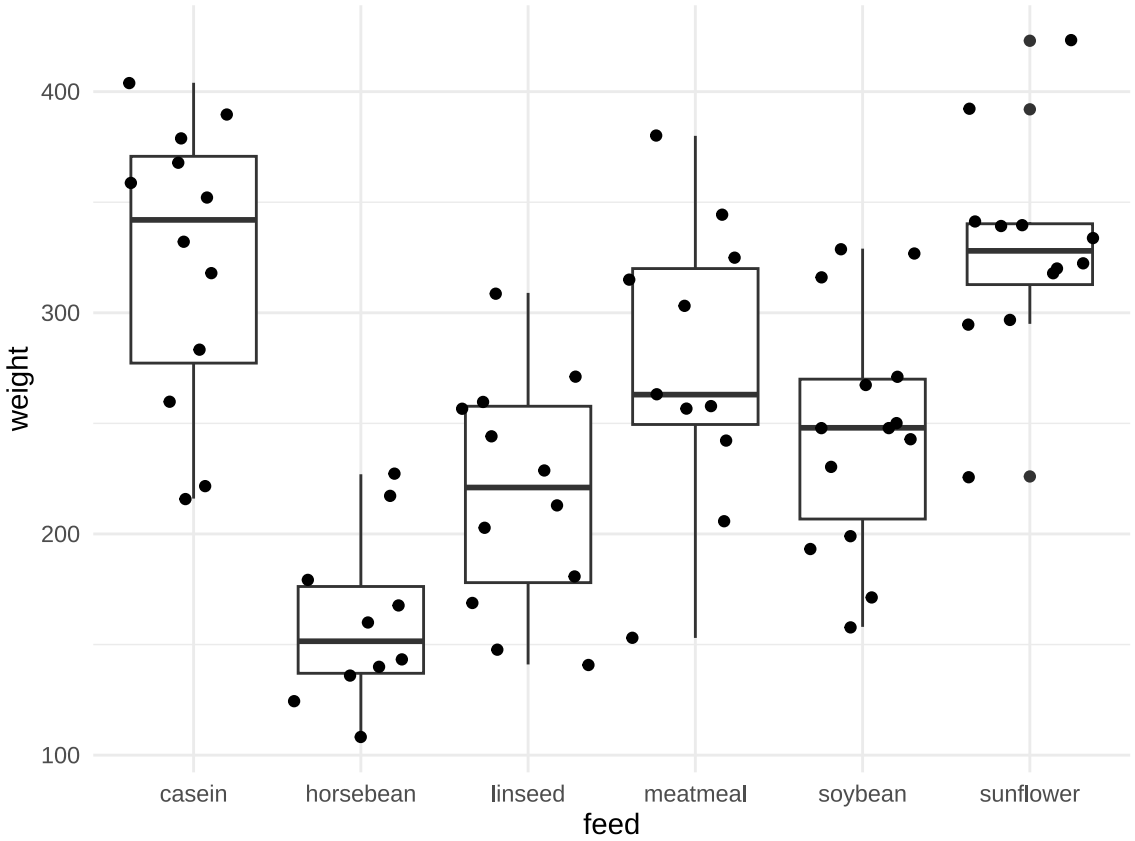


图 12.12: 不同喂食方式对小鸡的影响

3. 根据数据集 ChickWeight 分析 4 种喂食方式对小鸡体重有影响，每个小鸡本身对喂食方式的接受、吸收程度不一样、它们本身的素质不一样（个体差异），要考察喂食的方式的影响，应该剔除掉个体差异，才是喂食方式的真正影响。

```
ggplot(data = ChickWeight, aes(x = Time, y = weight, group = Chick, color = Diet)) +
  geom_point() +
  geom_line() +
  facet_wrap(~Diet) +
  theme_minimal()
```

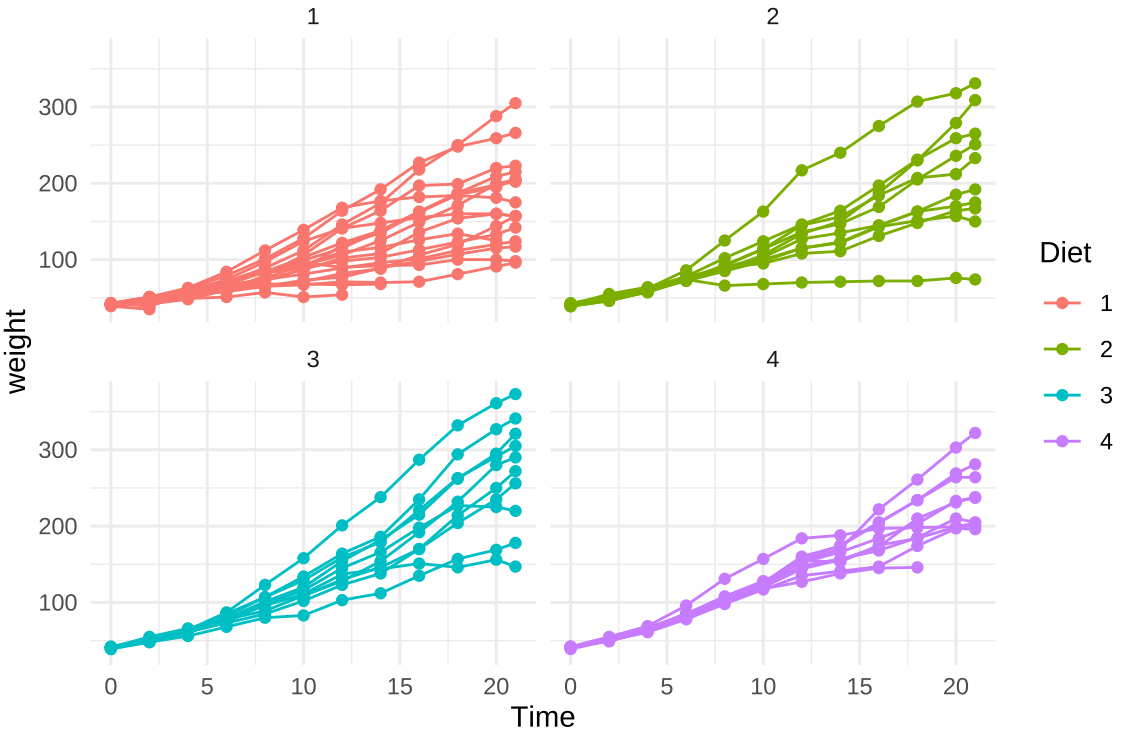


图 12.13: 不同喂食方式对小鸡的影响（续）

第十三章　回归与相关分析

13.1　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 13.1 展示该数据集的部分内容。

表格 13.1: 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)

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

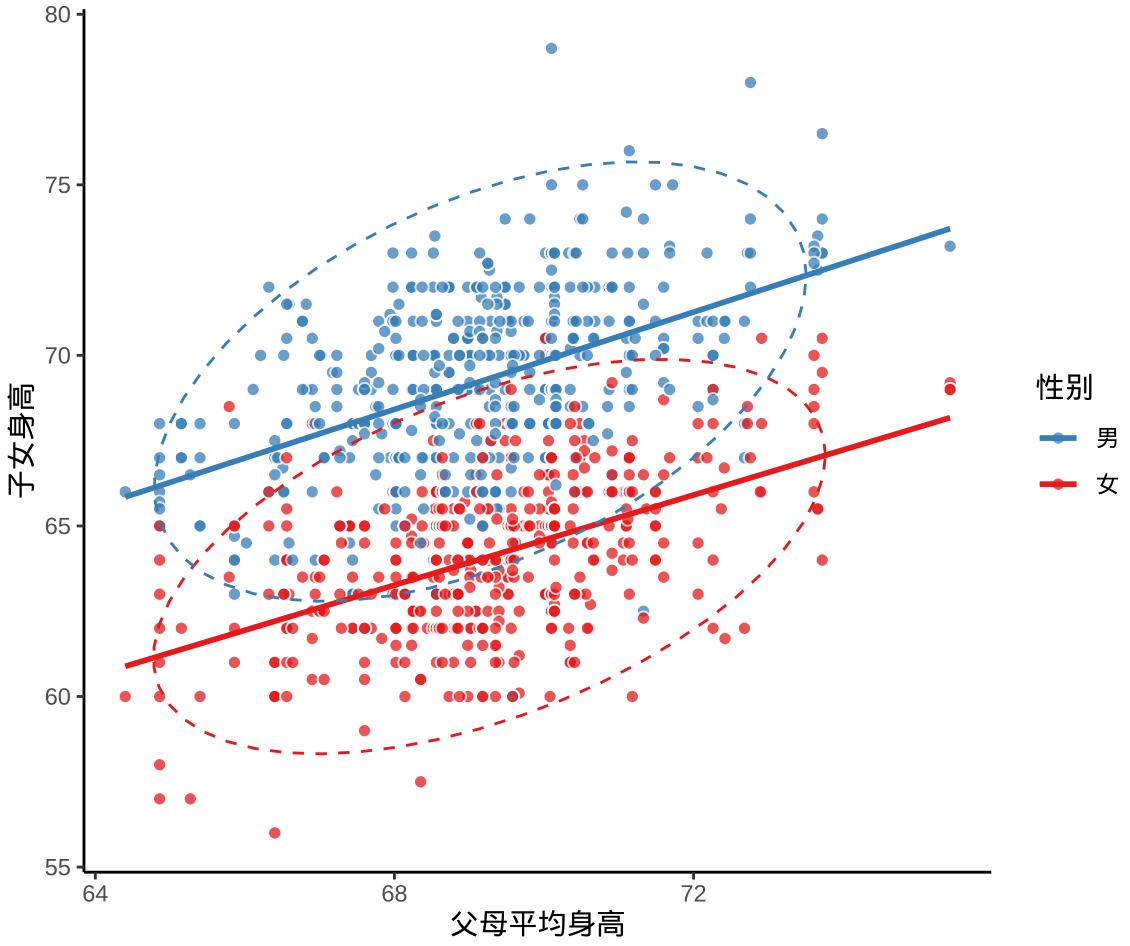


图 13.1: 子代身高与亲代身高的关系

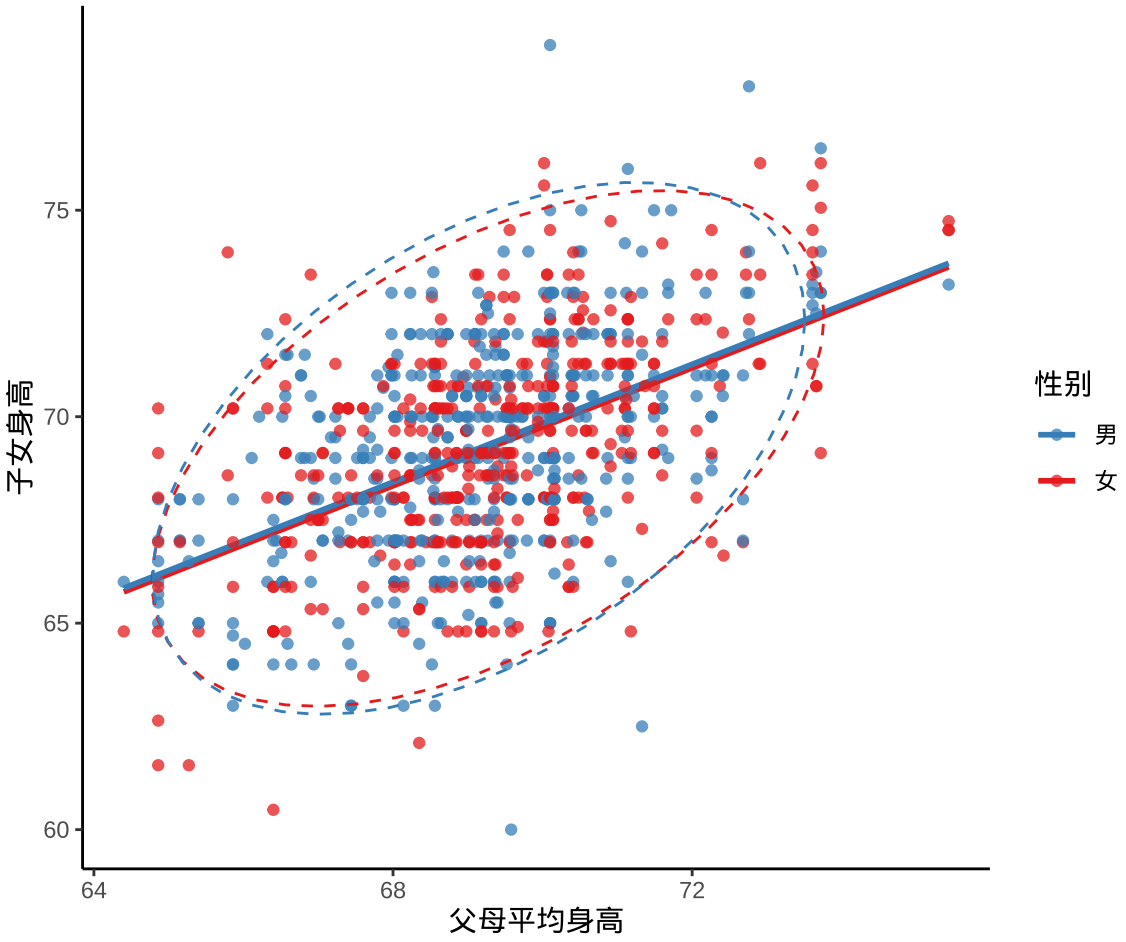


图 13.2: 子代身高与亲代身高的关系

表格 13.2: 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

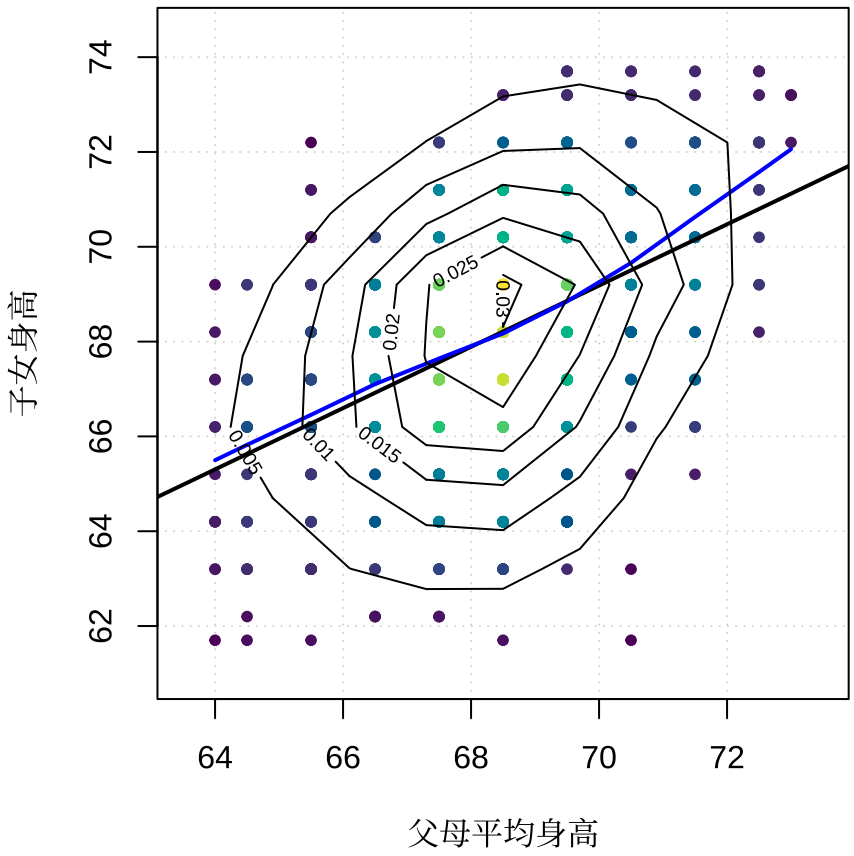


图 13.3: 二维核密度估计与二元正态分布

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。

13.2 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 13.3 展示美国各州的部分统计数据。

表格 13.3: 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 13.4 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 13.4 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 13.5 所示的分类气泡图。

整体来说，预期寿命与人均收入息息相关。

> 提示
>
> 从图 13.5 到图 13.6 ，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

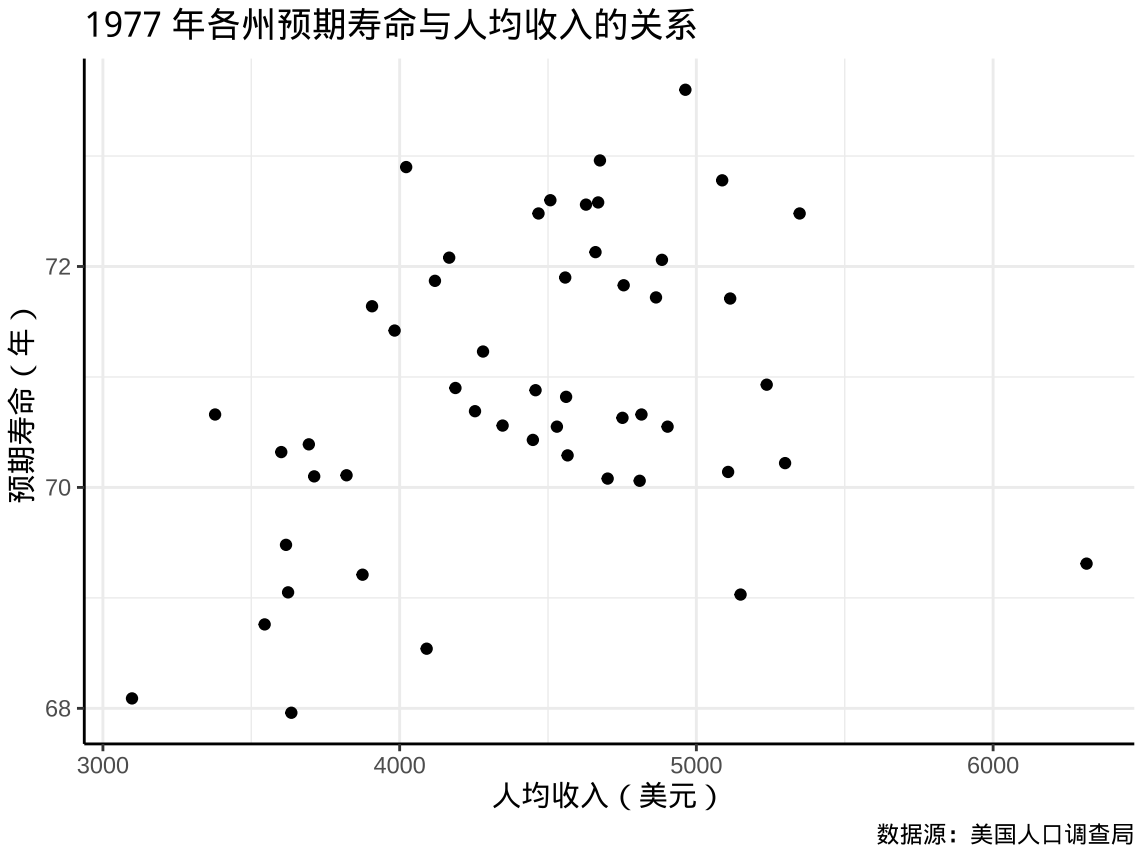


图 13.4: 预期寿命与人均收入的关系图

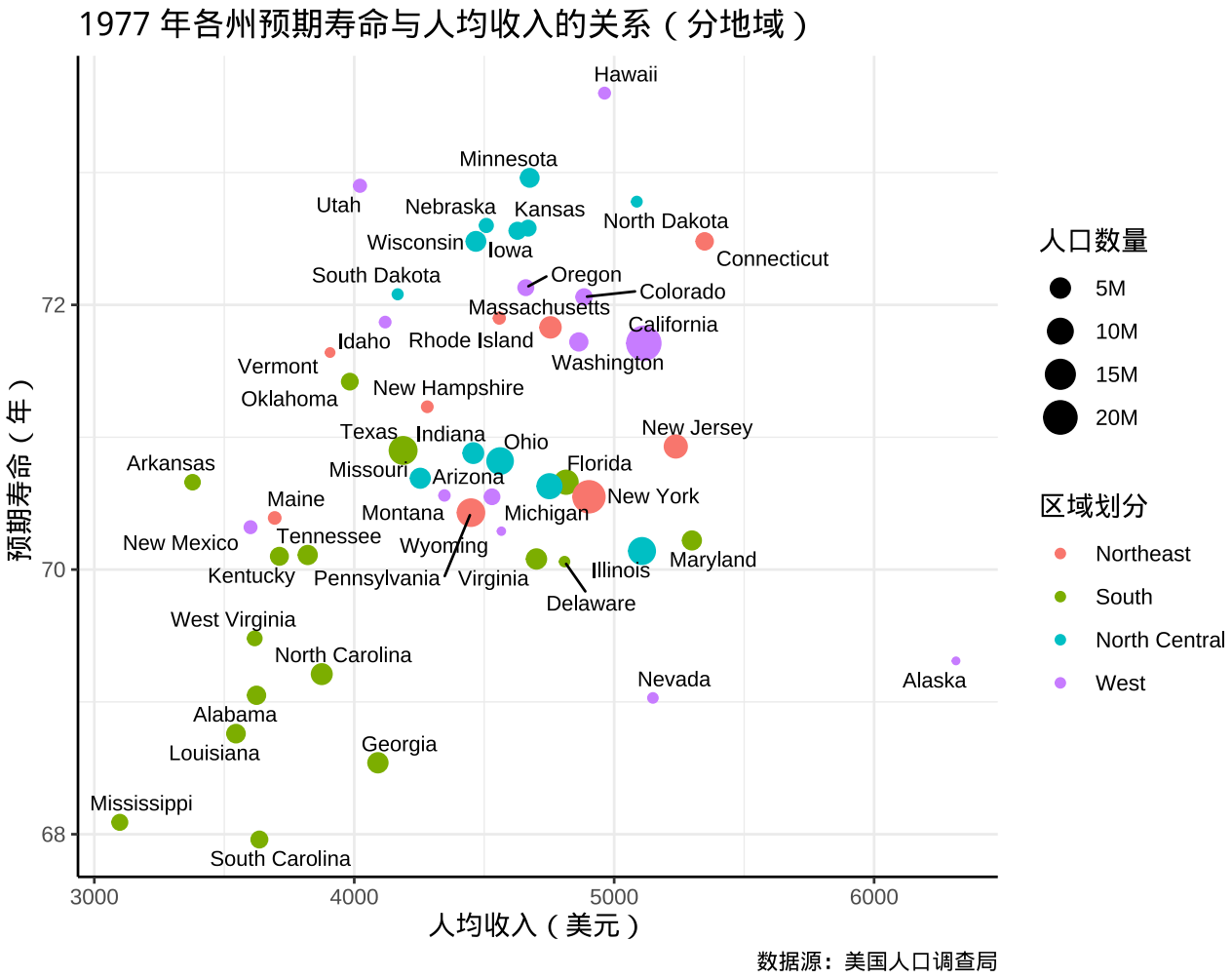


图 13.5: 分地域预期寿命与人均收入的气泡图

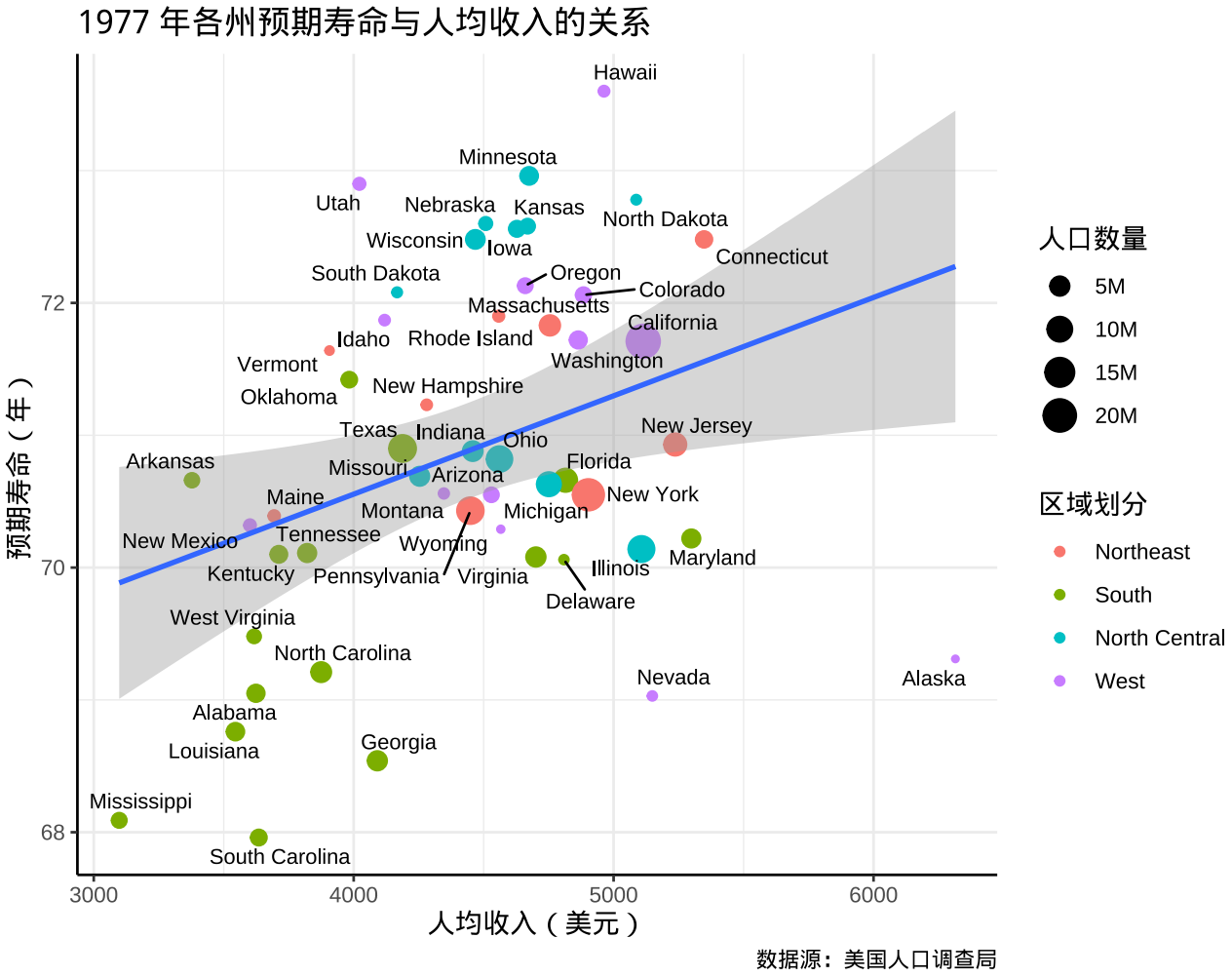


图 13.6: 1977 年美国各州预期寿命与人均收入的关系：回归分析

> 达州的数据偏离集体太远。那又是什么原因导致阿拉斯加州人均收入全美第一，而预期寿命倒数呢？同样的，内华达州的人均收入也不低，但预期寿命为什么上不去呢？

```
m <- lm(data = state_x77, `Life Exp` ~ Income)
summary(m)
```

```
#>
#> Call:
#> lm(formula = `Life Exp` ~ Income, data = state_x77)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -2.96547 -0.76381 -0.03428  0.92876  2.32951
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 6.758e+01  1.328e+00  50.906   <2e-16 ***
#> Income      7.433e-04  2.965e-04   2.507   0.0156 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.275 on 48 degrees of freedom
#> Multiple R-squared:  0.1158, Adjusted R-squared:  0.09735
#> F-statistic: 6.285 on 1 and 48 DF,  p-value: 0.01562
```

输出结果中各个量的计算公式及 R 语言实现，比如方差 Variance、偏差 Deviance/Bias、残差 Residual Error

13.3 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 ...
```

13.4 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。

第十四章　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。

14.1　比例检验

14.1.1　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)。

14.1.1.1　近似检验

14.1.1.2　精确检验

函数 `binom.test()` 来做二项检验，函数 `binom.test()` 用来检验伯努利试验中成功概率 p 和给定概率 p_0 的关系，属于精确检验 (Clopper 和 Pearson 1934)。

比例 p 的检验，做 n 次独立试验，样本 $X_1, \ldots, X_n \sim b(1, p)$，事件发生的总次数 $\sum_{i=1}^{n} X_i$。

```r
# 模拟一组样本
set.seed(20232023)
x <- sample(x = c(0, 1), size = 100, replace = TRUE, prob = c(0.8, 0.2))
```

二项分布中成功概率的检验

```r
binom.test(sum(x), n = 100, p = 0.5)
```

```
#>
#>  Exact binomial test
#>
#> data:  sum(x) and 100
#> number of successes = 23, number of trials = 100, p-value = 5.514e-08
#> alternative hypothesis: true probability of success is not equal to 0.5
#> 95 percent confidence interval:
#>  0.1517316 0.3248587
#> sample estimates:
#> probability of success
#>                   0.23
```

检验成功概率 p 是否等于 0.5，P 值 5.514×10^{-8} 结论是拒绝原假设

```r
binom.test(sum(x), n = 100, p = 0.2)
```

```
#>
#>  Exact binomial test
#>
#> data:  sum(x) and 100
#> number of successes = 23, number of trials = 100, p-value = 0.4534
#> alternative hypothesis: true probability of success is not equal to 0.2
#> 95 percent confidence interval:
#>  0.1517316 0.3248587
#> sample estimates:
#> probability of success
#>                   0.23
```

检验成功概率 p 是否等于 0.2，P 值 0.4534 结论是不能拒绝原假设

切比雪夫不等式（Chebyshev, 1821-1894）。设随机变量 X 的数学期望和方差都存在，则对任意常数 $\epsilon > 0$，有

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

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

14.1.2 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。

14.1.3 多样本检验

14.1.3.1 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。

14.1.3.2 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
```

14.2 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 $\lambda > 0$。

14.2.1 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

14.2.2 两样本

14.3 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
```

14.3.1 行列分组表格

```
# 长格式转宽格式
titanic_data <- reshape(
  data = as.data.frame(Titanic), direction = "wide",
  idvar = c("Class", "Sex", "Age"),
  timevar = "Survived", v.names = "Freq", sep = "_"
)
```

```
# 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 14.1: 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

14.3.2 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")
```

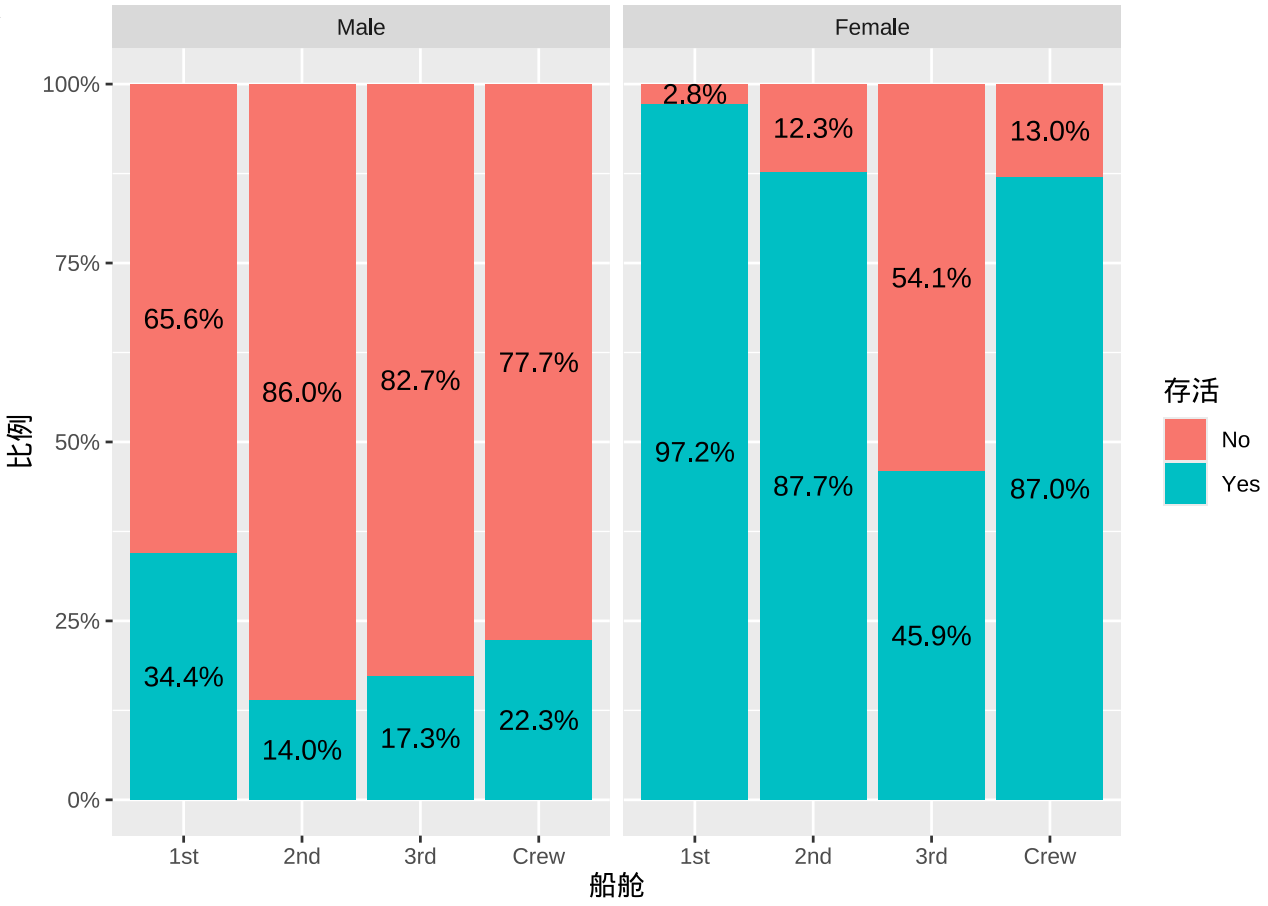


图 14.1: 百分比堆积柱形图展示多维分类数据

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

14.3.3 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"
  )
```

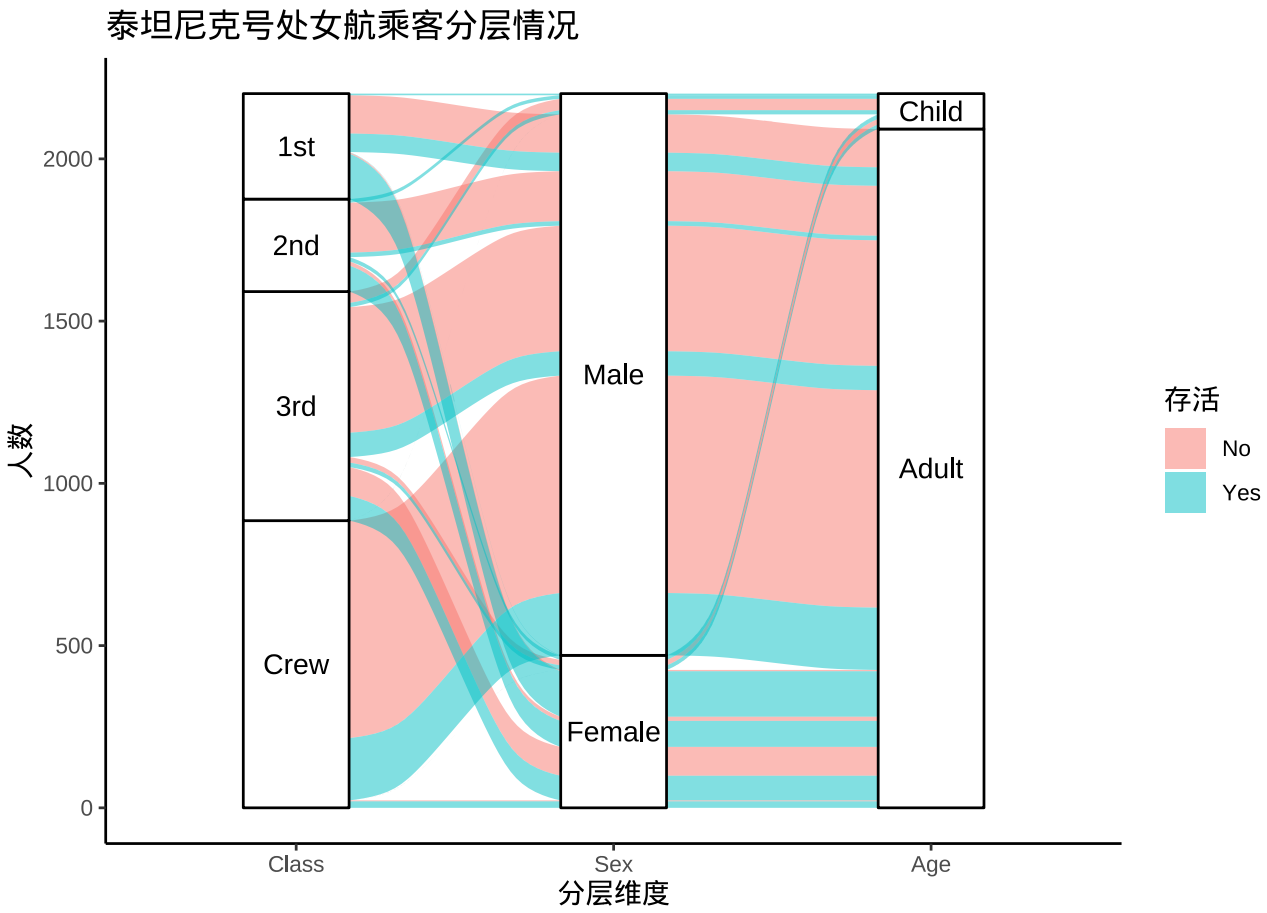


图 14.2: 桑基图展示多维分类数据

14.3.4 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)
```

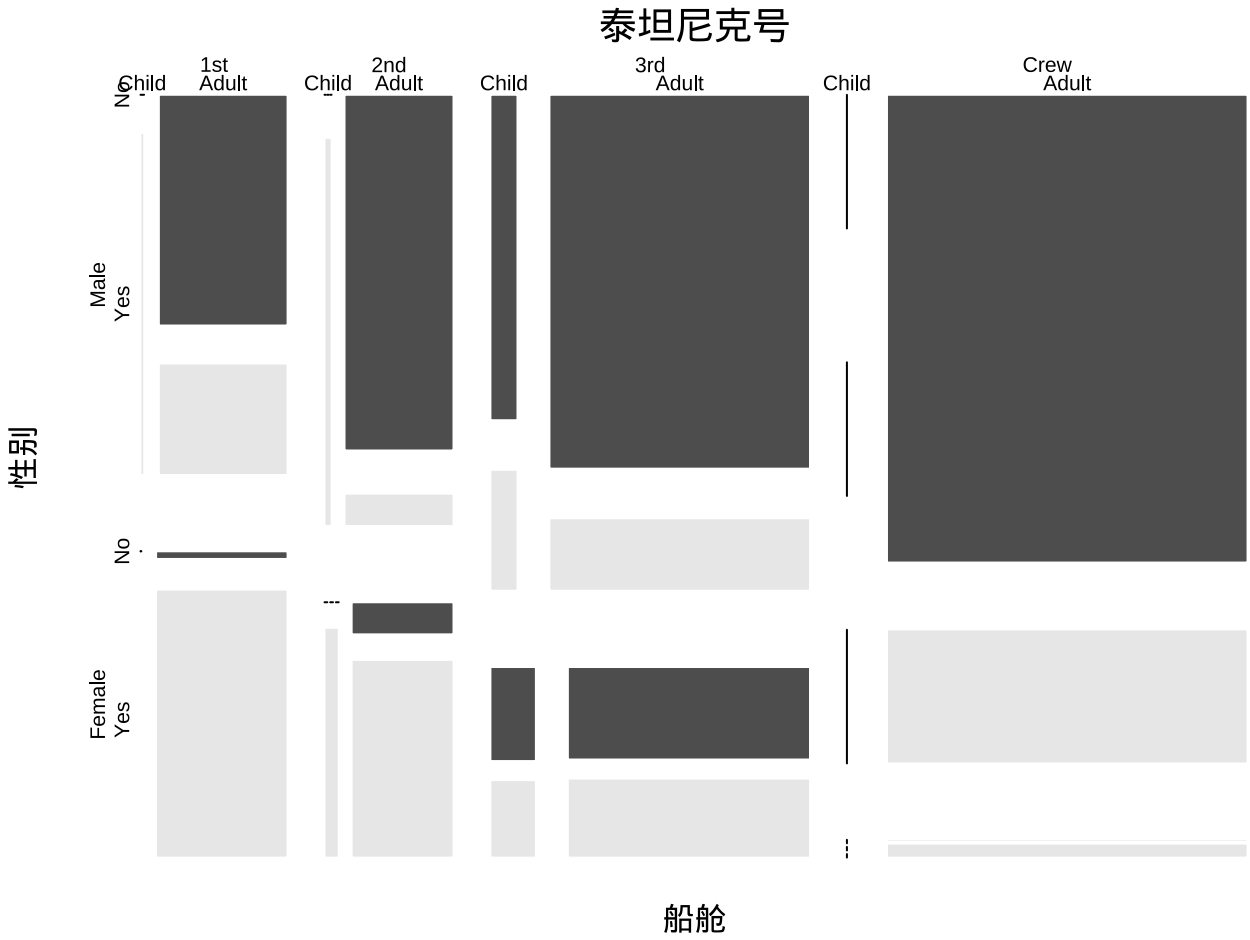


图 14.3: 马赛克图展示多维分类数据

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
)
```

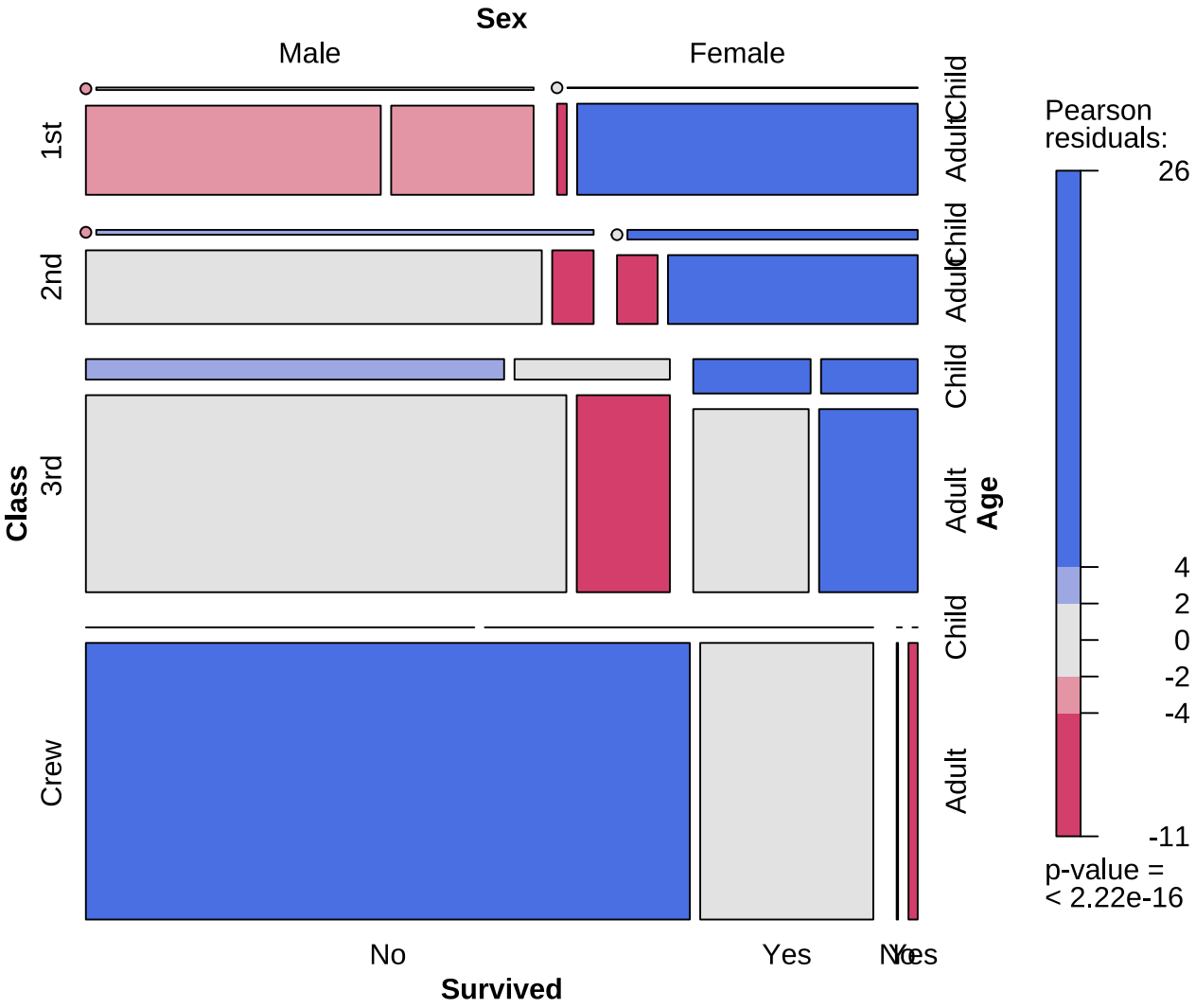


图 14.4: 马赛克图展示多维分类数据

14.4 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 14.2: 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

14.4.1 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 14.3: 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）。

14.4.2 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
```

```
# 双边 P 值
fisher.test(TeaTasting, alternative = "two.sided")
```

```
#>
#>  Fisher's Exact Test for Count Data
#>
#> data:  TeaTasting
#> p-value = 0.4857
#> alternative hypothesis: true odds ratio is not equal to 1
#> 95 percent confidence interval:
```

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

```
#>    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
```

14.4.3 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
```

14.4.4 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
```

14.5 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 14.4，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 14.4: 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 14.5 可以更加直观的看出数据中的比例关系。

接下来进行定量的分析，首先，按性别和录取情况统计人数，如下：

```
m <- xtabs(Freq ~ Gender + Admit, data = as.data.frame(UCBAdmissions))
m
```

```
#>          Admit
#> Gender   Admitted Rejected
#>   Male       1198     1493
#>   Female      557     1278
```

可以看到，申请加州伯克利分校的女生当中，只有 $557/(557+1278) = 30.35\%$ 录取了，而男生则有 $1198/(1198+1493) = 44.52\%$ 的录取率。根据皮尔逊 χ^2 检验：

```
# 不带耶茨矫正
chisq.test(m, correct = FALSE)
```

```
#>
#>  Pearson's Chi-squared test
#>
#> data:  m
#> X-squared = 92.205, df = 1, p-value < 2.2e-16
```

可知 χ^2 统计量的值为 92.205 且 P 值远远小于 0.05，差异达到统计显著性，不是随机因素导致的。因此，加州伯克利分校被指控在招生录取工作中存在性别歧视。然而，当我们细分到各个院系去看录取率（录取人数 / 申请人数），结果显示院系 A 的录取率为 64.41%，院系 B 的录取率为 63.24%，依次类推，各院系情况如下：

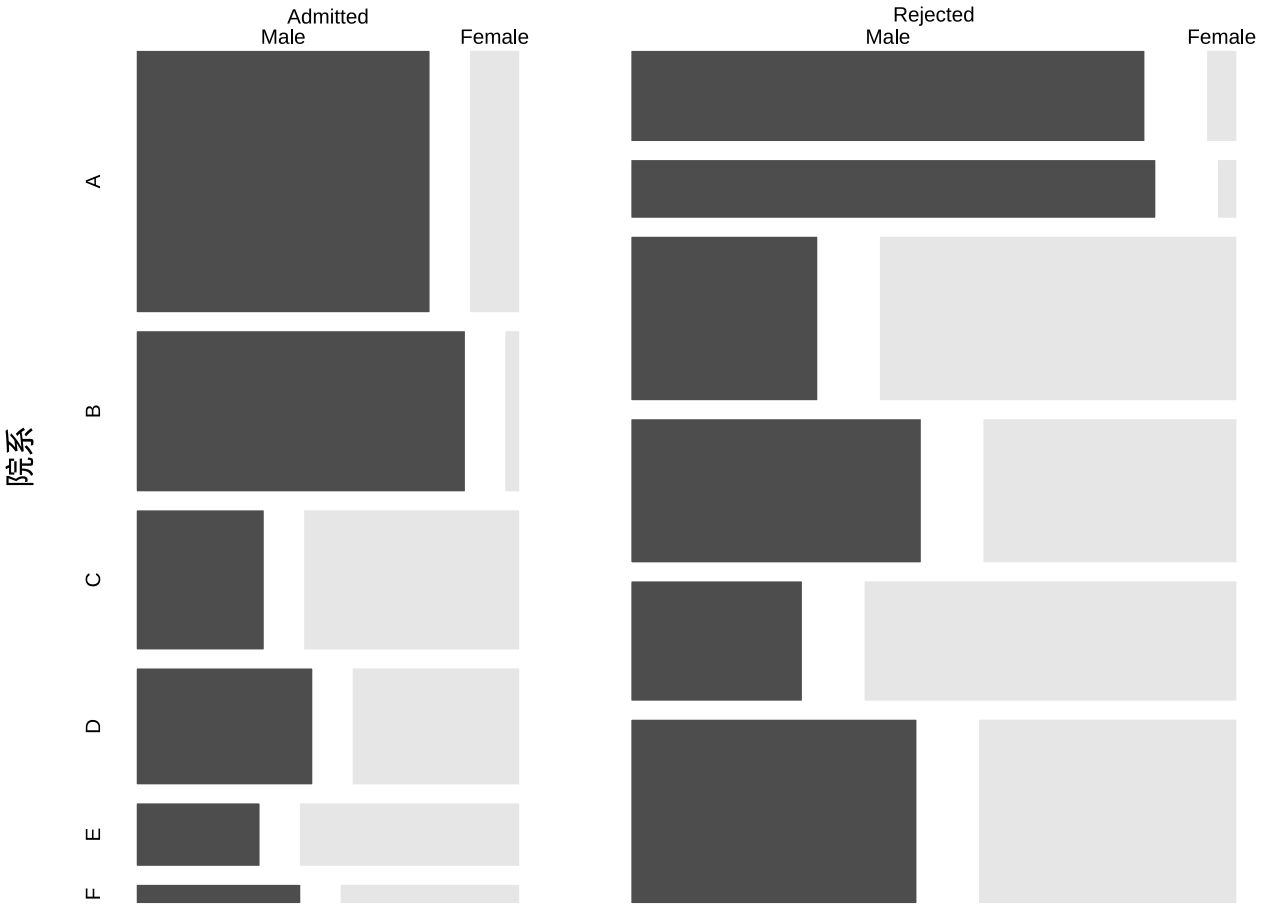


图 14.5: 加州伯克利分校院系录取情况

```
proportions(xtabs(Freq ~ Dept + Admit,
  data = as.data.frame(UCBAdmissions)
), margin = 1)
```

```
#>      Admit
#> Dept   Admitted   Rejected
#>    A 0.64415863 0.35584137
#>    B 0.63247863 0.36752137
#>    C 0.35076253 0.64923747
#>    D 0.33964646 0.66035354
#>    E 0.25171233 0.74828767
#>    F 0.06442577 0.93557423
```

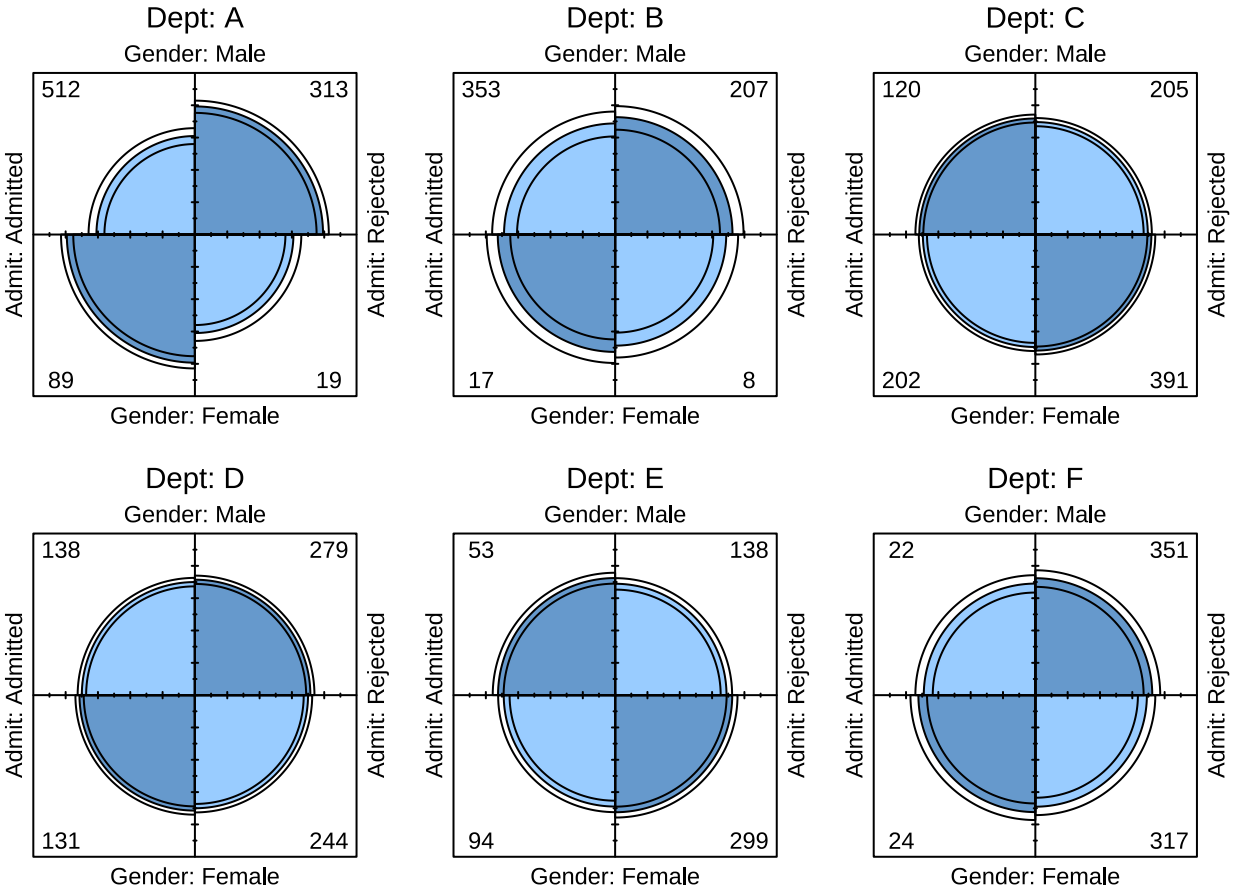


图 14.6: 加州伯克利分校各院系录取情况

对每个院系，单独使用皮尔逊 χ^2 检验，发现只有 A 系的男、女生录取率的差异达到统计显著性，其它系的差异都不显著。辛普森悖论在这里出现了，在分类数据的分析中，常常遇到。

```
# 以 A 系为例
ma <- xtabs(Freq ~ Gender + Admit,
  subset = Dept == "A",
```

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

```
#>
#>  Pearson's Chi-squared test
#>
#> data:  ma
#> X-squared = 17.248, df = 1, p-value = 3.28e-05
```

为了经一步说明此现象的原因，建立对数线性模型来拟合数据，值得一提的是皮尔逊卡方检验可以从对数线性模型的角度来看，而对数线性模型是一种特殊的广义线性模型，针对计数数据建模。

```
fit_ucb0 <- glm(Freq ~ Dept + Admit + Gender,
  family = poisson(link = "log"),
  data = as.data.frame(UCBAdmissions)
)
summary(fit_ucb0)
```

```
#>
#> Call:
#> glm(formula = Freq ~ Dept + Admit + Gender, family = poisson(link = "log"),
#>     data = as.data.frame(UCBAdmissions))
#>
#> Coefficients:
#>               Estimate Std. Error z value Pr(>|z|)
#> (Intercept)    5.37111    0.03964 135.498  < 2e-16 ***
#> DeptB         -0.46679    0.05274  -8.852  < 2e-16 ***
#> DeptC         -0.01621    0.04649  -0.349 0.727355
#> DeptD         -0.16384    0.04832  -3.391 0.000696 ***
#> DeptE         -0.46850    0.05276  -8.879  < 2e-16 ***
#> DeptF         -0.26752    0.04972  -5.380 7.44e-08 ***
#> AdmitRejected  0.45674    0.03051  14.972  < 2e-16 ***
#> GenderFemale  -0.38287    0.03027 -12.647  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for poisson family taken to be 1)
#>
#>     Null deviance: 2650.1  on 23  degrees of freedom
#> Residual deviance: 2097.7  on 16  degrees of freedom
#> AIC: 2272.7
```

```
#>
#> 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)
> ```

14.6 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
```

第十五章　统计检验的功效

15.1　三大检验方法

统计检验的一般方法。

15.1.1　Wald 检验

15.1.2　Wilks 检验

也叫似然比检验

15.1.3　Rao 检验

也叫得分检验

15.2　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"
)
```

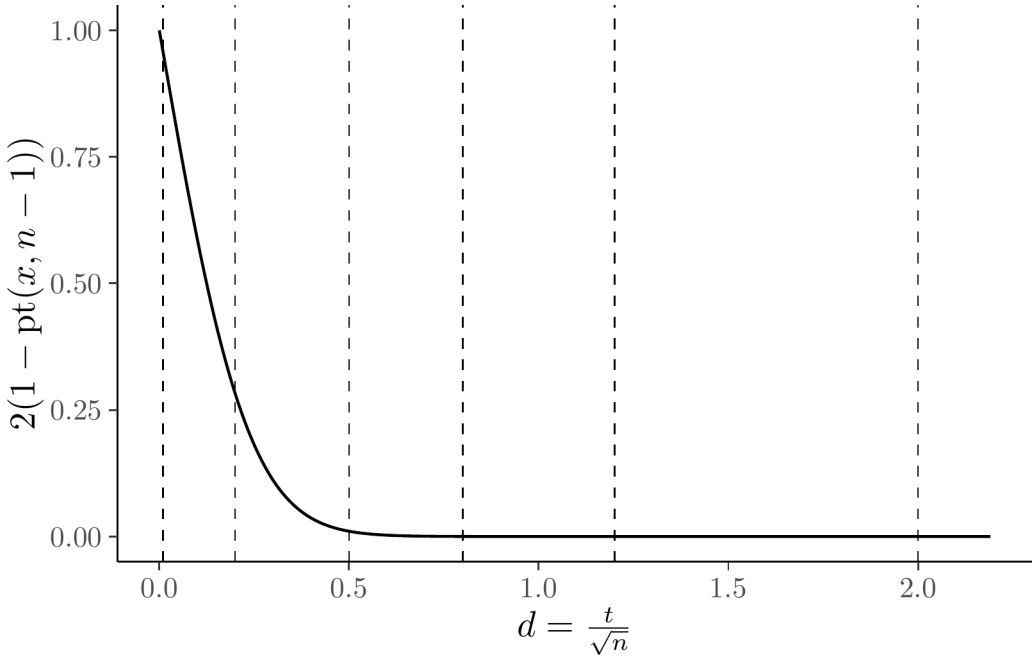


图 15.1: t 检验的功效

```
#>
#>      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 15.1: 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
```

```
# 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

```
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騄先生对此功效计算方法做出过巨大贡献。

15.3 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

15.4 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
```

第四部分

数据建模

第十六章　网络数据分析

网络数据分析又是另一个大话题，微信、微博、论坛、知乎、豆瓣、美团等等应用都或多或少自带了社交属性。人不能脱离社会存在，社交是人的一种本能，身处洪流之中，即是复杂社会网络中的一个节点。网络分析的内容有很多，比如社区探测、节点影响力分析等。网络图是表示节点之间关系的图，核心在于关系的刻画。用来表达网络关系的是稀疏矩阵，以及为处理这种矩阵而专门优化的矩阵计算库，如 **Matrix** 包、**rsparse** 包和 **RcppEigen** 包 (Bates 和 Eddelbuettel 2013) 等。图关系挖掘和计算的应用场景非常广泛，如社交推荐（社交 App）、风险控制（银行征信、企业查）、深度学习（图神经网络）、知识图谱（商户、商家、客户的实体关系网络）、区块链、物联网（IoT）、反洗钱（金融监管）、数据治理（数据血缘图谱）等。

本文将分析 R 语言社区开发者之间的协作关系网络。首先基于 CRAN（The Comprehensive R Archive Network）上发布的 R 包元数据信息，了解 R 语言社区 R 包及其维护者的规模，以及根据元数据中的信息发掘社区中的组织，最后，分析开发者在协作网络中的影响力，并将结果可视化。本文主要用到的工具有 igraph 包，操作图数据和图计算的 tidygraph 包，以及可视化图数据的 ggraph 包。

16.1　R 语言社区的规模

从 CRAN 上的 R 包及其开发者数量来看目前 R 语言社区规模。

```
# 设置就近的 CRAN 镜像站点
Sys.setenv(R_CRAN_WEB = "https://mirrors.tuna.tsinghua.edu.cn/CRAN")
# 获取 R 包元数据
pdb <- tools::CRAN_package_db()
```

截止 2022 年 12 月 31 日，CRAN 上发布的 R 包有 18976 个，CRAN 进入年末维护期 2022-12-22 至 2023-01-05。

```
pdb <- subset(
  x = pdb, subset = !duplicated(Package),
  select = c("Package", "Maintainer", "Title", "Authors@R", "Date", "Published")
)
```

距离上次更新的时间分布，有的包是一周内更新的，也有的是 10 多年未更新的。

```
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 16.1 所示，以 2020 年为例，总数 18976 个 R 包当中有 2470 个 R 包的更新日期停留在 2020 年，占比 2470 / 18976 = 13.02%。过去 1 年内更新的 R 包有 8112 个（包含新出现的 R 包），占总数 8112 / 18976 = 42.75%，过去 2 年内更新的 R 包有 11553 个，占总数 11553 / 18976 = 60.88%，这个占比越高说明社区开发者越活跃。

```
library(ggplot2)
aggregate(data = pdb, Package ~ published_year, FUN = length) |>
  ggplot(aes(x = published_year, y = Package)) +
  geom_col(fill = NA, color = "gray20") +
  theme_classic() +
  coord_cartesian(expand = F) +
  labs(x = "年份", y = "R 包数量")
```

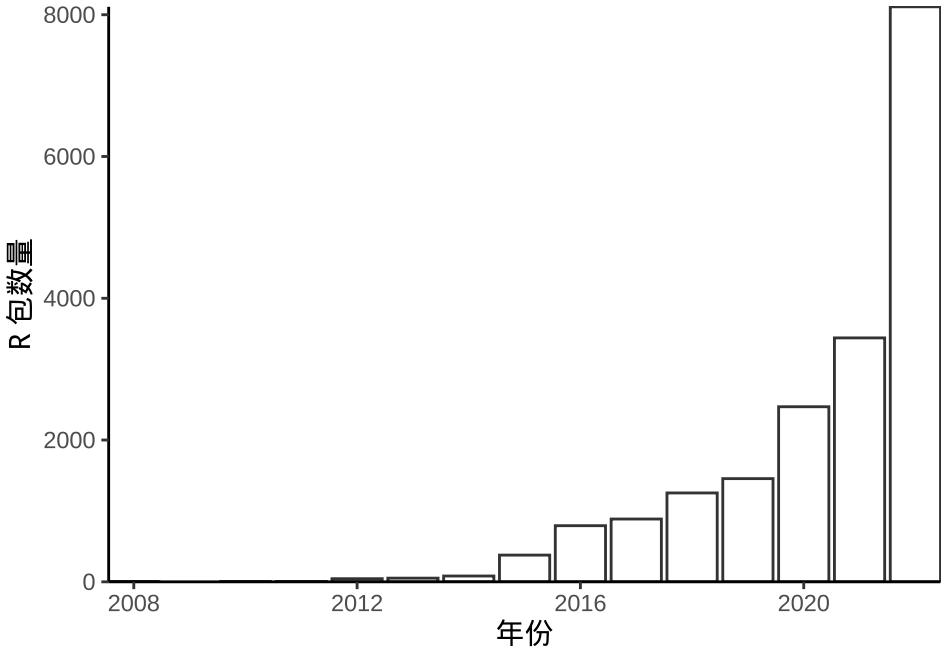


图 16.1: CRAN 上 R 包的更新情况

截止 2022-12-31，CRAN 上 R 包的维护者有 10067 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 16.2 按 R 包发布年份统计开发者数量。

```
# 清理维护者字段，同一个开发者可能有多个邮箱
extract_maintainer <- function(x) {
  x <- gsub(pattern = "<.*?>", replacement = "", x = x)
```

```
  trimws(x, which = "both", whitespace = "[ \t\r\n]")
}
# 只有 18 个维护者名字有大小写差别
pdb$Maintainer2 <- extract_maintainer(pdb$Maintainer)
# 维护者总数
length(unique(pdb$Maintainer2))

#> [1] 10067
```

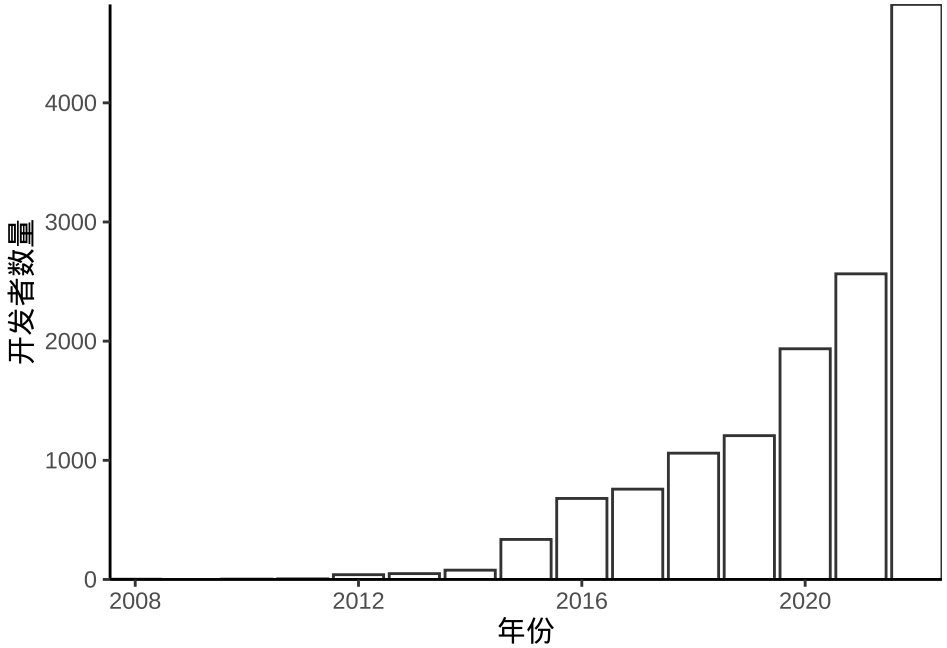


图 16.2: CRAN 上的维护者活跃情况

16.2 R 语言社区的组织

除了 RStudio 公司出品的 tidyverse (Wickham 等 2019) 和 tidymodels (Kuhn 和 Wickham 2020)，还有一些数据分析、建模的工具箱，如 mlr3verse (Lang 和 Schratz 2023)、easystats (Lüdecke 等 2022)、strengejacke (Lüdecke 2019) 和 DrWhy (Biecek 2023)。也有的组织基本停止了开发，如 Omegahat。还有的被商业公司收购后，不再活跃了，如 Revolution Analytics。它们作为解决方案大都属于一些组织，还有深藏功与名，有待笔者挖掘的。因不存在明显的规律，下面从开发者的邮箱出发，隶属企业、组织往往有统一的邮箱后缀。

```
str_extract <- function(text, pattern, ...) regmatches(text, regexpr(pattern, text, ...))
# 移除 ORPHANED
pdb <- subset(pdb, subset = Maintainer != "ORPHANED")
# 抽取邮件后缀
extract_email_suffix <- function(x) {
```

```
  x <- str_extract(text = x, pattern = "<.*?>")
  sub(x = x, pattern = ".*?@(.*?)>", replacement = "\\1")
}
pdb$Email_suffix <- extract_email_suffix(pdb$Maintainer)
```

按组织统计扩展包的数量（总的 R 包数量约 2 万），即各个组织开发的 R 包。

```
pdb_pkg <- aggregate(
  data = pdb, Package ~ Email_suffix, FUN = function(x) { length(unique(x)) }
)
head(pdb_pkg[order(pdb_pkg$Package, decreasing = TRUE), ], 20)
```

```
#>         Email_suffix Package
#> 876        gmail.com    6968
#> 2044     rstudio.com     208
#> 979      hotmail.com     185
#> 1825     outlook.com     152
#> 1971  R-project.org      106
#> 2           163.com       94
#> 210    berkeley.edu       91
#> 2559      umich.edu       91
#> 2819         uw.edu       74
#> 1927 protonmail.com       73
#> 2564        umn.edu       69
#> 581      debian.org       68
#> 2951      yahoo.com       68
#> 1828     outlook.fr       63
#> 2212   stanford.edu       58
#> 155  auckland.ac.nz       57
#> 887           gmx.de       55
#> 2911       wisc.edu       55
#> 895  googlemail.com       50
#> 1970  r-project.org       50
```

不难看出，至少有如下几类：

1. 邮件服务提供商。6968 个 R 包使用 gmail 邮箱作为联系维护者的方式，googlemail.com 也是谷歌提供的服务。hotmail.com 和 outlook.com 都是微软提供的邮箱服务，outlook.fr（法国）也是，除此之外，比较大的邮件服务提供商就是 163.com（网易）、protonmail.com 和 yahoo.com （雅虎）等。
2. 商业组织。208 个 R 包来自 RStudio 公司的员工，这些维护者使用 RStudio 公司提供的邮箱。
3. 开源组织。R-project.org 和 r-project.org 都是 R 语言组织的联系方式，自不必多说，R 语言核心团队成员不仅维护 R 软件源码，还维护了很多 R 包。debian.org 是 Debian 组织的联系方式，都

是开源组织（Open Source Org）。

4. 教育机构。berkeley.edu 、umich.edu 等以 edu 结尾的北美（国）的大学，gmx.de、posteo.de 等以 de 结尾的德国大学，ucl.ac.uk 等以 uk 结尾的英国的大学，auckland.ac.nz 等以 nz 结尾的新西兰的大学，uwaterloo.ca 等以 ca 结尾的加拿大的大学。

按组织统计开发者的数量（总的开发者数量约 1 万），即各个组织的 R 包开发者。

```
pdb_org <- aggregate(
  data = pdb, Maintainer2 ~ Email_suffix, FUN = function(x) { length(unique(x)) }
)
head(pdb_org[order(pdb_org$Maintainer2, decreasing = TRUE), ], 20)
```

```
#>        Email_suffix Maintainer2
#> 876       gmail.com        3800
#> 979     hotmail.com         110
#> 1825    outlook.com          87
#> 2           163.com          57
#> 2559      umich.edu          54
#> 2951      yahoo.com          51
#> 2564        umn.edu          47
#> 1927 protonmail.com          46
#> 2819         uw.edu          46
#> 887          gmx.de          34
#> 210    berkeley.edu          33
#> 2044    rstudio.com          30
#> 895  googlemail.com          28
#> 2212   stanford.edu          27
#> 468    columbia.edu          26
#> 1114       inrae.fr          26
#> 2451      ucl.ac.uk          25
#> 2964       yale.edu          25
#> 635        duke.edu          23
#> 1906      posteo.de          23
```

可见，大部分开发者采用邮件服务提供商的邮件地址。3800 个开发者使用来自谷歌的 gmail.com、197 个开发者使用来自微软的 hotmail.com 或 outlook.com，57 个开发者使用来自网易的 163.com，51 个开发者使用来自雅虎的 yahoo.com，46 个开发者使用来自 Proton 的 protonmail.com。

无论从开发者数量还是 R 包数量的角度看，都有两个显著特点。其一马太效应，往头部集中，其二，长尾分布，尾部占比接近甚至超过 50%。

16.2.1 美国、英国和加拿大

1666 个开发者来自以 edu 为后缀的邮箱。各个组织（主要是大学）及其 R 包开发者数据如下：

```
sum(pdb_org[grepl(pattern = "edu$", x = pdb_org$Email_suffix), "Maintainer2"])
```

```
#> [1] 1666
```

```
pdb_org_edu <- pdb_org[grepl(pattern = "edu$", x = pdb_org$Email_suffix), ]
pdb_org_edu[order(pdb_org_edu$Maintainer2, decreasing = TRUE), ] |> head(20)
```

```
#>         Email_suffix Maintainer2
#> 2559      umich.edu          54
#> 2564        umn.edu          47
#> 2819         uw.edu          46
#> 210    berkeley.edu          33
#> 2212   stanford.edu          27
#> 468    columbia.edu          26
#> 2964       yale.edu          25
#> 635        duke.edu          23
#> 2911       wisc.edu          23
#> 482     cornell.edu          22
#> 2444    ucdavis.edu          21
#> 1929        psu.edu          19
#> 2449   uchicago.edu          19
#> 2830 vanderbilt.edu          19
#> 1660       ncsu.edu          18
#> 1663         nd.edu          18
#> 1008    iastate.edu          17
#> 1919  princeton.edu          17
#> 1815        osu.edu          16
#> 2523      uiowa.edu          16
```

好吧，几乎全是美国各个 NB 大学的，比如华盛顿大学（uw.edu）、密歇根大学（umich.edu）、加州伯克利大学（berkeley.edu）等等。顺便一说，美国各个大学的网站，特别是统计院系很厉害的，已经帮大家收集得差不多了，有留学打算的读者自取，邮箱后缀就是学校/院官网。

有些邮箱后缀带有院系，但是并没有向上合并到学校这一级，比如 `stanford.edu` 、`stat.stanford.edu` 和 `alumni.stanford.edu` 等没有合并统计。实际上，使用 `edu` 邮箱的教育机构大部份位于美国。有的邮箱来自教育机构，但是不以 `edu` 结尾，比如新西兰奥克兰大学 `auckland.ac.nz` 、瑞士苏黎世联邦理工学院 `stat.math.ethz.ch` 等美国以外的教育机构。下面分别查看英国和加拿大的情况。

350 个开发者来自以 uk 为后缀的邮箱。各个组织（主要是大学）及其 R 包开发者数据如下：

```
sum(pdb_org[grepl(pattern = "uk$", x = pdb_org$Email_suffix), "Maintainer2"])
```

```
#> [1] 350
```

```
pdb_org_uk <- pdb_org[grepl(pattern = "uk$", x = pdb_org$Email_suffix), ]
pdb_org_uk[order(pdb_org_uk$Maintainer2, decreasing = TRUE), ] |> head(20)
```

```
#>            Email_suffix Maintainer2
#> 2451          ucl.ac.uk          25
#> 329           cam.ac.uk          17
#> 295       bristol.ac.uk          15
#> 1088     imperial.ac.uk          14
#> 658            ed.ac.uk          13
#> 1286    lancaster.ac.uk          11
#> 1363          lse.ac.uk           9
#> 1605  mrc-bsu.cam.ac.uk           9
#> 2878      warwick.ac.uk           9
#> 870       glasgow.ac.uk           8
#> 1364        lshtm.ac.uk           8
#> 1424   manchester.ac.uk           8
#> 636        durham.ac.uk           7
#> 744        exeter.ac.uk           7
#> 2260 statslab.cam.ac.uk           7
#> 2188        soton.ac.uk           6
#> 2972         york.ac.uk           6
#> 978       hotmail.co.uk           5
#> 1948         qmul.ac.uk           5
#> 248         bioss.ac.uk           4
```

258 个开发者来自以 ca 为后缀的邮箱。各个组织（主要是大学）及其 R 包开发者数据如下：

```
sum(pdb_org[grepl(pattern = "ca$", x = pdb_org$Email_suffix), "Maintainer2"])
```

```
#> [1] 258
```

```
pdb_org_ca <- pdb_org[grepl(pattern = "ca$", x = pdb_org$Email_suffix), ]
pdb_org_ca[order(pdb_org_ca$Maintainer2, decreasing = TRUE), ] |> head(10)
```

```
#>          Email_suffix Maintainer2
#> 2822     uwaterloo.ca          19
#> 1397   mail.mcgill.ca          14
#> 2123           sfu.ca          12
#> 2801      utoronto.ca          12
#> 2426      ualberta.ca          11
#> 2239      stat.ubc.ca           9
#> 2434           ubc.ca           9
```

表格 16.1: CRAN 团队开发维护 R 包数量情况

(a) 表

团队成员	R 包数量
Kurt Hornik	28
Simon Urbanek	26
Achim Zeileis	25
Martin Maechler	25
Torsten Hothorn	25
Paul Murrell	19
Toby Dylan Hocking	17
Brian Ripley	12
Thomas Lumley	12
Uwe Ligges	9
Duncan Murdoch	7
David Meyer	6
CRAN Team	5

(b) 续表

团队成员	R 包数量
Friedrich Leisch	5
Luke Tierney	5
Michael Lawrence	5
Stefan Theussl	5
Bettina Grün	3
John Chambers	3
Simon Wood	3
Bettina Gruen	2
Deepayan Sarkar	2
Douglas Bates	2
Martyn Plummer	2
Peter Dalgaard	1

```
#> 2813          uvic.ca          8
#> 952            hec.ca          7
#> 1416 mail.utoronto.ca          7
```

16.2.2 CRAN 和 RStudio

下面根据邮箱后缀匹配抽取 CRAN 团队及开发的 R 包，规则也许不能覆盖所有的情况，比如署名 CRAN Team 的维护者代表的是 CRAN 团队，**XML** 和 **RCurl** 包就由他们维护。再比如，Brian Ripley 的邮箱 ripley@stats.ox.ac.uk 就不是 CRAN 官网域名。读者若有补充，欢迎 PR 给我。

Kurt Hornik、Simon Urbanek、Achim Zeileis 等真是高产呐！除了维护 R 语言核心代码，还开发维护了那么多 R 包。以 Brian Ripley 为例，看看他都具体维护了哪些 R 包。

表格 16.2: Brian Ripley 维护的 R 包

Package	Title
boot	Bootstrap Functions (Originally by Angelo Canty for S)
class	Functions for Classification
fastICA	FastICA Algorithms to Perform ICA and Projection Pursuit
gee	Generalized Estimation Equation Solver
KernSmooth	Functions for Kernel Smoothing Supporting Wand & Jones (1995)
MASS	Support Functions and Datasets for Venables and Ripley's MASS

表格 16.2: Brian Ripley 维护的 R 包

Package	Title
mix	Estimation/Multiple Imputation for Mixed Categorical and Continuous Data
nnet	Feed-Forward Neural Networks and Multinomial Log-Linear Models
pspline	Penalized Smoothing Splines
RODBC	ODBC Database Access
spatial	Functions for Kriging and Point Pattern Analysis
tree	Classification and Regression Trees

震惊！有一半收录在 R 软件中，所以已经持续维护 **20** 多年了。下面继续根据邮箱后缀将 RStudio 团队的情况统计出来，结果见下表。

CRAN 和 RStudio 团队是 R 语言社区最为熟悉的，其它团队需借助一些网络分析算法挖掘了。

表格 16.3: RStudio 团队开发维护 R 包数量情况（部分）

(a) 表

团队成员	R 包数量
Hadley Wickham	48
Yihui Xie	22
Max Kuhn	18
Lionel Henry	15
Winston Chang	15
Daniel Falbel	13
Jennifer Bryan	13
Davis Vaughan	11
Carson Sievert	10
Tomasz Kalinowski	8
Barret Schloerke	6
Thomas Lin Pedersen	6
Hannah Frick	5
Christophe Dervieux	4
Joe Cheng	4
Julia Silge	4

(b) 续表

团队成员	R 包数量
Cole Arendt	3
Edgar Ruiz	3
JJ Allaire	3
Kevin Kuo	3
Kevin Ushey	3
Richard Iannone	3
Aron Atkins	2
Romain François	2
Yitao Li	2
Brian Smith	1
Emil Hvitfeldt	1
Garrick Aden-Buie	1
James Blair	1
Nathan Stephens	1
Nick Strayer	1

16.3 R 语言社区的开发者

16.3.1 最高产的开发者

继续基于数据集 pdb，将维护 R 包数量比较多的开发者统计出来。

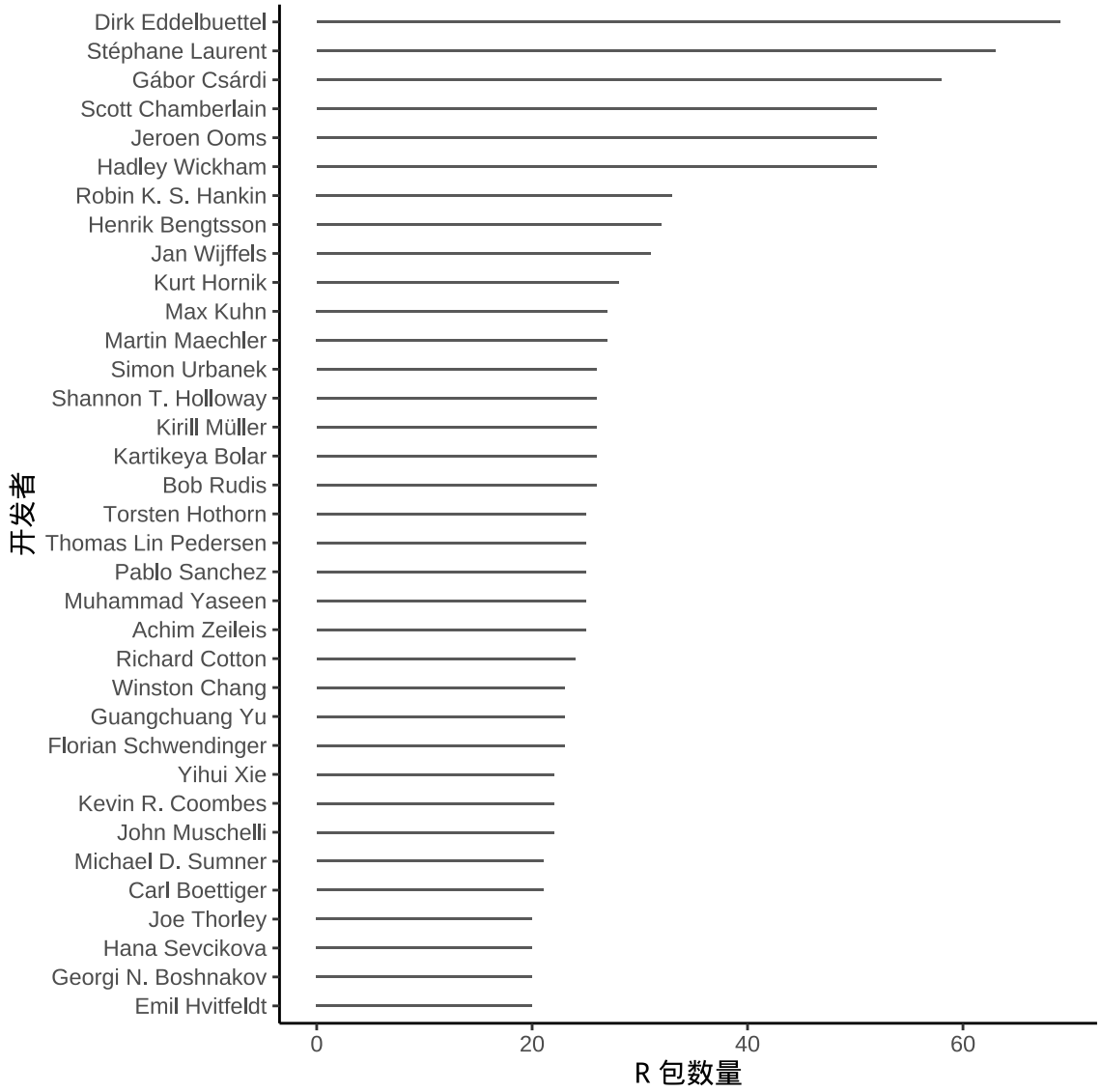


图 16.3: 高产的 R 包开发者

这些开发者的主页和主要的 R 社区贡献如下：

1. Dirk Eddelbuettel 维护了 Rcpp、RcppEigen 等流行的 R 包，通过 Rcpp 包将很多优秀的 C++ 库引入 R 语言社区。
2. Stéphane Laurent 维护了很多与 shiny、htmlwidgets 相关的 R 包，比如 rAmCharts4 包。
3. Gábor Csárdi 维护了 igraph 包以及大量帮助 R 包开发的基础设施，RStudio 雇员。
4. Hadley Wickham 维护了 ggplot2、dplyr、devtools 等流行的 R 包，RStudio 雇员。

5. Jeroen Ooms 维护了 magick、curl 以及大量帮助 R 包开发的基础设施。
6. Scott Chamberlain 维护了很多与 HTTP/Web 相关的 R 包，rOpenSci 联合创始人。
7. Robin K. S. Hankin 维护了很多与贝叶斯、多元统计相关的 R 包。
8. Henrik Bengtsson 维护了 future 和 parallelly 等流行的 R 包，在并行计算方面有很多贡献。
9. Jan Wijffels 维护了很多与自然语言处理、图像识别相关的 R 包，比如 udpipe 、BTM 和 word2vec 等包，Bnosac 团队成员。
10. Kurt Hornik 参与维护 R 软件代码并许多与自然语言处理相关的 R 包，R 核心团队成员。
11. Martin Maechler 维护了 Matrix 包，R 核心团队成员。
12. Max Kuhn 维护了 tidymodels 等包，RStudio 雇员。
13. Bob Rudis 维护了一些与 ggplot2 相关的 R 包，如 ggalt、hrbrthemes 和 statebins 等。
14. Kartikeya Bolar 维护了很多统计与 shiny 结合的 R 包，比如方差分析、逻辑回归、列联表、聚类分析等。
15. Kirill Müller 维护了 DBI 等大量与数据库连接的 R 包。
16. Shannon T. Holloway 维护了许多与生存分析相关的 R 包。
17. Simon Urbanek 维护了 rJava、Rserve 等流行的 R 包，R 核心团队成员，负责维护 R 软件中与 MacOS 平台相关的部分。
18. Achim Zeileis 维护了 colorspace 等流行的 R 包，R 核心团队成员。
19. Muhammad Yaseen 维护了多个与 Multiple Indicator Cluster Survey 相关的 R 包。
20. Pablo Sanchez 维护了多个与市场营销平台连接的 R 语言接口，Windsor.ai 组织成员。
21. Thomas Lin Pedersen 维护了 patchwork、gganimate 和 ggraph 等流行的 R 包，RStudio 雇员。
22. Torsten Hothorn 在统计检验方面贡献了不少内容，比如 coin 和 multcomp 等包，R 核心团队成员。
23. Richard Cotton 维护了 assertive 和 rebus 系列 R 包，代码可读性检查。
24. Florian Schwendinger 维护了大量运筹优化方面的 R 包，扩展了 ROI 包的能力。
25. Guangchuang Yu 维护了 ggtree 和 ggimage 等 R 包，在生物信息和可视化领域有不少贡献。
26. Winston Chang 维护了 shiny 等流行的 R 包，RStudio 雇员。
27. John Muschelli 维护了多个关于神经图像的 R 包。
28. Kevin R. Coombes 维护了多个关于生物信息的 R 包，如 oompaBase 和 oompaData 等。
29. Yihui Xie 维护了 knitr 、rmarkdown 等流行的 R 包，RStudio 雇员。
30. Carl Boettiger 维护了多个接口包，比如 rfishbase 等，rOpenSci 团队成员。
31. Michael D. Sumner 维护了多个空间统计相关的 R 包。
32. Emil Hvitfeldt 维护了多个统计学习相关的 R 包，如 fastTextR 包等，RStudio 雇员。
33. Georgi N. Boshnakov 维护了多个金融时间序列相关的 R 包，如 fGarch、timeDate 和 timeSeries 等包。
34. Hana Sevcikova 维护了多个与贝叶斯人口统计相关的 R 包。
35. Joe Thorley 维护了多个与贝叶斯 MCMC 相关的 R 包，Poisson Consulting 雇员。

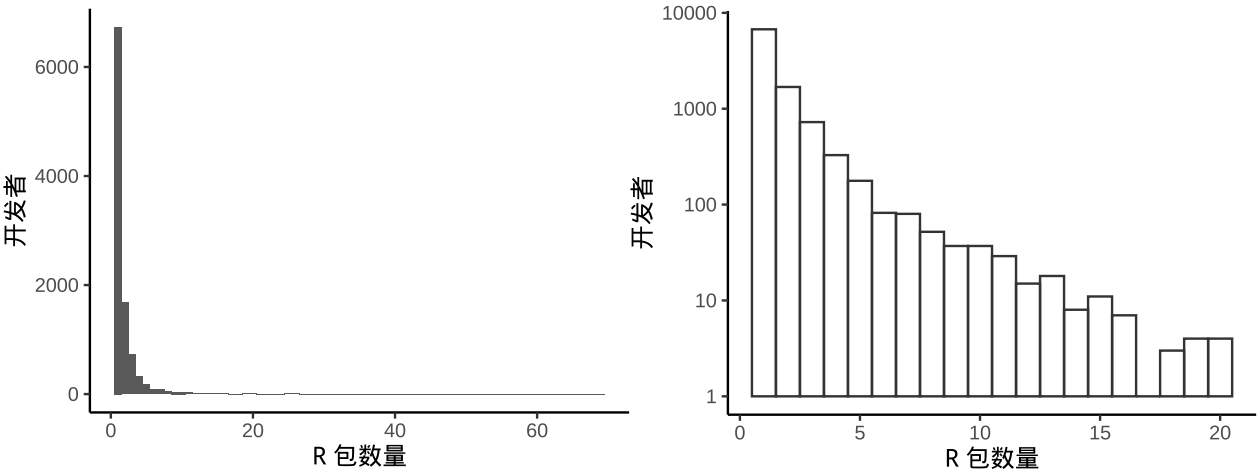
统计开发者数量随维护 R 包数量的分布，发现，开发 1 个 R 包的开发者有 6732 人，开发 2 个 R 包的开发者有 1685 人，第二名是第一名的五分之一，递减规律非常符合指数分布。

```
table(pdb_ctb$Package)
```

```
#>
#>    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
#> 6732 1685  725  328  177   82   80   52   37   37   29   15   18    8   11    7
#>   17   18   19   20   21   22   23   24   25   26   27   28   31   32   33   52
#>    1    3    4    4    2    3    3    1    5    5    2    1    1    1    1    3
#>   58   63   69
#>    1    1    1
```

过滤掉非常高产的开发者，可以发现变化规律服从幂律分布。

```
ggplot(data = pdb_ctb, aes(x = Package)) +
  geom_histogram(binwidth = 1) +
  theme_classic() +
  labs(x = "R 包数量", y = "开发者")
ggplot(data = pdb_ctb[pdb_ctb$Package <= 20, ], aes(x = Package)) +
  geom_histogram(binwidth = 1, fill = NA, color = "gray20") +
  scale_y_log10() +
  theme_classic() +
  labs(x = "R 包数量", y = "开发者")
```



(a) 直方图　　(b) 直方图（对数尺度）

图 16.4: 开发者数量的分布

最高产 Top 1% 的开发者 131 人（开发 R 包超过 10 个的开发者）贡献了 2329 / 18976 = 12.3% 的扩展包，高产的是商业公司、开源组织、大学机构。

```
dim(pdb_ctb[pdb_ctb$Package > 10, ])
```

```
#> [1] 131   2
```

```
sum(pdb_ctb[pdb_ctb$Package > 10, "Package"])
```

```
#> [1] 2329
```

最低产 Bottom 的开发者 6732 人（仅开发一个 R 包的开发者）占总开发者的比例 6732 / 10067 = 66.87%，贡献了 6732 / 18976 = 35.5 % 的扩展包，低产的人是主体。

16.3.2 开发者协作关系

如果一个开发者维护了一个 R 包，就成为维护者。一个 R 包有唯一的一个维护者，可能有一个至多个贡献者，这样，维护者和贡献者之间就形成了有向关系，贡献者可能又是另一个 R 包的维护者，也可能不是。不仅有向而且可能存在环。在一个 R 包中，A 是 B 的贡献者，而在另一个 R 包中，B 是 A 的贡献者，A 和 B 之间可能通过多个 R 包存在多次互相协作关系，这也表明 A 和 B 之间的关系密切。有向环的节点可能有 2 个以上，一个人可能同时属于多个环。

维护者 A 接受来自多个开发者的贡献，接受次数（所有贡献者人数的累和，A 的每个 R 包的贡献者人数相加）视为 A 的入度。维护者 A 作为开发者给多个维护者贡献，贡献次数（作为开发者给其它 R 包做贡献的次数，向外参与贡献的 R 包数目）视为 A 的出度。注意，A 作为维护者，必然包含 A 作为开发者，忽略 A 到 A 的贡献，只考虑贡献/协作关系。

```
# 过滤重复和缺失的记录
pdb <- subset(
  x = pdb, subset = !duplicated(Package) & !is.na(`Authors@R`),
  select = c("Package", "Maintainer", "Authors@R")
)
# 提取维护者的名字
pdb$Maintainer <- extract_maintainer(pdb$Maintainer)
```

有些包的元数据中没有 Authors@R 字段，有可能是没有贡献者，比如 mgcv 包、gam 包等，但也有可能是有贡献者，只是维护者没有填写这个字段，比如 Rcpp 包、RcppEigen 包等，因此将这些先过滤出来。总之，本文是以 Authors@R 字段作为贡献者的来源，共计 12503 个 R 包含有 Authors@R ，有 6000+ 个 R 包没有该字段，缺失约占 R 包总数的 1/3，在不那么考虑准确性的情况下，也可以使用。Author 字段是一段没有结构的文本，相比于 Author 字段，Authors@R 字段是以 R 语言中的 person 类型为存储结构的，比较规范，因此，提取贡献者的操作比较方便。作为示例，下面提取 Matrix 包的贡献者。

```
tmp <- eval(parse(text = pdb[pdb$Package == "Matrix", "Authors@R"]))
tmp <- unlist(lapply(tmp, function(x) format(x, include = c("given", "family"))))
# 返回一个整洁的数据框
tmp <- data.frame(Package = "Matrix", Maintainer = pdb[pdb$Package == "Matrix", "Maintainer"], Auth
# 去掉 Authors 是 Maintainer 的记录
subset(tmp, subset = Maintainer != Authors)
```

```
#>   Package      Maintainer           Authors
#> 1  Matrix Martin Maechler     Douglas Bates
#> 3  Matrix Martin Maechler      Mikael Jagan
#> 4  Matrix Martin Maechler  Timothy A. Davis
#> 5  Matrix Martin Maechler Jens Oehlschlägel
```

```
#> 6  Matrix Martin Maechler       Jason Riedy
#> 7  Matrix Martin Maechler       R Core Team
```

数据框包含 R 包（Package 字段）、及其维护者（Maintainer 字段）和贡献者（Authors 字段）。将上述过程写成一个函数，接着，将所有 R 包的贡献者提取出来，形成一个大的数据框。

```
extract_authors <- function(pkg) {
  sub_pdb <- pdb[pdb$Package == pkg, ]
  tmp <- eval(parse(text = sub_pdb[, "Authors@R"]))
  tmp <- unlist(lapply(tmp, function(x) format(x, include = c("given", "family"))))
  tmp <- data.frame(Package = pkg, Maintainer = sub_pdb[, "Maintainer"], Authors = tmp)
  subset(tmp, subset = Maintainer != Authors)
}
extract_authors("Matrix")
```

```
#>   Package      Maintainer           Authors
#> 1  Matrix Martin Maechler     Douglas Bates
#> 3  Matrix Martin Maechler      Mikael Jagan
#> 4  Matrix Martin Maechler  Timothy A. Davis
#> 5  Matrix Martin Maechler Jens Oehlschlägel
#> 6  Matrix Martin Maechler       Jason Riedy
#> 7  Matrix Martin Maechler       R Core Team
```

```
# lapply(c("Matrix", "gt"), extract_authors)
# 抽取所有 R 包的贡献者，运行需要1-2分钟时间
pdb_authors_list <- lapply(pdb[, "Package"], extract_authors)
# 合并列表
pdb_authors_dt <- data.table::rbindlist(pdb_authors_list)
```

最后整理出来的大数据框 **pdb_authors_dt** 含有近 26000 条记录，即边的规模大小。考虑到有些维护者和贡献者之间可能存在多次合作的情况，下面统计一下合作次数。

```
pdb_authors_dt[ ,.(cnt = length(Package)) , by = c("Maintainer", "Authors")
                ][cnt >= 10, ][order(cnt, decreasing = T), ]
```

```
#>                Maintainer              Authors   cnt
#>                    <char>               <char> <int>
#>  1:         Hadley Wickham              RStudio    36
#>  2:          Pablo Sanchez           Windsor.ai    25
#>  3:           Jan Wijffels               BNOSAC    24
#>  4:           Gábor Csárdi              RStudio    19
#>  5:               Hong Ooi            Microsoft    16
#>  6:               Max Kuhn              RStudio    14
#>  7:           Lionel Henry              RStudio    14
```

```
#>  8:      Robrecht Cannoodt        Wouter Saelens    13
#>  9:      Scott Chamberlain              rOpenSci    13
#> 10:            Joe Thorley    Poisson Consulting    13
#> 11:      Frederic Bertrand Myriam Maumy-Bertrand    12
#> 12:          Winston Chang               RStudio    12
#> 13:          Daniel Falbel               RStudio    12
#> 14:           David Kretch           Adam Banker    12
#> 15:           David Kretch      Amazon.com, Inc.    12
#> 16:         Victor Perrier           Fanny Meyer    11
#> 17:         Jennifer Bryan               RStudio    11
#> 18: William Michael Landau Eli Lilly and Company    11
#> 19:        Adrian Baddeley             Ege Rubak    11
#> 20:           Gábor Csárdi            Jim Hester    10
#> 21:          Kirill Müller               RStudio    10
#> 22:         Carson Sievert               RStudio    10
#> 23:    Thomas Lin Pedersen               RStudio    10
#> 24:           Lionel Henry        Hadley Wickham    10
#> 25:        Adrian Baddeley           Rolf Turner    10
#>                 Maintainer               Authors   cnt
```

Authors 字段出现了不少组织的名字，这是因为有许多 R 包的维护者受雇于该组织，版权归属于该组织，组织不仅提供持续的资金，而且还提供其它帮助。以 **dplyr** 包为例，Hadley Wickham 受雇于 RStudio 公司，在 dplyr 包的元数据中，字段 `Authors@R` 中 RStudio 的角色是 cph 和 fnd ，即版权所有和资金支持。角色 cre 就是维护者，负责与 CRAN 团队的沟通。角色 aut 就是对 R 包有实质贡献的人。

```
format(eval(parse(text = pdb[pdb$Package == "dplyr", "Authors@R"])),
       include = c("given", "family", "role"))
```

```
#> [1] "Hadley Wickham [aut, cre]" "Romain François [aut]"
#> [3] "Lionel Henry [aut]"        "Kirill Müller [aut]"
#> [5] "RStudio [cph, fnd]"
```

此外，同属于一个组织的维护者之间常常合作紧密，从上面的结果可以看到，Gábor Csárdi 和 Jim Hester ，Lionel Henry 和 Hadley Wickham，Carson Sievert 和 Joe Cheng ，Jennifer Bryan 和 Hadley Wickham 等同属于 RStudio 公司，常常协作开发项目。对 RStudio、CRAN Team 和 rOpenSci 不再赘述，下面对排名靠前的其它组织略作说明。

1. Windsor.ai 提供一系列可以连接各大营销平台，获取营销效果数据 R 包。
2. BNOSAC 提供一系列计算机视觉、图像识别、自然语言处理方面的 R 包，比如 udpipe、word2vec、doc2vec 等包。
3. Microsoft 提供一系列连接和操作 Azure 云套件的 R 包，比如 AzureR 包。
4. Wouter Saelens 提供一系列单细胞轨迹推理（single-cell trajectory inference）相关的 R 包，形成一个 **dynverse** 家族。

5. Poisson Consulting 提供一系列用于计算生物学和统计生态学的 R 包和相关研究论文。
6. Amazon.com, Inc. 提供一系列用于存储、管理、操作等 Amazon 云服务的 R 包，形成一个 paws 套件。
7. Eli Lilly and Company 可能是 rOpenSci 的一员，赞助了旗下的 targets 和 jagstargets 等 R 包。

最后，统计协作次数的分布，网络中边的权重的分布。

```
pdb_authors_net <- pdb_authors_dt[, .(cnt = .N), by = c("Maintainer", "Authors")]
table(pdb_authors_net$cnt)
```

```
#>
#>     1     2     3     4     5     6     7     8     9    10    11    12    13
#> 20420  1513   366   121    44    28    14     8     3     6     4     5     3
#>    14    16    19    24    25    36
#>     2     1     1     1     1     1
```

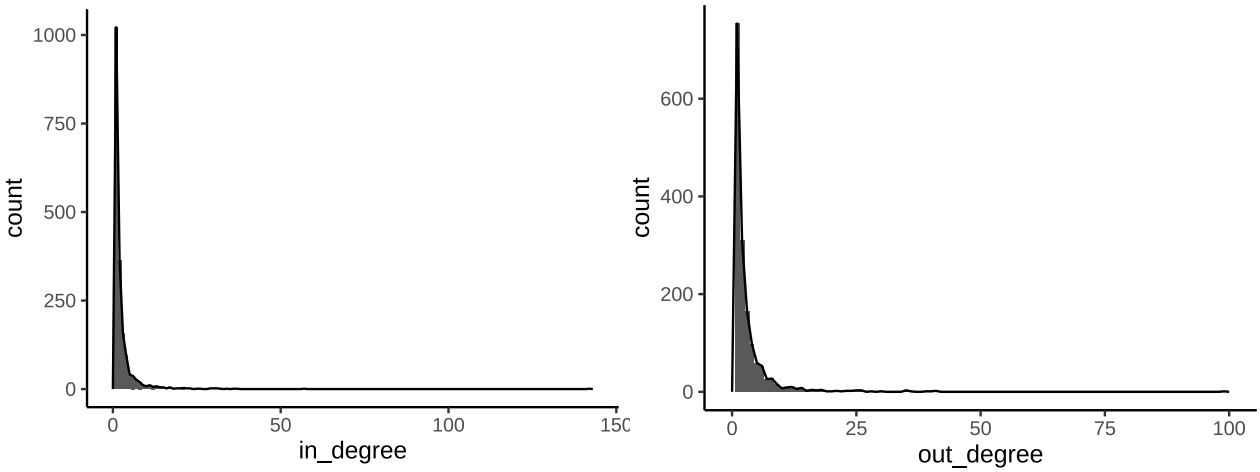
可以发现，绝大多数人之间协作只有一次。

16.3.3 节点出入度分布

下面简化这个网络，仅考虑贡献者也是维护者的情况，就是说网络中所有节点既是维护者也是贡献者，这会过滤掉组织机构、大量没有在 CRAN 发过 R 包的贡献者、从没给其它维护者做贡献的维护者。简化后，网络节点的出度、入度的分布图如下。

```
# Maintainer 的入度
pdb_authors_net_indegree <- pdb_authors_dt[Authors %in% Maintainer,
  ][, .(in_degree = length(Authors)), by = "Maintainer"]
# Authors 的出度
pdb_authors_net_outdegree <- pdb_authors_dt[Authors %in% Maintainer,
  ][, .(out_degree = length(Maintainer)), by = "Authors"]

ggplot(pdb_authors_net_indegree, aes(x = in_degree)) +
  geom_histogram(binwidth = 1) +
  geom_freqpoly(binwidth = 1) +
  theme_classic()
ggplot(pdb_authors_net_outdegree, aes(x = out_degree)) +
  geom_histogram(binwidth = 1) +
  geom_freqpoly(binwidth = 1) +
  theme_classic()
```



(a) 入度的分布

(b) 出度的分布

图 16.5: 节点的入度和出度的分布

16.3.4 可视化协作网络

节点的大小以维护者维护的 R 包数量来表示，边的大小以维护者之间协作次数来表示。为了美观起见，更为了突出重点，仅保留协作次数大于 1 的边。

```
# 边
pdb_authors_net_edge <- pdb_authors_dt[Authors %in% Maintainer,
  ][, .(edge_cnt = .N), by = c("Authors", "Maintainer")][edge_cnt > 1, ]
pdb_authors_net_edge[order(edge_cnt, decreasing = TRUE),]
```

```
#>                      Authors            Maintainer edge_cnt
#>                       <char>                <char>    <int>
#>   1:              Jim Hester          Gábor Csárdi       10
#>   2:          Hadley Wickham          Lionel Henry       10
#>   3:               Joe Cheng        Carson Sievert        9
#>   4:          Hadley Wickham        Jennifer Bryan        8
#>   5: Steven Andrew Culpepper James Joseph Balamuta        8
#>  ---
#> 528:             Aaron Wolen     Scott Chamberlain        2
#> 529:               Bob Rudis         Simon Garnier        2
#> 530:           Marco Sciaini         Simon Garnier        2
#> 531:          Carlos Morales           Martin Chan        2
#> 532:               Md Yeasin     Ranjit Kumar Paul        2
```

```
# 顶点
pdb_authors_net_vertex <- pdb_authors_dt[, .(vertex_cnt = length(unique(Package))), by = "Maintaine
  ][Maintainer %in% c(pdb_authors_net_edge$Maintainer, pdb_authors_net_edge$Authors),]
```

```
pdb_authors_net_vertex[order(vertex_cnt, decreasing = TRUE),]
```

```
#>                Maintainer vertex_cnt
#>                    <char>      <int>
#>   1:       Hadley Wickham         43
#>   2:         Gábor Csárdi         33
#>   3:          Jeroen Ooms         28
#>   4:    Scott Chamberlain         28
#>   5:            Yihui Xie         21
#>  ---
#> 579:    Katriona Goldmann          1
#> 580:        Carlo Pacioni          1
#> 581:       Michael Scholz          1
#> 582: Javier Roca-Pardinas          1
#> 583:         Xianying Tan          1
```

这是一个有向图，其各个字段含义如下。

- Maintainer 维护者（代表流 to）
- Authors 贡献者（代表源 from）
- edge_cnt 边的大小表示维护者 Maintainer 和贡献者 Authors 的协作次数
- vertex_cnt 顶点大小表示维护者 Maintainer 维护的 R 包数量

下面先考虑用 igraph 包可视化这个复杂的有向带权网络。pdb_authors_net_edge 和 pdb_authors_net_vertex 都是数据框，首先调用 igraph 包的函数 graph_from_data_frame() 将其转化为网络类型 igraph ，然后使用函数 plot() 绘制网络图。

协作关系弱的开发者占大部分，构成一个「月亮」的造型，其中，不乏维护多个 R 包的开发者，这些人要么单干，要么在专业小领域、小组织内协作。与之相对应的是协作关系较强的开发者，人数虽少，影响力却大，构成一个「太阳」的造型。协作得多往往意味着维护的 R 包也不少，甚至同属于一个组织，因此，高产的开发者、影响力大的组织聚集在一起，如 R Core Team、RStudio、rOpenSci 等。

```
eb <- cluster_edge_betweenness(pdb_authors_graph)
eb
```

```
#> IGRAPH clustering edge betweenness, groups: 181, mod: 0.88
#> + groups:
#>   $`1`
#>   [1] "Matt Nunes"           "Daniel Grose"         "Guy Nason"
#>   [4] "Rebecca Killick"      "Idris Eckley"         "Alessandro Cardinali"
#>
#>   $`2`
#>   [1] "Jin Zhu"    "Shiyun Lin"
#>
```

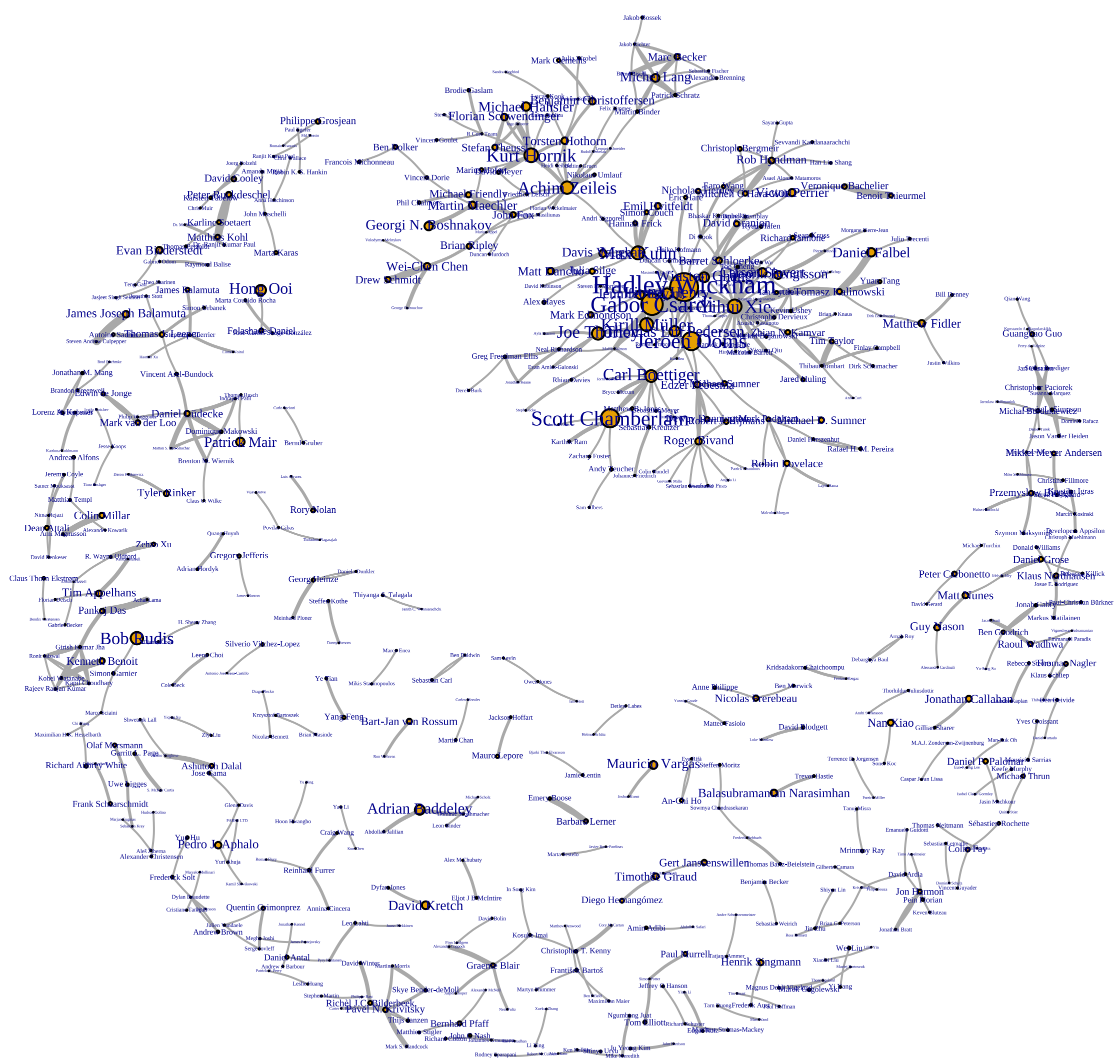


图 16.6: 开发者的协作关系网络

```
#>   $`3`
#>    [1] "Julio Trecenti"       "Henrik Bengtsson"     "Morgane Pierre-Jean"
#>    [4] "Zhian N. Kamvar"      "Pierre Neuvial"       "Michal Bojanowski"
#>   + ... omitted several groups/vertices
```

igraph 包提供多种社区探测的算法，上面简单使用函数 `cluster_edge_betweenness()` 来探测，结果显示有 181 个社区。社区 1 包含的成员如下：

```
eb$names[eb$membership == 1]
```

```
#> [1] "Matt Nunes"           "Daniel Grose"         "Guy Nason"
#> [4] "Rebecca Killick"      "Idris Eckley"         "Alessandro Cardinali"
```

社区 3、14、21、34、46、52、75 的成员是比较多的。其中，社区 3 是以 RStudio 为核心的大社区，社区 14 是以 CRAN 为核心的大社区。

```
# RStudio 为核心的大社区
eb$names[eb$membership == 3]
```

```
#>  [1] "Julio Trecenti"       "Henrik Bengtsson"     "Morgane Pierre-Jean"
#>  [4] "Zhian N. Kamvar"      "Pierre Neuvial"       "Michal Bojanowski"
#>  [7] "Ian Lyttle"           "Thomas Lin Pedersen"  "Yihui Xie"
#> [10] "Dirk Schumacher"      "Jeroen Ooms"          "Gábor Csárdi"
#> [13] "Sean Kross"           "Carl Boettiger"       "Neal Richardson"
#> [16] "Ryan Hafen"           "Matthew Fidler"       "Hadley Wickham"
#> [19] "Mark Edmondson"       "Kirill Müller"        "Richard Iannone"
#> [22] "Carson Sievert"       "Winston Chang"        "Lionel Henry"
#> [25] "Jennifer Bryan"       "Michael Sumner"       "Scott Chamberlain"
#> [28] "Garrick Aden-Buie"    "Daniel Falbel"        "Matthew B. Jones"
#> [31] "Hiroaki Yutani"       "Taiyun Wei"           "Jim Hester"
#> [34] "Romain François"      "Greg Freedman Ellis"  "Rhian Davies"
#> [37] "Bryce Mecum"          "Steph Locke"          "Christophe Dervieux"
#> [40] "Jonathan Keane"       "Thibaut Jombart"      "Dewey Dunnington"
#> [43] "Anne Cori"            "Bill Denney"          "Jared Huling"
#> [46] "Wush Wu"              "Atsushi Yasumoto"     "Barret Schloerke"
#> [49] "Yuan Tang"            "Duncan Garmonsway"    "Edzer Pebesma"
#> [52] "Sebastian Meyer"      "Derek Burk"           "Tim Taylor"
#> [55] "Alicia Schep"         "Tomasz Kalinowski"    "Michael Rustler"
#> [58] "Joe Cheng"            "Bhaskar Karambelkar"  "Sebastian Kreutzer"
#> [61] "JJ Allaire"           "JooYoung Seo"         "Zachary Foster"
#> [64] "Malcolm Barrett"      "Aaron Wolen"          "Bruno Tremblay"
#> [67] "Justin Wilkins"       "Yixuan Qiu"           "Johannes Friedrich"
#> [70] "Kevin Ushey"          "Steven M. Mortimer"   "Karthik Ram"
```

```
#> [73] "Jorrit Poelen"        "Maëlle Salmon"        "Aron Atkins"
#> [76] "Ramnath Vaidyanathan" "Thomas Leeper"        "Dirk Eddelbuettel"
#> [79] "Xianying Tan"
```

```
# CRAN 为核心的大社区
eb$names[eb$membership == 14]
```

```
#>  [1] "Achim Zeileis"        "Michael Hahsler"      "Michel Lang"
#>  [4] "Nikolaus Umlauf"      "Vincent Dorie"        "Bettina Gruen"
#>  [7] "Bernd Bischl"         "Ben Bolker"           "Marc Becker"
#> [10] "Friedrich Leisch"     "Brian Ripley"         "Michael Friendly"
#> [13] "John Fox"             "Kurt Hornik"          "Patrick Schratz"
#> [16] "Volodymyr Melnykov"   "Martin Maechler"      "George Ostrouchov"
#> [19] "Drew Schmidt"         "Georgi N. Boshnakov"  "Wei-Chen Chen"
#> [22] "Stefan Theussl"       "David Meyer"          "Jakob Bossek"
#> [25] "Francois Michonneau"  "Marius Hofert"        "Florian Schwendinger"
#> [28] "Felix Zimmer"         "Martin Binder"        "Phil Chalmers"
#> [31] "Lukas Sablica"        "Sebastian Fischer"    "Lennart Schneider"
#> [34] "Jakob Richter"        "Florian Wickelmaier"  "Rudolf Debelak"
#> [37] "Duncan Murdoch"       "Alexander Brenning"   "Ingo Feinerer"
```

同时，在 RStudio 这个大社区下，有一些与之紧密相关的小社区，比如 Rob Hyndman 等人的时间序列社区、Roger Bivand 等人的空间统计社区。

```
# 时间序列 Rob Hyndman
eb$names[eb$membership == 52]
```

```
#>  [1] "Asael Alonzo Matamoros"   "Nicholas Tierney"
#>  [3] "Sevvandi Kandanaarachchi" "Rob Hyndman"
#>  [5] "Di Cook"                  "Mitchell O'Hara-Wild"
#>  [7] "Han Lin Shang"            "Sayani Gupta"
#>  [9] "Earo Wang"                "Christoph Bergmeir"
```

```
# 空间统计 Roger Bivand
eb$names[eb$membership == 75]
```

```
#> [1] "Sebastian Jeworutzki" "Roger Bivand"         "Colin Rundel"
#> [4] "Angela Li"            "Gianfranco Piras"     "Patrick Giraudoux"
#> [7] "Giovanni Millo"
```

结合前面的图 16.6 ，知道有很多小圈圈，这些放一边，重点关注那些大的圈圈，见下图。

下面使用 **tidygraph** 包构造图数据、计算节点中心度，**dplyr** 包操作数据。中心度代表节点（开发者）的影响力（或者重要性）。最后，借助 **ggraph** 包绘制维护者之间的贡献网络，节点的大小代表维护者影响力的强弱。

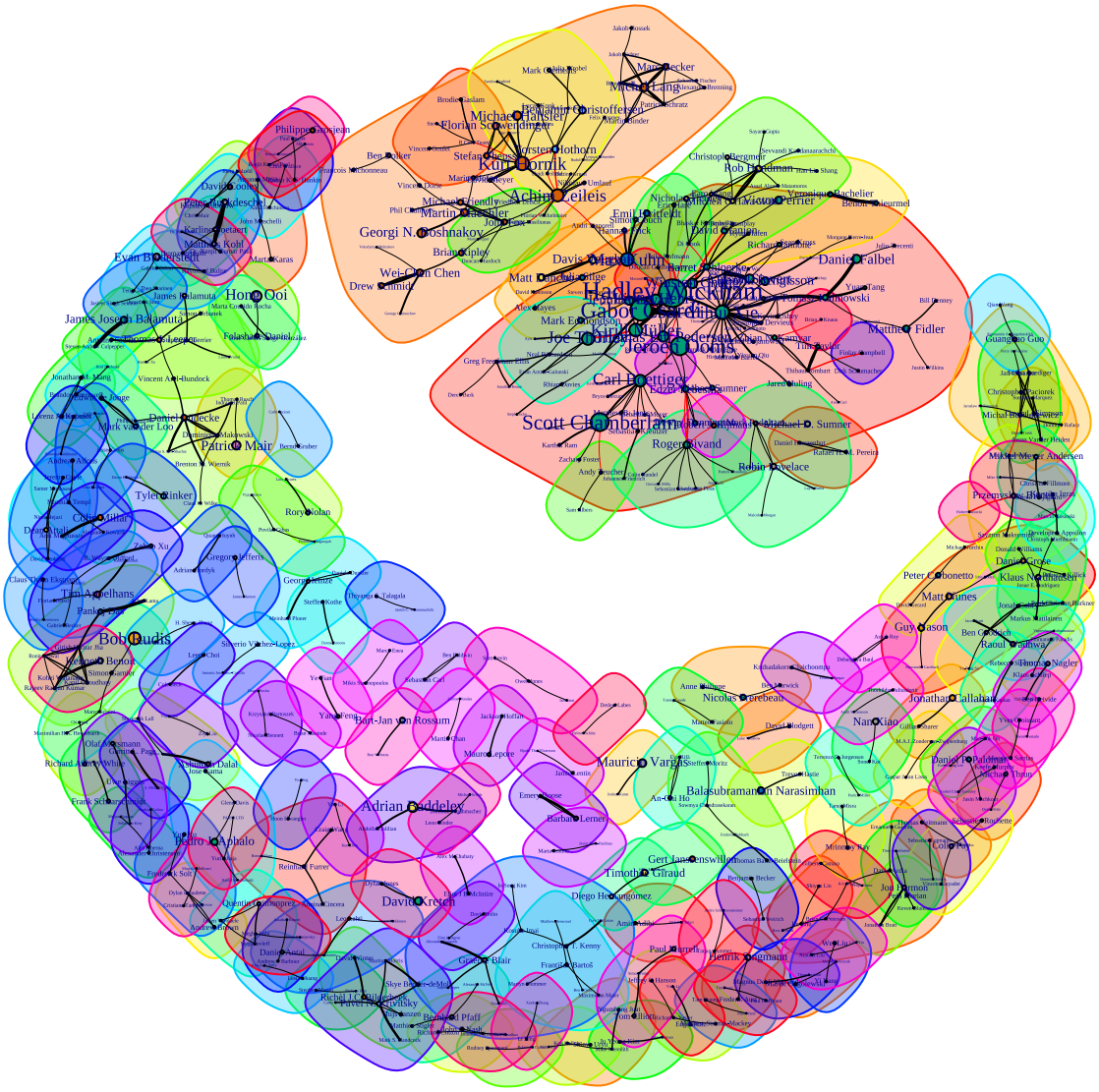


图 16.7: 探测协作关系网络中的社区

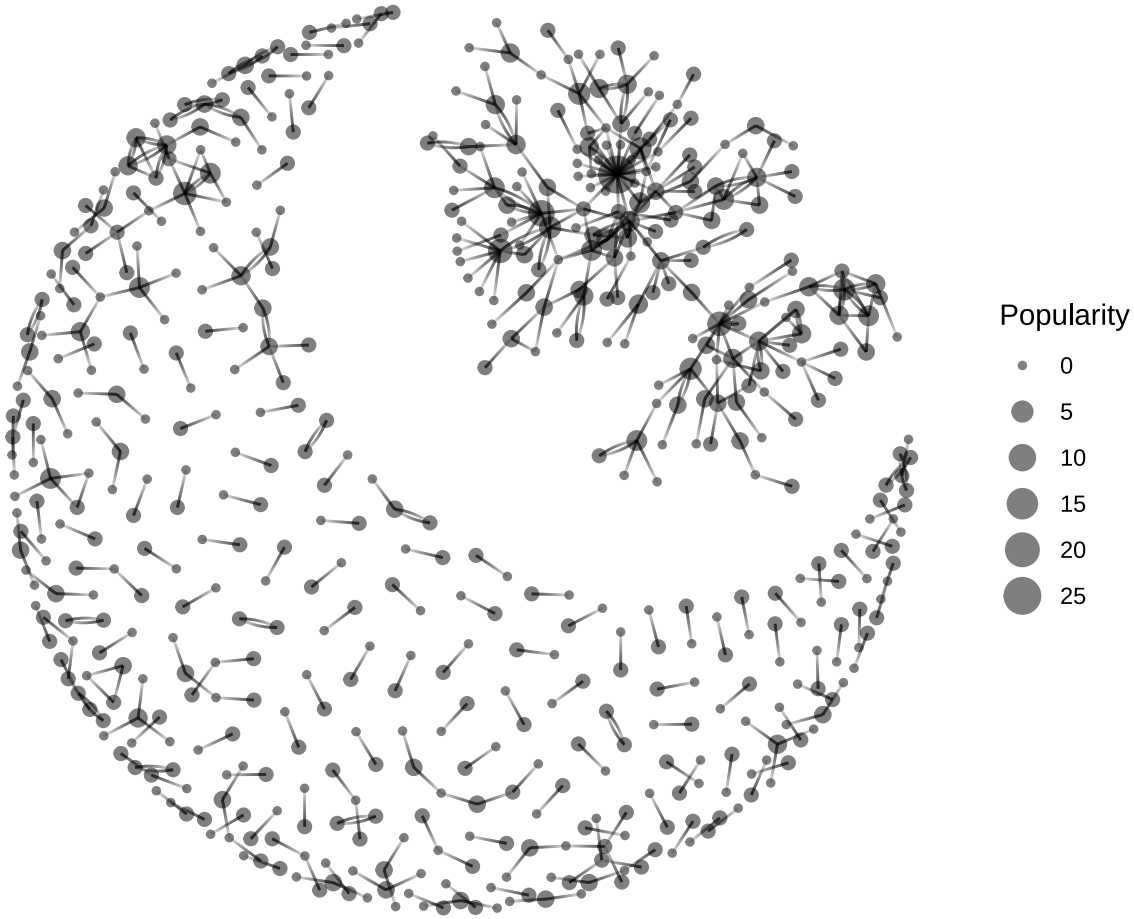


图 16.8: 开发者的影响力网络

前面两个网络图基于同一份数据、同样的网络布局算法，得到非常类似的结果。静态图上的标签相互重叠，影响细节的观察和探索，比如连接 CRAN 和 RStudio 两大阵营的通道。下面使用 visNetwork 包制作交互式网络图形，它是 JS 库 vis-network 的 R 语言接口，使用 **visNetwork** 包绘制交互式网络图后，可以在图上使用鼠标放大、拖拽。可以发现在 CRAN 社区的 Achim Zeileis 和 RStudio 社区的 Max Kuhn 之间是由 Andri Signorell 牵线搭桥。此外，读者若有兴趣，可以使用 Richard Iannone 开发的 DiagrammeR 包制作静态的矢量网页图形。

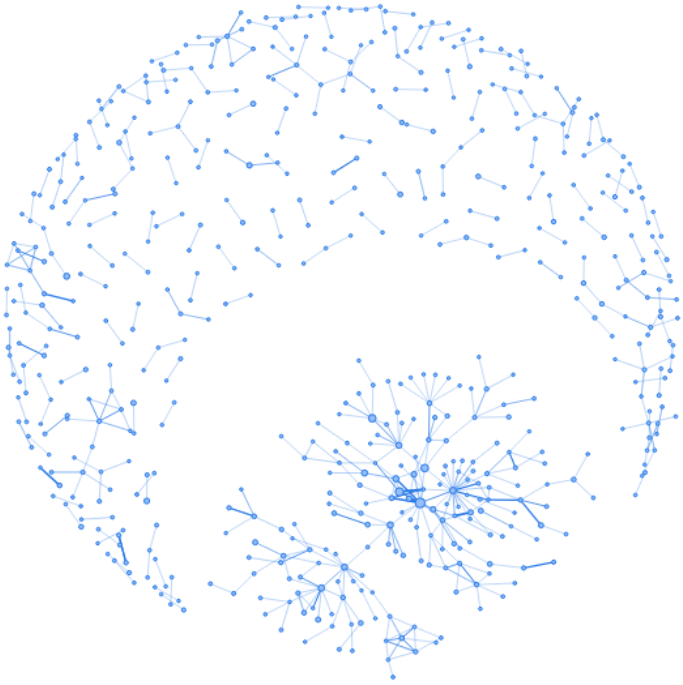


图 16.9: 开发者的影响力网络（visNetwork）

16.4 扩展阅读

R 语言网络分析方面的著作有 Erick Kolaczyk 的书籍《Statistical Analysis of Network Data with R》(Kolaczyk 和 Csárdi 2020)，网络可视化方面，推荐 Hadley Wickham 的著作《ggplot2: Elegant Graphics for Data Analysis》(Wickham, Navarro, 和 Pedersen 2024) 的第七章，Sam Tyner 等人的文章《Network Visualization with ggplot2》(Tyner, Briatte, 和 Hofmann 2017) 也值得一看。

在网络数据分析方面，igraph 是非常流行的分析框架，它是由 C 语言写成的，非常高效。同时，它提供多种语言的接口，其 R 语言接口 igraph 包在 R 语言社区也是网络数据分析的事实标准，被很多其它做网络分析的 R 包所引用。开源的 Gephi 软件适合处理中等规模的网络分析和可视化。大规模图计算可以用 Apache Spark 的 GraphX。R 语言这层，主要还是对应数据分析和数据产品，用在内部咨询和商业分析上。

企业级的图存储和计算框架，比较有名的是 Neo4j ，它有开源版本和商业版本。Nebula Graph 开源分

布式图数据库，具有高扩展性和高可用性，支持千亿节点、万亿条边、毫秒级查询，有中文文档，有企业应用案例（美团图数据库平台建设及业务实践）。阿里研发的 GraphScope 提供一站式大规模图计算系统，支持图神经网络计算。

16.5 习题

1. 类似开发者协作关系的分析，可以统计 R 包被多少 R 包依赖，依赖数量的分布。统计 R 包被依赖的深度（若 R 包 A 被 R 包 B 依赖，R 包 B 被 R 包 C 依赖，以此类推）。进而，构建、分析、可视化依赖关系网络，分析 R 包的影响力。
2. 本文基于 2022 年 12 月 31 日的 R 包元数据进行分析，请与 2023 年 12 月 31 日的数据比较。

第十七章　文本数据分析

R 语言任务视图中以自然语言处理（Natural Language Processing）涵盖文本分析（Text Analysis）的内容。R 语言社区中有两本文本分析相关的著作，分别是《Text Mining with R》(Silge 和 Robinson 2017) 和《Supervised Machine Learning for Text Analysis in R》(Hvitfeldt 和 Silge 2021)。

本文有两个目的：其一分析谢益辉 10 多年日志，挖掘写作主题及其变化；其二挖掘湘云的日志主题，计算与益辉日志的风格相似度。事实上，益辉在个人主页中是明确说了自己的兴趣范围的，本文借用文本挖掘中的主题建模方法，不过是一点实证，熟悉文本建模的操作过程。

从谢益辉公开的日志中，探索成功人士的经历，从中汲取一些经验、教训。最近才知道他有 300 多万字的日志，数字惊讶到我了，遂决定抽取最近 10 年的日志数据进行分析。中英文分开，首先处理、分析中文日志。文本操作、分析的内容有数据清理、文本分词、词频统计、词云展示、词向量化、主题建模、相似度度量等。对作者来说，感兴趣的主题与写作的内容有直接的关系。益辉的日志都是没有分类标签的，主题挖掘可以洞察作者的兴趣。

```
library(jiebaRD)  # 词库
library(jiebaR)   # 分词
library(ggplot2)  # 绘图
library(ggrepel)
library(ggwordcloud) # 词云
library(text2vec)    # LDA 算法
```

17.1　数据获取

- 总体规模：益辉每年的日志数量、日志平均字数，益辉发布书籍的年份
- 过程细节：发布时间、日志字数的日历图、日志年度主题

下载益辉的日志数据

```
git clone git@github.com:yihui/yihui.org.git
```

经过整理后，打包成 Rdata 数据供 R 软件使用。

```
# 加载益辉的日志数据
load(file = "data/text/yihui.Rdata")
```

17.2 日志概况

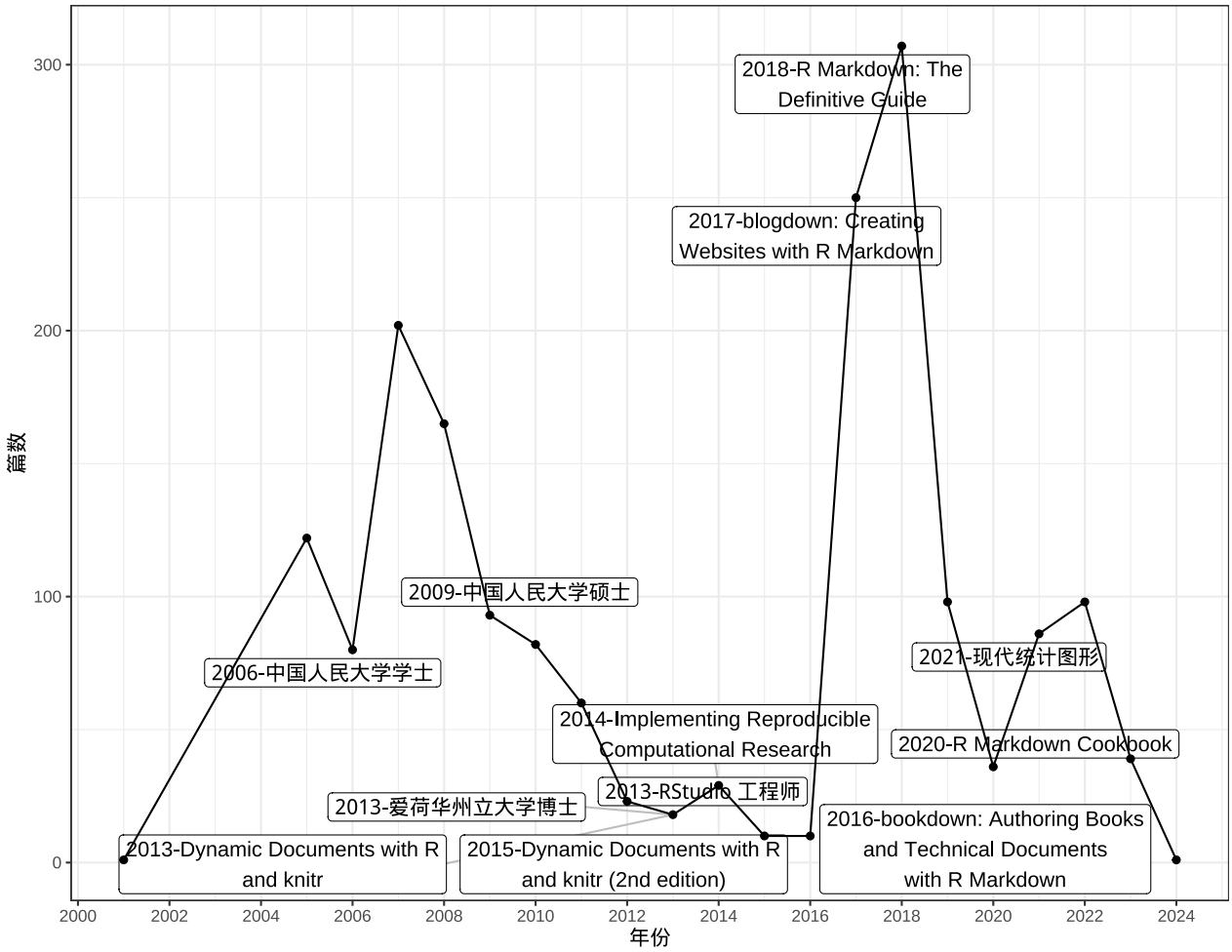


图 17.1: 益辉每年发布的日志数量

2006 年获得中国人民大学学士学位，2009 和 2013 年分别获得中国人民大学硕士和爱荷华州立大学博士学位，在校期间，日志数量持续增加，又陆续创立统计之都，举办中国 R 语言大会。在毕业那年需要完成毕业论文，因此，日志数量明显减少。2013 -2016 年，每年都有书籍出版，期间，有博士毕业、找工作、安家等重要事情，因此，日志数量持续处于低位。稳定后，2017-2018 年除了正常出两本书以外，写了大量的日志，迎来第二个高峰，2018 年，中英文日志数量超过 300 篇。2019-2020 年集中精力在写一本食谱。2021 年第一本中文书《现代统计图形》在 10 年后出版，这主要是 2007-2011 年的工作。2021-2023 年日志数量（2023 年中文日志未发布）处于较低水平。

17.3 数据清洗

以 2001 年的一篇日志为例，展开数据清洗的过程。移除文章的 YAML 元数据，对于文本分析来说，主要是没啥信息含量。

```
remove_yaml <- function(x) {
  x[(max(which(x == "---")) + 1):length(x)]
}
x <- remove_yaml(x)
```

移除「我」「是」「你」「的」「了」「也」等高频的人称、助词、虚词。这些词出现的规律对表现个人风格很重要，且看红楼梦关于后 40 回作者归属的研究，通过比较一些助词、虚词的出现规律，从而看出作者的习惯、文风。这种东西是在长期的潜移默化中形成的，对作者自己来说，都可能是无意识的。

```
library(jiebaR)
# jieba_seg <- worker(stop_word = "data/text/stop_word.txt")
jieba_seg <- worker(stop_word = "data/text/cn_stopwords.txt")
```

添加新词，比如「歪贼」、「谢益辉」等，主要是人名、外号等实体。

```
new_words <- readLines(file("data/text/new_word.txt"))
new_user_word(worker = jieba_seg, words = new_words)
```

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

```
# 分词
x_seg <- segment(x, jieba_seg)
```

分词后，再移除数字和英文

```
remove_number_english <- function(x) {
  x <- x[!grepl("\\d{1,}", x)]
  x[!grepl("[a-zA-Z]", x)]
}
xx <- remove_number_english(x = x_seg)
```

词频统计

```
tmp <- freq(x = xx)
tmp <- tmp[order(tmp$freq, decreasing = T), ]
head(tmp)
```

```
#>       char freq
#> 285   一个    7
#> 397   同学    5
#> 178   一篇    4
#> 356   没有    4
#> 67    鼻音    3
#> 106 田春雨    3
```

ggwordcloud 包绘制词云图可视化词频统计的结果。

```
library(ggwordcloud)
head(tmp, 150) |>
  ggplot(aes(label = char, size = freq)) +
  geom_text_wordcloud(seed = 2022, grid_size = 8, max_grid_size = 24) +
  scale_size_area(max_size = 10)
```

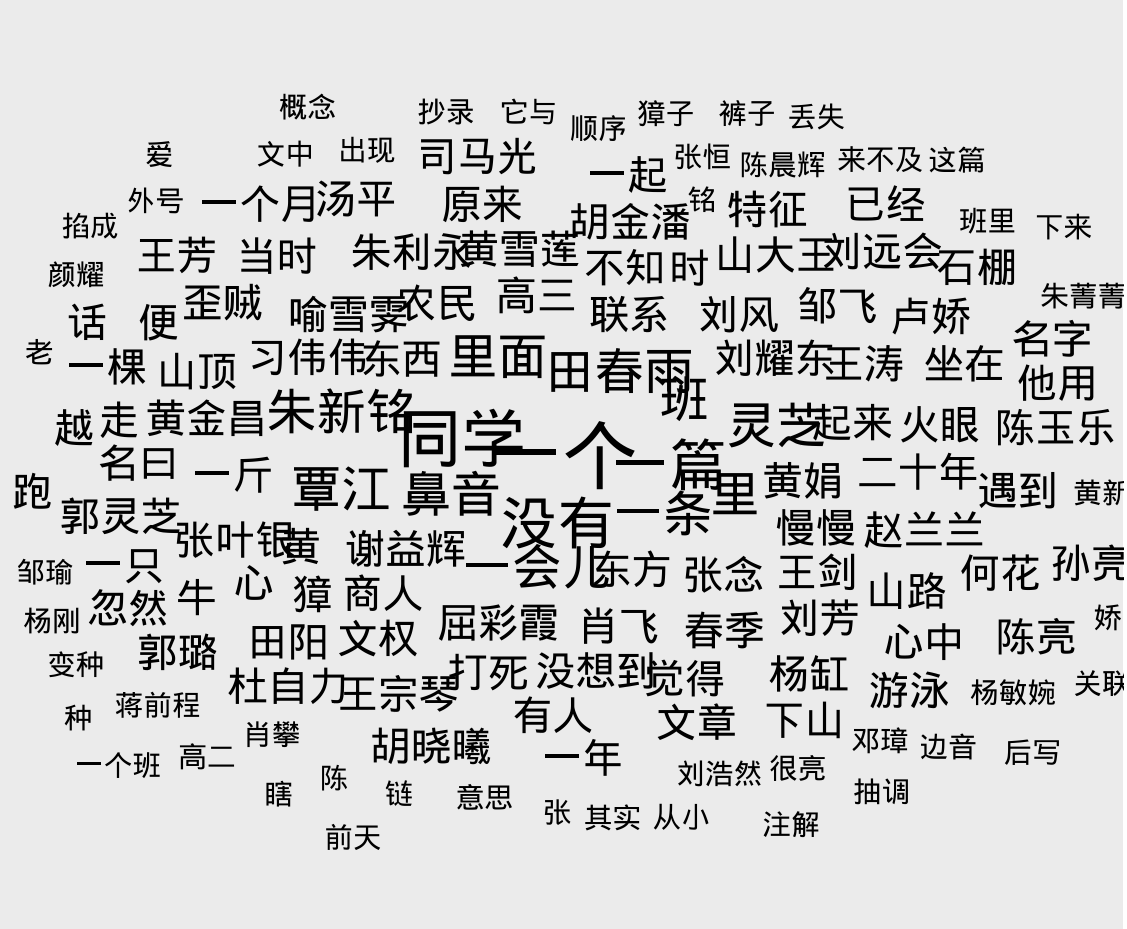


图 17.2: 词云可视化词频结果

计算 TF-IDF 值

```
# tmp = get_idf(x = list(xx))
get_idf(x = list(xx)) |> head()
```

```
#>     name count
#> 1   黄新     0
#> 2   邹瑜     0
#> 3   张恒     0
#> 4 蒋前程     0
#> 5   肖攀     0
```

```
#> 6 刘浩然     0
```

17.4 主题的探索

益辉的日志是没有分类和标签的，所以，先聚类，接着逐个分析每个类代表的实际含义。然后，将聚类的结果作为结果标签，再应用多分类回归模型，最后联合聚类、分类模型，从无监督转化到有监督模型。

topicmodels (Grün 和 Hornik 2011) 基于 tm (Feinerer, Hornik, 和 Meyer 2008) 支持潜在狄利克雷分配 (Latent Dirichlet Allocation，简称 LDA) 和 Correlated Topics Models (CTM) 文本主题建模，这一套工具比较适合英文文本分词、向量化和建模。text2vec 包支持多个统计模型，如 LDA 、LSA 、GloVe 等，文本向量化后，结合统计学习模型，可用于分类、回归、聚类等任务，更多详情见 https://text2vec.org。

接下来使用 David M. Blei 等提出 LDA 算法做主题建模，详情见 LDA 算法原始论文。

```
library(text2vec)
```

首先将所有日志分词、向量化，构建文档-词矩阵 document-term matrix (DTM)

```
# 移除链接
remove_links <- function(x) {
  gsub(pattern = "(<http.*?>)|(\\(http.*?\\))|(<www.*?>)|(\\(www.*?>\\))", replacement = "", x)
}
# 清理、分词、清理
file_list1 <- lapply(file_list, remove_yaml)
file_list1 <- lapply(file_list1, remove_links)
file_list1 <- lapply(file_list1, segment, jiebar = jieba_seg)
file_list1 <- lapply(file_list1, remove_number_english)
```

去掉没啥实际意义的词（比如单个字），极高频词和极低频词。

```
# Token 化
it <- itoken(file_list1, ids = 1:length(file_list1), progressbar = FALSE)
v <- create_vocabulary(it)
# 去掉单个字 减少 3K
v <- v[nchar(v$term) > 1,]
# 去掉极高频词和极低频词 减少 1.4W
v <- prune_vocabulary(v, term_count_min = 10, doc_proportion_max = 0.2)
```

采用 LDA（Latent Dirichlet Allocation）算法建模

```
# 词向量化
vectorizer <- vocab_vectorizer(v)
# 文档-词矩阵 DTM
dtm <- create_dtm(it, vectorizer, type = "dgTMatrix")
#  10 个主题
```

```
lda_model <- LDA$new(n_topics = 9, doc_topic_prior = 0.1, topic_word_prior = 0.01)
# 训练模型
doc_topic_distr <- lda_model$fit_transform(
    x = dtm, n_iter = 1000, convergence_tol = 0.001,
    n_check_convergence = 25, progressbar = FALSE
  )
```

```
#> INFO  [08:15:09.422] early stopping at 175 iteration
#> INFO  [08:15:09.728] early stopping at 50 iteration
```

下图展示主题的分布，各个主题及其所占比例。

```
barplot(
  doc_topic_distr[1, ], xlab = "主题", ylab = "比例",
  ylim = c(0, 1), names.arg = 1:ncol(doc_topic_distr)
)
```

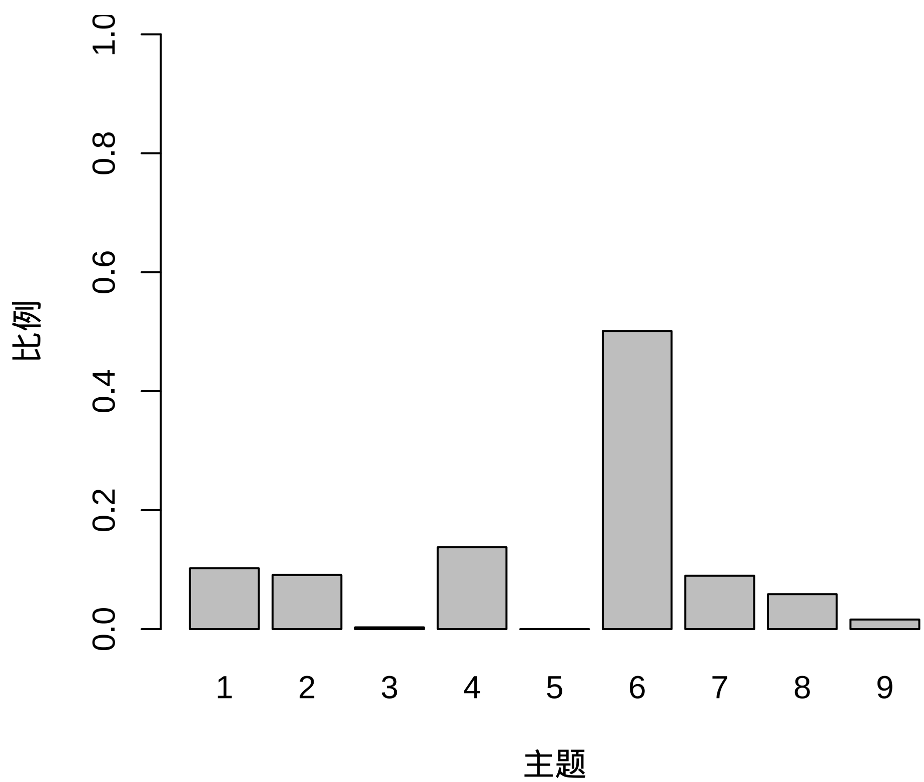


图 17.3: 主题分布

将 9 个主题的 Top 12 词分别打印出来。

```
lda_model$get_top_words(n = 12, topic_number = 1L:9L, lambda = 0.3)
```

```
#>       [,1]   [,2]   [,3]   [,4]     [,5]   [,6]   [,7]   [,8]   [,9]
#>  [1,] "例子" "一首" "社会" "统计"   "代码" "记得" "吱吱" "时代" "网站"
```

```
#>  [2,] "翻译" "歌词" "观点" "会议"   "函数" "不知" "照片" "意义" "数据"
#>  [3,] "字符" "手机" "痛苦" "模型"   "文档" "同学" "好吃" "媒体" "图形"
#>  [4,] "特征" "这首" "教育" "论文"   "文件" "阿姨" "我家" "文化" "域名"
#>  [5,] "作品" "首歌" "人类" "老师"   "变量" "居然" "家里" "现实" "软件"
#>  [6,] "中文" "遗憾" "追求" "分布"   "字体" "看见" "味道" "社交" "服务器"
#>  [7,] "排版" "艺术" "思考" "小子"   "元素" "学校" "厨房" "社区" "邮件"
#>  [8,] "意思" "小说" "强烈" "统计学" "语法" "路上" "在家" "眼中" "提供"
#>  [9,] "风格" "生活" "成功" "参加"   "编译" "听说" "黄瓜" "避免" "编辑"
#> [10,] "主题" "诗词" "接受" "报告"   "图片" "印象" "包子" "造成" "系统"
#> [11,] "伟大" "鸡蛋" "工作" "检验"   "参数" "当时" "叶子" "事实" "浏览器"
#> [12,] "表示" "人间" "努力" "学生"   "生成" "名字" "辣椒" "政治" "注册"
```

结果有点意思，说明益辉喜欢读书写作（主题 1、3、8）、诗词歌赋（主题 2）、统计图形（主题 4）、代码编程（主题 5）、回忆青春（主题 6）、做菜吃饭（7）、倒腾网站（主题 9）。

> **注释**
>
> 提示：参考论文 (L. Zhang, Li, 和 Zhang 2023) 根据 perplexities 做交叉验证选择最合适的主题数量。

17.5 相似性度量

我与益辉日志的相似性度量

17.6 习题

1. **text2vec** 包内置的电影评论数据集 `movie_review` 中 sentiment（表示正面或负面评价）列作为响应变量，构建二分类模型，对用户的一段评论分类。（提示：词向量化后，采用 **glmnet** 包做交叉验证调整参数、模型）
2. 根据 CRAN 上发布的 R 包元数据分析 R 包的描述字段，实现 R 包主题分类。
3. 接习题 2，根据任务视图对 R 包的标记，建立有监督的多分类模型，评估模型的分类效果，并对尚未标记的 R 包分类。（提示：一个 R 包可能同时属于多个任务视图，考虑使用 xgboost 包）

第十八章　时序数据分析

预测是非常古老的话题，几乎人人都想拥有预测未来的能力，唐朝袁天罡和李淳风的故事至今还广为流传。事实上，古时候只有至高无上的皇帝才可以去问钦天监了解星辰大海和国运命脉。时间序列数据的分析，以及根据分析得到的一般规律进行预测是经久不衰的命题。预测既包含一般规律指向的确定性，又有无法预知的不确定性，且**同时**包含认知局限带来的不确定性，后者往往更大。无休止地渴求往往伴随着巨大的挑战，而更大的挑战则是预测效果常常不能满足期待。

```
library(quantmod)    # 获取数据
library(ggplot2)     # 可视化
library(ggfortify)   # 静态展示
library(lmtest)      # 格兰杰因果检验
library(dygraphs)    # 交互展示
```

本章主要从以下几个方面展开：数据获取、数据探索、平稳性诊断、时间序列分解、模型拟合和预测。

18.1　数据获取

Joshua M. Ulrich 开发维护的 **quantmod** 包可以下载国内外股票市场的数据。本节主要以美团股价数据为例，美团自 2018-09-20 在香港挂牌上市，股票代码 3690.HK。首先用 **quantmod** 包 (Ryan 和 Ulrich 2022) 获取美团上市至 2023-11-24 每天的股价数据，包含 Open 开盘价、High 最高价、Low 最低价、Close 闭市价、Adjusted 调整价和 Volume 成交量数据。

```
library(quantmod)
# 美团股票代码 3690
meituan <- getSymbols("3690.HK", auto.assign = FALSE, src = "yahoo")
```

先来看数据的类型，数据类型颇为复杂，是由 `xts` 和 `zoo` 两种类型复合而成，`xts` 类型是继承自 `zoo` 类型的。

```
class(meituan)
```

```
[1] "xts" "zoo"
```

```
str(meituan)
```

```
An xts object on 2018-09-20 / 2023-11-24 containing:
```

```
  Data:    double [1275, 6]
Columns: 3690.HK.Open, 3690.HK.High, 3690.HK.Low, 3690.HK.Close, 3690.HK.Volume ... with 1 more column
 Index:   Date [1275] (TZ: "UTC")
 xts Attributes:
   $ src    : chr "yahoo"
   $ updated: POSIXct[1:1], format: "2023-11-27 06:31:12"
```

数据集 meituan 是一个 xts 类型的时间序列数据对象，时间范围是 2018-09-20 至 2023-11-24，包含 4 个成分，分别如下

- Data 部分显示为 906 行 6 列的双精度浮点存储的数值。
- Columns 部分显示列名，依次是 3690.HK.Open、3690.HK.High、3690.HK.Low 和 3690.HK.Close 等，当列数很多时，显示时会省略。
- Index 部分表示索引列，有序是时间序列数据的本质特点。示例中索引存储数据点产生的先后顺序，索引是用日期来表示的，日期所在的时区是 “UTC”。
- xts 部分是数据类型的一些属性（元数据），说明数据集的来源，什么时候制作的数据。示例中数据是从雅虎财经下载的，下载时间是 2023-11-27 14:31:12。

与时间序列数据相关的数据类型有很多，比如 Base R 提供的 Date 和 POSIX 等，扩展包 **timeDate** 和 **chron** 也都有自己的一套数据类型及处理方法。**xts** 包是处理时间序列数据的主要工具之一，xts 是 eXtensible Time Series 的缩写。为了进一步了解用法，下面举个例子，使用该 R 包的函数 xts() 构造时间序列对象。

```
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
```

18.2 数据探索

18.2.1 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")
```

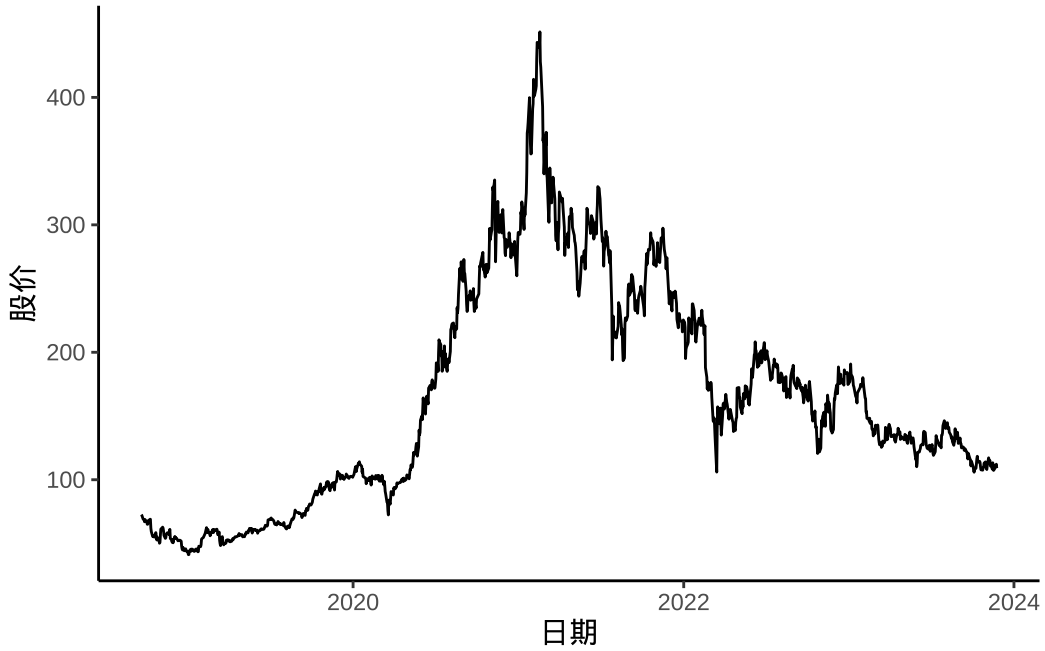


图 18.1: 美团在香港上市以来的股价走势

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")
```

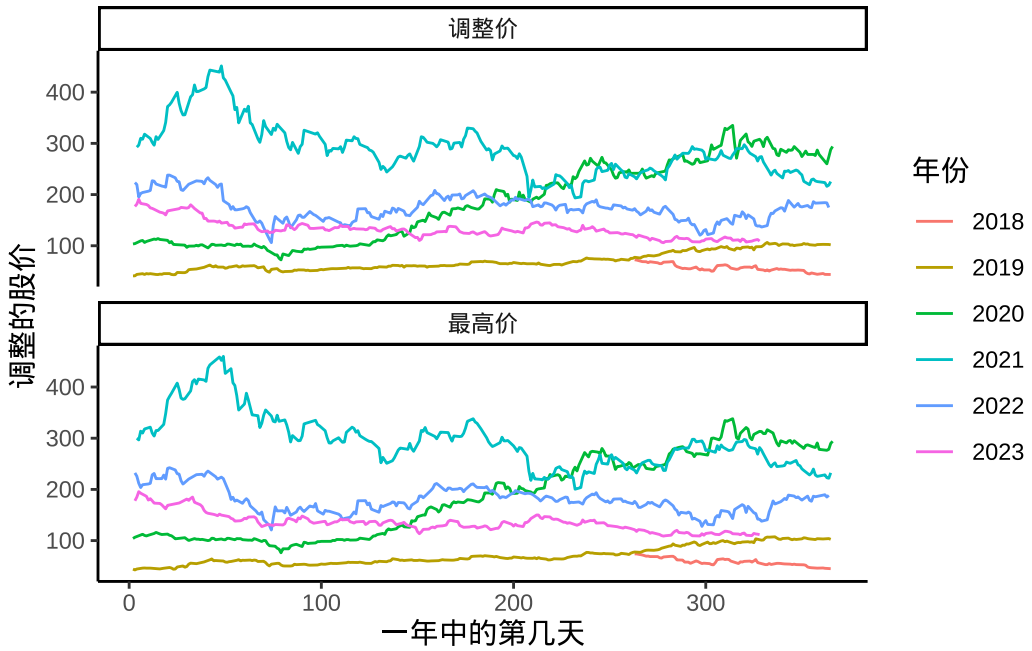


图 18.2: 美团调整的股价逐年走势

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。

18.2.2 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")
```

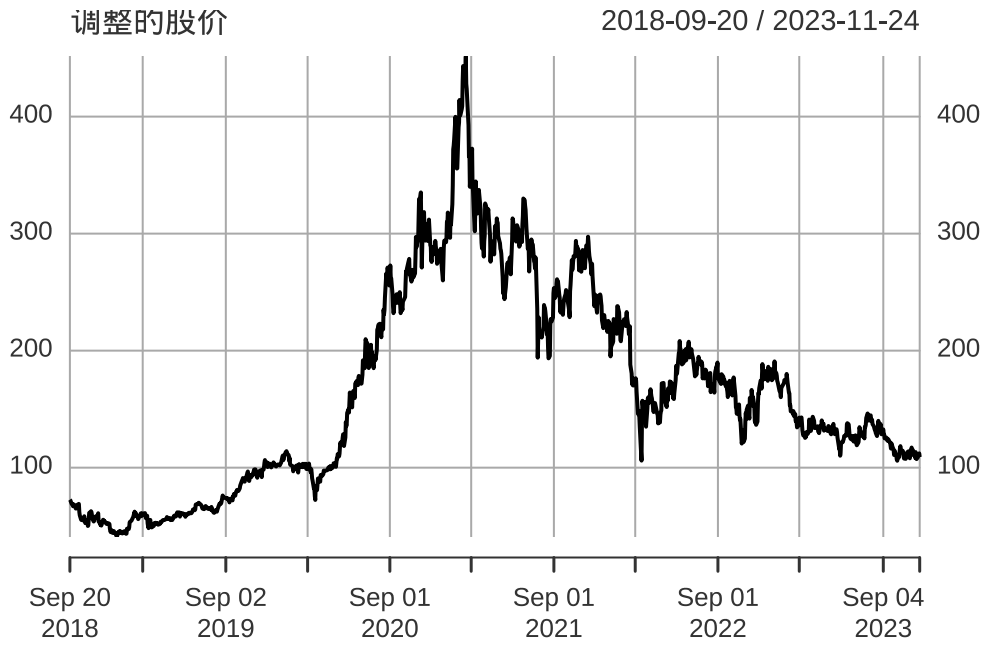


图 18.3: 美团在香港上市以来的股价走势

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 18.4: 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。

18.2.3 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()
```

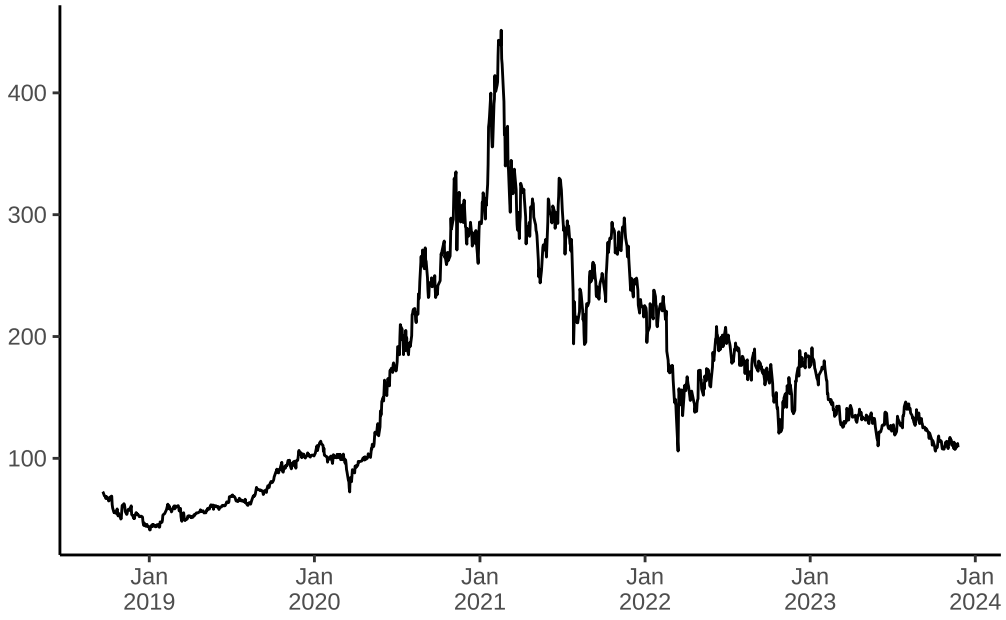


图 18.5: 美团股价走势

18.2.4 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("2023-01-01", "2023-11-24")) |>
  dyAxis(name = "x", axisLabelFormatter = getYearMonth) |>
  dyAxis("y", valueRange = c(0, 500), label = "美团股价") |>
  dyEvent("2020-01-23", "武汉封城", labelLoc = "bottom") |>
  dyShading(from = "2020-01-23", to = "2020-04-08", color = "#FFE6E6") |>
  dyAnnotation("2020-01-23", text = "武汉封城", tooltip = "武汉封城", width = 60) |>
  dyAnnotation("2020-04-08", text = "武汉解封", tooltip = "武汉解封", width = 60) |>
  dyHighlight(highlightSeriesOpts = list(strokeWidth = 2)) |>
  dySeries(label = "调整股价") |>
  dyLegend(show = "follow", hideOnMouseOut = FALSE) |>
  dyOptions(fillGraph = TRUE, drawGrid = FALSE, gridLineColor = "lightblue") |>
  dyUnzoom()
```

图 18.6: 美团股价变化趋势

上图默认展示 YTD 数据，在一个动态的时间窗口内显示数据，假如今天是 2023-07-15，则展示 2023-01-01 至 2023-07-15 的股价数据。在函数 `dyRangeSelector()` 中设定时间窗口参数 `dateWindow`，实现数据范围的筛选。

18.3 平稳性诊断

18.3.1 自相关图

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

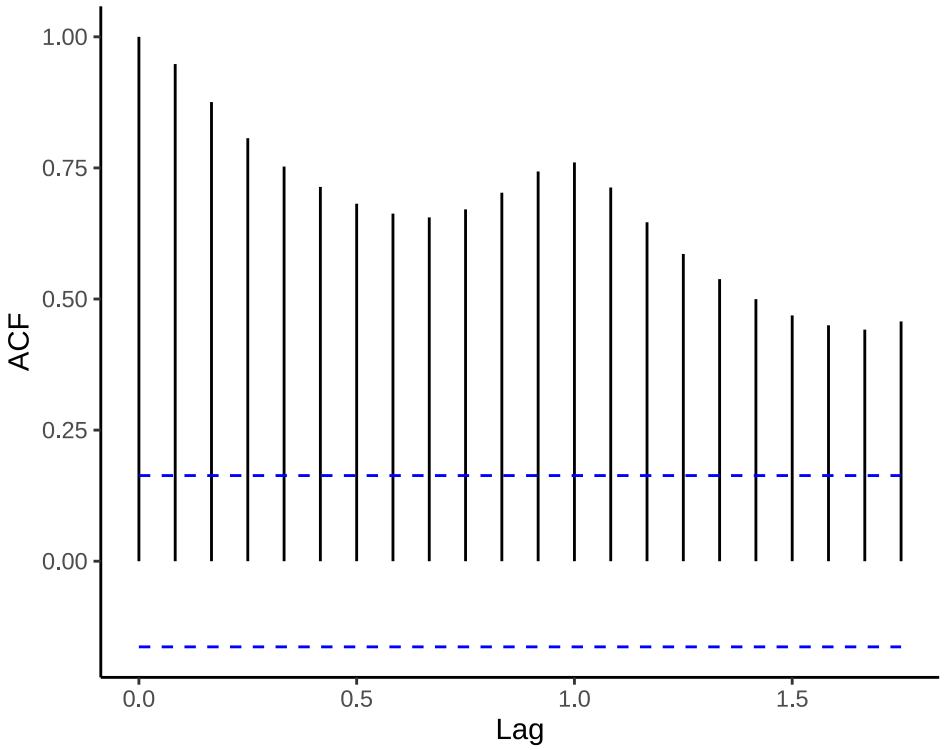


图 18.7: 乘客数量自相关图

18.3.2 偏自相关图

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

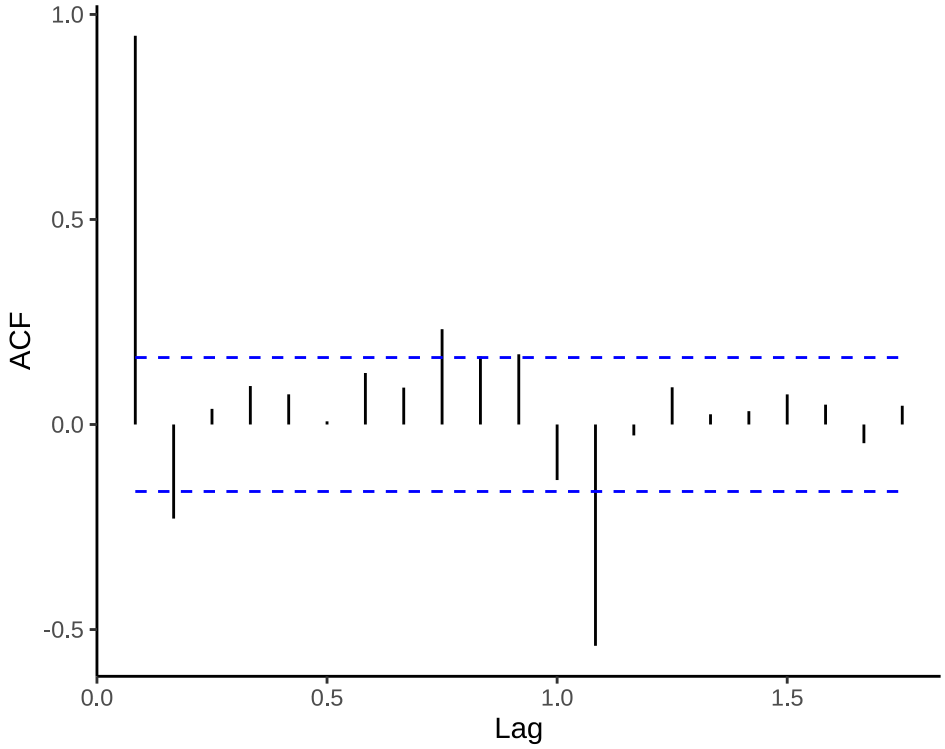


图 18.8: 乘客数量偏自相关图

18.3.3 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
```

18.3.4 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
```

18.3.5 单位根检验

18.3.6 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。

18.4 指数平滑模型

18.4.1 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。

18.4.2 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）

$$y_i = x_i + f_1 y_{i-1} + \cdots + f_p y_{i-p} \tag{18.1}$$

- 卷积过滤（移动平均）

$$y_i = f_1 x_{i+o} + \cdots + f_p x_{i+o-(p-1)} \tag{18.2}$$

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 18.1 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：

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

比照模型方程式 18.2 ，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
```

18.4.3 简单指数平滑

当时间序列不含趋势和季节性成分的时候，可以用简单指数平滑模型来拟合和预测。简单指数平滑模型如下：

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

其中，周期 p

```
air_passengers_exp <- HoltWinters(AirPassengers, gamma = FALSE, beta = FALSE)
air_passengers_exp
```

```
Holt-Winters exponential smoothing without trend and without seasonal component.

Call:
HoltWinters(x = AirPassengers, beta = FALSE, gamma = FALSE)

Smoothing parameters:
 alpha: 0.9999339
 beta : FALSE
 gamma: FALSE

Coefficients:
      [,1]
a 431.9972
```

预测的残差平方和 SSE sum-of-squared-errors

```
air_passengers_exp$SSE
```

```
[1] 162510.6
```

```
# plot(air_passengers_exp)
autoplot(air_passengers_exp) +
  theme_classic()
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_line()`).
```

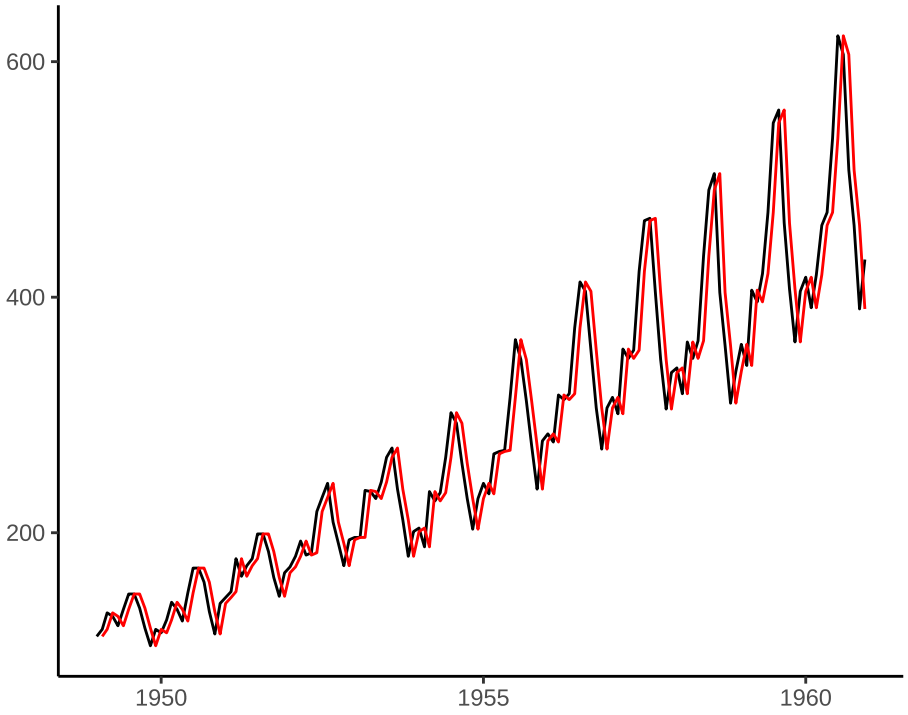


图 18.9: 简单指数平滑模型

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)
```

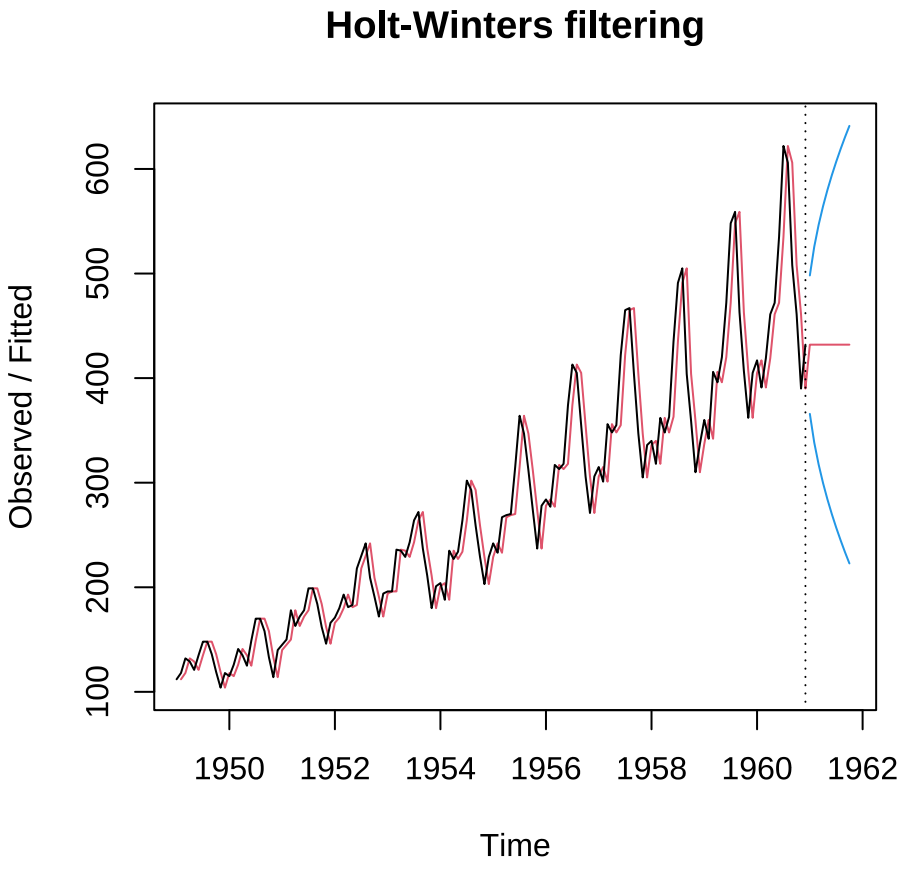


图 18.10: 简单指数平滑模型预测

18.4.4 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) 。

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

```
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)
```

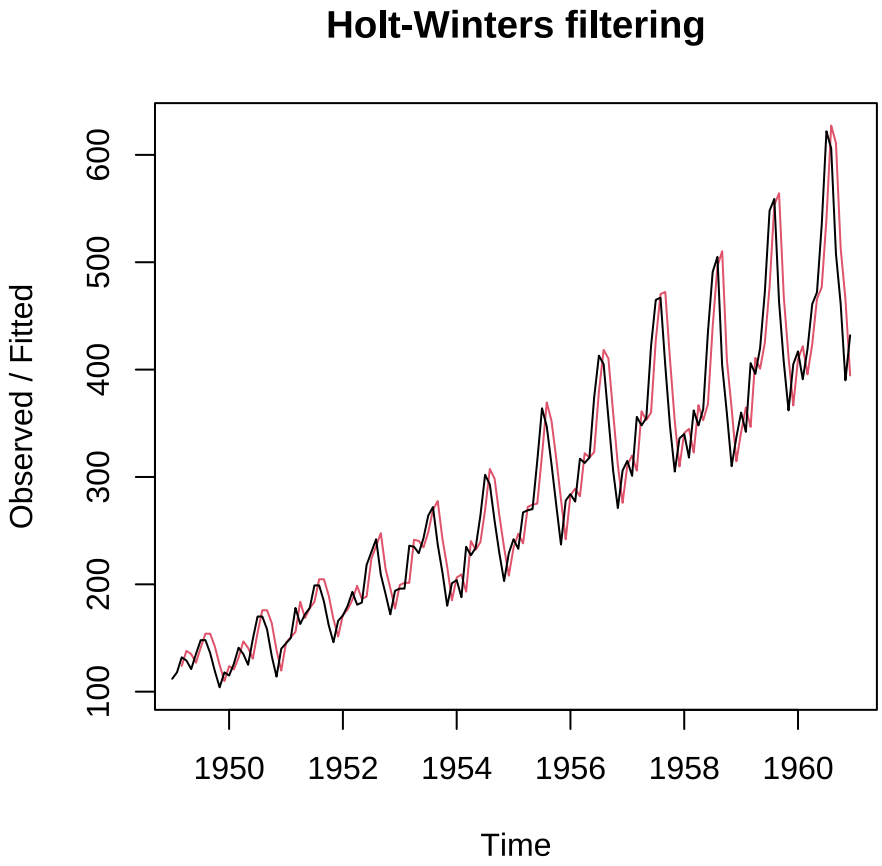


图 18.11: holt 指数平滑模型

18.4.5 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 or values outside the scale range
(`geom_line()`).
```

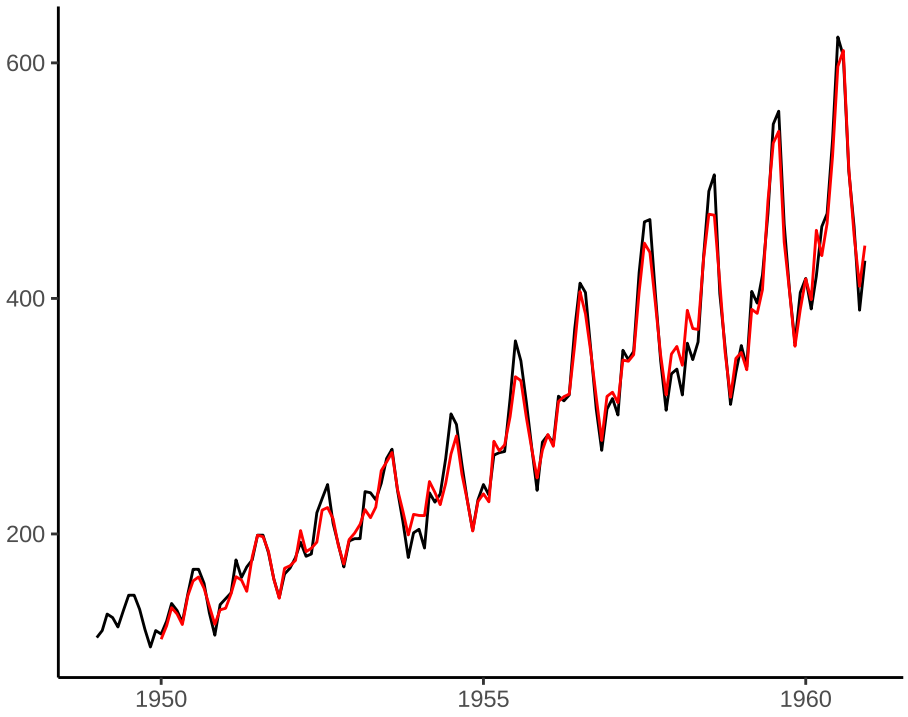


图 18.12: 可加 Holt-Winters 平滑模型拟合

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

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

```
Warning: Removed 12 rows containing missing values or values outside the scale range
(`geom_line()`).
```

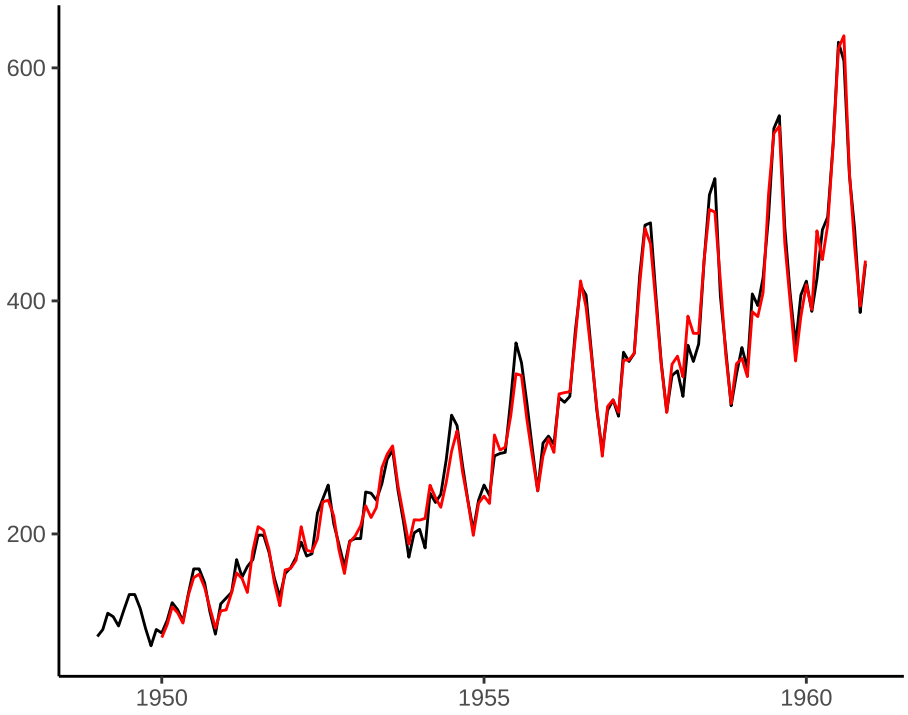


图 18.13: 可乘 Holt-Winters 平滑模型拟合

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

18.5 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

18.5.1 函数 decompose()

函数 decompose() 分解

```
air_decomp_add <- decompose(x = AirPassengers, type = "additive")
```

函数返回一个列表，包含 6 个元素，分别是 x 原始序列，seasonal 季节性成分，figure 估计的季节图，trend 趋势成分，random 剩余成分，type 分解方法。

```
# plot(air_decomp_add)
autoplot(air_decomp_add) +
  theme_classic()
```

```
Warning: Removed 24 rows containing missing values or values outside the scale range
(`geom_line()`).
```

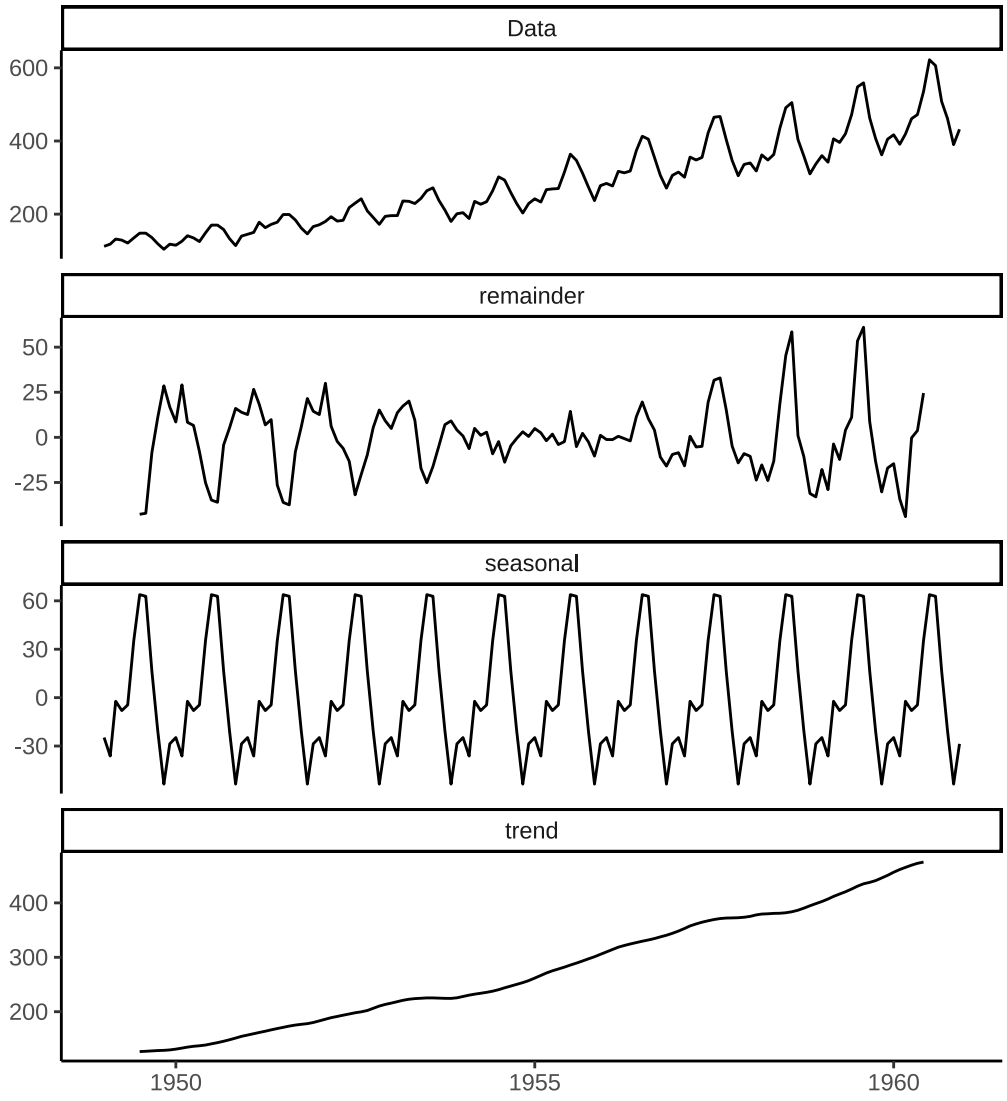


图 18.14: 变化趋势的分解

去掉季节性部分

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

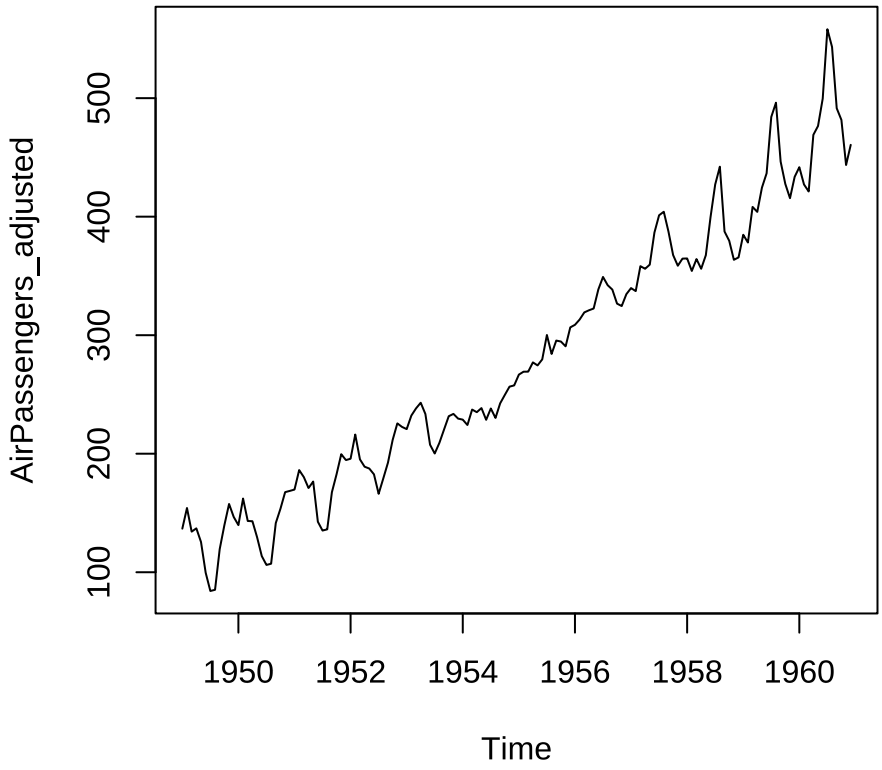


图 18.15: 季节性调整

18.5.2 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()
```

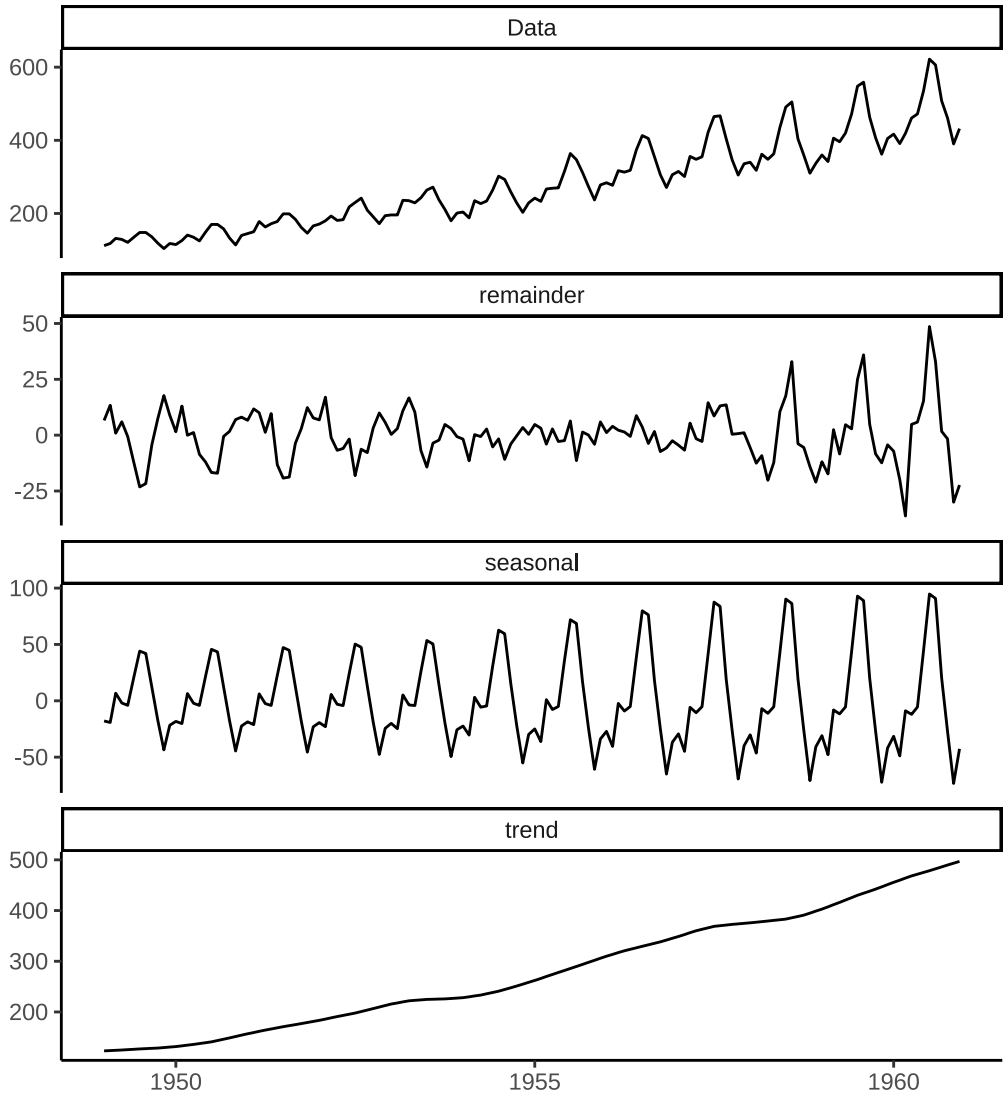


图 18.16: 变化趋势的分解

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
```

18.6 经典时间序列模型

18.6.1 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
```

18.6.2 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
```

18.6.3 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
```

18.7 总结

方法没有好坏，只有适合与否。Holt-Winter 适合预警任务，算法简单，可以及时出预测结果，仅需要一步预测，不需要给出多步预测，要求快，以便迅速作出反应。Prophet 实现的贝叶斯结构可加模型适合短期预测任务，只要在可容许的时间范围内出结果即可，可以迅速出结果当然更好，需要给出多步预测结果，且结果需要强解释性，以便提前做一些商家供给、平台资源的分配。商分模型常常需要比较强的可解释性，算法策略模型重在预测精准度，对可解释性要求不高。

在时间序列数据的可视化方面，除了 Base R 提供的绘图方法外，静态的时序图 **lattice** 和 **ggplot2** 都不错，而交互式图形推荐使用 **plotly** 和 **dygraphs**。

PortfolioAnalytics 包做投资组合优化，均值-方差，收益和风险权衡。Rmetrics 提供系列时间序列数据分析和建模的 R 包，包括投资组合优化 **fPortfolio**、多元分析 **fMultivar**、自回归条件异方差模型 **fGarch**、二元相依结构的 Copulae 分析 **fCopulae** 、市场和基础统计 **fBasics** 。

fable 一元到多元时间序列预测问题，提供 ETS、ARIMA、TSLM 等模型，并有书籍时间序列预测原则。值得一提，**forecast** 包开发者 Rob J Hyndman 称已不再开发新的功能，推荐大家使用 **fable** 包。**feasts** 包辅助特征抽取、序列分解、汇总统计和绘制图形等，插件包 **fable.prophet** 接入 Prophet 的预测能力。**timetk** 时间序列数据处理、分析、预测和可视化工具箱，提供一致的操作方式，试图形成完成的解决方案。The Rmetrics Association 开发了一系列 R 包专门处理金融时间序列数据，比如 **fGarch** 包提供条件自回归异方差模型。

从时间序列中寻找规律，这样才是真的数据建模，从数据到模型，而不是相反 Finding Patterns in Time Series，识别金融时间序列的模式和统计规律。

第五部分

优化建模

第十九章　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。

19.1　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) (C.-H. 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。

19.2　对数似然与损失函数

19.2.1　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

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

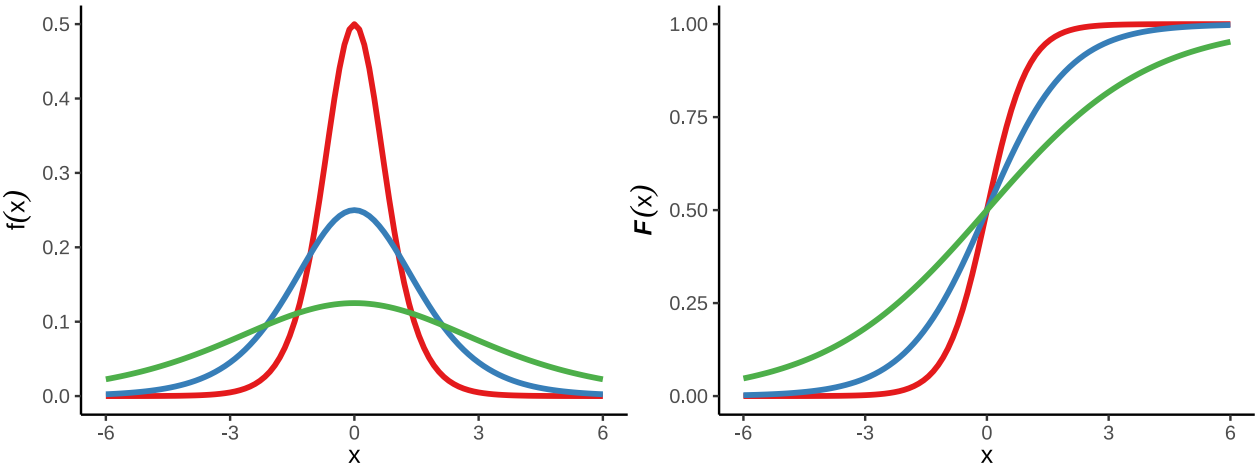
密度函数的形式为

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

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 19.1: 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。

19.2.2 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：

$$\begin{aligned} \mathcal{L}(\alpha, \boldsymbol{\beta}) &= \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i} \\ &= \prod_{i=1}^{n} \Big(\frac{e^{\alpha+\boldsymbol{x}_i^\top\boldsymbol{\beta}}}{1+e^{\alpha+\boldsymbol{x}_i^\top\boldsymbol{\beta}}}\Big)^{y_i} \Big(\frac{1}{e^{\alpha+\boldsymbol{x}_i^\top\boldsymbol{\beta}}}\Big)^{1-y_i} \end{aligned} \tag{19.1}$$

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

$$\begin{aligned} \ell(\alpha, \boldsymbol{\beta}) &= \log \mathcal{L}(\alpha, \boldsymbol{\beta}) \\ &= \sum_{i=1}^{n} \Big[y_i \log(p_i) + (1-y_i)\log(1-p_i)\Big] \\ &= \sum_{i=1}^{n} \Big[y_i \log\Big(\frac{e^{\alpha+\boldsymbol{x}_i\boldsymbol{\beta}}}{1+e^{\alpha+\boldsymbol{x}_i\boldsymbol{\beta}}}\Big) + (1-y_i)\log\Big(\frac{1}{e^{\alpha+\boldsymbol{x}_i\boldsymbol{\beta}}}\Big)\Big] \end{aligned} \tag{19.2}$$

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

$$\begin{aligned} \frac{\partial \ell(\alpha, \boldsymbol{\beta})}{\partial \alpha} &= \sum_{i=1}^{n} \Big[\big(\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}\big)\frac{\partial p_i}{\partial \alpha}\Big] \\ \frac{\partial \ell(\alpha, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^{n} \Big[\big(\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}\big)\frac{\partial p_i}{\partial \beta}\Big] \\ &= \sum_{i=1}^{n} \Big[\big(\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}\big)p_i(1-p_i)\boldsymbol{x}_i^\top\Big] \end{aligned} \tag{19.3}$$

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。

19.3 数值优化问题求解器

19.3.1 `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 19.2 ，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 19.2 所示，二维情形下的逻辑回归模型的负对数似然函数曲面。

当用 Base R 函数 `optim()` 来求解时，发现 Nelder-Mead 算法收敛慢，易陷入局部最优解，即使迭代 10000 次，与真值仍然相去甚远。当用 SANN （模拟退火算法）求解此 11 维非线性无约束优化问题时，迭代 10000 次后，比较接近真值。

```
optim(
  par = rep(1, 11), # 初始值
  fn = log_logit_lik, # 目标函数
  method = "SANN",
  control = list(maxit = 10000)
)
```

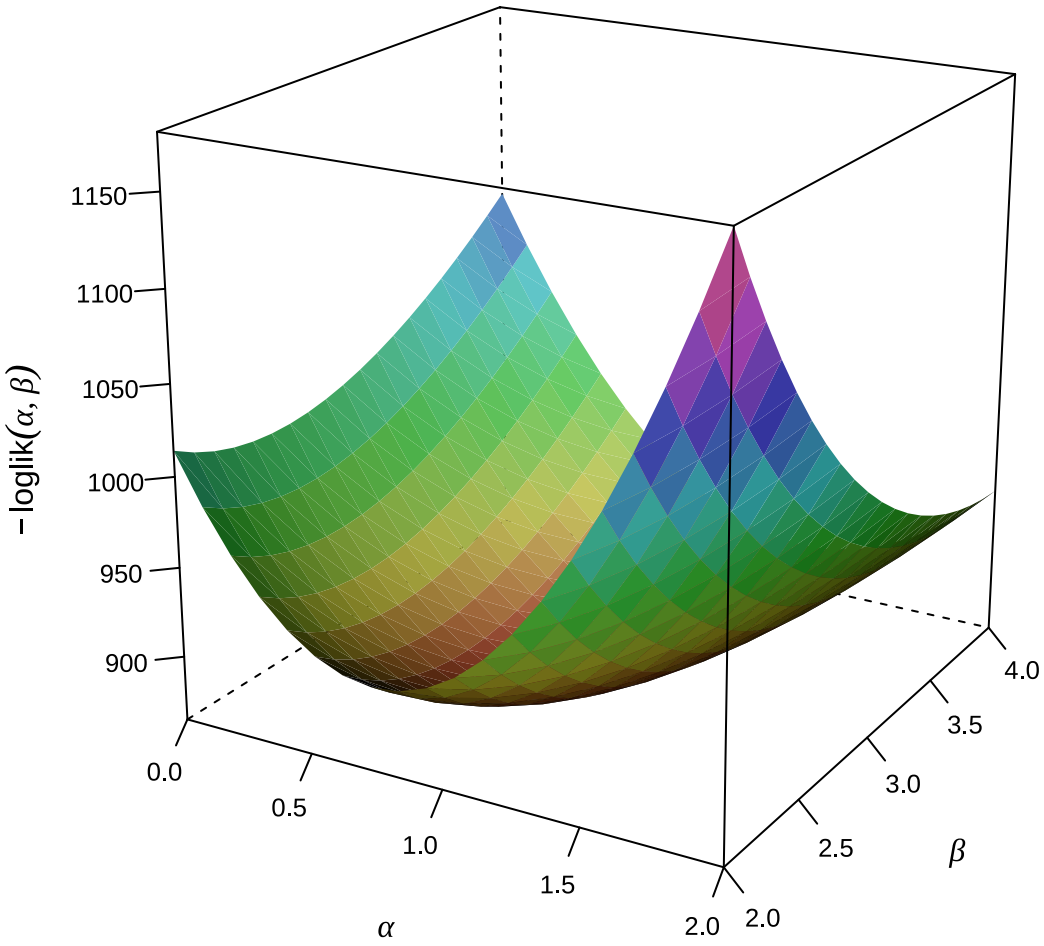


图 19.2: 二维情形下的逻辑回归模型的负对数似然函数曲面

```
#> $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。

19.3.2 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()`。

19.3.3 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")
```

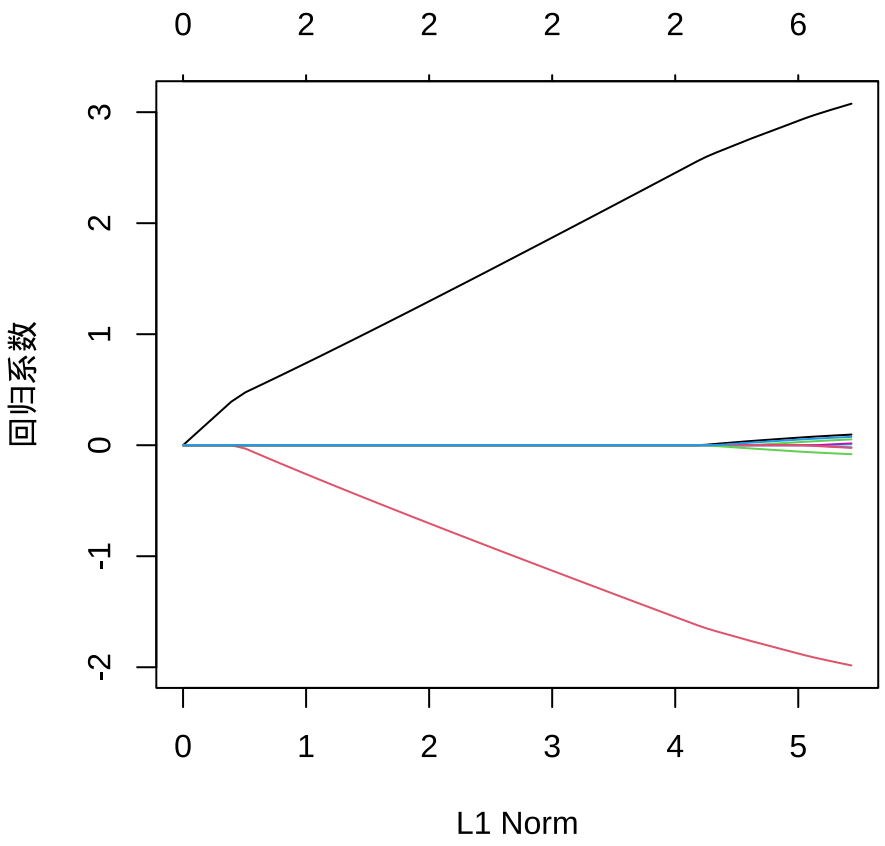


图 19.3: 回归系数的迭代路径

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"
)
```

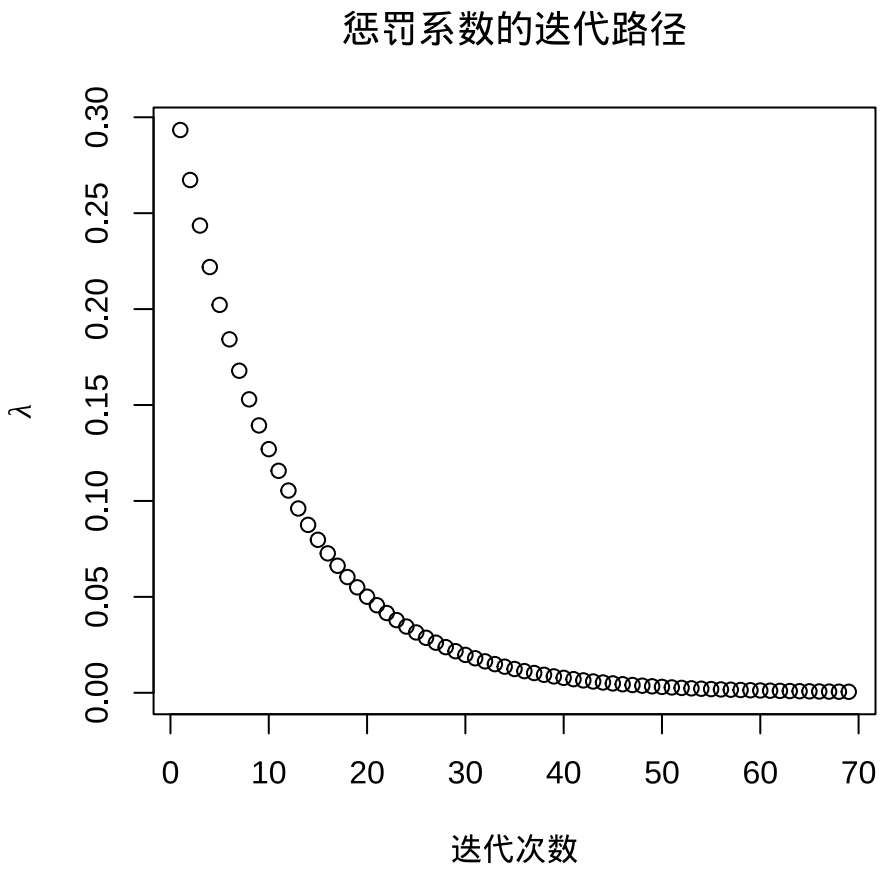


图 19.4: 惩罚系数的迭代路径

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。

19.4 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。

19.4.1 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
```

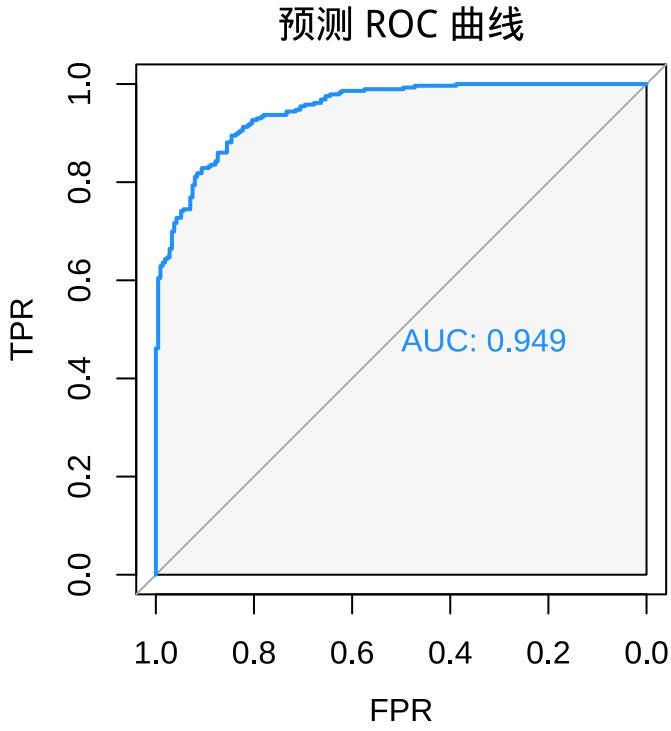


图 19.5: ROC 曲线

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。

19.4.2 Wilcoxon 检验

对每个标签的预测概率指定服从均匀分布，相当于随机猜测，所以最后 ROC 会接近对角线，而且样本量越大越接近，AUC 会越来越接近 0.5。如果预测结果比随机猜测要好，Wilcoxon 检验会显著，预测

效果越好检验会越显著，表示预测 pred 和观测 y 越接近。

```
wilcox.test(pred ~ y, data = dat_test)
#>
#>  Wilcoxon rank sum test with continuity correction
#>
#> data:  pred by y
#> W = 3140, p-value < 2.2e-16
#> alternative hypothesis: true location shift is not equal to 0
```

第二十章　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 (刘浩洋等 2020) 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。

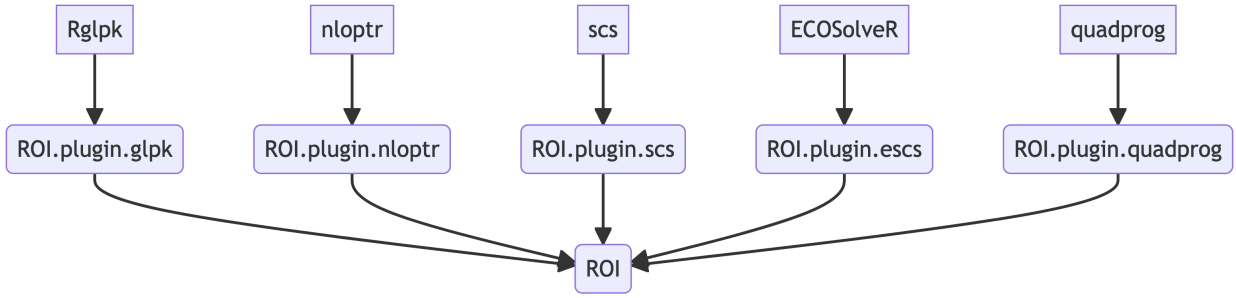


图 20.1: 数值优化扩展包、插件包和 ROI 包的关系图

```
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
  )
}
```

20.1 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：

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

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 20.1: **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()`

20.2 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 20.2 展示无约束和有约束条件下的解。图中橘黄色线围成的三角形区域是可行域，红点表示无约束下求解器 nloptr.slsqp 获得的解 $(0, 1)$，真正的无约束解是蓝点所在位置为 $(-4/3, 1/3)$，黄点表示线性约束下求解器 nloptr.slsqp 获得的解 $(1/6, 11/6)$。所以，不能用二次优化的求解器去求解无约束的二次优化问题。

quadprog 包在求解约束条件下的严格凸二次优化问题时，同时给出无约束条件下的解。这个包自定义

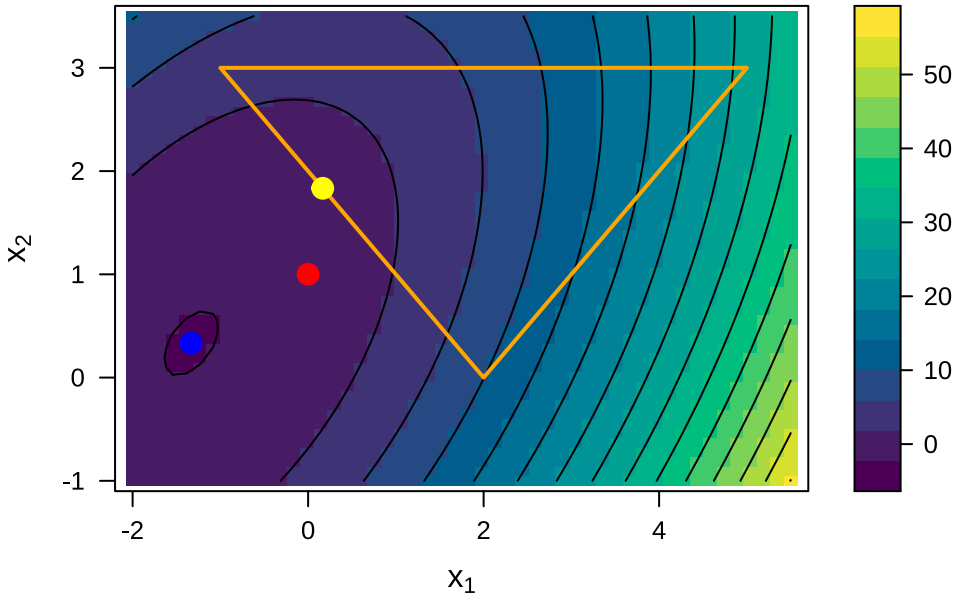


图 20.2: 对比无约束和有约束条件下的解

了一套二次优化问题的符号，查看求解函数 `solve.QP()` 的说明，略作对应后，求解上述优化问题的代码如下。

```
library(quadprog)
sol <- solve.QP(
  Dmat = Dmat, dvec = -dvec, Amat = t(-Amat), bvec = -bvec
)
sol
```

```
#> $solution
#> [1] 0.1666667 1.8333333
#>
#> $value
#> [1] -0.08333333
#>
#> $unconstrained.solution
#> [1] -1.3333333  0.3333333
#>
#> $iterations
#> [1] 2 0
#>
#> $Lagrangian
#> [1] 1.5 0.0 0.0
#>
#> $iact
```

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

其中，返回值的 `unconstrained.solution` 表示无约束下的解，与预期的解一致，这就没有疑惑了。可见，约束二次优化问题和无约束二次优化问题的求解器不同。

20.3 凸锥优化

20.3.1 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 20.3 所示。

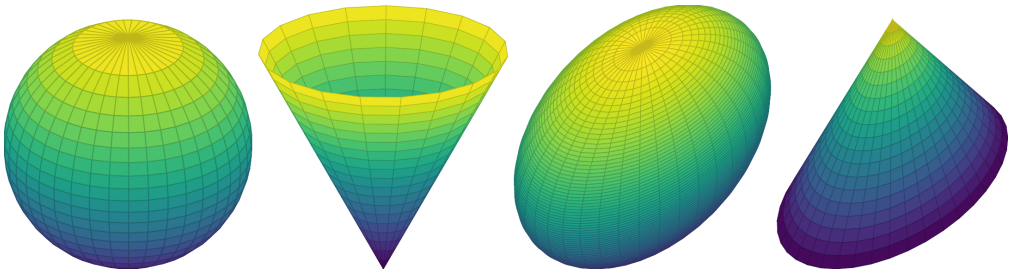


图 20.3: 常见的三维凸锥

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 20.3 所示，边界是一个三维曲面，曲面及其内部构成一个凸锥。

20.3.2 零锥

零锥的定义如下：

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

常用于表示线性等式约束。

20.3.3 线性锥

线性锥（Linear Cone）的定义如下：

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

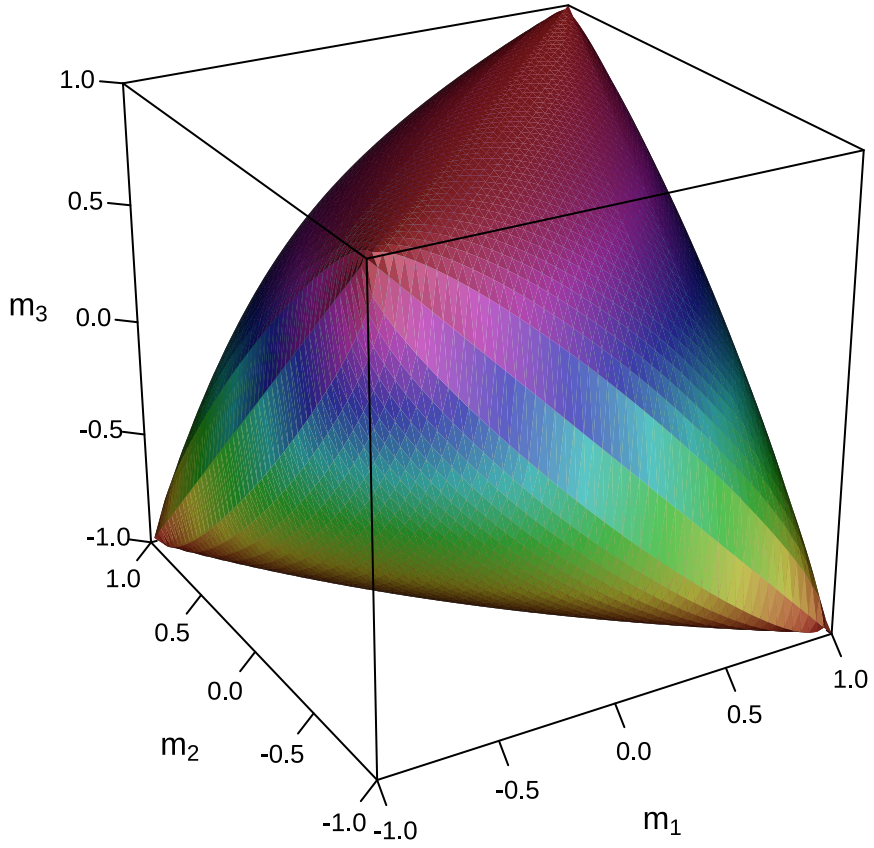


图 20.4: 锥

常用于表示线性不等式约束。

20.3.4 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)
```

```r
    ),
    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
```

```r
# 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。

```r
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
```

```r
# 目标函数值
sol$objval
```

```
#> [1] 25.35542
```

20.3.5 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
```

20.3.6 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
```

20.3.7 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
```

20.4 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 20.2: 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。

20.4.1 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 20.5 ，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)
```

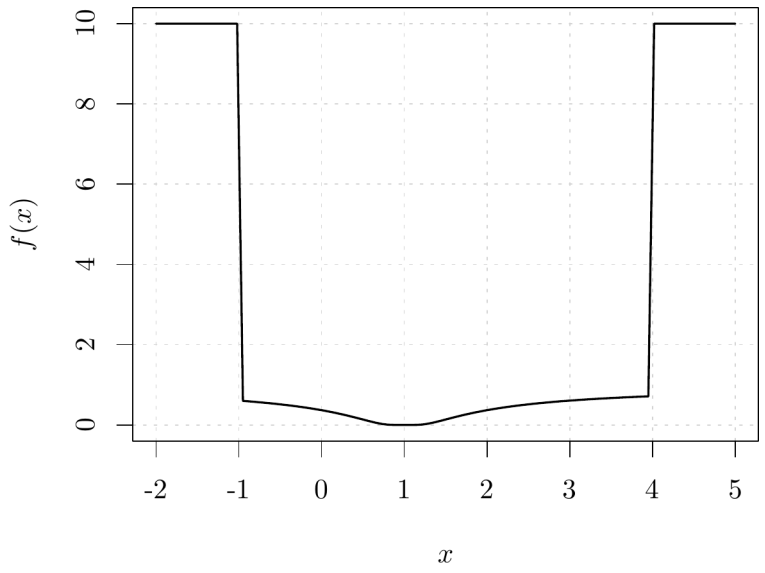


图 20.5: 一维函数图像

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。

20.4.2 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 20.6 所示，可以获得关于隐函数的大致情况。

最后，获得暴力网格搜索的结果，目标函数在 $(2.8,-0.9)$ 处取得最小值 -0.02159723。总的来说，这是一个近似结果，如果进一步缩小搜索区域，将网格划分得越细，搜索的结果将越接近全局最小值。

```
df[df$fn == min(df$fn), ]
```

```
#>      x1   x2          fn
#> 930 2.8 -0.9 -0.02159723
```

将求隐函数极值的问题转为含非线性等式约束的非线性优化问题。

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & y \\ \text{s.t.} \quad & f(x_1, x_2, y) = 0 \end{aligned}$$

由于等式约束非常复杂，手动计算等式约束的雅可比矩阵不可行，可以用 **numDeriv** 包的函数 `jacobian()` 计算等式约束的雅可比矩阵。考虑到本例中仅含有一个等式约束，雅可比矩阵退化为梯度向量，这可以用 **numDeriv** 包的另一个函数 `grad()` 计算。

```
# 等式约束
heq <- function(x) {
  f1 <- (x[1] * x[3] - 0.5)^2 + 2 * x[1] * x[2]^2 - x[3] / 10
  f2 <- (x[1] - 0.5 - exp(-x[2] + x[3]))^2 + x[2]^2 - x[3] / 5 + 3
  x[3] - sin(f1) * exp(-f2)
```

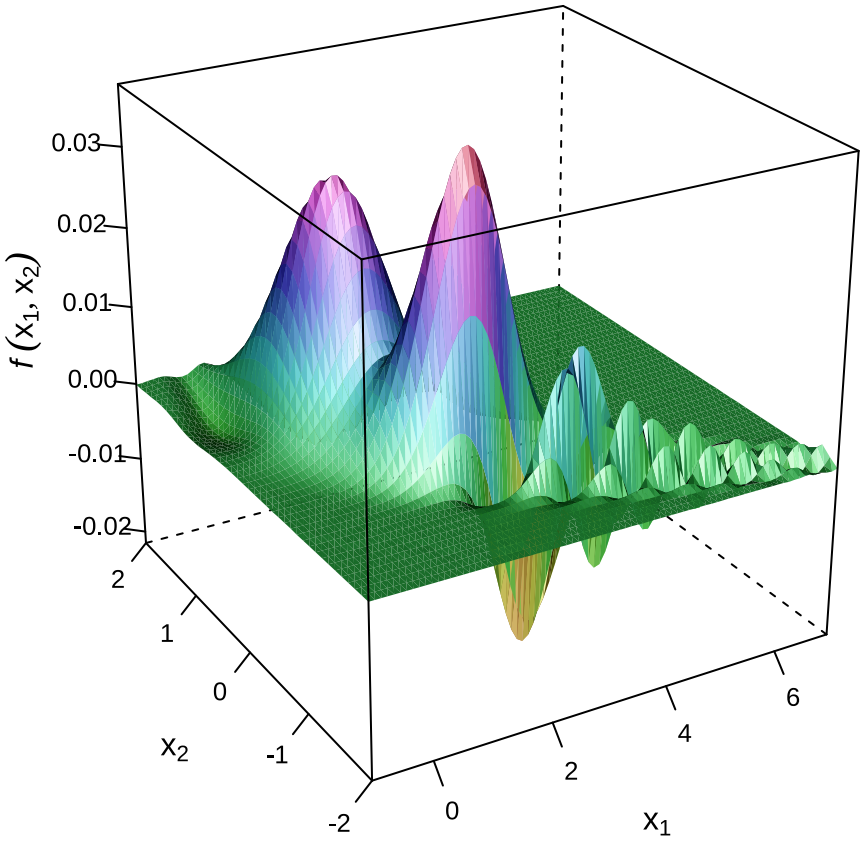


图 20.6: 隐函数图像

```
}
# 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。

20.4.3 多元无约束优化

20.4.3.1 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 20.7 所示，这是一个多模态的函数，有许多局部极小值。如果采用 BFGS 算法寻优容易陷入局部极值点。

不失一般性，考虑函数维数 $n = 20$ ，决策变量 $x_i \in [-50, 50], i = 1, 2, \ldots, n$ 的情况。

```
op <- OP(
  objective = F_objective(fn, n = 20L),
  bounds = V_bound(ld = -50, ud = 50, nobj = 20L)
)
```

调全局优化器求解优化问题。

```
nlp <- ROI_solve(op, solver = "nloptr.directL")
# 最优解
nlp$solution
```

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

```
# 目标函数值
nlp$objval
```

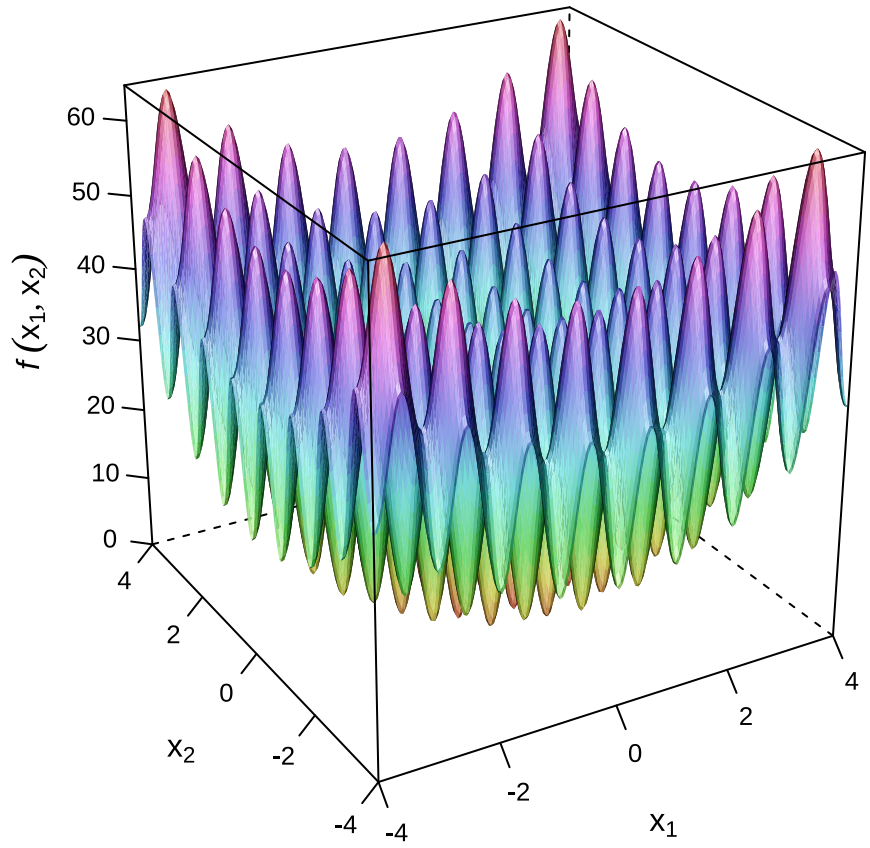


图 20.7: 二维 Rastrigin 函数图像

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

20.4.3.2 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 20.8a 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 20.8b 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
```

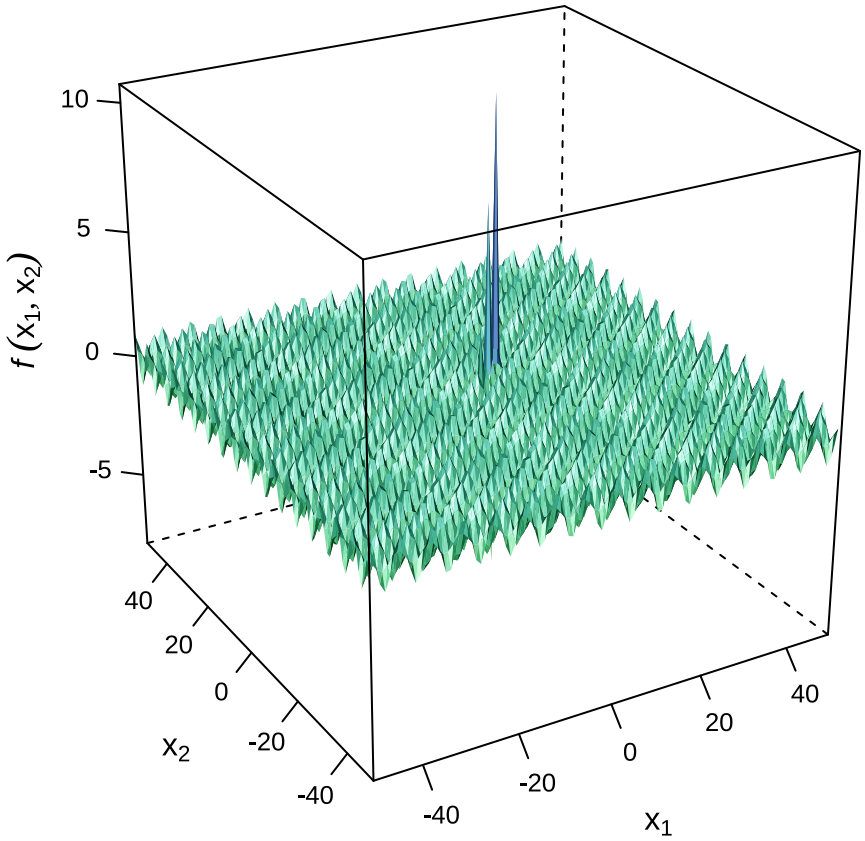
```
#> [1]  0.00000 22.22222
```

```
nlp$objval
```

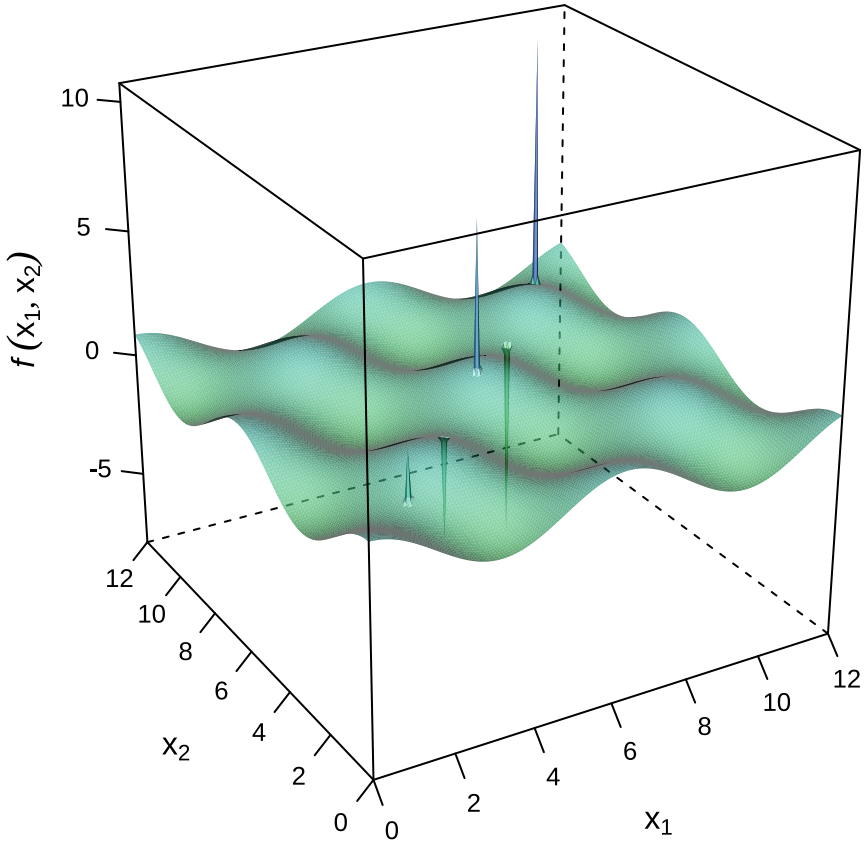
```
#> [1] -0.9734211
```

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
```



(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 20.8: 局部放大前后的函数图像

```
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。

20.4.4 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。

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

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 20.9 ，函数曲面和香蕉函数有些相似。

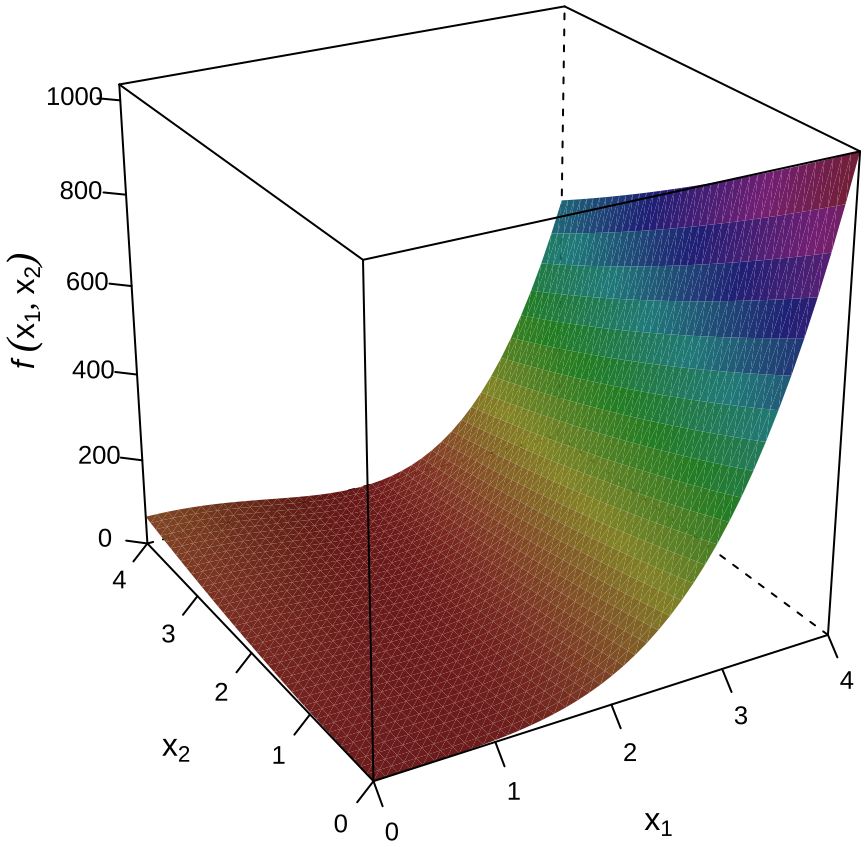


图 20.9: 类香蕉函数的曲面图

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，

20.4.4.1 `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。

20.4.4.2 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
#>     3719      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。

20.4.4.3 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。

20.4.4.4 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] 368.1061
```

```
# 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
```

20.4.5 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
#>      405       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。

20.4.6 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 20.3: 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

20.4.6.1 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。

20.4.6.2 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] 1.062118 4.632624 3.958203 1.320844
```

```
nlp$objval
```

```
#> [1] 17.50024
```

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。

20.5 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。

20.5.1 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。

20.5.2 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。

20.5.3 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
```

20.5.4 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。

20.5.5 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
```

20.6 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。

20.7 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 20.10 所示。（提示：容错能力低的求解器一般无法求解。Lingo 给出一个局部最优解 $(-46.14402, -0.8879601)$，目标函数值为 -2.645518，仅供参考。）

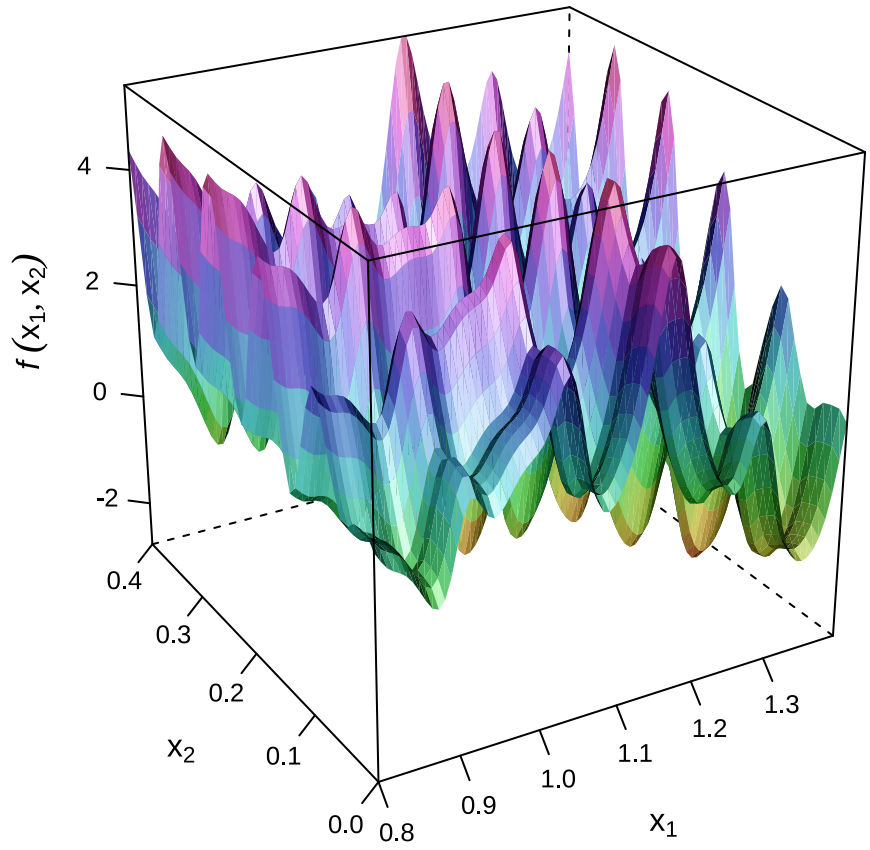


图 20.10: 目标函数的曲面图

7. 求解如下非线性约束优化问题。

$$\min_{\boldsymbol{x}} \quad x_1^2 \sin(x_2) + x_2^2 \cos(x_1)$$

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

第二十一章　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
  )
}
```

21.1　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 21.1 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？

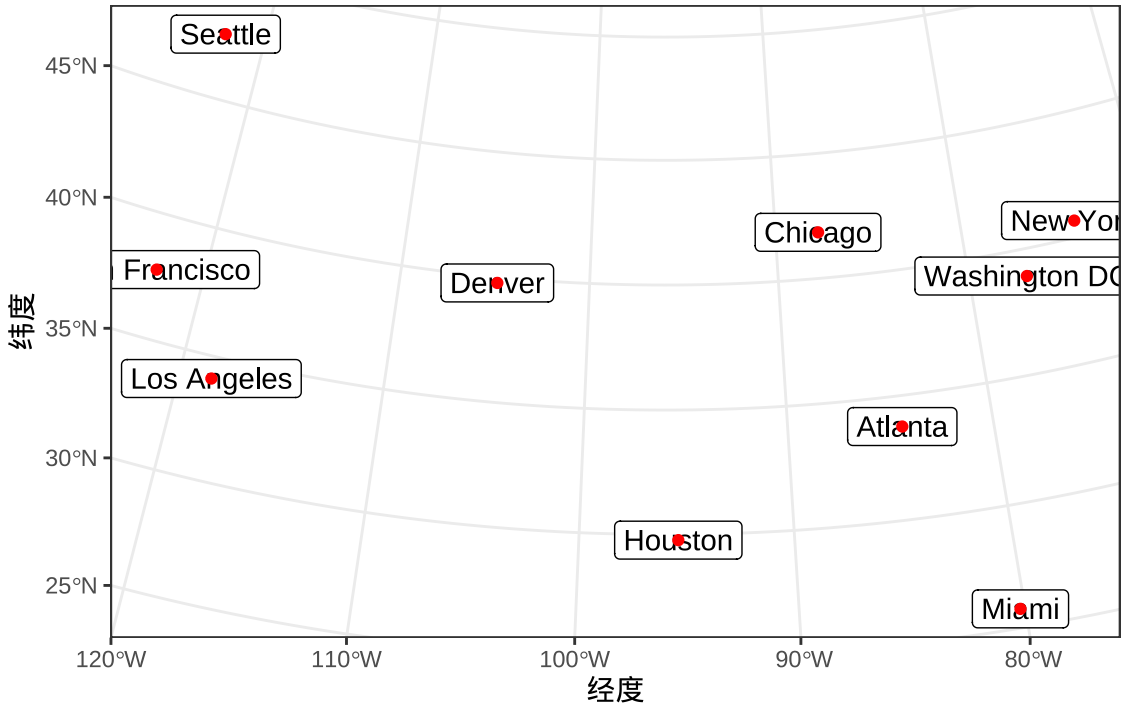


图 21.1: 10 个城市的分布图

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 21.2 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。

21.2 投资组合问题

作为一个理性的投资者，希望回报最大而风险最小，给定投资和回报的约束条件下，选择风险最小的组合。一个简单的马科维茨投资组合优化问题如下：

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \boldsymbol{w}^\top \hat{\Sigma} \boldsymbol{w} \\ \text{s.t.} \quad & A\boldsymbol{w}^\top \leq \boldsymbol{b} \end{aligned}$$

其中，$\boldsymbol{w}$ 是权重向量，每个分量代表对投资对象的投资比例，$\hat{\Sigma}$ 是关于投资对象的协方差矩阵，约束条件中包含两个部分，一个是权重之和为 1，一个是投资组合的收益率达到预期值。下面基于 12 个科技公司公开的股价数据介绍此组合优化问题。

首先利用 **quantmod** 包获取微软、谷歌、亚马逊、惠普、甲骨文、英特尔、威瑞森、eBay、AT&T、Apple、Adobe 和 IBM 等 12 支股票的历史股价数据。根据 2022-11-01 至 2022-12-01 期间的股票调整价，计算各支股票天粒度的收益率。收益率可以看作一个随机变量，收益率的波动变化，即随机变量的方差，可以看作风险。

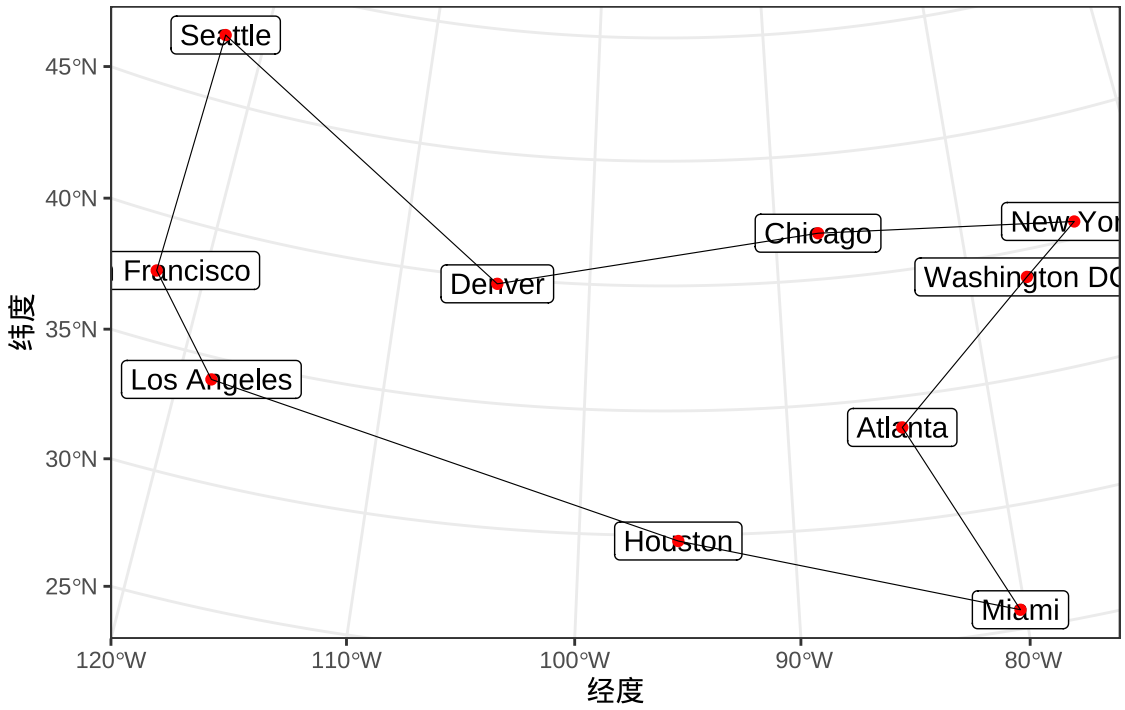


图 21.2: 10 个城市的路线图

```
# 12 支股票的收益率
tech_stock_return <- readRDS(file = "data/tech_stock_return.rds")
DD <- 100 * tech_stock_return
# 平均收益率
r <- mean(DD)
r
```

```
#> [1] 0.3476413
```

```
# 目标函数
foo <- Q_objective(Q = cov(DD), L = rep(0, ncol(DD)))
# 投资约束
full_invest <- L_constraint(rep(1, ncol(DD)), "==", 1)
# 回报约束
target_return <- L_constraint(apply(DD, 2, mean), "==", r)
# 目标规划
op <- OP(objective = foo, constraints = rbind(full_invest, target_return))
op
```

```
#> ROI Optimization Problem:
#>
#> Minimize a quadratic objective function of length 12 with
#> - 12 continuous objective variables,
```

```
#>
#> subject to
#> - 2 constraints of type linear.
#> - 0 lower and 0 upper non-standard variable bounds.
```

求解器 nloptr.slsqp 需要给初值和等式约束的梯度，而求解器 quadprog 不需要给初值。下面使用 quadprog 来求解组合优化问题。

```
library(ROI.plugin.quadprog)
sol <- ROI_solve(op, solver = "quadprog")
# 最优解：投资组合
w <- sol$solution
# 保留 4 位小数
round(w, 4)
```

```
#>  [1] 0.0000 0.0000 0.0000 0.0000 0.3358 0.0000 0.0000 0.0000 0.3740 0.0000
#> [11] 0.0000 0.2902
```

```
# 目标函数值：投资风险
sqrt(t(w) %*% cov(DD) %*% w)
```

```
#>           [,1]
#> [1,] 0.9860861
```

求解出来的投资组合是甲骨文、AT&T 和 IBM，投资比例分别是 33.58% 、37.40% 和 29.02% 。以上 12 支股票都属于科技公司，收益率具有非常高的相关性，因此，最终选出来 3 支。

与给定预期回报而风险最小的组合优化问题相对应的是另一个问题：给定风险的约束条件下，获得预期回报最大的组合。即求解如下组合优化问题：

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

其中，目标函数中 $\hat{\boldsymbol{\mu}}$ 表示根据历史数据获得的投资对象的收益率，约束条件中 σ 表示投资者可以接受的投资风险，其他符号的含义同前。在给定风险约束 σ 下，求取回报最大的组合。线性约束也可以用函数 Q_constraint() 来表示，这样线性约束和二次约束可以整合在一起，代码如下：

```
# 风险阈值
sigma <- sqrt(t(w) %*% cov(DD) %*% w)
sigma
```

```
#>           [,1]
#> [1,] 0.9860861
```

```
# 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
```

21.3 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：

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

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] 63.934143  6.121323
```

```
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 21.3 。

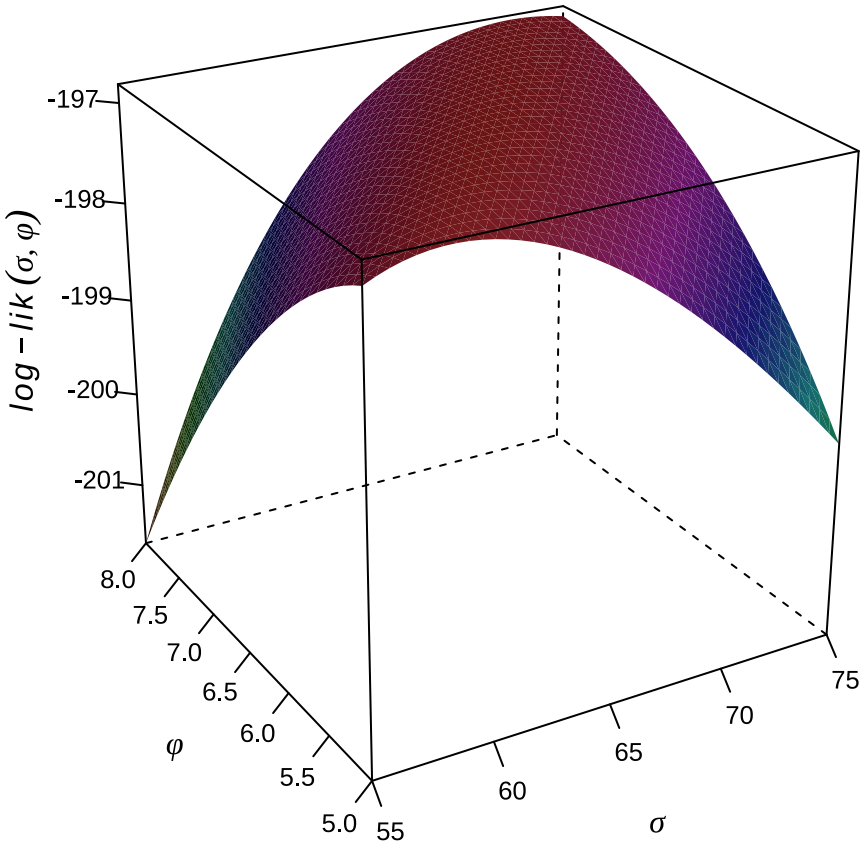


图 21.3: 对数似然函数的曲面图

等高线图呈现一道非常长且平滑的山岭 long flat ridge，山岭上布满许多局部极大值，普通的数值优化求解器常常陷入其中，只有全局优化求解器才可能找到全局极大值点。高斯过程回归模型的对数似然函数是非凸的，多模态的。

上图中没有看到许多局部极小值，与作者论文中的图 1 似乎不符。原因是什么？似然函数中涉及到的矩阵运算不精确，应该设计精度更高的运算方式？**lattice** 包绘图引擎无法展示更加细微的差异？还有一种解释，上图是对的，算法迭代时，对不同的初值，常常收敛到不同的结果，而这些不同的结果都位于岭上不同位置，对应的对数似然值却又几乎一样。

作为验证，下面调用 **nlme** 包的 `gls()` 函数拟合数据，参数的极大似然估计结果与全局优化求解器的结果比较一致。参数估计结果 $(\sigma, \phi) = (63.93429, 6.121352)$ ，对数似然函数值为 -244.6006 ，自编的似然函数 `log_lik()` 在最优解处的值为 -196.8158，再加上之前扔掉的常数项 `-52 / 2 * log(2 * pi)` ，就是 -244.6006 ，丝毫不差。

```
library(nlme)
fit_topo_ml <- gls(z ~ 1,
  data = topo, method = "ML",
  correlation = corExp(value = 65, form = ~ x + y)
```

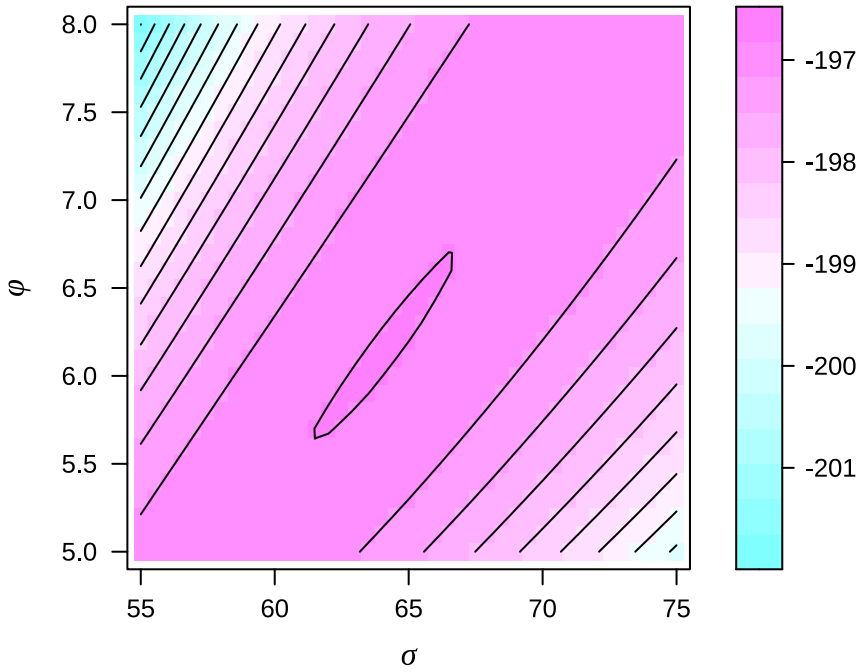


图 21.4: 对数似然函数的等高线图

```
)
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
```

21.4 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 $\text{Poisson}(\lambda_1)$ ，而以概率 $1-p$ 服从泊松分布 $\text{Poisson}(\lambda_1)$ 。

$$p \times \text{Poisson}(\lambda_1) + (1-p) \times \text{Poisson}(\lambda_2)$$

泊松混合分布的概率密度函数 $f(x; p, \lambda_1, \lambda_2)$ 如下：

$$f(x; p, \lambda_1, \lambda_2) = p \times \frac{\lambda_1^x \exp(-\lambda_1)}{x!} + (1-p) \times \frac{\lambda_2^x \exp(-\lambda_2)}{x!}$$

随机变量 X 服从参数为 p 的伯努利分布 $X \sim \text{Bernoulli}(1, p)$ ，随机变量 Y 服从泊松混合分布，在伯努利分布的基础上，泊松混合分布也可作如下定义：

$$Y \sim \begin{cases} \text{Poisson}(\lambda_1), & \text{当 } X = 1 \text{ 时,} \\ \text{Poisson}(\lambda_2), & \text{当 } X = 0 \text{ 时.} \end{cases}$$

对数似然函数如下：

$$\ell(p, \lambda_1, \lambda_2) = \sum_{i=0}^{n} y_i \log \left(p \times \exp(-\lambda_1) \times \frac{\lambda_1^{x_i}}{x_i!} + (1-p) \times \exp(-\lambda_2) \times \frac{\lambda_2^{x_i}}{x_i!} \right)$$

下表格 21.1 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 21.1: 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:  -2331.369  pgrad:  3.922366
#> iter:  10  f-value:  -2002.196  pgrad:  5.702759
#> iter:  20  f-value:  -1998.674  pgrad:  1.953276
#> iter:  30  f-value:  -1989.972  pgrad:  2.180855
#> iter:  40  f-value:  -1989.946  pgrad:  0.0003387868
#>   Successful convergence.
```

```
ans
```

```
#> $par
#> [1] 0.3598824 1.2560893 2.6634010
#>
#> $value
#> [1] -1989.946
#>
#> $gradient
#> [1] 0.0003979039
#>
#> $fn.reduction
#> [1] -341.4234
#>
#> $iter
```

```
#> [1] 47
#>
#> $feval
#> [1] 149
#>
#> $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,] -907.1119  270.2287  341.2547
#> [2,]  270.2287 -113.4793  -61.6818
#> [3,]  341.2547  -61.6818 -192.7824
```

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

```
#> [1] 0.1946838 0.3500315 0.2504773
```

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,] -1999.853      0.3657      2.3827      2.0169
#> [2,] -1989.946      0.6401      2.6634      1.2560
#> [3,] -1989.946      0.6401      2.6634      1.2561
#> [4,] -2000.232      0.7376      2.0582      2.4061
#> [5,] -1989.946      0.3599      1.2561      2.6634
```

21.5 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 21.5 展示两个伽马分布的概率密度函数，形状参数分别为 5 和 9，尺度参数均为 1，即伽马分布 $f(x;5,1)$ 和 $f(x;9,1)$ 。

给定一组来自伽马分布的样本 $x_1,x_2,\ldots,x_n$ ，关于参数 α 和 σ 的似然函数如下：

$$\mathcal{L}(\alpha,\sigma)=(\frac{1}{\sigma^{\alpha}\Gamma(\alpha)})^n(\prod_{i=1}^n x_i)^{\alpha-1}\exp(-\frac{\sum_{i=1}^n x_i}{\sigma})$$

则，其对数似然函数如下：

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

对数似然函数关于参数 α 和 σ 的偏导数如下：

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

根据第二个式子可得 $\sigma=\frac{1}{n\alpha}\sum_{i=1}^n x_i$ ，将其代入第一个式子可得

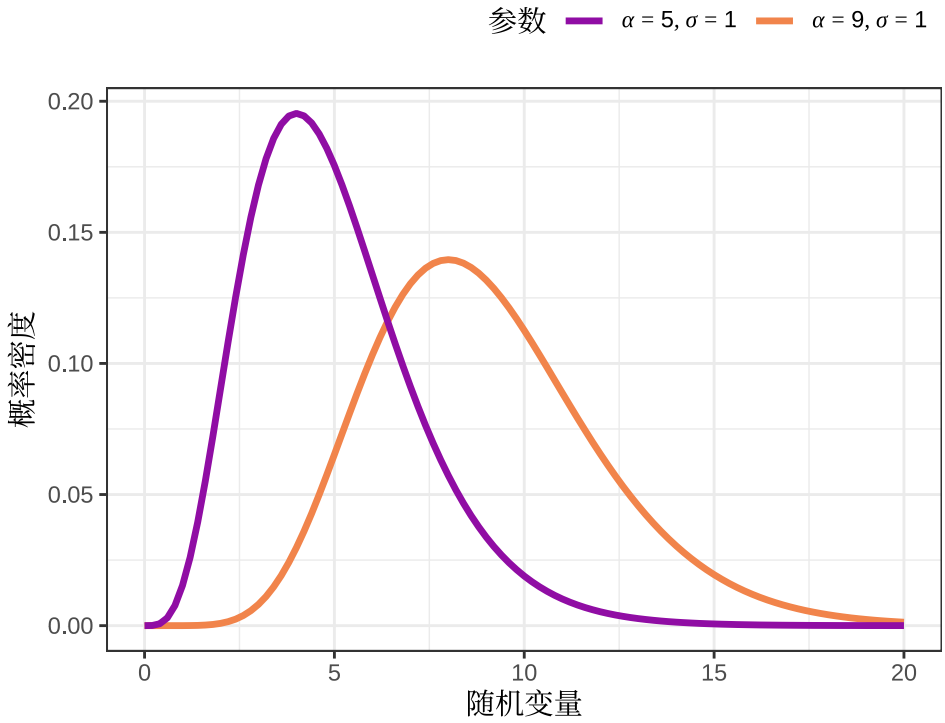


图 21.5: 伽马分布的概率密度函数

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

```
set.seed(20232023)
x <- rgamma(1000, shape = 1.5, scale = 2)
# 形状参数和尺度参数的矩估计
c(mean(x)^2 /var(x), var(x)/mean(x))
```

```
#> [1] 1.636030 1.902239
```

```
# 极大似然估计
# 常量
cc <- log(mean(x)) - mean(log(x))
# 方程
fun <- function(alpha){
  log(alpha) - digamma(alpha) - cc
}
# 找根
uniroot(f = fun, interval = c(1, 3))
```

```
#> $root
#> [1] 1.610272
#>
```

```
#> $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
```

21.6 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 21.2: 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

参考文献

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

Ansari, A. R., 和 R. A. Bradley. 1960. 《Rank-Sum Tests for Dispersions》. *The Annals of Mathematical Statistics* 31 (4): 1174–89. https://doi.org/10.1214/aoms/1177705688.

Anscombe, F. J. 1973. 《Graphs in Statistical Analysis》. *The American Statistician* 27 (1): 17. https://doi.org/10.2307/2682899.

Bates, Douglas, 和 Dirk Eddelbuettel. 2013. 《Fast and Elegant Numerical Linear Algebra Using the RcppEigen Package》. *Journal of Statistical Software* 52 (5): 1–24. https://doi.org/10.18637/jss.v052.i05.

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.

Biecek, Przemyslaw. 2023. *DrWhy: Explain, Explore and Debug Predictive Machine Learning Models*. https://github.com/ModelOriented/DrWhy.

Brandao, Filipe. 2023. *rAMPL: AMPL API for R*. https://github.com/ampl/rAMPL.

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.

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.

Davies, Rhian, Steph Locke, 和 Lucy D'Agostino McGowan. 2022. *datasauRus: Datasets from the Datasaurus Dozen*. https://CRAN.R-project.org/package=datasauRus.

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.

Feinerer, Ingo, Kurt Hornik, 和 David Meyer. 2008. 《Text Mining Infrastructure in R》. *Journal of Statistical Software* 25 (5): 1–54. https://doi.org/10.18637/jss.v025.i05.

Fligner, Michael A., 和 Timothy J. Killeen. 1976. 《Distribution-Free Two-Sample Tests for Scale》. *Journal of the American Statistical Association* 71 (353): 210–13. https://doi.org/10.1080/01621459.1976.10481517.

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.

Grün, Bettina, 和 Kurt Hornik. 2011. 《topicmodels: An R Package for Fitting Topic Models》. *Journal of Statistical Software* 40 (13): 1–30. https://doi.org/10.18637/jss.v040.i13.

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.

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.

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, 和 Julia Silge. 2021. *Supervised Machine Learning for Text Analysis in R*. New York: Chapman; Hall/CRC. https://smltar.com/.

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.

Kim, Seock-Ho, 和 Allan S. Cohen. 1998. 《On the Behrens-Fisher Problem: A Review》. *Journal of*

Educational and Behavioral Statistics 23 (4): 356–77. https://doi.org/10.2307/1165281.

Kim, Yongdai, Hosik Choi, 和 Hee-Seok Oh. 2008. 《Smoothly Clipped Absolute Deviation on High Dimensions》. *Journal of the American Statistical Association* 103 (484): 1665–73. https://doi.org/10.1198/016214508000001066.

Kolaczyk, Eric D., 和 Gábor Csárdi. 2020. *Statistical Analysis of Network Data with R*. 2nd 本. Springer, New York, NY. https://doi.org/10.1007/978-3-030-44129-6.

Kuhn, Max, 和 Hadley Wickham. 2020. *Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles*. https://www.tidymodels.org.

Lang, Michel, 和 Patrick Schratz. 2023. *mlr3verse: Easily Install and Load the mlr3 Package Family*. https://CRAN.R-project.org/package=mlr3verse.

Lüdecke, Daniel. 2019. *strengejacke: Load Packages Associated with Strenge Jacke!* https://github.com/strengejacke/strengejacke.

Lüdecke, Daniel, Mattan S. Ben-Shachar, Indrajeet Patil, Brenton M. Wiernik, Etienne Bacher, Rémi Thériault, 和 Dominique Makowski. 2022. 《easystats: Framework for Easy Statistical Modeling, Visualization, and Reporting》. *CRAN*. https://easystats.github.io/easystats/.

Meyer, David, Achim Zeileis, 和 Kurt Hornik. 2006. 《The Strucplot Framework: Visualizing Multi-Way Contingency Tables with vcd》. *Journal of Statistical Software* 17 (3): 1–48. https://doi.org/10.18637/jss.v017.i03.

Mood, A. M. 1954. 《On the Asymptotic Efficiency of Certain Nonparametric Two-Sample Tests》. *The Annals of Mathematical Statistics* 25 (3): 514–22. https://doi.org/10.1214/aoms/1177728719.

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.

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.

Shapiro, S. S., 和 M. B. Wilk. 1965. 《An analysis of variance test for normality (complete samples)》. *Biometrika* 52 (3-4): 591–611. https://doi.org/10.1093/biomet/52.3-4.591.

Silge, Julia, 和 David Robinson. 2017. *Text Mining with R*. New York: O'Reilly Media, Inc. https://www.tidytextmining.com/.

"Student". 1908. 《The probable error of a mean》. *Biometrika* 6: 1–25.

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.

Tyner, Sam, François Briatte, 和 Heike Hofmann. 2017. 《Network Visualization with ggplot2》. *The R Journal* 9 (1): 27–59. https://doi.org/10.32614/RJ-2017-023.

Varadhan, Ravi, 和 Paul Gilbert. 2009. 《BB: An R Package for Solving a Large System of Nonlinear Equations and for Optimizing a High-Dimensional Nonlinear Objective Function》. *Journal of Statistical Software* 32 (4): 1–26. https://www.jstatsoft.org/v32/i04/.

Warnes, J. J., 和 B. D. Ripley. 1987. 《Problems with likelihood estimation of covariance functions of spatial gaussian processes》. *Biometrika* 74 (3): 640–42.

Wickham, Hadley. 2016. *ggplot2: Elegant Graphics for Data Analysis*. 2nd 本. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, 等. 2019. 《Welcome to the tidyverse》. *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Wickham, Hadley, Mine Çetinkaya-Rundel, 和 Garrett Grolemund. 2023. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. 2nd 本. Sebastopol, California: O'Reilly Media, Inc. https://r4ds.hadley.nz/.

Wickham, Hadley, Danielle Navarro, 和 Thomas Lin Pedersen. 2024. *ggplot2: Elegant Graphics for Data Analysis*. 3rd 本. Springer-Verlag New York. https://ggplot2-book.org/.

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.

Xie, Yihui. 2015. *Dynamic Documents with R and knitr*. 2nd 本. Boca Raton, Florida: Chapman; Hall/CRC. https://yihui.org/knitr/.

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.

Zhang, Lijin, Xueyang Li, 和 Zhiyong Zhang. 2023. 《Variety and Mainstays of the R Developer Community》. *The R Journal* 15: 5–25. https://doi.org/10.32614/RJ-2023-060.

刘浩洋, 户将, 李勇锋, 和文再文. 2020. 最优化：建模、算法与理论. 北京: 高等教育出版社. http://faculty.bicmr.pku.edu.cn/~wenzw/optbook.html.

宋泽熙. 2011.《两个二项总体成功概率的比较》. 中国校外教育（理论）z1: 81. https://doi.org/10.3969/j.issn.1004-8502-B.2011.z1.0919.

赵鹏, 谢益辉, 和黄湘云. 2021. 现代统计图形. 北京: 人民邮电出版社. https://bookdown.org/xiangyun/msg.

韦博成. 2009.《《红楼梦》前 80 回与后 40 回某些文风差异的统计分析（两个独立二项总体等价性检验的一个应用）》. 应用概率统计 25 (4): 441–48. https://doi.org/10.3969/j.issn.1001-4268.2009.04.012.

附录 A　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。

A.1　安装配置

A.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。

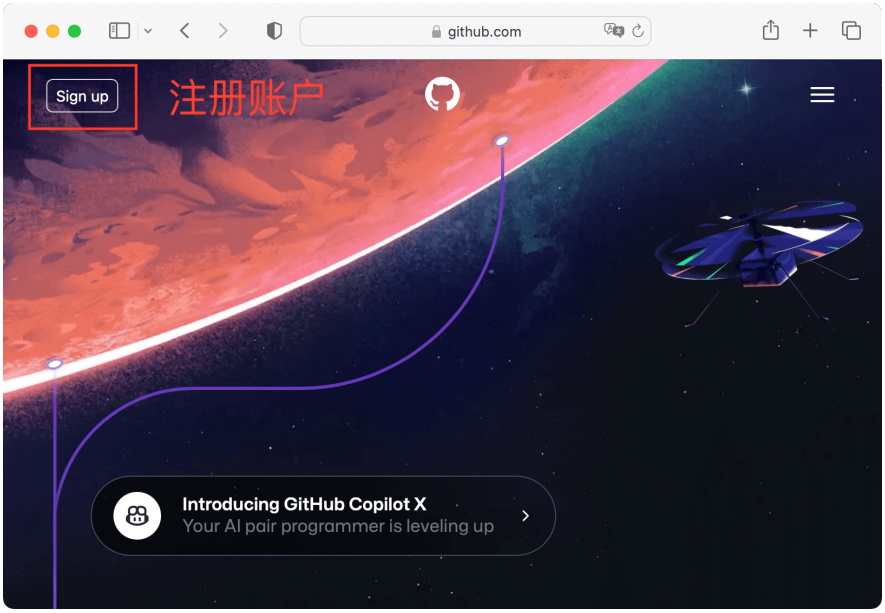


图 A.1: 点击注册

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。

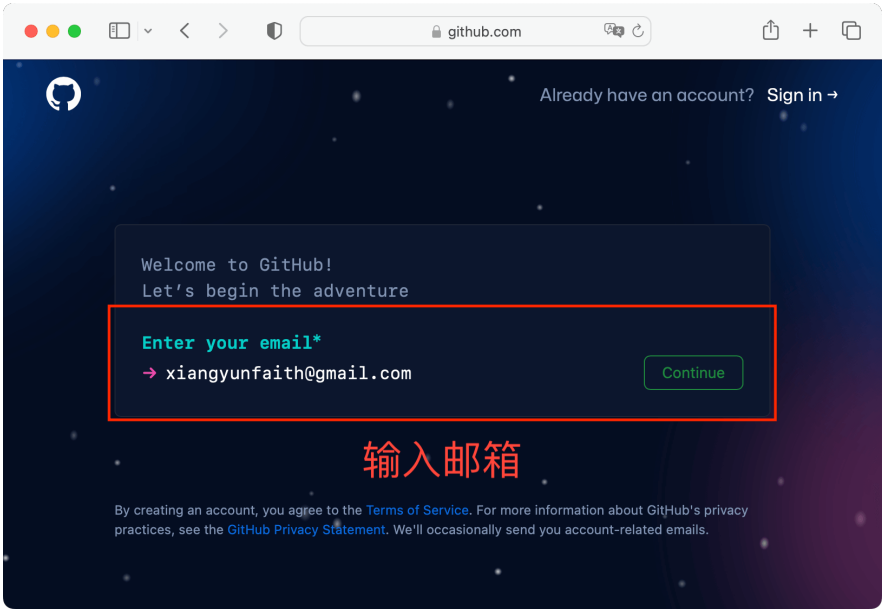


图 A.2: 输入邮箱

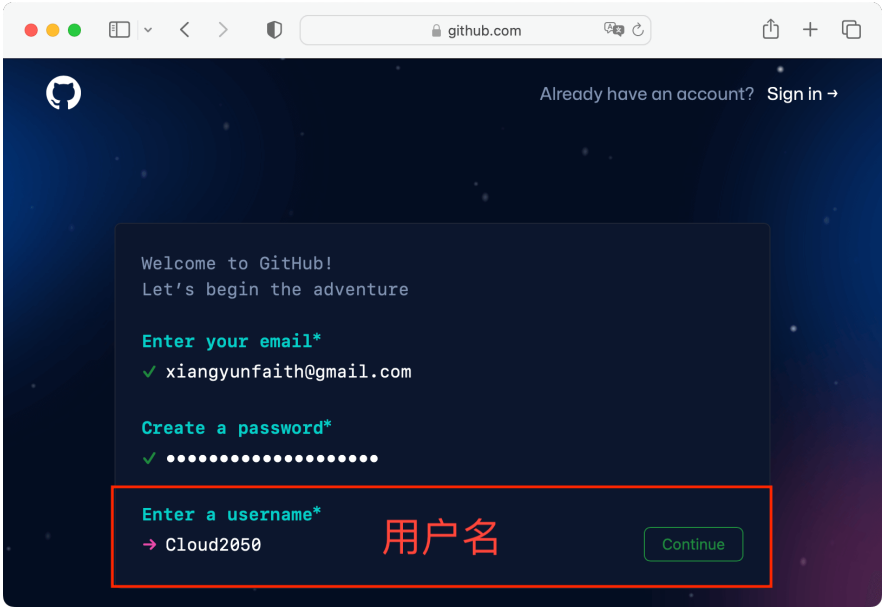


图 A.3: 输入用户名

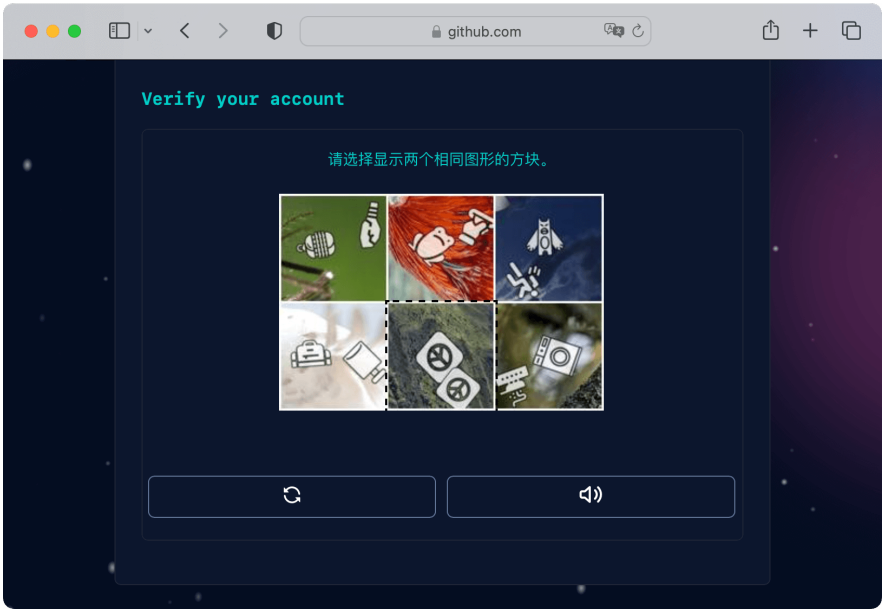


图 A.4: 回答问题

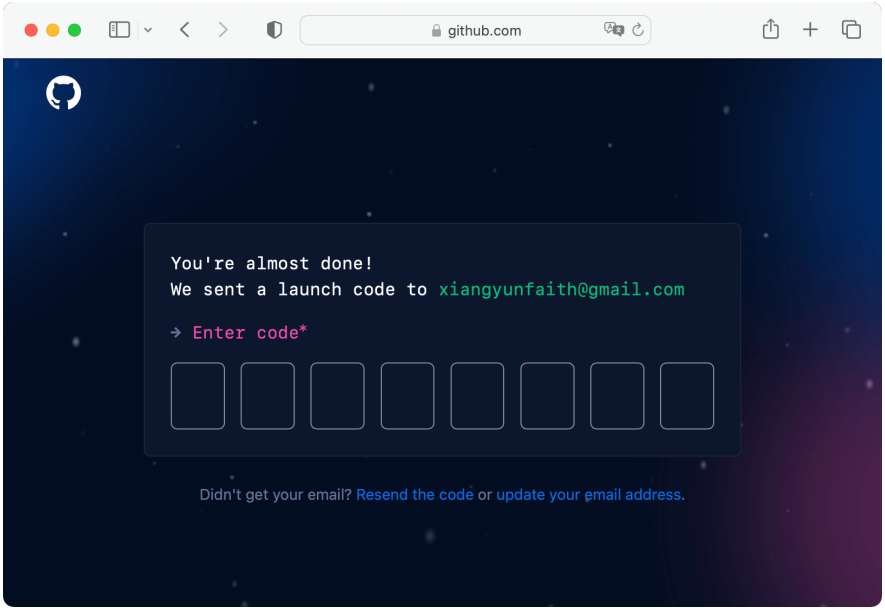


图 A.5: 输入验证码

将收到的验证码输入进去，完成账户验证。

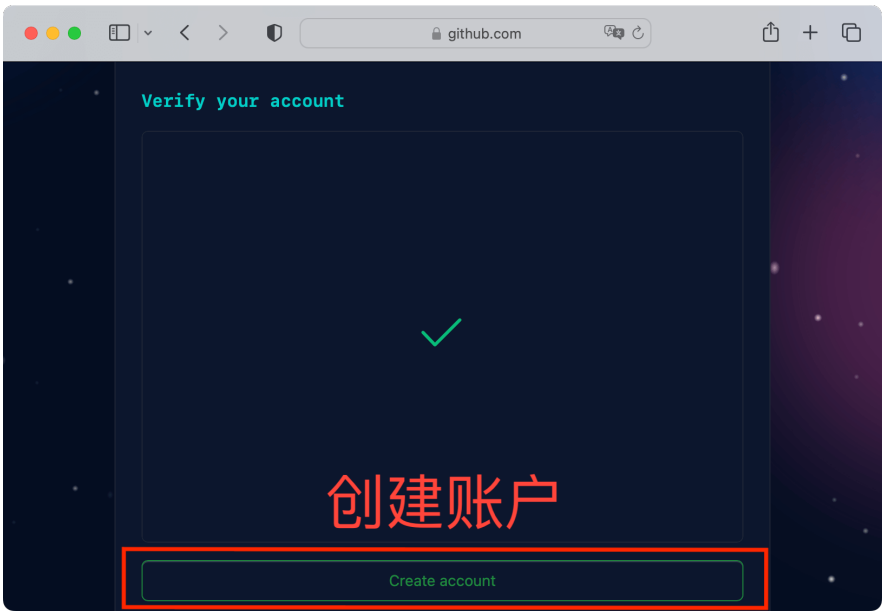


图 A.6: 验证账户

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。

A.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
```

A.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"
```

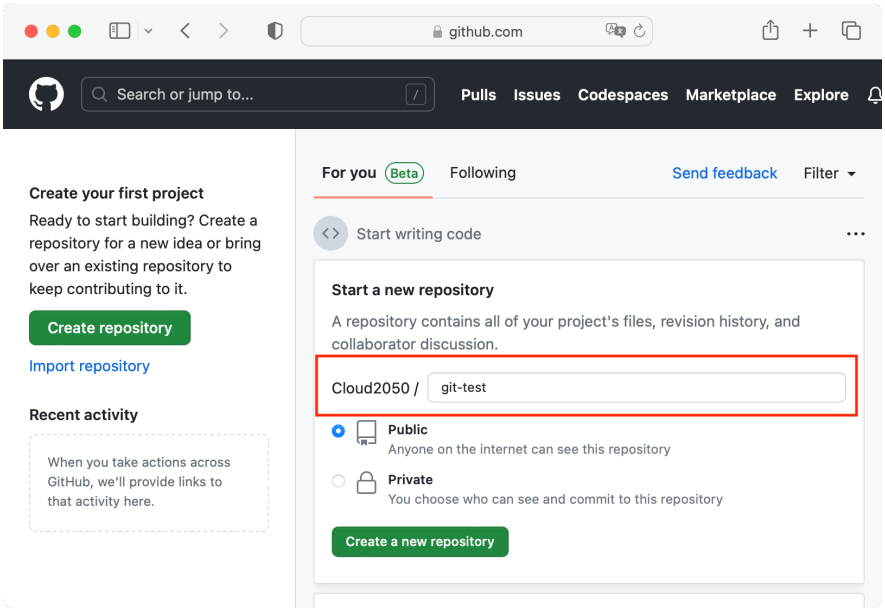


图 A.7: 创建代码仓库

A.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.
```

A.2 基本操作

A.2.1 初始化仓库

git init

A.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
```

A.2.3 记录修改

git commit

```
git commit -m "添加提交说明"
```

A.2.4 推送修改

git push

```
git push -u origin master
```

A.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
```

A.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 A.8: Git 分支操作

A.3.1 创建分支

```
git checkout -b 分支名称
```

A.3.2 分支切换

```
git checkout 分支名称
```

A.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
```

A.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

A.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

A.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()
```

A.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
```

A.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。

A.5.1 语法高亮

git-delta

```
brew install git-delta
```

对 `git diff` 的输出提供语法高亮

A.5.2 文本接口

在 MacOS 上，推荐用 Homebrew 安装

```
brew install tig
```

A.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