# 序言（一）

受作者邀请为此书写序，感到惶恐和高兴。多年来，我们研究组很多同学都在学习和使用 R 语言，作为老师的我却从来没有用过，要为一本关于 R 语言的书写序，自然感到很惶恐。而当我打开这本书，没有看到晦涩难懂的概念、公式，在风趣易懂的语言、引人入胜的描述下，很快阅读完了前几章，不仅迅速跨越了零基础的障碍，更为两名作者为广大的数据统计分析和 R 语言学习人员提供这样一本独特、有趣的入门书而感到高兴。

随着科技的高速发展，对大量数据的处理和分析已成为科研甚至日常生活的必需，掌握一门强大的数据分析工具非常必要。作为一种开源、免费且不断更新的语言，R 不仅拥有数据分析、统计、作图等强大功能，其应用范围还在不断扩大：可以用来撰写学术论文、做备忘录、制作幻灯片、整理读书笔记、整理照片、写书、记录吉他谱、写博客、做网站等。R 语言拥有越来越多的使用者和爱好者，也使得其功能越来越广泛、强大。

与其它介绍 R 语言的书籍不同，这本书以两名作者自身学习的经验娓娓道来，没有多少枯燥的术语和公式，是一本非常适合自学 R 语言和统计学的入门书，尤其适合零基础和不懂统计学的读者。这本书可作为大学本科生和研究生自学用，也可作为高校教师教案。相信读者们不仅会发现这本书很有用，更会发现关于 R 语言和统计学的书可以写得很有趣。

—朱彤[1]，2017 年 6 月 13 日

[1]朱彤，北京大学教授、博士生导师，北京大学环境科学与工程学院院长。

# 序言（二）

每当人找我写序的时候我都恨得牙痒痒，因为我十年前就开始写一本关于 R 图形的中文书，但后来因为各种事情搁下了，现在的时间只够给人写序。不过这次我倒是很乐意为《学 R》写序，因为这本书实现了我的一个小小愿望：我希望技术书籍能变得有趣一些。我在另一本书《R 包开发》的序言里写道：

> ‘‘我也很期待国内能出现更多本土作者的中文 R 作品。讲真啊，我认为中文的表现力比英文不知道要高到哪里去了。’’

我讲这句话的原因是我们翻译的外文 R 书籍太多了，而翻译的书真的是很难做到语言自然流畅。本书的作者语言功力很强，而且处处透露着幽默感，至少在文字方面我觉得几乎无可挑剔。

我头一次注意到本书的作者之一大鹏是 2013 年他在统计之都论坛上的第一个帖子：`https://d.cosx.org/d/109820`。当时我就跟我们统计之都的人说，我们得把这厮拉进我们的队伍，因为他的文风很别致，也很有热情。聪明的人我见过不少，但内心有小火苗的人我真心觉得少见。那时候他都还没用过 Github，对 Pandoc 也不熟，举着一把 R Markdown 板斧、骑着一头 RStudio 小毛驴就杀将出来；在燕小六[2]一般哇呀呀坚持几年过后，终于开始横刀立马来检阅战果了。对《学 R》一书的出版，我感到非常欣慰。

老实讲，我并没有通读书稿，只看了几个样章，但我经常看大鹏的博

[2]燕小六：电视剧《武林外传》里的一名捕快。

客，所以我觉得这几个样章应该很有代表性。这本书最大的亮点应该是作者真心倾注了热情在里面，从字里行间可以感受到他们总是在鼓励你试试这个、尝尝那个，仿佛一直在拿胳膊肘子捅你：喂，这里很好玩哟，这里是见证奇迹的时刻哟。我相信这样的书读起来一定会很吸引人。

在内容涵盖范围方面，我感觉这本书尤其适合学术界以及那些对 R 语言有钻研爱好的读者，当然我也相信它对其他读者一定也会有很多帮助。它介绍了很多“经世致用”的技能，可能那些圣贤书上都不会讲但又很实用，可以以很快的速度学以致用，这一点在这个所谓的“数据科学”时代尤其重要。这本书涵盖的统计学主题相对少一些，但作为一本入门读物也无妨，因为当你玩上瘾之后，继续深挖其它主题就会势如破竹。

这几年我一直忙着做开发工作，没多少时间在中文社区解释和宣传我的工作，所以我很感激本书作者帮我用中文形式介绍了 knitr、bookdown（“不可挡”真乃神翻译）和 blogdown 等包。希望它们对大家有用，也祝大家能在阅读本书的过程中感受到学习 R 的乐趣。

— 谢益辉[3]，2017 年 8 月 23 日

[3]谢益辉（`https://yihui.name`），中国人民大学统计学院经济学学士和硕士，美国爱荷华州立大学统计学博士，“统计之都”创办人，中国 R 语言会议发起人，rmarkdown、knitr、bookdown、blogdown、animation 等诸多 R 扩展包的作者，现为 RStudio 公司软件工程师。

# 前言：思 R 不学则殆

学 R 不思则罔，思 R 不学则殆。

—《论语 • 为政》[4]

《论语》开篇头两个字就是 ‘‘学 R’’。可是，R 语言 (R Core Team, 2016) 的古怪却是我平生头一回遇见。一些好友对 R 的火爆津津乐道，而另一些朋友却闻所未闻；我努力钻研专家们推荐的教材，却屡战屡 ‘‘殆’’，死活入不了门；出国读书前，身边没有一个人使用 R，但到了德国，研究组里每个人都在用 R 处理科研数据，R 代码满天飞，里面隐藏着最新的科研方法，而我抓住一个，打开却像看天书，‘‘罔’’ 了。

多年之后，我费尽周折算是入了 R 的门。除了科研必需的数据分析、统计计算及作图外，我现在用 R 来做很多以前意想不到的事情。我发现，R 不仅像人们说的那样，是一门顶级的数据分析语言和一个极其人性化的编程环境，R 还是一件功能强大的工具和一种充满惊喜的生活方式。每个人都能从中构建自己独特的美丽新世界。通过 R 语言，我甚至结识了很多志同道合的朋友。

然而，对于我这样的外行来说，R 的入门并不容易。

市面上的 R 语言书籍琳琅满目，大部分是专业人士从内行的角度写成的，讲了很多的 ‘‘是什么’’（what）和 ‘‘为什么’’（why），而我更关心的是 ‘‘能不能’’（if）以及 ‘‘怎么做’’（how）。外行学 R，就像小学生学外语，当然应该在游戏和场景中从口语学起；如果上来就读语法书，门槛太高。此

[4]据说是林祯舜博士在中国 R 语言会议的一次发言里将原文的 ‘‘而’’ 改成了 ‘‘R’’。

外，R 语言的中文书多数从外文翻译而来，充斥着大量专业词汇，晦涩难懂，让人望而生畏。

菜鸟应该有菜鸟的学法。当我以菜鸟的视角把学 R 的历程写在个人博客[5]之后，才发现原来遭遇 R 学习困境的不只是我一个人。网友们或留言或来信，表达了类似的心路历程，并且兴奋地告诉我，他们通过我的博客终于进入了 R 的世界。

在博客的基础上，诞生了《学 R》这本书。

本书适合大学本科以上使用。如果你完全不懂 R，或者完全不懂统计学，但渴望掌握一个好用的数据分析工具，那么，这本书最适合你；如果你已经掌握了 R 的初步应用技能，想进一步地探讨和应用 R 的另一些有用又有趣的功能，比如绘制地图（包括动态地图）、批量处理文件、搭建个人主页、自动从网络下载并整理归类数据、解决不同时间格式的换算、在 R 和常用的办公软件中自如地切换，甚至用 R 设计一些好玩的好游戏，那么，本书是你最好的工具；如果你使用过诸如 SPSS、SAS 等软件，但是其灵活性不能满足需求，那么，使用本书来学习 R 会很轻松；如果你并无理工科背景，仅仅想为生活增添一些别样的趣味，那么，本书是很个性的选择。

本书各章节的难度前后相继。对编程或 R 语言零基础的读者，可以从头读起。同时，各章相对独立成篇，读者大可挑感兴趣的章节跳跃阅读。有一定基础的读者，可以将本书常备手头，参照书前目录和书后索引，来查阅合适的章节、小贴士和函数示例。

在体例上，本书每个章节配有 ‘‘卷首语’’、‘‘小贴士’’、‘‘练习’’、‘‘思考’’、‘‘课外活动’’。‘‘卷首语’’ 一般是与该章节有关的言论，旨在增加学习 R 语言的趣味；‘‘小贴士’’ 是对实用小技巧的总结，方便读者快速查阅；‘‘习题’’ 多是仿照书中的实例来设计的，依葫芦画瓢就可以解答，在书后配有参考答案；‘‘思考’’ 多为启发性的问题，没有固定答案，用来启发读者骑着思想的野马随意驰骋；‘‘课外活动’’ 提供了一些有趣的话题，是该章的实践和延伸。书中的所有示例代码，连同勘误表，全部放在本书的主页[6]上。想省事的话，只需将主页上的代码拷贝粘贴到 R 的界面运行即可。不过，我们仍

[5]大鹏志：http://dapengde.com/archives/tag/r

[6]本书主页：http://xuer.pzhao.net

然建议你亲手敲一遍代码，获得第一手的经验。本书中的 R 函数名称后面均带有圆括号，与 R 代码和运行结果都以等宽字体格式展示。丰富的格式得益于 R 语言的 bookdown (Xie, 2016a,b) 扩展包。

也许你会问：听说 R 是专门搞统计学的。我虽然想用 R 语言做出那些漂亮的图，但是不懂统计学，那可怎么办？

放心，这并不是学习 R 的障碍。R 最初确实是统计学专用的，但发展至今，早已用于各行各业，遍地开花。翻翻本书的目录，你就会发现，本书大部分章节跟统计学关系不大。我们把统计学的内容集中放在了最后几章，读完之后就会知道如何应用 R 做一些基础的统计检验了。

R 语言的世界地大物博，区区一本小书必然挂一漏万。本书只是牵针引线，陪伴你跨过 R 语言的门槛，把门推开一条缝，从门缝里一窥 R 世界的绚丽。“鱼” 不如 “渔”，请在阅读时注意学习如何自助、如何在互联网寻找和筛选答案。为了便于读者理解，我们尽量避免使用专业术语，但这种妥协必然会牺牲表述上的严谨和准确。如果有不得不使用却用错之处，请大家不吝赐教。欢迎 R 爱好者来信[7]，共叙 R 学习和使用过程中的悲喜。

拜罗伊特大学生态地理模拟研究组的 Björn Reineking 教授开设的 Introduction to R 课程，是本书整体风格的滥觞。感谢 Björn 的慷慨，允许我根据需要把讲义里的示例代码跟大家分享。同时，感谢微气象学系 Thomas Foken 教授研究组的同事们，在我学习 R 的过程中给我很多帮助。在拜罗伊特大学的学习和生活，是我迄今最为怀念的一段时光。

很荣幸邀请到谢益辉博士为本书作序。益辉是一位对工作精益求精的 “强迫症” 患者。他对本书原稿的肯定，令我们对本书的出版信心倍增。益辉还对本书的写作工具 bookdown 给予了最大程度的技术支持，并为书稿的格式提出了中肯的修改意见。

外行写 R 属于班门弄斧，刘思喆等来自 “统计之都” 的专业人士给予了我莫大的宽容和鼓励。同时感谢很多热心网友的反馈，他们用火眼金睛在博客原稿里挑出了一些难以察觉的错误。

[7]联系邮箱：xuer@pzhao.net

衷心感谢我的研究生导师朱彤教授和我的中学语文老师范晓太先生。朱老师于百忙之中抽时间为本书作序，而范老师于遣词造句方面提供的意见让我豁然开朗。两位先生皆是我的人生导师，从他们那里得到的收获让我受益终生。

本书书名主要来自韩蕾博士的提议。此外，她还以读者身份指出了书稿中的一些疏漏。兜兜和他的朋友小语各自为封面画了插图，尽管最终因风格不符而未被采纳，但我仍然为两个孩子的热心参与感到欣慰。当然也要感谢轩轩，我写书时他经常在身边不声不响地玩耍，时间长了就跑过来盯着我的眼睛说：‘‘别看电脑啦，爱护眼睛。’’

— 赵鹏，2017 年 2 月 14 日于因斯布鲁克

# 目录

# 第一章　初见

有些人从没听说过 R，也照样过得无比快乐，而实际上我的工作之一就是把 R 交给他们。快乐不等于能力和效率。有些情况下，效率对一个人的好处，比短暂的快乐要强得多。

— Patrick Burns, April 2005

世间所有的相遇，都是久别重逢。

— 电影《一代宗师》

## 1.1　结缘：下载安装

R 是跨平台的开源免费自由软件，Windows、Linux、macOS 都有对应的安装文件可以下载。本书均以 Windows 系统为例作介绍。截至本书成稿时，R 的最新版是 3.4.1，安装程序文件只有 75 M。我们去 R 的服务器 CRAN[1]，点击 'Download R for Windows'，在打开的新网页最上方点击 'base'，就找到下载链接了。下载完毕后，一路 ''下一步'' 的傻瓜式安装即可。安装完毕之后，从开始菜单中找到 R，运行就可以了。

R 的默认界面是控制台窗口（图 1.1），展示的是代码和运行结果。在这个窗口逐条输入下面的代码，每换一次行就会执行一条。

[1]CRAN：https://cran.r-project.org/

```
co2
summary(co2)
```

我们输入的 R 代码很简单。第一行展示了一个名叫 co2 的变量，内容是夏威夷 Mauna Loa 观测站的大气二氧化碳浓度数据，是 R 自带安装的。这个数据举世闻名，我们现在常说的全球变暖、节能减排，经常会拿这个数据来说事儿。第二行计算的是 co2 数据的统计量, 依次是最小值、第一分位数、中值、平均值、第三分位数、最大值。敲几个字母就算出这么多结果，是不是很方便？这只是 R 牛刀小试而已。

需要注意的是，退出 R 之后，这个窗口的代码不会保存。想保存的话，可以点击菜单栏的 File – New script，就会出现一个新的编辑窗口，这里可以输入代码。需要执行的话，光标移到代码所在行，按 ctrl+r （表示同时按住 ctrl 键和 r 键。下同）即可。编辑窗口的代码可以保存成文本文件，方便以后重复使用。

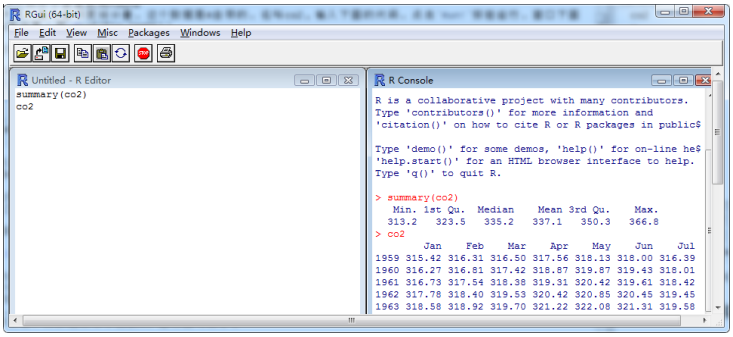



图 1.1: 裸奔的 R 默认界面

不出意料的话，新手对 R 的第一印象一定是：怎么这么简陋！

是的，此刻的 R 在裸奔，虽然能用，但不好看，而且不方便。我们可以给 R 穿上一件衣服遮遮羞，这只需再安装一个软件就行。

---

**思考 1.1.** 你还见过哪些软件，界面简陋却功能强大？哪些软件界面花哨却功能差强人意？

R 能穿的免费衣服很多，人们各有所爱。我们用过 Notepad++、Tinn-R、RKward，也用过 Vim 配合 R 插件，最后选定了 RStudio，因为有诸多好处，最明显的就是把 R 常用的界面整合到了一起（图 1.2）。看吧，华丽丽堪比貂皮大衣。默认有四个面板：左上面板输入代码，左下面板查看代码的运行结果，右上面板展示工作环境，右下面板显示作图结果和帮助信息。我第一次用 RStudio 的时候一时惊为天人，就像《天龙八部》里的段誉看见了神仙姐姐的雕像，《神雕侠侣》里的郭襄看见了摘去面具的杨过，从此不可自拔。RStudio 还有很多好处，我们以后慢慢讲。将来会发现，RStudio 岂止是件貂皮大衣，简直是一栋豪华别墅。

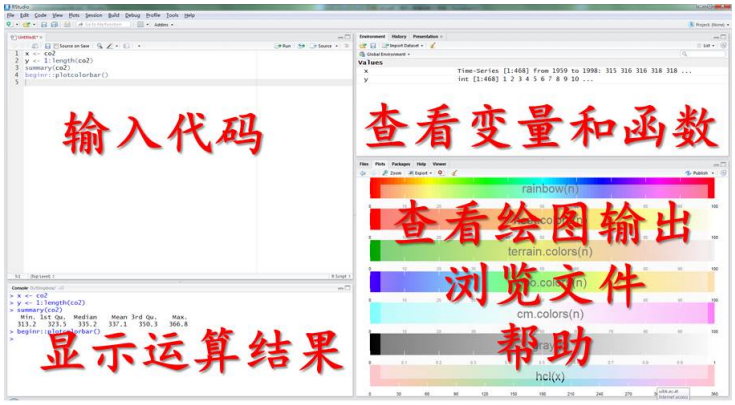



图 1.2: RStudio:R 的豪宅

RStudio 的安装很简单。上 RStudio 官网[2]，下载安装文件运行即可。安装完毕后运行，然后选择菜单 File – New — R script ，或按快捷键 ctrl+shift+n，会新建一个.r 文件。

以后我们只需运行 RStudio 就行了，它会自动调用 R。那个裸奔的 R 界面，就让它像泰坦尼克号一样永远埋葬在电脑的深处。忘了它吧。

现在，我们正式开始 R 的奇幻之旅。

[2]RStudio：`https://www.rstudio.com/`

## 1.2 第一次畅谈：计算

R 最简单的功能，是用作计算器。先在左上面板窗口输入以下代码，然后按窗口上方的运行（Run）按钮，或按快捷键 ctrl+ 回车（这个快捷键会经常用），就会运行光标所在行的整行代码：

```
3 * (2 + 2)
```

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

上面第一行是输入的代码示例。第二行用两个 # 号开头，表示是运行结果，默认显示在 RStudio 的左下面板。如不另作说明，本书都用这种格式来区分代码和运行结果。我们暂时不管 # 号后面的 [1] 是什么，先来试试 R 的数学基本运算符：加 +，减 -，乘 *，除 /，乘方 ^，整除的商 %/%，整除的余数 %%。

---

**练习 1.1.** 计算 365 除以 7 得到的整除商和余数。

下面，我们开个平方。输入并运行

```
9 ^ 0.5 # 开平方
```

```
## [1] 3
```

或者

```
sqrt(9)  # 也是开平方
```

```
## [1] 3
```

上面两条语句的作用等同，只是形式不同。这里，sqrt() 是开平方的函数，被开方的数必须放在圆括号里，这是 R 语法的基本规则之一。# 号后面一直到这一行的末尾是注释，注释部分不会被运行，这样是为了方便将来理解这句代码的用途。当然，我们可以用注释随便写点什么，比如说“# 哇塞我的第 1 行代码太帅了”或者“# 今天心情不大好就写到这儿吧”等

等。如果你乐意，那么完全可以在注释里偷偷写一部小说，就像《倚天屠龙记》里有人在《楞伽经》夹缝处写下《九阳真经》一样。

有人读到这里，可能会退缩了：sqrt，开玩笑，我怎么记得住啊！注意 R 入门第一秘诀：**不要被 R 吓住**！现在，我们请出我们的第一位人气小助理：tab 键。试试只输入 s，然后按 tab 键，就会看到 RStudio 给出的贴心提示，所有以 s 打头的函数和变量都列在里边了，用鼠标或箭头键选取就行了。在 s 后面接着输入 q 之后再按 tab 键试试。这个 "tab 小助理" 我们以后天天时时分分秒秒都会用。

其实，常用的函数就那么几个，用几次就不需要贴心提示了。而且函数名称都很好记，sqrt 就是 square root 的缩写，顺便练了英文。实在记不住，那就用基本运算符来求乘方好了，`9 ^ 0.5` 即可。将来学了自定义函数之后，你甚至可以把 sqrt 改名叫做 kaipingfang。我们在后面的学习中，会经常针对同一个问题给出多个解决方案，条条道路通罗马，R 很灵活的，随便挑一个你喜欢的方案拿去用就行了。

---

**小贴士 1.1.** *R 菜鸟入门三大秘诀*

*第一秘诀：不要害怕！学 R 非难事，谁都可以 R（Anyone can R）。*

*第二秘诀：能用就行！只要能完成工作，R 代码写得漂亮与否并不重要。如果你有两个解决办法，那就选用你熟悉的那个。将来时间有富余的话再试另一个。*

*第三秘诀：与人分享！如果你的 R 代码是一把刀，那么分享就是磨刀，越磨越快。*

常用函数都可以顾名思义：四舍五入 `round()`，截取整数 `trunc()`，开平方 `sqrt()`，求绝对值 `abs()`，指数函数 `exp()`，自然对数函数 `log()`，以 10 为底的对数函数 `log10()`，三角函数 `sin()`，`cos()`，`tan()`，`asin()`，`acos()`，`atan()` 等等。

有些常数在 R 中已经定义好了，例如圆周率 $\pi$，只要输入 `pi` 并运行

```
pi
```

```
## [1] 3.141593
```

怎么只有这几位有效数字？有些读者上幼儿园时就背下来了，精确度不够高啊。要提高精确度，需要用选项函数 options()：

```
options(digits = 22)  # 最大支持 22 位
pi
```

```
## [1] 3.1415926535897931
```

options() 函数运行一次后，以后的数字都会是指定的位数，直到重新运行一次，或者退出 R。下面我们把位数改为默认值，7 位：

```
options(digits = 7)
pi
```

```
## [1] 3.141593
```

位数就变回来了。

有的常数，虽然没有定义好，但很容易算出来，例如自然对数的底 e：

```
exp(1)  # 计算 e
```

```
## [1] 2.718282
```

可以像 pi 一样，我们自己定义一个名叫 e 的变量，把 exp(1) 的值保存在 e 里，方便以后调用：

```
e = exp(1)
```

或者

```
e <- exp(1)
```

两种办法的赋值效果完全等同。<- 是个箭头，表示把右边的值赋给左边。如果你去看别人写的代码，会发现有人爱用箭头，有人爱用等号，这完

全取决于个人喜好。箭头的灵活之处在于，可以把左边的值赋给右边：

```
exp(1) -> e
```

本书的赋值符号统一用箭头。RStudio 中输入箭头有个快捷键：按 alt +_ 就行了。

---

**思考 1.2.** 箭头和等号的作用完全等同吗？什么情况下只能用等号，不能用箭头？上网搜搜答案。

好了，以后可以用 e 来代表自然对数的底了。查看 e 的值，可以看 RStudio 的右上面板，也可以在左上面板代码窗口输入变量名 e，然后 ctrl+回车，

```
e
```

```
## [1] 2.718282
```

就会在左下面板的结果窗口出现 e 的值。e 可以用来做后续计算，比如：

```
x <- round(e)^2
x
```

```
## [1] 9
```

注意，R 中大小写字母是有区别的，'E' 和 'e' 是不同的两个变量名。这叫做 ''大小写敏感''。

---

**小贴士 1.2.** 变量名的约定（三可三不可）

☺可以是一个或多个字母，如 *'e'*, *'x'*, *'mydata'*；

☺可以包括数字，如 *'a1'*, *'a2'*;

☺可以包括句点和下划线，如 *'temperature_air'*, *'humidity.max'*。

☹不可以包含空格，如 *'my data'*；

☹不可以用数字或小数开头，如 *'2x'*，*'.3y'*；

☹不可以用中文。

此外，你的变量名不能跟 R 的内置变量重名。这个倒是不必担心，遇见的时候 R 会自动发警告。一般来说，我们只要注意变量名不要加空格，不要用中文，就不会犯大错。

> 不要给你的矩阵变量取名为“矩阵”。你会给你的狗狗起名字叫“狗狗”吗？
>
> — Barry Rowlingson, October 2004

一个变量名可以存储很多数据。比如说，本市的月降水量从一月到十二月依次是：61, 45, 55, 46, 56, 79, 86, 57, 56, 56, 57, 71 mm。可以把这十二个数据赋值给一个变量 $x$，这种变量叫做向量：

```
x <- c(61, 45, 55, 46, 56, 79, 86, 57, 56, 56, 57, 71)
x
```

```
##  [1] 61 45 55 46 56 79 86 57 56 56 57 71
```

如果需要查看四月的降水量，就用方括号来指定 ‘‘下标’’。下面方括号中的 4 就是下标，表示调用 $x$ 中的第四个数值。

```
x[4]
```

```
## [1] 46
```

再比如前面提到的二氧化碳数据，变量名就是 ‘‘co2’’，这一个变量里存的是多年二氧化碳的浓度。我们可以将它转存到另一个变量里：

```
y <- co2 # 转存
y[10] # 看看第十个数据
```

```
## [1] 313.18
```

R 支持向量运算。试试输入：

```
x + 100
```

```
##  [1] 161 145 155 146 156 179 186 157 156 156 157 171
```

$x$ 里的每一个数都加上了 100 。这就是向量运算的好处：简单的代码，避免逐个计算。

现在我们可以回答 RStudio 左下方窗口里显示的结果开头那个 `[1]` 了，它表示的是这一行开头显示的是 $x$ 的第一个值。如果显示的向量长，需要折行，那么下一行开头的方括号里显示的就是该行第一个元素在 $x$ 中的位置，省得我们从头数。

## 1.3　第一张留影：作图

下面，让我们作出第一个图形来：Mauna Loa 观测站的二氧化碳浓度时间序列。这是张全球闻名的明星图。承接前面的数据，我们只需敲 7 个键就行了（图 1.3）：

```
plot(y)  # 作图
```

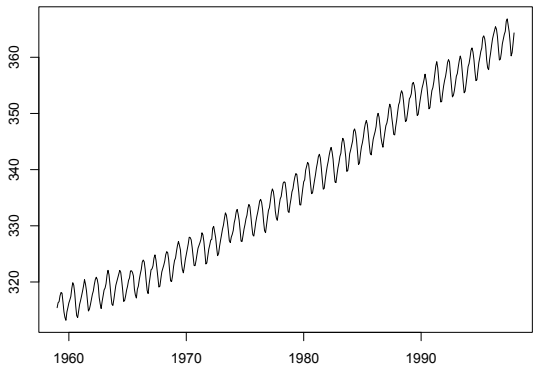



图 1.3: Mauna Loa 二氧化碳浓度

是不是很简单？有没有很激动？简单的东西人见人爱。

本市的月降水量，也可以这样画出来：因为已经保存在变量 $x$ 里了，所以 plot(x) 就可以了。

再进一步，我们来做统计运算，看看本市月降水量的平均值是多少。

```
(x[1] + x[2] + x[3] + x[4] + x[5] + x[6] + x[7] + x[8] +
   x[9] + x[10] + x[11] + x[12]) / 12  # 计算平均值
```

```
## [1] 60.41667
```

这个式子很长，我们把这条指令强行写成了两行，R 读完第一行发现指令不完整，就会自动读下一行。由于受版式的约束，本书的代码都会采用这种换行方式。实际写代码时不必这样换行。

$x$ 的 12 个元素逐个敲起来太麻烦了，可以用求和函数 sum() 以及求向量长度的函数 length()，来简化代码：

```
sum(x) / length(x)
```

```
## [1] 60.41667
```

或者直接用平均值函数 mean()：

```
mean(x)
```

```
## [1] 60.41667
```

更厉害的是 summary() 函数：

```
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   45.00   55.75   56.50   60.42   63.50   86.00
```

得到的六个数是最小值、25% 分位数、中位数、均值、75% 分位数和最大值。记住 summary() 这个明星函数吧，将来我们会反复享受使用这个函数的快乐。

上文我们用四种方法计算平均值，算出来的结果都一样，条条大路通

罗马，想起哪个用哪个。

常用统计函数有：求和 `sum()`，平均值 `mean()`，最大值 `max()`，最小值 `min()`，范围 `range()`，中位数 `median()`，分位数 `quantile()`，标准差 `sd()`，方差 `var()`，总结报告 `summary()`。你可以把这些函数用在 $x$ 上，看看结果都是什么。

---

**练习 1.2.** 2003 年 8 月北京城区测得的 $PM_{2.5}$ 的质量浓度日变化[3], 从 0 时到 23 时依次是 97, 80, 64, 91, 87, 100, 128, 144, 150, 150, 150, 106, 78, 68, 62, 46, 55, 68, 84, 92, 95, 108, 128, 138 微克每立方米。做出北京 $PM_{2.5}$ 的日变化图。计算 $PM_{2.5}$ 出现的最大值、最小值、平均值。最大值出现在几点钟？

最后，请在 RStudio 菜单栏点击 File – Save，或按快捷键 ctrl+s，把刚才输入的代码保存到一个扩展名为.r 的文件里，下一节接着用。这个.r 文件其实就是文本文件，用 Windows 记事本打开就能看，只不过里面放的是 r 代码罢了。如果装了 RStudio，双击.r 文件就会用 RStudio 打开。

好啦，以上就是 R 的基本操作和运算、作图、统计分析，你全都掌握了！R 就差不多学完了！喝一杯庆祝一下吧。

**小贴士 1.3.** *新手学 R 第一步*

| 项目 | 内容 |
|---|---|
| 安装 | CRAN, RStudio |
| 数学基本运算符 | `+`, `-`, `*`, `/`, `^`, `%%`, `%/%` |
| 常用数学函数 | `round()`, `trunc()`, `sqrt()`, `abs()`, `exp()`, `log()`, `log10()`, `sin()`, `asin()` |
| 常用统计函数 | `sum()`, `mean()`, `max()`, `min()`, `range()`, `median()`, `sd()`, `var()`, `summary()` |
| 作图 | `plot()` |

[3]数据来源：Chan et al., 2005. Atmospheric Environment 39 (28) : 5113-5124

## 1.4 课外活动：表白

经过了初步的相处，你对 R 的印象如何？有没有相见恨晚或者一见倾心的感觉？

R 给我的印象，说得文雅一点，那就是：

> 关关雎鸠，在河之洲。窈窕淑女，君子好逑。
> 参差荇菜，左右流之。窈窕淑女，寤寐求之。
> 求之不得，寤寐思服。悠哉悠哉，辗转反侧。
>
> —《诗经·国风·周南·关雎》

说得通俗一点：我想和 R 在一起。

跟很多理科生一样，我本科论文中使用的是 Excel，硕士论文使用的是 OriginLab，但博士期间换用了 R 之后，从此死心塌地跟 R 永结同心。

那么，R 窈窕在哪里？

仁者见仁，智者见智，一千个人心中有一千个哈姆雷。R 是一个取之不尽用之不竭的宝藏，我们各取所需便是。比如我，贪图便宜，看上 R 是看上了它的免费和随心所欲。当然，盗版的 Excel，OriginLab，Matlab 也免费，但盗版毕竟是见不得光的事儿，还是少干吧。

不光免费和灵活，还有 R 功能的强大，R 社区的友好等等。从我的角度来说，如果没有学习 R 和使用 R 带来的乐趣，那么我的博士研究生活必定会枯燥很多。几年过去了，我依然记得当年为论文做出一张图（图 1.4）时的兴奋。有前人定义好的函数，花了不到一分钟，只用了一个语句，就画出了 7 个变量的直方图（对角线）、两两之间的散点图和 loess 拟合曲线（对角线左下半部分），并标出了两两之间的相关系数（对角线右上半部分，正负用数字的颜色区别，相关程度用字体的大小表示）。那种激动和快乐，至今历历在目。

---

**思考 1.3.** 如果使用你熟悉的作图软件，那么图 *1.4*这种图该怎么做？

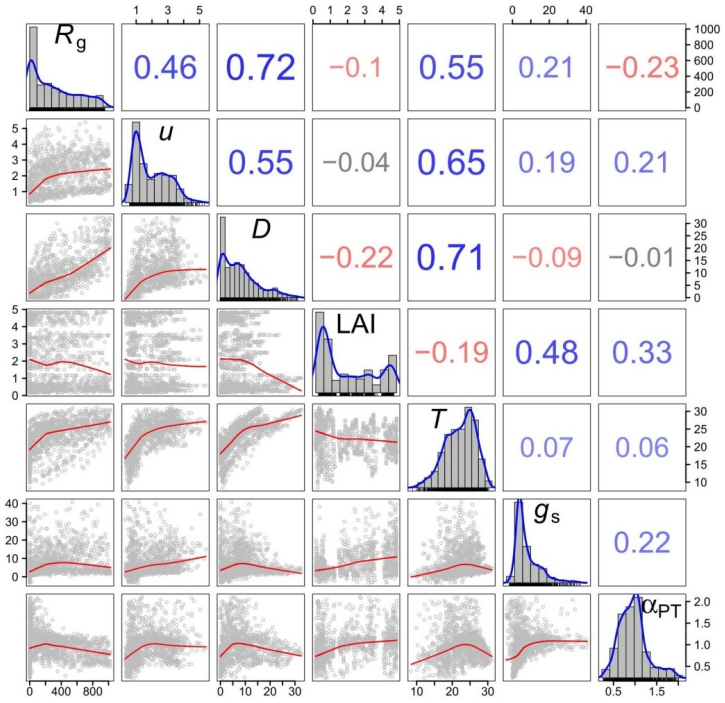



图 1.4: 增强版两两散点图

我们将在第九章学习这种图的作法。

不光是论文作图，R 还能很容易做出 3D 动画来演示。不光是枯燥的科技作图和演示，R 还可以娱乐。比如可以画一颗立体中国心（图 1.5）。

图 1.5: R 绘出的中国心

当然可以很容易地把国旗换成别的。写本文时正值情人节，那就换成

她或他的照片好啦。这种图的作法同样是在第九章。

总地来说，我学 R 的理由，说得文雅一点，那就是：

> 桃之夭夭，灼灼其华。之子于归，宜其室家。
> 桃之夭夭，有蕡其实。之子于归，宜其家室。
> 桃之夭夭，其叶蓁蓁。之子于归，宜其家人。
>
> —《诗经·国风·周南·桃夭》

说得通俗一点：和 R 在一起真好。

下面轮到你了。请勇敢表白一下：你是为什么要学 R 呢？

# 第二章　数据

数据！数据！数据！没有泥的话我没法做出砖。

—夏洛克·福尔摩斯

要是你的数据烂，不管啥统计程序都没救。

— Berton Gunter, April 2005

在第一章里，我们学会了用 R 进行常规的数学运算和统计计算，并且做出了三张图：大气二氧化碳浓度时间序列，降水的季节变化图，北京的 $PM_{2.5}$ 日变化图。好像已经把 R 语言学完了。只是，总不能每次都把数据一个一个敲到代码里吧，也不能只使用 R 自带的数据自娱自乐吧。要是处理你自己文件里的大量数据呢？本篇就解决这个问题。

本书里我们会使用示例数据，方便你跟我们同步操作。示例数据文件可以请 R 来自动生成。运行下面这两行命令：

```
dir.create('c:/r4r')
write.csv(as.data.frame(t(matrix(
  co2, 12, dimnames = list(
    month.abb, unique(floor(time(co2)))))),
  file = 'c:/r4r/co2.csv')
```

将来我们会解释这两条代码的含义，现在你大可略去，只管去 c 盘找到一个名叫 “r4r” 的文件夹，里面有个名叫 “co2.csv” 的文件。这就是我们做示范的示例数据文件，你就当作是从我们的光盘上拷贝过来的吧。请

在参照本书运行示例代码的过程中保留 “c:/r4r” 这个文件夹，我们在后续章节所做的示范都要用到它。

请用 Excel 或记事本打开 co2.csv 这个文件。这是个数据表，内容是 1959 年 1 月到 1997 年 12 月夏威夷 Mauna Loa 观测站的大气二氧化碳浓度，以年份为行，以月份为列。不要做任何修改，我们现在假定这是你即将处理的数据文件，看看如何对其中的数据进行操作。

## 2.1 输入：读取文件

读取文件，就是让 R 把数据读进 R 的脑子里。

如果你喜欢拷贝粘贴的方式，那么可以用 Excel 打开数据文件 co2.csv，用鼠标选中全部数据区（ctrl+a），拷贝，然后在 R 中用下面的代码读取剪贴板里的数据（边输入边试一下 “tab 小助理”。以后每个长命令都用一下，养成习惯）：

```
mydata1 <- read.table(file = "clipboard", header = TRUE)
```

这条指令的含义是：读取剪贴板里的数据，保存到 mydata1 这个数据框变量里。`header = TRUE` 翻译过来就是 “文件头是**真**有啊”，意思是数据表的第一行是列名称。

这时，注意观察 RStudio 的右上窗，出现了 mydata1 的信息。鼠标单击可以查看内容，也可以用输入代码并运行：

```
mydata1
```

```
##       X    Jan    Feb    Mar    Apr    May    Jun    Jul
## 1  1959 315.42 316.31 316.50 317.56 318.13 318.00 316.39
## 2  1960 316.27 316.81 317.42 318.87 319.87 319.43 318.01
## 3  1961 316.73 317.54 318.38 319.31 320.42 319.61 318.42
## 4  1962 317.78 318.40 319.53 320.42 320.85 320.45 319.45
## 5  1963 318.58 318.92 319.70 321.22 322.08 321.31 319.58
```

```
## 6  1964 319.41 320.07 320.74 321.40 322.06 321.73 320.27
## ......
## 38 1996 362.09 363.29 364.06 364.76 365.45 365.01 363.70
## 39 1997 363.23 364.06 364.61 366.40 366.84 365.68 364.52
```

好了，读取数据就是这么简单。如果这已经满足你的需求，那么就可以跳至第 2.2 节，进行后续操作了。不过，我们建议你耐心把本节读完。因为，用拷贝粘贴的方式读取数据，优点是简单灵活易上手，适合临时用一下；缺点是重复性差，下回你可能忘了上次拷贝的是哪个区域的数据，这不是 R 的做事风格。更多情况下，我们要告诉 R，数据文件保存在哪里，只需把上面命令的剪贴板 `clipboard` 换成数据文件的路径即可。下面我们详细介绍这种方法。

不习惯命令行的用户，可以通过下面的指令获取这个文件的路径（请在敲入时练习一下前面说过的 ‘‘箭头快捷键’’ 和 ‘‘tab 小助理’’）：

```
myfile1 <- file.choose()
```

在弹出的窗口中选择文件 c 盘下 r4r 文件夹里的 co2.csv。

好了，现在我们看看 myfile1 的值是什么。在 RStudio 运行：

```
myfile1
```

```
## [1] "c:/r4r/co2.csv"
```

是刚才选取文件的路径。这是获取路径的方法之一，比较符合很多人喜欢鼠标选择文件的习惯，但比较麻烦，每次使用这个代码时都得点一次。一般来说，我们存放数据的路径是固定不变的，所以更常用的方法，是在代码里直接敲入文件路径：

```
myfile2 <- "c:/r4r/co2.csv"
myfile2
```

```
## [1] "c:/r4r/co2.csv"
```

跟鼠标选取文件的结果完全相同。

注意：

- 路径的名称前后要用引号（单双都行，但要成对儿），表示这是一个字符串。
- 文件路径中上下级文件夹之间的斜线必须是斜线（/ ）而不是反斜线（\），Windows 用户一定要注意！其中的道理我们暂不深究。

myfile2 里存储的文件路径，并不是文件内容。R 现在知道文件在哪里，却不知道里面是什么内容。现在，我们让 R 读取文件的内容。

```
mydata2 <- read.table(file = myfile2,
                      header = TRUE, sep = ",")
mydata2
```

sep 参数表示数据列的分隔符，这里设置为逗号，表示读取逗号分隔的数据。

你也许会说，read.table() 括号里那么多东西，用起来也太复杂了吧？怎么记得住？对，谁都记不住，现在我们有请助理团的第二位成员隆重登场！只需要把光标放到代码 read.table 的任何一个字符处，按键盘 F1 键，RStudio 此时会在右下面板显示帮助信息，有详细的解释和实例。好好读读帮助吧，以后你会发现，F1 小助理是仅次于 tab 的常用操作。除了 tab 小助理和 F1 小助理外，以后我们会介绍更多的小助理跟你见面。

---

**小贴士 2.1.** *R 菜鸟入门三大法宝*

- *第一法宝：助理！以 F1 和 tab 键为首的豪华助理团，简直就是身边的诸葛亮，可以随时方便地寻求帮助。*
- *第二法宝：猎狗！就是搜索引擎。遇见问题不懂就上网搜，你会发现，早就有人提出类似的问题并解决了。*
- *第三法宝：顾问！就是论坛。内事不决问统都*[1]*（中文论坛），外事不*

[1]统计之都 R 语言论坛：https://d.cosx.org/t/r

决问爆栈[2]（英文论坛）。

为了省事儿，我们可以用 read.table() 的瘦身简化版 read.csv() 函数，用来专门读取逗号分隔的.csv 文件：

```
mydata2 <- read.csv(file = myfile2)
```

跟上一条指令的效果完全相同。到此为止，数据文件中的数据就被 R 读进了他的脑子里。mydata1 和 mydata2 这种二维表格数据，叫做“数据框”。

你可能会觉得麻烦，怎么在 Excel 里双击一下就搞定的事，在 R 里边却这么麻烦？是的，R 对数据的读入并非“傻瓜”操作，也许在读数据上 R 比 Excel 麻烦 10 倍，但只要读进去了，后面会省事百倍千倍。而且，如果需要读入千百个数据文件，那么配合第五章的循环语句可以轻松搞定，而不必双击千百次。相信我们，磨刀不误砍柴工。

其实，上面的过程是一套分解动作，让我们容易理解读取数据的过程。实际应用时，只需一行代码：

```
mydata2 <- read.csv(file = "c:/r4r/co2.csv")
```

---

**思考 2.1.** *细心的你也许会留意，本章开头有个 write.csv() 函数，跟 read.csv() 有什么关系？既然 read.csv() 是 read.table() 的瘦身版，那么会不会有个 write.table() 函数呢？要弄清楚这些问题，请试试你的三大法宝。*

在 RStudio 中，像 ctrl+shift+n、tab 和 F1 这样的快捷键操作还有很多。从 RStudio 菜单栏选择 Tools – Keyboard Shortcuts Help，或者直接按 alt+shift+k 键，就会弹出一本快捷键魔法书。

[2]爆栈网 R 语言论坛：http://stackoverflow.com/questions/tagged/r

## 2.2 计算：数据处理和作图

R 把数据读入脑子后，就可以开始干活儿了。

我们先让 R 对数据画个图，看起来更直观（图 2.1）：

```
plot(mydata2)
```

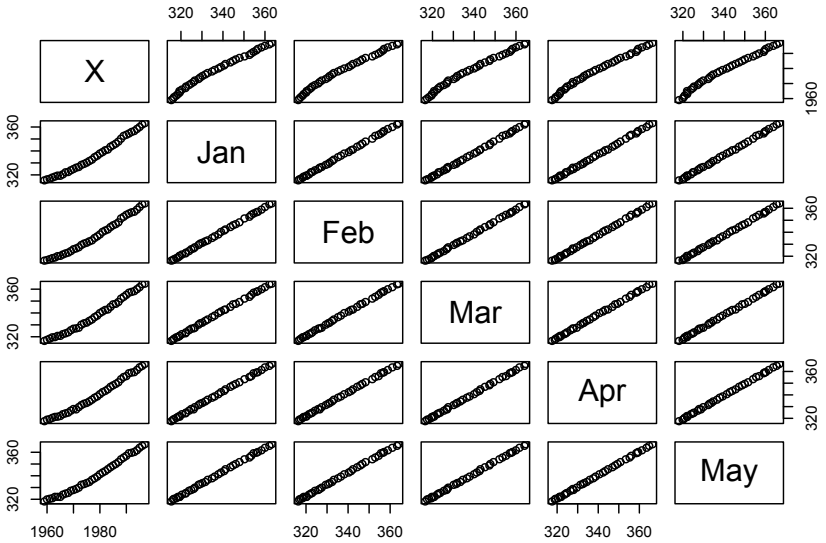



图 2.1: 多张小图一次完成（局部展示）

Bingo！多张小图一次完成！这就是事半功倍的效果。

你想多懒就多懒，并不耽误高效率。

— Kevin Murphy, September 2003

这是任意两列的散点图。画的小图表示的是什么呢？比如从上数第 2 行第 4 列的小图，跟它排在同一行的文字是左边的 Jan，这是小图的纵坐标标签；跟小图排在同一列的文字是下方的 Mar，这是小图的横坐标标签。所以这个小图展示的是以 3 月的二氧化碳浓度为 $x$，1 月份的二氧化碳浓

度为 $y$ 的散点图。其他小图可以类推。那么，第 1 列小图，展示的就是各月二氧化碳浓度的逐年变化。

这么复杂的图，简单一条指令就轻松做出来了。追求高效人士的利器！

---

**思考 2.2.** *使用 Excel 或其他常用的作图软件，如何做出图 2.1?*

我们一般在读入数据文件后的第一件事就是 `plot()` 一下，对数据有个整体的感觉。第二件事，一般是用 `summary()` 看看这个文件的总结报告，这个函数我们在第一章已经打过照面了：

```
summary(mydata2)
```

```
##        X              Jan             Feb
##  Min.   :1959   Min.   :315.4   Min.   :316.3
##  1st Qu.:1968   1st Qu.:323.1   1st Qu.:323.6
##  Median :1978   Median :334.8   Median :335.2
##  Mean   :1978   Mean   :336.4   Mean   :337.2
##  3rd Qu.:1988   3rd Qu.:349.0   3rd Qu.:349.9
##  Max.   :1997   Max.   :363.2   Max.   :364.1
##       Mar             Apr             May
##  Min.   :316.5   Min.   :317.6   Min.   :318.1
##  1st Qu.:324.6   1st Qu.:325.7   1st Qu.:326.3
##  Median :336.5   Median :337.6   Median :337.8
##  Mean   :338.1   Mean   :339.3   Mean   :339.9
##  3rd Qu.:350.6   3rd Qu.:352.1   3rd Qu.:352.9
##  Max.   :364.6   Max.   :366.4   Max.   :366.8
##       Jun             Jul             Aug
##  Min.   :318.0   Min.   :316.4   Min.   :314.6
##  1st Qu.:325.9   1st Qu.:324.9   1st Qu.:322.7
##  Median :337.7   Median :336.4   Median :334.5
##  Mean   :339.3   Mean   :337.9   Mean   :336.0
##  3rd Qu.:352.3   3rd Qu.:350.8   3rd Qu.:349.1
```

```
##  Max.   :365.7   Max.   :364.5   Max.   :362.6
##       Sep             Oct             Nov
##  Min.   :313.7   Min.   :313.2   Min.   :314.7
##  1st Qu.:321.2   1st Qu.:320.9   1st Qu.:321.9
##  Median :332.6   Median :332.4   Median :333.8
##  Mean   :334.2   Mean   :334.2   Mean   :335.5
##  3rd Qu.:347.4   3rd Qu.:347.4   3rd Qu.:348.8
##  Max.   :360.2   Max.   :360.8   Max.   :362.5
##       Dec
##  Min.   :315.4
##  1st Qu.:323.3
##  Median :334.8
##  Mean   :336.7
##  3rd Qu.:350.0
##  Max.   :364.3
```

得到的是每一列数据（也就是各年各月二氧化碳浓度）的最大值、最小值、中位数、平均值函数，顺便还附送了四分位数（1st Qu.，3rd Qu.）。什么是四分位数？试试猎狗。

> 放狗去搜。
>
> — 谢益辉

如果我们想看看数据框里某个指定值，比如 1995 年 9 月二氧化碳的浓度，该怎么选取单元格呢？你还记得上一节我们是如何选取 4 月的降水量吗？`x[4]`。类似地，1995 年是第 37 行，9 月是第 10 列，所以：

```
mydata2[37, 10]
```

```
## [1] 358.11
```

要选取多个行呢？若你还记得用 `c()` 生成一个向量，那就好办了，比如选取第 10 列的全部偶数行：

```
mydata2[c(2, 4, 6, 8, 10, 12, 14, 16, 18, 20,
          22, 24, 26, 28, 30, 32, 34, 36, 38), 10]
```

```
##  [1] 314.00 316.11 316.54 318.48 320.18 322.93 324.68
##  [8] 327.27 329.14 332.60 335.72 337.81 340.90 344.68
## [15] 348.55 350.82 352.94 355.84 359.51
```

行数太多，逐个敲起来太麻烦了吧，高效人士的办法是用 `seq()` 函数生成个数列：

```
mydata2[seq(from = 2, to = 39, by = 2), 10]
```

`seq(from = 2, to = 39, by = 2)` 表示以 2 为起点，39 为终点，间隔（即步长）为 2 生成一个数列。比如步长是 1 的话，就用 `seq(from = 2, to = 39, by = 1)`，等同于 `2:39`：

```
mydata2[2:39, 10]
```

```
##  [1] 314.00 314.83 316.11 316.05 316.54 317.66 318.48
##  [8] 319.10 320.18 322.22 322.93 323.20 324.68 327.35
## [15] 327.27 328.40 329.14 331.42 332.60 333.75 335.72
## [22] 336.52 337.81 339.69 340.90 342.92 344.68 346.27
## [29] 348.55 349.64 350.82 352.05 352.94 353.67 355.84
## [36] 358.11 359.51 360.24
```

如果是选取整行或整列的话, 为了省事儿，可以空出来相应位置：

```
mydata2[, 10]  # 第 10 列全部。
```

```
##  [1] 313.68 314.00 314.83 316.11 316.05 316.54 317.66
##  [8] 318.48 319.10 320.18 322.22 322.93 323.20 324.68
## [15] 327.35 327.27 328.40 329.14 331.42 332.60 333.75
## [22] 335.72 336.52 337.81 339.69 340.90 342.92 344.68
## [29] 346.27 348.55 349.64 350.82 352.05 352.94 353.67
## [36] 355.84 358.11 359.51 360.24
```

```
mydata2[37, ]  # 第 37 行全部。
```

```
##       X    Jan    Feb    Mar    Apr    May    Jun
## 37 1995 359.98 361.03 361.66 363.48 363.82 363.3
##       Jul    Aug    Sep    Oct    Nov    Dec
## 37 361.94 359.5 358.11 357.8 359.61 360.74
```

如果列数太多，总不能老去数第几列吧？别急，也可以用行或列名称来替代列数：

```
mydata2[, 'Sep']
```

或者用美元符号后面跟着列的名称：

```
mydata2$Sep
```

列名称都有哪些呢？其实在输入 $ 符号后，RStudio 就立刻贴心地把所有列名称列出来了，供我们选择。如果没有出现，请咨询 tab 小助理。除此之外，用 names() 函数或 colnames() 函数可以查看列名称：

```
names(mydata2) # 或 colnames(mydata2)
```

```
##  [1] "X"   "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul"
##  [9] "Aug" "Sep" "Oct" "Nov" "Dec"
```

第一列在原始文件中没有名字，所以 R 自动起了个名字叫做 “X”。我们可以把它的列名称改为 “year”：

```
names(mydata2)[1] <- 'year'
names(mydata2)
```

```
##  [1] "year" "Jan"  "Feb"  "Mar"  "Apr"  "May"  "Jun"
##  [8] "Jul"  "Aug"  "Sep"  "Oct"  "Nov"  "Dec"
```

类似的，用 rownames() 函数可以查看行名称。由于我们并没有给各行起名字，R 默认按数字顺序命名。我们可以将年份列作为行名称：

```
rownames(mydata2)
```

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

```
rownames(mydata2) <- mydata2$year
rownames(mydata2)
```

```
##  [1] "1959" "1960" "1961" "1962" "1963" "1964" "1965"
##  [8] "1966" "1967" "1968" "1969" "1970" "1971" "1972"
## [15] "1973" "1974" "1975" "1976" "1977" "1978" "1979"
## [22] "1980" "1981" "1982" "1983" "1984" "1985" "1986"
## [29] "1987" "1988" "1989" "1990" "1991" "1992" "1993"
## [36] "1994" "1995" "1996" "1997"
```

这样，如果要查看 1995 年 9 月的数据，就不用再数第几行第几列了，用行名称和列名称更方便：

```
mydata2['1995', 'Sep']
```

```
## [1] 358.11
```

这对于处理大型的表格尤其方便，省得瞪着眼睛去数数寻找单元格了，很大程度上减少了对视力的伤害。

任何一行或一列，都可以作为向量来计算。我们仿照上一章介绍的向量计算方法，来算一下 1995 年全年二氧化碳的平均浓度，只要对该行的第 2 到 13 列求平均：

```
sum(mydata2['1995', 2:13]) / 12
```

```
## [1] 360.9142
```

**思考 2.3.** 直接使用平均值函数 *mean()* 行不行？为什么？

如果求 1996 年比 1995 年二氧化碳浓度增加了多少，可以料想，只需把两行的平均值相减：

```
sum(mydata2['1996', 2:13]) / 12 -
  sum(mydata2['1995', 2:13]) / 12
```

```
## [1] 1.7725
```

那么，1996 年每个月比 1995 年同比增加了多少二氧化碳？

```
mydata2['1996', ] - mydata2['1995', ]
```

```
##      year  Jan  Feb Mar  Apr  May  Jun  Jul  Aug Sep
## 1996    1 2.11 2.26 2.4 1.28 1.63 1.71 1.76 2.04 1.4
##       Oct  Nov  Dec
## 1996 1.85 1.19 1.64
```

这里我们可以体会向量计算的特点，是逐个对应相减的。

现在，你可以试着对任意一行或一列来做向量计算了。

---

**练习 2.1.** 请计算 1995 年二氧化碳浓度的最大值和最小值。

整行或整列的计算，在 R 中有更方便的方法，可以这样：

```
colMeans(mydata2[, 2:13]) # 排除掉第一列后，对整列求平均
```

```
##      Jan      Feb      Mar      Apr      May      Jun
## 336.4308 337.2033 338.0546 339.2944 339.8821 339.3282
##      Jul      Aug      Sep      Oct      Nov      Dec
## 337.9164 335.9579 334.2428 334.1692 335.4679 336.6946
```

类似的函数还有整行求和 `rowSum()` 和整列求和 `colSum()`。函数名很好记，就是在原来函数前面加个 `row` 或 `col`，并且注意大小写。

下面，我们对整行求平均值，并且把结果作为一列添加到 mydata2 中，列名称叫做 mean：

```
mydata2$mean <- rowMeans(mydata2[, 2:13])
```

如果不是求和或求平均值，而是求其他函数值，比如中位数、最大值、最小值呢？难道也是在原函数名前边添个 row 或者 col 就行了吗？呃，理论上当然是可以做到的（见第九章），但实际上没那个必要，因为那样就需要太多的新函数。我们可以用功能更强大的 apply() 函数：

```
mydata2$median <- apply(X = mydata2[, 2:13],
                        FUN = median, MARGIN = 1)
```

这条代码，表示计算对象是 mydata2[, 2:13] 这个数据框，按行操作（MARGIN = 1），操作的函数是求中位数 median。如果把 median 换成 sum 或 mean，那么就跟 rowSums() 和 rowMeans() 的效果完全一样了。想了解 apply() 函数更多详细情况，请咨询 F1 小助理。

现在，我们可以用上一章认识的各种数学函数，对这个表格进行随便折腾了。

任意相邻年份某个月的二氧化碳浓度增量，也就是相邻两行的差，可以用 diff() 函数：

```
diff(mydata2$Sep)
```

---

**练习 2.2.** 请解释下面代码的含义，猜猜计算出来的结果是什么，再运行，看看跟你猜的是否一致。

```
apply(X = mydata2, FUN = diff, MARGIN = 2)
```

上面我们演示了如何让 R 读取一个数据文件，并简单分析和作图。从此以后，你就可以依葫芦画瓢来处理真正属于你自己的数据了！

回过头来看，我们上面处理的是表格数据。如果要处理其他格式的数据文件，尽管用我们前面介绍的鼠标选定拷贝到剪贴板的方法也可以达到

目的，但是我们仍然建议你用 Excel 或别的软件将它保存或导出为.csv 格式，方便 R 读取和进一步处理，这是首选方案。并且，我们建议你以后把所有数据都尽量从一开始就保存为.csv 格式，因为它就像全世界语言里的英语，到哪里都通用，几乎任何软件都能打开。如果这仍然满足不了你的需求，那么

- 请 F1 小助理搬出 `read.table()` 和 `scan()` 函数的帮助文件；
- 请阅读《R 数据的导入和导出》这本书，来自开源社区，网上免费获取[3]；
- 如果经常处理 Excel 的 xls 或 xlsx 文件，请使用 R 相关的扩展包。这个方法我们会在第九章介绍。

> R 说明文档的水平已经远高于开源软件的平均水平了，甚至高过商业软件（特别是 SPSS，老是摆出一张臭脸，说“如果你不懂输出的东西是什么含义，那么点击帮助按钮，我们会弹出 5 行你仍然不会弄懂的火星语。”）
>
> — Peter Dalgaard, April 2002

## 2.3 输出：保存文件

数据文件的保存比读取要简单多了，用 `write.csv()` 函数即可。下面的语句把 mydata2 这个数据保存到 c:/r4r 文件夹下面，文件命名为“mydata2.csv”。

```
write.csv(mydata2, file = 'c:/r4r/mydata2.csv')
```

---

**思考 2.4.** 用 *Excel* 或记事本打开这个新生成的文件，同时也打开原有的 *co2.csv* 文件，比较两者有什么不同。

现在我们可以解释本章开头两条代码的含义了：先用 `dir.create()`

[3] R 数据的导入和导出：http://pem.freeshell.org/math/R_data_import_export_zh.pdf

函数在电脑里创建了个文件夹，然后用 `write.csv()` 函数将一个名为 co2 的数据经过一系列格式转换后存成了 co2.csv 文件。

那么 co2 这个数据最初是怎么跑到我们的电脑里的？是 R 安装时捆绑自带的。我们可以运行

```
data()
```

就可以看到 R 自带的很多数据文件。这些文件可以在学习 R 的过程里用来做各种测试。

在后面的章节中，如无例外，我们都使用 R 自带的数据做示例，省却读入数据的步骤。当然，如果想练习数据的读入，那么可以用本章开头的方法，把 R 自带数据存成数据文件，然后装作自己的文件从头操作。

今天的活儿干完了。我们关闭 RStudio，收工。RStudio 会弹出一个窗口，问你两件事（图 2.2）：

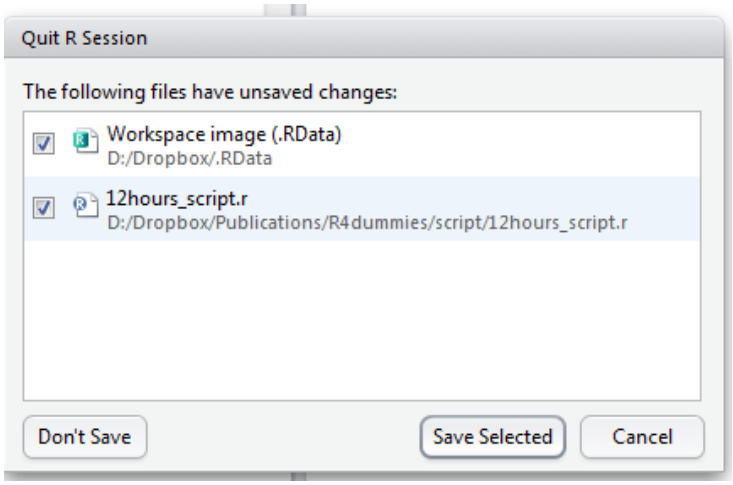



图 2.2: RStudio 退出前保存

1. 你要不要把工作区存到一个.RData 文件里？翻译过来就是：“主人，你要不要我把今天算出来这些数据记在我脑子里，就是右上面板工作区 Environment 标签下列出的里那些东西，下回打开我的时候直接就能用这些数据？” 保不保存随你心意。我们一般选择不保存，让 RStudio

把数据忘得一干二净，下回打开时右上方的窗口是空的，只需重新读入数据和计算就行了，反正原始数据在，而数据的处理方法都在代码里。

2. 你要不要保存.r 代码？也就是左上方窗口这些东东？当然保存了。

现在，才算是真正收工了。

**小贴士 2.2.** 常用数据操作

| 操作 | 提示 |
|---|---|
| 设定文件路径 | 路径里不要空格，不要中文，用斜线分隔，`file.choose()` |
| 读取数据 | `read.table()`, `read.csv()` |
| 保存文件 | `write.csv()` |
| 选取单元格 | `x[2, 3]`, `x[2, ]`, `x[, 3]`, `x$Sep`, `x['1995', 'Sep']` |
| 快速掌握数据 | `plot()`, `summary()` |
| 行列名称 | `rownames()`, `colnames()`, `names()` |
| 行列计算 | `rowMeans()`, `colMeans()`, `rowSum()`, `colSum()`, `apply()` |

## 2.4 课外活动：有 R 伴我走天涯

R 不仅可以安装在本地，还可以从网络访问。在你的手机或电脑里打开浏览器，访问下面介绍的网站，就可以使用 R 语言了。

我们先来试试 Ideone[4]。打开这个网站后，输入图 2.3 上方窗口里的代码，点击 `Run!` 按钮运行，窗口下面就立刻显示运算结果。

Ideone 可以让我们用浏览器来进行 R 编程和调试，非常方便。喜欢的话，可以注册个免费账号，这样就可以将一些代码保存在自己名下，方便

[4]Ideone：`http://ideone.com`

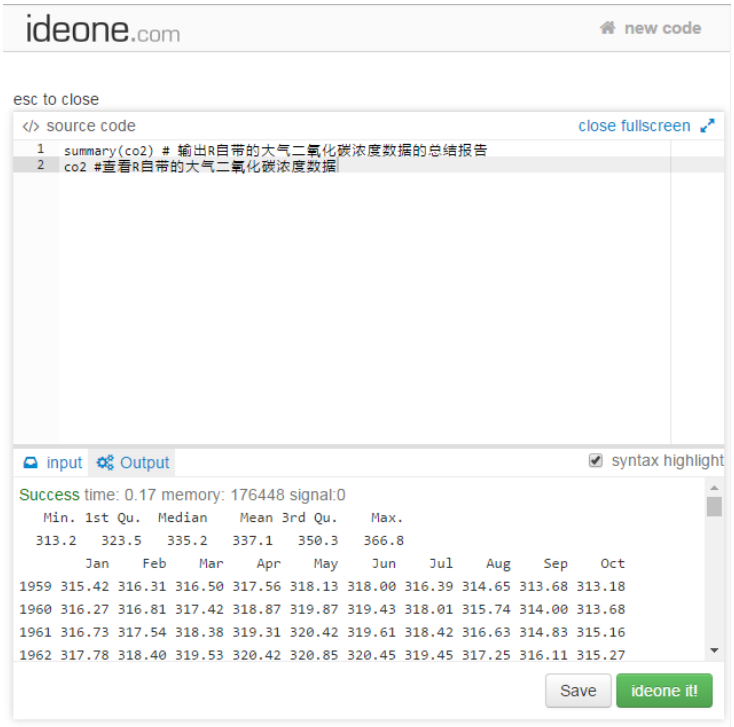



图 2.3: R 在线调试网站 Ideone

以后调用。它还支持把代码嵌入到网页里，以便分享。

由于输入输出接口的限制，Ideone 里的 R 语言不能读入外部数据，也不能实现作图。没关系，还有很多类似网站，能够满足不同的需求，例如 r-fiddle[5]等。

如果你有自己的服务器，那么还可以用 RStudio 的服务器版来搭建自己专属的 R 网站，你的地盘你做主。

有了这些在线的 R 网站，随时随处都可以免费使用 R，没有电脑就用手机。有 R 伴我走天涯，走到哪里都不怕。

[5]r-fiddle: `http://www.r-fiddle.org`

# 第三章　作图

从优雅的角度来讲，R 是精准的、有品位的、美丽的。等我长大了，我要娶 R。

— Andy Bunn, May 2005

一图胜千言。

— 俗语

在上两章中，我们都用到了 `plot()` 函数来作图。如果说 Excel 的作图方法是《秘密花园》那种书，让你在已经画好的图案里涂涂改改，很受约束的话，那么 R 作图的流程更加自由：就像铺开一张白纸，自己打好格，画数据点，画坐标轴，加图例，最后把纸收起来。作图的每一步，都清清楚楚掌控在你手里。

这里，我们用第二章读取的二氧化碳数据 mydata2，画一些更漂亮的图。为了兼顾本章的独立性和跟上一章节的连续性，我们预先把数据读入到 mydata2 中 (当然，也可以用 `read.csv()` 函数把 co2 数据读取进来)：

```
mydata2 <- as.data.frame(t(matrix(
  co2, 12,
  dimnames = list(month.abb, unique(floor(time(co2)))))))
mydata2$year <- as.numeric(rownames(mydata2))
```

## 3.1 控制图像：线型，点状，颜色

我们先做一张最简单的图，只画各年 9 月份二氧化碳的浓度（图 3.1）。

```
plot(mydata2$Sep)
```

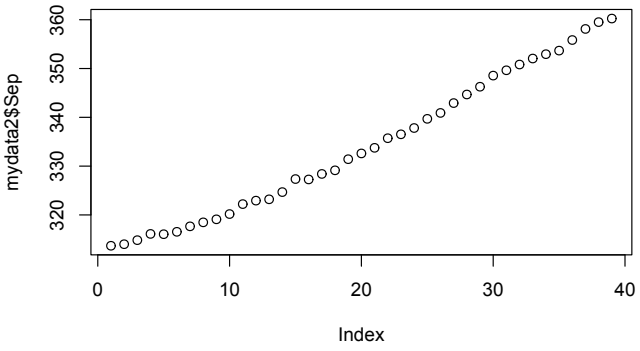



图 3.1: plot() 函数第一种用法示例：一维数据散点图

这是 plot() 函数的第一种用法，也是最简单用法：绘制一维数据散点图。如果 plot() 的作图对象只是一个数值型的向量，那么画出的图纵向是这个向量，横向是数据的序号。

plot() 函数还有别的什么用法呢？可以请 F1 小助理打开帮助文件。不过，今天我们请出我们的新助理：example() 函数:

```
example(plot)
```

运行这条代码，并在 RStudio 的左下面板里按照提示按回车键，就会看到很多示例。以后，想不起来某个函数的作用的时候，除了 F1 之外，小助理 example() 函数也是个很好的选择。

下面我们指定以年份为横坐标 $x$，9 月份的二氧化碳浓度为纵坐标 $y$，做 $xy$ 散点图（图 3.2）：

```
plot(x = mydata2$year, y = mydata2$Sep)
```

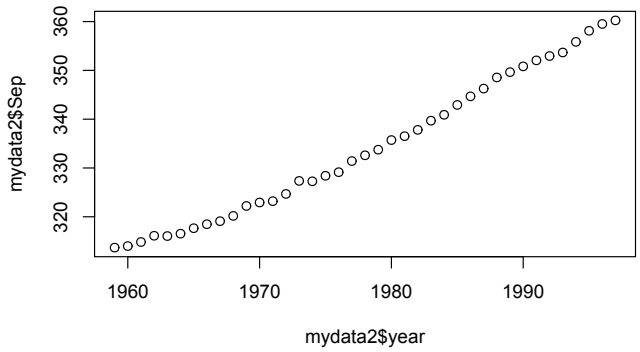



图 3.2: plot() 函数第二种用法示例：二维数据散点图

这是 plot() 函数的第二种用法：绘制二维数据散点图。比较一下，跟第一种用法有什么区别？

plot() 函数的第三种用法，其实在前面已经出现过了，我们重复一次：

```
plot(mydata2)
```

在这种用法里，plot() 的作图对象是个多行多列的数据框（mydata2），画出的是任意两列分别作为 $x$ 和 $y$ 的散点图。这时，plot() 函数等同于 pairs() 函数：

```
pairs(mydata2)
```

喜欢刨根问底的初学者可能对 plot() 函数的多种用法感到困惑。我们打个比方就容易理解了，这就好比佛教里的观音菩萨有 32 种化身，应众生的需要而以不同面孔示人。需要救人参果树的时候，菩萨就持杨柳枝；需要收鲤鱼精的时候，菩萨就编个鱼篮；适当的时候，菩萨还会伸出千手，或者送上个娃。plot() 函数也是如此，根据你的需要来发挥不同的作用。除了这三种化身外，还有第四第五以及更多化身，我们在以后的章节里遇到

再说。

当然，菩萨不止观音一位，R 的作图函数除了 `plot()` 外还有很多，见小贴士 3.1。他们的用法大同小异，可以咨询 `example()` 小助理。不过，我们这次有请 `example()` 助理的小姐妹——示范函数 `demo()` 来帮忙：

```
demo(graphics)
```

就像 `example()` 小助理一样，按照提示按回车键，就会看到各种作图函数的示范了。

---

**小贴士 3.1.** *常用作图函数（请使用 example() 函数来查看，如 example(plot)，或运行 demo(graphics)）*

| 函数 | 用途 |
|---|---|
| `plot()` | 主要用作散点图 |
| `pairs()` | 散点图矩阵 |
| `symbols()` | 气泡图 |
| `hist()` | 直方图 |
| `curve()` | 函数曲线图 |
| `barplot()` | 柱状图 |
| `boxplot()` | 箱式图 |
| `coplot()` | 条件散点图 |
| `dotchart()` | 点图（克利夫兰点图） |
| `stripchart()` | 一维散点图 |
| `image()` | 矩阵方格图 |
| `contour()` | 等高线图 |

在前面这些绘图操作里，我们没有对 R 额外要求什么，于是 R 就按默认值自行标注了坐标轴的名称、取值范围、数据点的类型。下面我们重新画一张图，来指定横坐标名称为 “Year”，纵坐标名称为 “CO2 in Sep”，图

形类型为线形，纵坐标的展示范围为 300 到 400 ppm（图 3.3）。

```
plot(x = mydata2$year, y = mydata2$Sep,
     xlab = "Year", ylab = "CO2 in Sep",
     ylim = c(300, 400), type = "l")
```

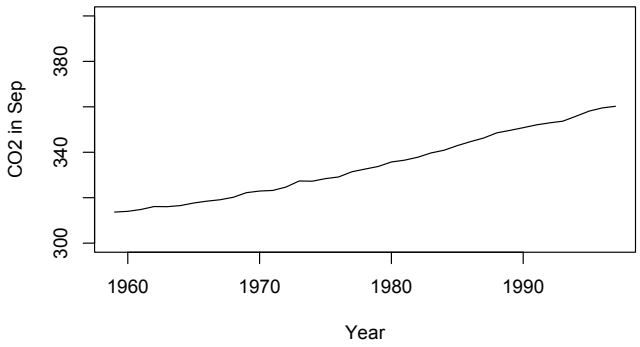



图 3.3: plot() 函数 type 参数的用法示例

我们看到，plot() 函数里除了用 x = 和 y = 两个参数来指定数据点横纵坐标外，还用 xlab（x label 的缩写）、ylab 等参数来指定作图的细节。

R 所有的函数都是这样使用的。比如，我们前面见过的读取数据函数 read.table()，就是用 header 参数来指定要不要把第一行当作列名称，用 sep（separation 的缩写）参数来指定列与列之间用什么符号分隔：

```
mydata2 <- read.table(file = myfile2,
                      header = TRUE, sep = ",")
```

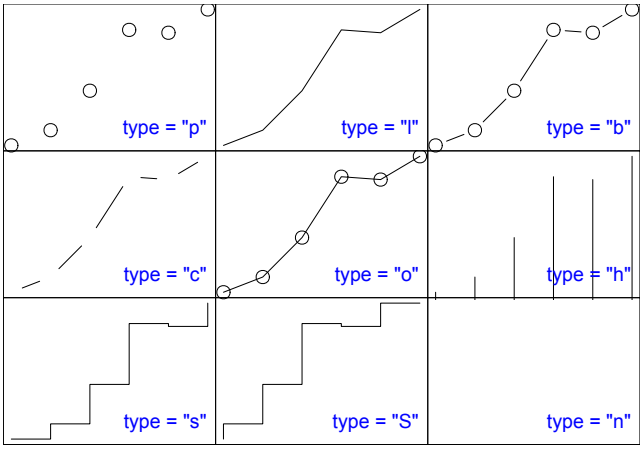
那么，一个函数里允许指定哪些参数呢？我们怎么才能记住这么多不同函数的不同参数呢？

没人记得住，也没必要记住。我们都是用 tab 小助理来调出参数列表后选择一个，或者 F1 小助理来查看帮助文件。下面，我们介绍 plot() 几个常用的参数。

type 参数用来指定把数据点画成点还是画成线。最常用的是 l 表示线（line），p 表示点（point）。此外还可以是 b，c，o，h，s，S，n。见小贴士 3.2。

---

**小贴士 3.2.** *plot() 函数的 type 参数*




我们可以试着把上一句作图命令改为：

```
plot(x = mydata2$year, y = mydata2$Sep, type = "p")
```

当数据点类型设置为 p 时，默认画出来的数据点是个小圆圈（图 3.4）。不喜欢的话，可以用参数 pch（point character 的缩写）来指定数据点的形状（图 3.5）。

```
plot(x = mydata2$year, y = mydata2$Sep, type = "p", pch = 20)
```

pch = 20 表示采用 20 号字符。20 号字符是什么？见小贴士 3.3。

当然，pch 也可以随意是用你喜欢的任何字符，比如我们用字母 “z”（图 3.6）：

```
plot(x = mydata2$year, y = mydata2$Sep,
     type = "p", pch = 'z')
```

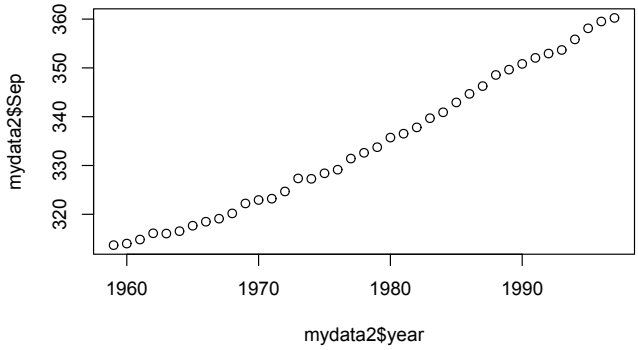



图 3.4: `plot()` 函数：`pch` 默认值

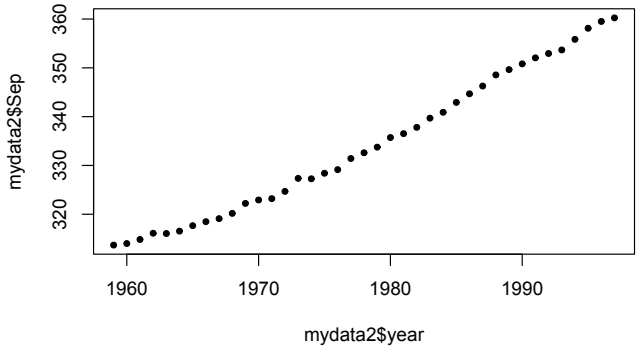



图 3.5: `plot()` 函数示例：`pch = 20`

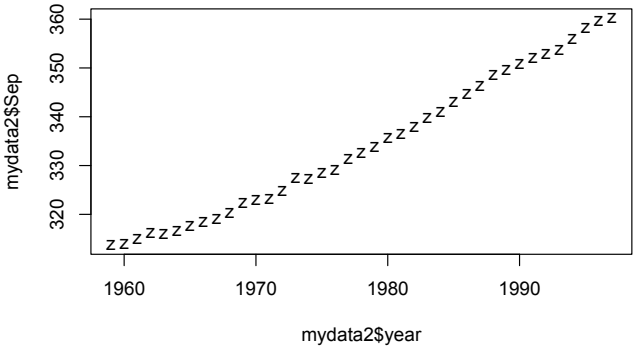



图 3.6: plot() 函数示例：pch = "z"

类似的，当数据点类型是 l（line，线）时，默认是实线。不喜欢的话，可以用 lty 参数（line type 的缩写）指定是虚线还是实线，比如（图 3.7）：

```
plot(x = mydata2$year, y = mydata2$Sep, type = "l", lty = 2)
```

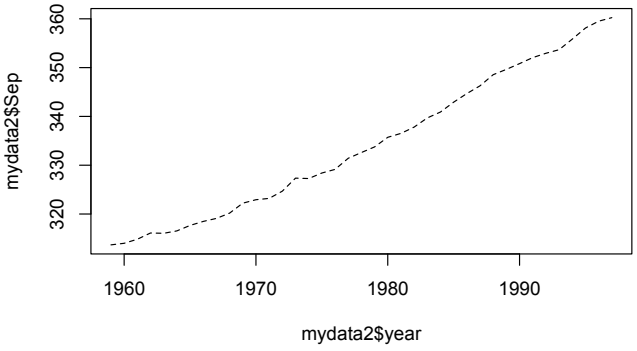
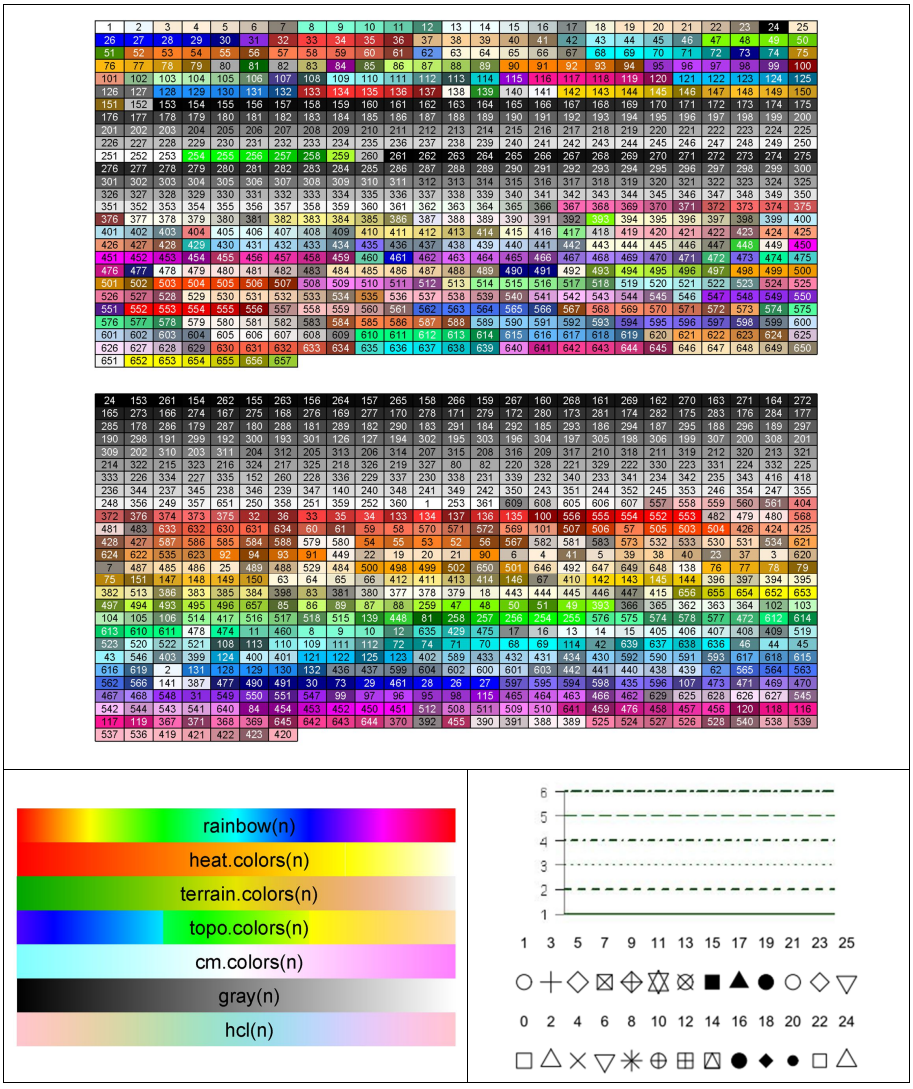



图 3.7: plot() 函数示例：lty 参数

**小贴士 3.3.** *R* 的颜色代码和常用绘图参数。上面板是按编号 $n$ 排序，中面板是按照色调和饱和度排序，每个色块上标出了 $n$ 值。调用时 $col = colors[n]$ 即可。下面板左边是几个常用的色彩函数，右边是绘图参数的线性（*lty*）和点状（*pch*）的取值。



颜色参数 col（color 的缩写）可以设为颜色的名称，比如蓝色（图 3.8）。

```
plot(x = mydata2$year, y = mydata2$Sep, type = "l", lty = 2,
     col = 'blue')
```

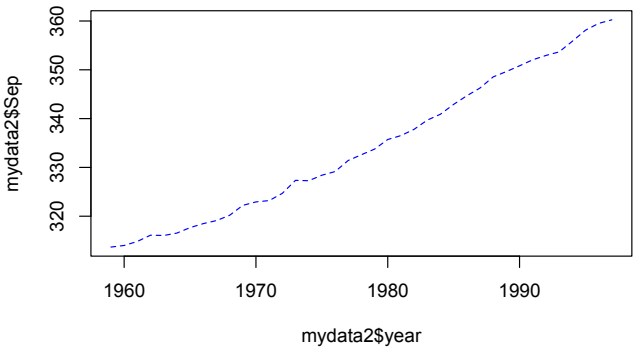



图 3.8: plot() 函数示例：蓝色数据点

R 认识哪些颜色呢？输入并运行：

```
colors()
```

```
##   [1] "white"                "aliceblue"
##   [3] "antiquewhite"         "antiquewhite1"
##   [5] "antiquewhite2"        "antiquewhite3"
##   [7] "antiquewhite4"        "aquamarine"
##   [9] "aquamarine1"          "aquamarine2"
##  [11] "aquamarine3"          "aquamarine4"
## ......
## [653] "yellow1"              "yellow2"
## [655] "yellow3"              "yellow4"
## [657] "yellowgreen"
```

有 657 种颜色名称可以使用。趁此机会学一下外语吧，有些颜色名称

听都没听说过。

我们看看这 657 种颜色中第 26 个颜色叫什么名字：

```
colors()[26]
```

```
## [1] "blue"
```

正是蓝色。`col = colors()[26]` 这个参数设置跟 `col = 'blue'` 是完全等同的。我们可以用颜色编号的方法，给上图的数据点染上不同的颜色。我们有 39 个数据点，那么我们依次染上第 27 到 65 个颜色（图 3.9）：

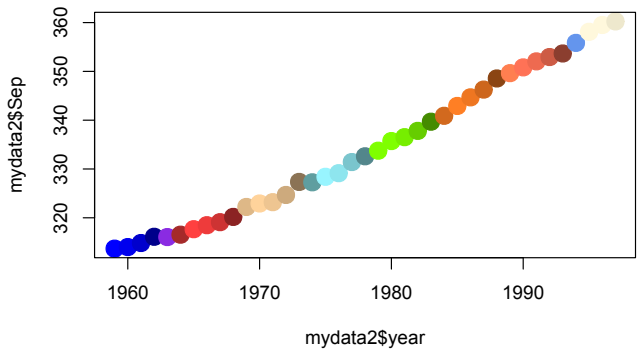



图 3.9: `plot()` 函数示例：多色数据点

从这一刻起，R 开始显得惊艳了。

这里我们用 `cex` 参数（character expansion 的缩写）来设定数据点的绘制尺寸，值越大，点画得就越大。

从名字挑颜色不够直观，经常是想起了颜色却叫不出名字，看见了英文名字却不知道是什么颜色。没关系，我们再次请 `demo()` 小助理来帮忙：

```
demo(colors)
```

按照提示按回车键，就会看到颜色连同名称的展示。

demo() 函数够酷。然而，并非所有函数都可以请 demo() 来帮忙的。运行：

```
demo()
```

会看到一个表格，列出了可供展示的函数。例如，我们看到其中有 persp， 那么可以运行：

```
demo(persp)
```

将来我们学到函数和扩展包时，demo() 小助理还会发挥更大的作用，到时候再说吧。

刚才我们给数据点染上颜色后，发现选的这些颜色对比不是很明显。R 提供了多种配色方案函数可以选用，比如彩虹函数 rainbow()。

```
rainbow(n = 39)
```

```
##  [1] "#FF0000FF" "#FF2700FF" "#FF4E00FF" "#FF7600FF"
##  [5] "#FF9D00FF" "#FFC400FF" "#FFEB00FF" "#EBFF00FF"
##  [9] "#C4FF00FF" "#9DFF00FF" "#76FF00FF" "#4EFF00FF"
## [13] "#27FF00FF" "#00FF00FF" "#00FF27FF" "#00FF4EFF"
## [17] "#00FF76FF" "#00FF9DFF" "#00FFC4FF" "#00FFEBFF"
## [21] "#00EBFFFF" "#00C4FFFF" "#009DFFFF" "#0076FFFF"
## [25] "#004EFFFF" "#0027FFFF" "#0000FFFF" "#2700FFFF"
## [29] "#4E00FFFF" "#7600FFFF" "#9D00FFFF" "#C400FFFF"
## [33] "#EB00FFFF" "#FF00EBFF" "#FF00C4FF" "#FF009DFF"
## [37] "#FF0076FF" "#FF004EFF" "#FF0027FF"
```

每个字符串代表一个颜色。彩虹函数要求给参数 n 设定一个整数值，只有这样 R 才知道该把彩虹的色带平均分割成几种颜色。我们现在有 39 个点，那么就让 n = 39，看看能画出怎样的彩虹（图 3.10）。

```
plot(x = mydata2$year, y = mydata2$Sep, type = "p", pch = 20,
     col = rainbow(n = 39), cex = 3)
```

请用 example() 小助理来调出 rainbow() 函数的示例。可以看到，除

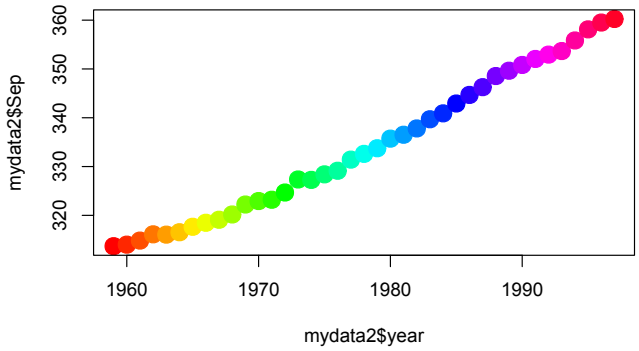



图 3.10: `plot()` 函数示例：`col` 参数彩虹数据点

了 `rainbow()` 外，还有几个类似的色彩函数，用法也类似。小贴士 3.3 展示了他们的色带范围。

有了颜色参数，我们的作图本领更加强大，可以重新考虑一下 mydata2 这个数据框了。

实际上，mydata2 的每个数据点都包含了三个内容：“年份”，“月份”，“$CO_2$ 浓度”。这是个三维数据。我们前面由于刚刚入门，受作图能力所限，只能把数据 “降维打击”[1]成了二维，只保留了 “年份” 和 “浓度” 两个维度（当然也可以只保留 “月份” 和 “浓度”，或 “年份” 和 “月份”，但后者实在无趣）来作图。既然每个点都可以指定一个特有的颜色，那么就可以用颜色来表示这些数据点的第三个维度。

下面，我们将作一个散点图，图上每个点的横坐标是 “月份”，纵坐标是 “年份”，每个点的颜色表示 “浓度”，在色带上，越偏向紫色就表示浓度越大（紫移），越偏向红色就表示浓度越小（红移）。

我们先用 `unlist()` 函数（见第 8.1 节）把 “浓度” 这个数据框转换成一个向量，作为第一个维度：

[1] 降维打击：见刘慈欣《三体 3：死神永生》。

```
myco2 <- unlist(mydata2[, 1:12])
myco2 <- round(myco2)
```

为了简化后面的操作，我们用 round() 函数对浓度数据进行了四舍五入。

然后，用一个向量 myyear 来存储第二个维度“年份”：

```
myyear <- rep(mydata2$year, 12)
```

rep() 是重复函数（repeat 的缩写）。由于每年 12 个月，所以，每个年份重复 12 次。

最后，用向量 mymonth 来存储第三个维度“月份”，每个月份重复的次数是数据里出现的年数，也就是行数：

```
mymonth <- rep(1:12, each = nrow(mydata2))
```

三个维度的数据准备完毕，下面准备一下每个数据点的颜色。我们用 rainbow() 函数根据浓度数值的大小给数据点确定一个颜色：

```
n <- diff(range(myco2)) # 彩虹分割的颜色数量
mycolor <- rainbow(n)[myco2 - min(myco2) + 1]
```

一切准备就绪，可以作图了：

```
plot(x = mymonth, y = myyear,
     col = mycolor, cex = 10, pch = 15)
```

得到的图 3.11 里，沿着横向看就是 $CO_2$ 浓度的逐月变化，沿纵向看就是年际变化。我们的图升维了。

事实上，R 里有另外一个函数 image()，可以一步做出类似的颜色图（图 3.12）。

```
image(t(as.matrix(mydata2[1:12])), col = rainbow(n))
```

image() 函数将在第 5.4 节再度露面，届时我们再做进一步介绍。

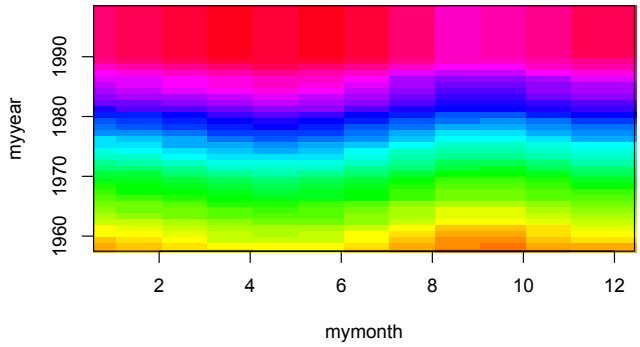



图 3.11: `plot()` 函数示例：三维图

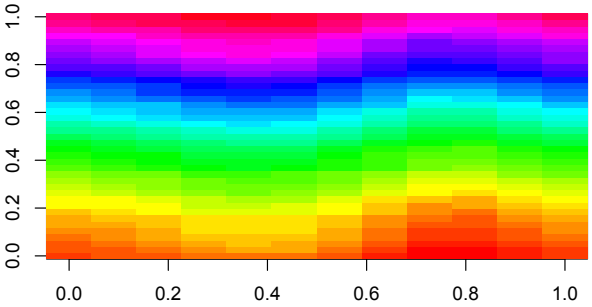



图 3.12: `image()` 函数示例

plot() 函数的参数很多，那么就允许数据有更多维度。例如，把地理位置作为第四个维度，那么我们可以用数据点的不同大小来表示（cex 参数），亚洲的数据用大号数据点，欧洲用中号等等。我们可以随意对图进行维度改造了！

要想对作图进行更精细的控制，可以使用 plot() 函数的其他参数、par() 函数、axis() 函数等。我们将在下文详细介绍，如果你迫不及待的话，请你自己找 F1 小助理问问吧。

---

**练习 3.1.** 请绘制 1959 年到 1997 年二氧化碳全年平均浓度的逐年变化散点图，线形为 "p"，数据点形状为实心三角形，数据点颜色为黄色。

## 3.2 丰富内容：直线，网格，图例

为了让图更容易被读懂，我们经常需要给图像添加直线、网格、图例等，这超出了 plot() 函数的职能范围，需要别的函数来实现。

在现有的图上添加直线，可以用 abline() 函数（图 3.13）：

```
plot(x = mydata2$year, y = mydata2$Sep)
abline(h = 350)
abline(h = 360, v = 1980, col = 'red')
abline(h = seq(from = 320, to = 340, by = 5),
       v = seq(from = 1970, to = 1990, by = 5),
       col = 'grey')
```

h 参数（horizontal）表示水平线，v 参数（vertical）表示垂直线。seq() 函数（sequence 的缩写）用来生成指定的数列。网格线其实就是一组间隔相等的直线，用 seq() 函数生成一组 h 或 v 值即可。

abline 的全称是 "截距 $a$ 和斜率 $b$ 的直线"。顾名思义，除了用 h 和 v 来画水平和垂直线外，还可以用参数 a 和 b 来画斜线。

```
abline(a = -2240, b = 1.3)
```

往图中添加图例，一般使用 `legend()` 函数：

```
legend(x = 1970, y = 350, legend = 'Sep', pch = 1)
```

图例放置的位置有多种设置方法。上面的例子是用坐标位置 $(x, y)$ 确定的。还可以用内置模板来指定，例如用 `topleft` 设置在左上角，`center` 设置在中央等等：

```
legend("topleft", legend = 'Sep', pch = 1)
```

还可以用鼠标确定位置，指哪儿打哪儿：

```
legend(locator(1), legend = 'Sep', pch = 1)
```

上面这个命令运行后，R 会等待你用鼠标左键在图上点击一下，再按 Esc 键，R 就知道点击位置的坐标了。

`legend()` 函数的参数很丰富灵活，可以用来控制图例的点形、线状、颜色、边框等。请咨询 F1 小助理吧。

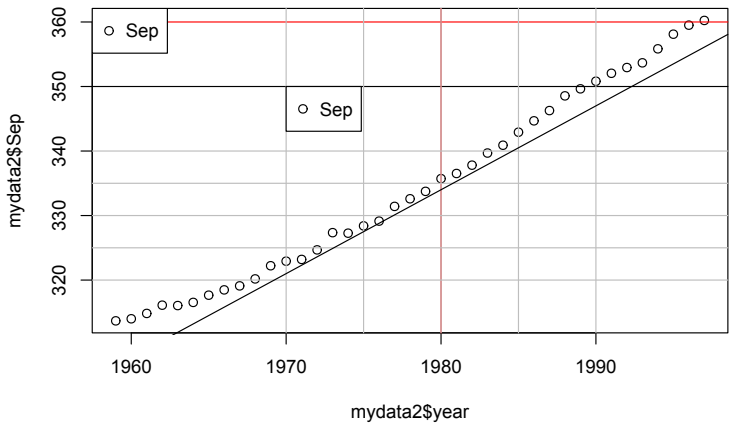



图 3.13: 给图上增添直线、网格和图例

除了增添直线、网格和图例，我们将在第四章里介绍如何用 `text()` 函数增添文字、用 `expression()` 函数添加公式。我们还可以用 `box()` 函数添加边框，`rect()` 函数添加矩形，`arrow()` 函数添加箭头或误差线等。小助理那里什么资料都有，本书就不做详细介绍了。

---

**练习 3.2.** 在图中画满网格线，竖线间隔为 1 年，横线间隔为 10 ppm。

## 3.3 多图合一：三种布局

我们经常需要把多张图画在一起，有时候是为了做对比，有时候为了节省空间。多图合一常见的页面布局大体分三种情况。

第一种情况，是在同一个坐标系里绘制多组 $(x, y)$ 数据点。比如图 3.14，以月份为 $x$，1959 年和 1997 年各月二氧化碳浓度为 $y$。

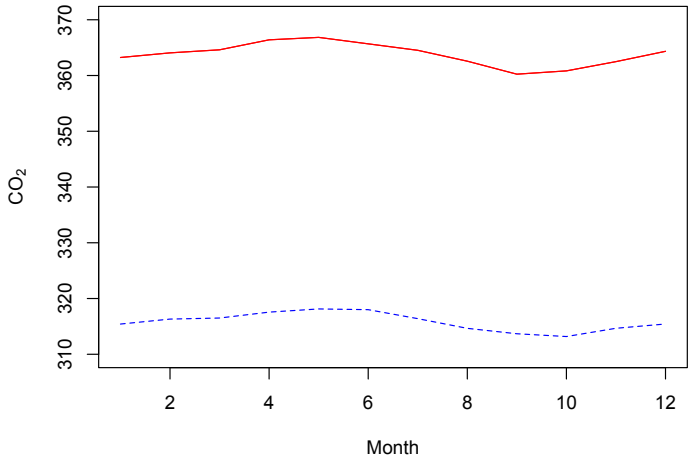



图 3.14: 多图合一布局 1

该布局的特点是两条曲线共享同一个 $x$-$y$ 坐标系。这种图的作法很简单，先按我们前面学过的 `plot()` 函数画出一组 $y_1$ 图线，

```
plot(x = 1:12, y = mydata2['1959', 1:12],
     xlab = 'Month', ylab = expression(CO[2]),
     ylim = c(310, 370),
     type = "l", lty = 2, col = 'blue')
```

再用 points() 或者 lines() 函数添加 $y_2$ 的图线。

```
lines(x = 1:12, y = mydata2['1997', 1:12], col = 'red')
# 或者
points(x = 1:12, y = mydata2['1997', 1:12], col = 'red',
       type = 'l')
```

---

**思考 3.1.** 上面两条代码的作用完全相同。那么这两个函数难道完全一样吗？请搜一下帮助信息，自己比较一下。

刚才画这张图里，两年的二氧化碳浓度的取值范围相差比较大，季节变化显示得不够明显，不如使用另一个新的纵坐标。这就是多图合一的第二种情况，即同样是多组 $xy$，横坐标相同，但纵坐标不同，我们以副坐标轴的形式来展示。方法是各自独立绘图，并且在画两个图之间加一个 par(new = TRUE) 命令，让 R 知道后一张图是重叠在前一张图上：

```
plot(x = 1:12, y = mydata2['1959', 1:12],
     xlab = 'Month', ylab = '1959',
     type = "l", lty = 2, col = 'blue')
par(new = TRUE)
plot(x = 1:12, y = mydata2['1997', 1:12], ylab = '1997',
     type = "l", lty = 1, col = 'red')
```

这回我们没有指定 y 轴的作图范围，让 R 自动选。图画是画出来了，但怎么坐标轴的刻度和标签都重叠在一起了（图 3.15）？

对呀，是我们让 R 重叠的，R 完全是按照我们的指令做的。做得不合意，是因为我们没有把我们的心意对 R 说清楚（请记住这句话，以后会用

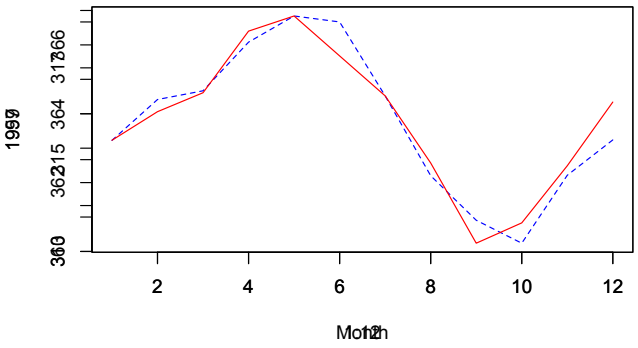



图 3.15: 多图合一布局 2：错误的重叠

得着）。

现在我们重新对 R 交代清楚我们想要的东西：把第二张图的 y 刻度和标签放在右侧，作为副坐标轴。作图分 6 步完成（图 3.16）：

```
# 1. 告诉 R 在右侧为副坐标轴留出空间。
par(mar = c(5,4,4,4))
# 2. 画第一张图。
plot(x = 1:12, y = mydata2['1959', 1:12],
     xlab = 'Month', ylab = '1959',
     type = "l", lty = 2, col = 'blue')
# 3. 告诉 R，下一张图跟第一张图叠加。
par(new = TRUE)
# 4. 画第二张图，但暂时不画坐标轴，也不标标签。
plot(x = 1:12, y = mydata2['1997', 1:12],
     type = "l", lty = 1, col = 'red', axes = FALSE,
     ylab = '', xlab = '')
# 5. 在右侧画出副坐标轴。
axis(side = 4, col = 'red')
# 6. 为副坐标轴添加名称。
```

```
mtext(side = 4, text = '1997', line = 3, col = 'red')
```

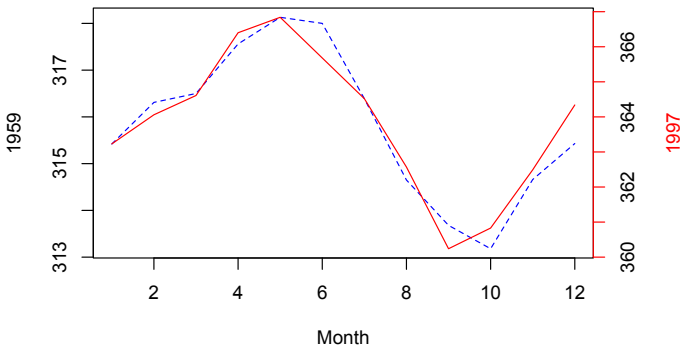



图 3.16: 多图合一布局 2: 正确的重叠

R 是个听话的仆人，吩咐他做什么就做什么；没吩咐的，他按事先的约定（默认值）来做。

上面的第一条和第三条的 `par()` 函数，用参数 `mar`（margin 的缩写）来指定图表下方、左方、上方、右方的空白有多大，用参数 `new` 来指定下一条作图命令跟上一条的作图区域相同。第四条 `plot()` 指令的 `axes` 参数设置为“假”，来禁止显示默认的坐标轴。第五条和第六条指令，先用坐标轴函数 `axis()` 在右侧画一条红色的坐标轴，然后用 `mtext()` 函数在右侧添加文字，作为新坐标轴的标签。“右侧”是 `side` 参数来确定的，它的取值可以是 1、2、3、4，分别表示在图表区的下、左、上、右方向。`line` 参数表示文字与图表区的距离。有了 `axis()` 函数和 `mtext()` 函数，我们就可以对坐标轴为所欲为了。限于篇幅，我们不再做详细介绍，请咨询 F1 小助理。

多图合一的第三种情况，是绘制多个各自独立的小图，把他们拼贴在一张大图上。这仍然可以用 `par()` 函数的参数来实现。比如，我们把上图拆成左右两个小图，只需在作图前用 `par()` 的 `mfrow` 参数（matrix of figures entered row-wise 的缩写）告诉 R，“请把这两张图放在一个表格里，这个

表格是 1 行 2 列”（图 3.17）：

```
par(mfrow = c(1, 2))
plot(x = 1:12, y = mydata2['1959', 1:12],
     xlab = 'Month', ylab = '1959',
     type = "l", lty = 2, col = 'blue')
plot(x = 1:12, y = mydata2['1997', 1:12],
     xlab = 'Month', ylab = '1997',
     type = "l", lty = 1, col = 'red')
```

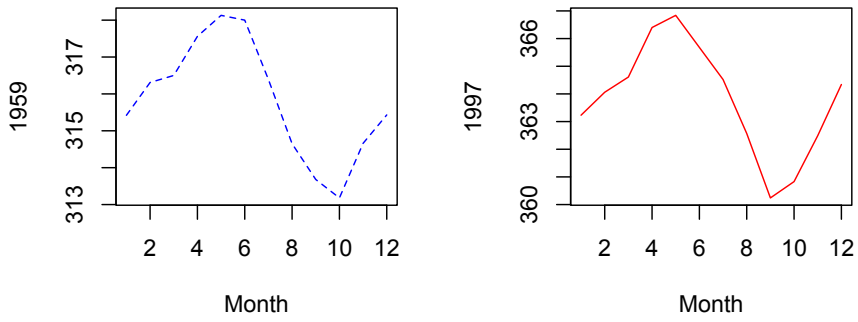



图 3.17: 多图合一布局 3

`par()` 函数的 `mfrow` 参数是将绘图区平均分割成几块来按顺序绘图。如果需要在大图里嵌小图，则需要用 `par()` 函数的 `fig` 参数和 `new` 参数；更复杂的拼贴，可以用 `layout()` 函数来实现，简直可以随心所欲。限于篇幅，这里按下不表，感兴趣的话，请找 F1 小助理看看函数的帮助和示例吧。

## 3.4　保存图片：pdf，png，jpg

我们绘制的图片展示在 RStudio 右下窗绘图区，并没有以独立文件的形式保存，一旦关掉 RStudio，图就随之消失了，除非保存为指定的文件。

把图片保存到电脑里的最简单方法，是在 RStudio 的绘图区点击 Export，选择保存的格式和路径就可以了。不过，我们更常用的方法，是用代码，可以保证下一次运行同一段代码而输出同样的图片。

代码保存图片的流程是这样的：

```
 # 打开一张宽度为 8，高度为 4 的白纸：
pdf('c:/r4r/fig2_13.pdf', width = 8, height = 4)
 # 在白纸上画图：
plot(x = mydata2$year, y = mydata2$Jan)
# 画完了，把纸张收起来：
dev.off()
```

pdf() 函数将图片保存成 pdf 格式。常用的保存图片的指令，除 pdf() 外，还有 png()、jpg() 等。

---

**小贴士 3.4.** 常用作图指令

| 项目 | 内容 |
|---|---|
| 作图函数 | plot(), boxplot() |
| 点形状 | pch = 1, pch = 'z' |
| 线形状 | lty = 2 |
| 颜色 | col = 'blue', rainbow(), colors() |
| 添加要素 | lines(), points(), abline(), axis(), box(), legend(), text(), mtext(), expression() |
| 多图布局 | par(), layout() |
| 保存图片 | pdf(), jpg(), png() |

**练习 3.3.** 做出本章小贴士 3.2 里的那张 9 张小图拼成的图，并保存为 pdf 文件。

## 3.5 课外活动：R 的毛坯房与精装修

你也许已经发现，我们使用的这些作图函数虽然强大灵活，但看起来有些简陋。其实这是很正常的事，因为简陋，才能灵活。好比一套毛坯房，你可以随心所欲地装修成你想要的样子。至于装修之后好看难看有用没用，就看你的水平了。

如果你不喜欢毛坯房，R 也给你提供了多套精装修的房子，那就是我们将在第 9.3 节讲到的扩展包（package）。这里，先给大家简单介绍一下三个作图的扩展包，展示一下 R 的精装房。这里涉及的函数先不必深究，只管运行即可。

首先，我们先安装这三套样板房。

```
install.packages(c('ggplot2', 'lattice', 'plotrix'))
```

我们先来到 ggplot2 (Wickham, 2009) 的展厅，请 `example()` 小助理带我们看看这个展厅的 `qplot()` 函数展台：

```
library(ggplot2)
example(qplot)
```

下面再去 lattice (Sarkar, 2008) 展厅，请 `demo()` 小助理带我们参观：

```
library(lattice)
demo(lattice)
```

最后是 plotrix (Lemon, 2006) 展厅：

```
library(plotrix)
demo(plotrix)
```

感觉怎么样？只要学会了 R，这些图将来我们都能轻松作出来。

你不信？现在我们就把之前做过的图用这几个扩展包重做一次。

为了方便作图，我们先准备一下数据。

```
mydatasub <- t(mydata2[as.character(
  seq(1960, by = 5, length.out = 8)), 1:12])
x <- rep(1:12, 8)
y <- as.vector(mydatasub)
group <- rep(colnames(mydatasub), each = 12)
```

好了，我们先用一下 lattice 扩展包的 `xyplot()` 函数（图 3.18）：

```
library(lattice)
xyplot(y ~ x|group, type = c('p', 'l'),
       xlab = 'Month', ylab = expression(CO[2]))
```

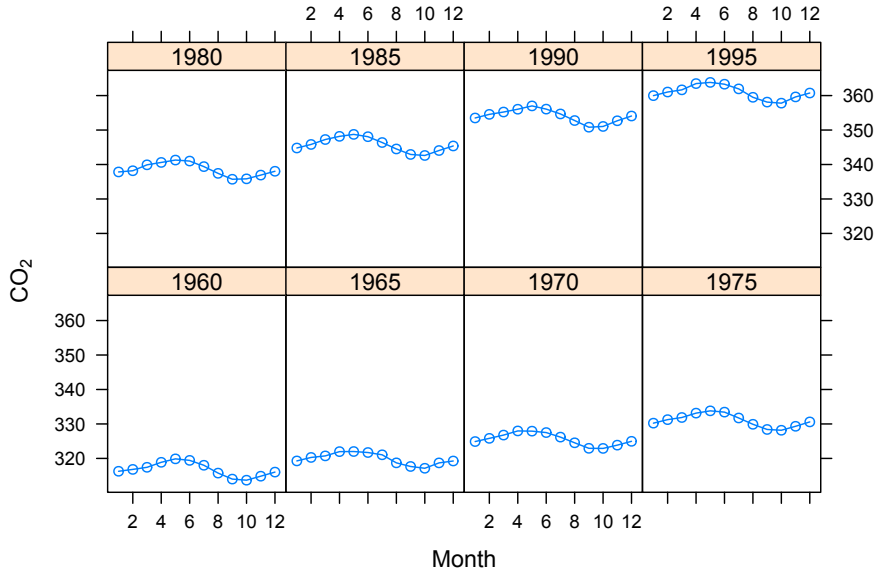



图 3.18: lattice 扩展包作图示例

把选中年份的数据一下就画出来了。

再试试 ggplot2（图 3.19）：

```
library(ggplot2)
qplot(x, y, col = group, geom = c("point", "line"),
      xlab = 'Month', ylab = expression(CO[2]))
```

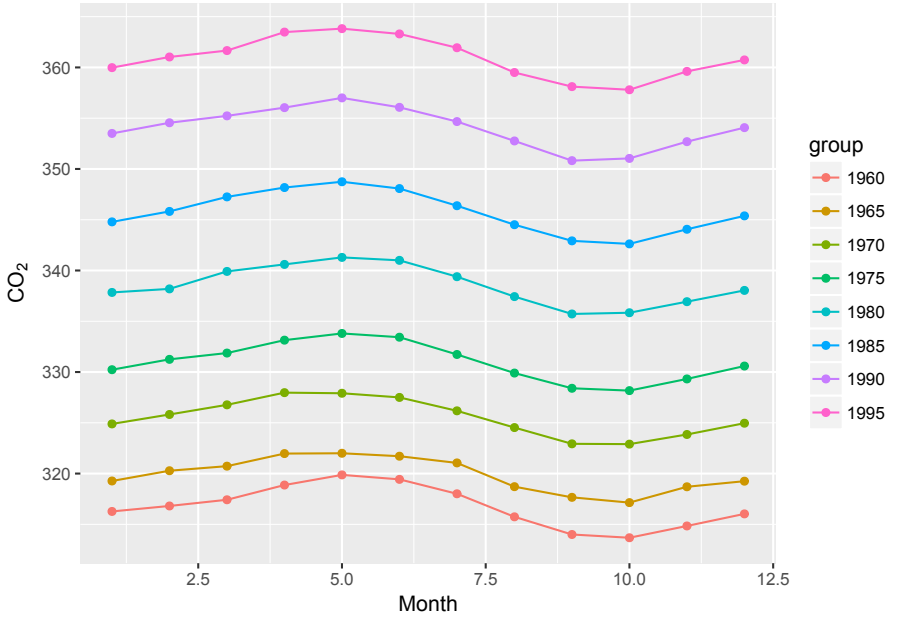



图 3.19: ggplot2 扩展包作图示例

第三个包 plotrix，我们就不演示了，你自己试试吧。

前边我们那些多图合一的方法，在用了扩展包之后，只需简单的指令，就可以做出漂亮的图来了。

那么，R 这些扩展包是怎么做出这么复杂的图的？我们有请助理团最后一位助理，神秘的 `vignette()` 函数隆重登场：

```
vignette('ggplot2-specs')
```

vignette 这个词在中文里实在难以找到恰当的翻译[2]。`vignette()` 小

[2]vignette: 词典上的释义是“小插图，简介短文”，R 社区将它译作“小品文”。

助理掌管的档案跟其他助理都不同。他提供的有时是一些技术文档，有时是论文，有时是示例。我们可以用下面的指令，看看他掌管了哪些资料：

```
vignette(all = TRUE)
```

对于这些资料，现在不明白不要紧，等我们学到第九章就清楚了。

---

**小贴士 3.5.** *R 的人气助理团*

| 名称 | 作用 |
|---|---|
| F1 | 展示函数的帮助信息 |
| F2 | 演示函数源代码 |
| tab | 提示和自动补全 |
| `example()` | 函数示例，圆括号里填函数名 |
| `demo()` | 演示，圆括号为空时查看所有可用演示 |
| `vignette()` | 技术文档，`vignette(all = TRUE)` 展示所有可用文档 |

# 第四章　拟合

你大可以边用边学啊！需要用到的先学，其它的就放一边，只要能善用一些常用到的功能，又何必要那么深入呢？而且您在使用当中经常会发现一些新功能，这又会马上让您给赚到了。

—《大家来学 VIM》

我们在 Excel 里经常需要对散点图做线性或非线性拟合，添加个趋势线，在图上显示个拟合公式。这个操作用 R 可以很轻松地完成，而且能输出更多有用有趣的结果。

## 4.1　线性拟合：散点图的趋势线

我们进行线性拟合示例所用的数据，是 R 自带的世界电话数据 World Phones。我们先准备一下数据，把 WorldPhones 保存为一个数据框，并增加一列年代数据。

```
wp <- as.data.frame(WorldPhones)
wp$year <- as.numeric(rownames(wp))
```

下面，我们看看亚洲跟欧洲的电话数量有没有线性关系。R 内置的线性拟合函数是 `lm()`，用起来很简单：

```
m <- lm(wp$Asia ~ wp$Europe)
```

lm() 函数里参数的格式是因变量 ~ 自变量，也就是 y ~ x。有时候，我们需要强制直线通过原点，只需在拟合时指定自变量加上 0 即可：

```
m0 <- lm(wp$Asia ~ wp$Europe + 0)
```

---

**思考 4.1.** 何时需要强制拟合直线过原点？

线性拟合的主要结果，也就是拟合直线的斜率和截距，都保存在了 m 这个变量里。

```
m  # 查看模型，显示斜率和截距。
```

```
##
## Call:
## lm(formula = wp$Asia ~ wp$Europe)
##
## Coefficients:
## (Intercept)    wp$Europe
##  -3782.5348       0.2915
```

信息量太少了，至少得给个 $R^2$ 和 $p$ 值吧。好办，还记得前面见过的 summary() 函数吗？可以直接用到模型结果上：

```
msum <- summary(m)
msum  # 模型的详细总结报告。
```

```
##
## Call:
## lm(formula = wp$Asia ~ wp$Europe)
##
## Residuals:
##      1      2      3      4      5      6      7
##  369.3 -252.2 -464.8  177.8 -322.1  242.3  249.7
##
```

```
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.783e+03  7.278e+02  -5.197  0.00348 **
## wp$Europe    2.915e-01  2.081e-02  14.012 3.33e-05 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 366.7 on 5 degrees of freedom
## Multiple R-squared:  0.9752, Adjusted R-squared:  0.9702
## F-statistic: 196.3 on 1 and 5 DF,  p-value: 3.33e-05
```

这个信息就丰富多了，给出了残差的分布情况，斜率和截距的拟合值，标准差，$t$ 检验的 $t$ 值、$Pr$ 值和显著性，残差的标准差，$R^2$，$F$ 检验的 $F$ 统计量和 $p$ 值等。关于 $t$ 检验和 $F$ 检验，详见第十七到第十九章。

---

**小贴士 4.1.** 获取拟合结果中各种统计量的函数

| 函数返回 | 值 |
|---|---|
| `summary()` | 主要统计量 |
| `anova()` | 方差分析表 |
| `coef()`, `coefficients()` | 回归系数 |
| `confint()` | 回归系数的置信区间 |
| `deviance()` | 残差平方和 |
| `effects()` | 正交效应向量（Vector of orthogonal effects ） |
| `fitted()` | 拟合的 Y 值向量 Vector of fitted y values |
| `residuals()`, `resid()` | 模型残差 Model residuals |
| `vcov()` | 主参数的协方差矩阵 |

要想从模型报告中提取需要的统计量，例如拟合系数和 $R^2$，那么运行：

```
msum$r.squared
```

```
## [1] 0.975165
```

```
msum$coefficients
```

```
##                   Estimate   Std. Error   t value
## (Intercept) -3782.5348374 727.84976513 -5.196862
## wp$Europe       0.2915207   0.02080546 14.011737
##                  Pr(>|t|)
## (Intercept) 3.476111e-03
## wp$Europe   3.329895e-05
```

记得在 $ 后用 tab 小助理看看都能提取出哪些统计量。

前文我们讲过作图函数 plot() 的三种化身。现在，我们介绍第四种化身。先请 example() 小助理调出 lm() 函数的示例来看看：

```
example(lm)
```

example() 函数实际运行的是 F1 帮助信息里是示例代码，这些代码在 RStudio 的左下面板逐个显示。仔细观察这些代码，可以发现，拟合结果图是用 plot() 函数直接画出来的。是的，这就是 plot() 的第四种化身：当作图对象是个拟合模型时，plot() 函数作出了四张模型诊断图。

下面我们把我们自己的拟合结果画出来。由于共有 4 张图，我们用上一章学到的作图布局方法，把四张图拼在一起：

```
par(mfrow = c(2, 2), mar = c(4, 4.2, 2, 1))
plot(m)
```

plot() 再次展示了强大的威力。这四张图里，第一张是以拟合值为横坐标、残差为纵坐标的对比图，用来展示残差是否均匀分布在直线 $y = 0$ 两侧；第二张是残差 Q-Q 图，用来展示残差是否符合正态分布；第三张是以拟合值为横坐标、标准残差平方根为纵坐标的对比图，仍然是用来展示残差分布状况的；第四张是以杠杆值为横坐标、标准残差为纵坐标的对比

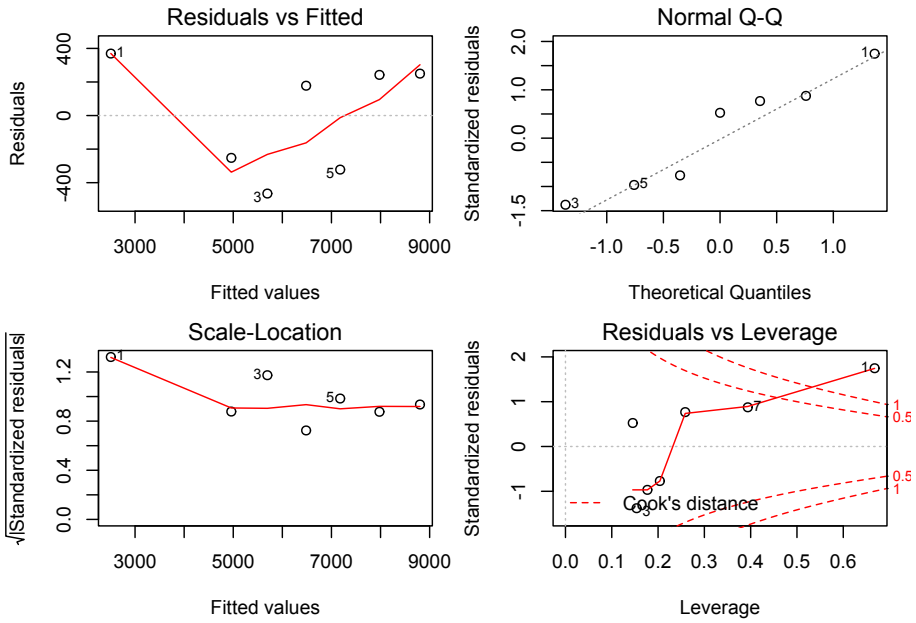



图 4.1: `plot()` 函数展示线性拟合模型的结果

图，展示影响回归结果的异常点。

做统计，R 确实比 Excel 不知高到哪儿去了。只是，信息量有点大，术语有点多。不懂或者忘了的话，不要忘了你的三大法宝。

---

**练习 4.1.** 请在本节示例的散点图中添加强制过原点的拟合直线。

## 4.2　在绘图区添加数学表达式

在展示线性拟合结果时，我们往往只需做个 $xy$ 散点图，添加趋势线和回归方程就够了。对于散点图和趋势线，我们可以按照前文所学的绘图指令来完成。

```
plot(x = wp$Europe, y = wp$Asia, pch = 19)
abline(m, col = "blue")
legend("bottomright", pch = c(19, NA), lty = c(NA, 1),
       legend = c("Data", "Linear fit"),
       col = c("black", "blue"), bty = 'n')
```

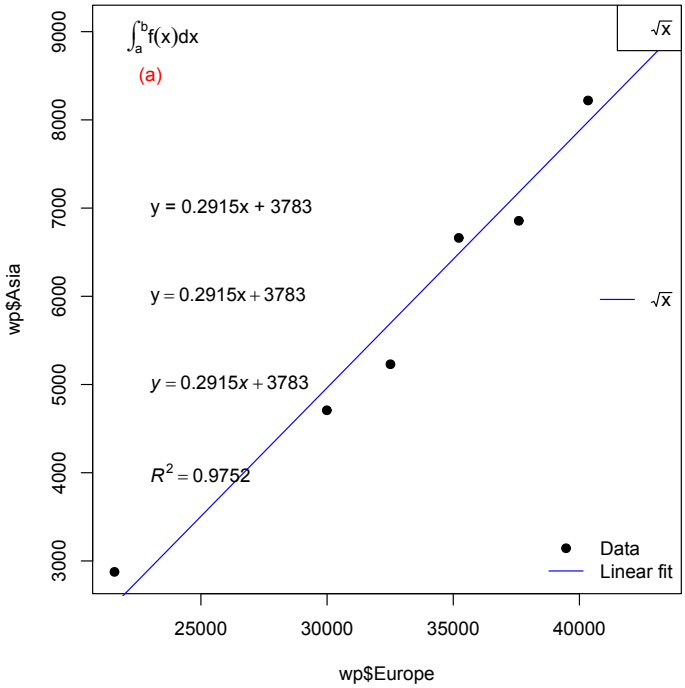



图 4.2: 给散点图添加趋势线和拟合方程

---

**思考 4.2.** *在这里，我们并未给 abline() 函数设定参数 $a$ 和 $b$，那么拟合直线是怎么作出来的？*

对于添加回归方程，我们先了解一下如何在绘图区内添加文字。

绘图区添加文字，一般用 `text()` 函数。它跟 `legend()` 函数的用法很像。比如，我们在图的左上角 (23000, 8500) 这个坐标位置添加文本 “(a)”，颜色为红色。

```
text(x = 23000, y = 8500, labels = '(a)', col = 'red')
```

除了使用指定的坐标位置外，跟 legend() 一样，放置文字的位置也可以 locate() 函数以鼠标点击的方式来确定。请自己试一下吧。

---

**思考 4.3.** *看起来 legend() 跟 text() 有很多相似之处，那么，什么情况下他们两个的用处完全相同？*

下一步，我们将回归方程以添加文本的方式添加到图上。从拟合结果里，可以找到拟合直线的斜率是 0.2915，截距是 3783。那么，方程可以写成一个字符串，然后用 text() 函数把这个字符串添进去：

```
eqlm1 <- 'y = 0.2915x + 3783'
text(x = 23000, y = 7000, labels = eqlm1, adj = 0)
```

要求不高的话，这就算是把公式添进图里了。

当然，严格来讲，这个表达式里的 x 和 y 应该用斜体来表示。这个需求是字符串满足不了的，我们需要用表达式函数 expression() 生成一个表达式：

```
eqlm2 <- expression(y == 0.2915 * x + 3783)
text(x = 23000, y = 6000, labels = eqlm2, adj = 0)
```

注意，在表达式里，乘号不能省略，而等号要用双等号表示。

eqlm2 跟 eqlm1 在图上貌似没差别呀？别急，还没完。在expression()函数里，可以增添需要的格式，例如斜体，用 italic() 函数：

```
eqlm3 <- expression(italic(y) == 0.2915 * italic(x) + 3783)
text(x = 23000, y = 5000, labels = eqlm3, adj = 0)
```

这回有区别了吧？

如果你觉得区别不明显，下面我们来添加 $R^2$ 值到图里：

```
eqr2 <- expression(italic(R) ^ 2 == 0.9752)
text(x = 23000, y = 4000, labels = eqr2, adj = 0)
```

^ 符号是个格式标记，expression() 函数看到这个标记之后，就知道后面紧跟的是上标了。如果没有 expression() 函数，仅凭字符串是很难得到上标符号的。

---

**思考 4.4.** *如果我偏要用字符串，坚持不用 expression() 函数，那该怎么输入上标？*

expression() 函数生成的表达式变量，不仅可以用在 text() 里作为文字插入绘图区，还可以用在 legend() 函数里插进图例，用在 mtext() 函数放在坐标轴外，用在 plot() 的 xlab 或 ylab 参数上作为坐标轴标签（见第 3.5 节）。

expression() 函数提供了一个强大的数学环境，允许插入任何复杂的数学符号。例如，我们添加一个开平方表达式：

```
txt1 <- expression(sqrt(x))
```

然后，我们用 legend() 函数，将 txt1 添加到图例里（当然，图中并没有这条开平方曲线，这里仅仅作为练习）。

```
legend('topright', legend = txt1)
```

可见，expression() 函数把 sqrt 符号自动转换成了开方符号。我们为上面这个图例增加线型，去除边框：

```
legend('right', legend = txt1, lty = 1, col = 'blue',
       bty = 'n')
```

下面, 我们再插入一个积分符号，用 integral() 函数：

```
txt2 <- expression(integral(f(x) * dx, a, b))
legend('topleft', legend = txt2, col = 'blue', bty = 'n')
```

常见的数学符号，见小贴士 4.2。

---

**小贴士 4.2.** 常用公式符号

| | | | | | |
|---|---|---|---|---|---|
| x %+−% y | $x \pm y$ | x %==% y | $x \equiv y$ | x %<−% y | $x \leftarrow y$ |
| x%/%y | $x \div y$ | x %prop% y | $x \propto y$ | x %up% y | $x \uparrow y$ |
| x %*% y | $x \times y$ | x %~% y | $x \sim y$ | x %down% y | $x \downarrow y$ |
| x %.% y | $x \cdot y$ | plain(x) | $x$ | Alpha − Omega | $A - \Omega$ |
| x[i] | $x_i$ | italic(x) | $x$ | alpha − omega | $\alpha - \omega$ |
| x^2 | $x^2$ | bold(x) | $\mathbf{x}$ | phi1 + sigma1 | $\varphi + \varsigma$ |
| x * y | $xy$ | bolditalic(x) | $\boldsymbol{x}$ | Upsilon1 | $\Upsilon$ |
| paste(x, y, z) | $xyz$ | underline(x) | $\underline{x}$ | infinity | $\infty$ |
| sqrt(x) | $\sqrt{x}$ | hat(x) | $\hat{x}$ | 32 * degree | $32^\circ$ |
| sqrt(x, y) | $\sqrt[y]{x}$ | tilde(x) | $\tilde{x}$ | 60 * minute | $60'$ |
| x != y | $x \neq y$ | ring(x) | $\mathring{x}$ | 30 * second | $30''$ |
| x < y | $x < y$ | bar(xy) | $\overline{xy}$ | sum(x[i], i = 1, n) | $\sum_{i}^{n} x_i$ |
| x <= y | $x \leq y$ | widehat(xy) | $\widehat{xy}$ | prod(plain(P)(X == x), x) | $\prod_{x} P(X = x)$ |
| x >= y | $x \geq y$ | widetilde(xy) | $\widetilde{xy}$ | integral(f(x) * dx, a, b) | $\int_{a}^{b} f(x)dx$ |
| x %~~% y | $x \approx y$ | x %<−>% y | $x \leftrightarrow y$ | lim(f(x), x %−>% 0) | $\lim_{x \to 0} f(x)$ |
| x %=~% y | $x \cong y$ | x %−>% y | $x \rightarrow y$ | | |

**练习 4.2.** 图 3.3 的纵坐标标签 “CO2”，这样写不规范，请将其中的 “2” 改成下标。

## 4.3　非线性拟合：一个指数递减模型

实验观测到一组 $x$ 值和 $y$ 值，假定 $y$ 对 $x$ 的响应关系用一个非线性方程来表达，那么一般可以使用 `nls()` 函数来得到方程的系数。nls 就是 Non-Linear Simulation（非线性模拟）的缩写。

下面举个例子。为了描述方便，我们随机生成一组数据[1]：

```
x <- seq(0, 50, 1)
y <- runif(1, 5, 15) * exp(-runif(1, 0.01, 0.05) * x) +
  rnorm(51, 0, 0.5)
```

[1]此处 $y$ 是随机生成的，所以重复运行这段代码得到的 $y$ 会不同，拟合的结果自然也不尽相同。若要重现本书的结果，请先运行 `set.seed(123)`。

假定上面这组数据是我们实验观测得到的，并且假定我们已知二者之间的关系可以用下面这个方程来描述：

$$y = ae^{bx}$$

我们作图看看他们的关系：

```
plot(x,y)
```

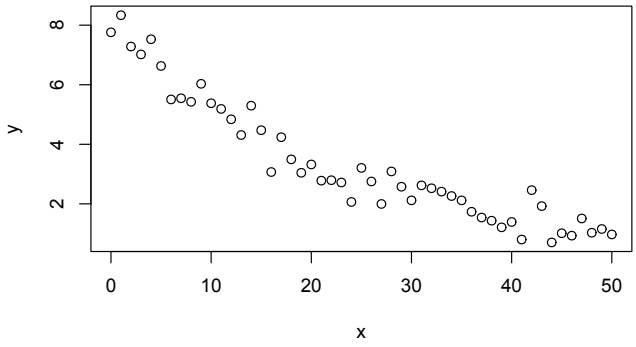



图 4.3: 用作非线性拟合的散点图示例

从两者响应的方程和图来估计，$a$ 是当 $x = 0$ 时的 $y$ 值，约为 8；当 $x$ 约为 50 时，$y$ 约为 1，那么根据对 $a$ 的估计，我们可以来估计 $b \approx -\log(1/a)/50$，并以此估计值作为 $a$ 和 $b$ 的初始值，来拟合得到精确值。

```
a_start <- 8
b_start <- - log(1/a_start) / 50
m <- nls(y ~ a * exp(-b * x),
         start = list(a = a_start, b = b_start))
m
```

```
## Nonlinear regression model
##   model: y ~ a * exp(-b * x)
```

```
##    data: parent.frame()
##       a       b
## 8.04087 0.04253
##  residual sum-of-squares: 10.3
##
## Number of iterations to convergence: 3
## Achieved convergence tolerance: 1.437e-06
```

可见，$a$ 的拟合值为 8.04，$b$ 的拟合值为 0.0425。拟合曲线的预测值跟实测值的相关系数为

```
cor(y,predict(m))
```

```
## [1] 0.9758937
```

要想查看总的拟合报告，我们再度使用人见人爱的 `summary()` 函数:

```
summary(m)
```

```
##
## Formula: y ~ a * exp(-b * x)
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## a 8.040872   0.190289   42.26   <2e-16 ***
## b 0.042525   0.001612   26.37   <2e-16 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4584 on 49 degrees of freedom
##
## Number of iterations to convergence: 3
## Achieved convergence tolerance: 1.437e-06
```

显示结果的格式跟前面的线性拟合是类似的。

---

**练习 4.3.** 请上面的指数递减模型的散点图上添加拟合曲线，并在绘图区添加图例和拟合方程。

## 4.4 课外活动：助理团与自助餐

求人之前先求己，人助之前先自助。我们在第三章介绍了 R 的人气助理团，并且频繁提醒你使用他们。相信你现在对 F1、`example()` 和 `demo()` 已经从陌生到熟悉了吧？只要再进一步，深入了解一下他们的工作方式，我们就能学到更多有价值的东西。

例如，我们先敲个函数名 `lm()`，再将光标移到 `lm` 函数名的任意位置按 F1，RStudio 右下面板就弹出了 `lm()` 的完整帮助信息。注意看其中的最后一部分 Examples。

同时，我们请 `example()` 小助理调出线性拟合函数 `lm()` 的示例:

```
example(lm)
```

RStudio 左下面板会提示你按回车键，每按一次就多显示一些示例。左右面板对比一下，你就会惊喜地发现，左下面板的黑色代码和右下面板帮助信息的 Examples 部分完全相同！左下面板只是多显示了运行结果而已。只要把帮助信息里的代码拷贝粘贴到代码窗口运行，跟 `example()` 函数运行的结果是一样的。

`example()` 小助理顿时哭了：主人，这秘密都被你发现了……以后你是不是只要 F1 不要我了……

其实 `example()` 比按 F1 再拷贝粘贴还是方便很多了。当然，这个因人而异，你喜欢哪个就用哪个好了。

下面一个要哭的是 `demo()`：

```
demo(plotmath)
```

这里，demo() 将表达式函数里的数学符号在 RSudio 右下面板的作图区展示出来。那么这些图是怎么作出来的？只要详细研究一下 demo() 运行后在左下面板出现的代码就知道了。

事实上，小贴士 4.2 就是从中摘录绘制而成的。

---

**思考 4.5.** *借用 demo(plotmath) 演示的代码，从中挑出你常用的数学符号，可以自制自己专属的小贴士备忘录。想想该如何制作。*

好了，现在你可以用 example() 和 demo() 试试别的函数，从中偷学一些绝技了。

关于 F1、example()、demo() 和 vignette() 这些助理更多的背景和秘密，我们将在第 9.4 节进一步深挖。

学习 R 语言，不愁没资料，不愁身边没人帮忙。F1，F2，example，demo，vignette，stackoverflow，google，这是一桌丰盛的自助餐。根本不用讨饭吃，好吃的都摆在你眼前，就看你愿不愿动筷子夹到自己碗里。

# 第五章　循环

男人至少要擅长一项运动，一种乐器，一种编程，和拿手的几个小炒。

—《小鸟学 AHK》

男女都一样。

— 大鹏赶紧补充

在我们的生活被阿尔法狗完全接管之前，机器比较适合做重复性的工作。比如你有 1000 组数据要做线性拟合，比如你有 1000 张照片文件要加上个拍摄日期，这种操作费时费力并且毫无乐趣，就应该交给机器来解决，这是高效人士的必备技能。本篇就来说说来进行重复工作的基本结构：循环。

## 5.1　假如没有循环

循环就是兜圈子。先从一个简单例子开始。在练习 3.3 中，我们做出了一张 3 行 3 列的组合图。那时我们还没有学到循环函数，那时的世界过的是这样的苦日子：

```
mydata2 <- as.data.frame(t(matrix(co2, 12,
    dimnames = list(month.abb, unique(floor(time(co2)))))))
```

```
mydata2$year <- as.numeric(rownames(mydata2))
x <- mydata2$year
y <- mydata2$Sep
par(mfrow = c(3,3), cex = 1.2, mar = c(3, 2, 0.5, 1))
plot(x = x, y = y, type = 'p')
legend('topleft', legend = 'p', cex = 0.8,
       bty = "n", text.col = 'blue')
plot(x = x, y = y, type = 'l')
legend('topleft', legend = 'l', cex = 0.8,
       bty = "n", text.col = 'blue')
plot(x = x, y = y, type = 'b')
legend('topleft', legend = 'b', cex = 0.8,
       bty = "n", text.col = 'blue')
plot(x = x, y = y, type = 'c')
legend('topleft', legend = 'c', cex = 0.8,
       bty = "n", text.col = 'blue')
plot(x = x, y = y, type = 'o')
legend('topleft', legend = 'o', cex = 0.8,
       bty = "n", text.col = 'blue')
plot(x = x, y = y, type = 'h')
legend('topleft', legend = 'h', cex = 0.8,
       bty = "n", text.col = 'blue')
plot(x = x, y = y, type = 's')
legend('topleft', legend = 's', cex = 0.8,
       bty = "n", text.col = 'blue')
plot(x = x, y = y, type = 'S')
legend('topleft', legend = 'S', cex = 0.8,
       bty = "n", text.col = 'blue')
plot(x = x, y = y, type = 'n')
legend('topleft', legend = 'n', cex = 0.8,
       bty = "n", text.col = 'blue')
```

虽然是以拷贝粘贴的方式，但也够烦的，不仅无趣，而且容易出错，还浪费纸张，占了本书整整一页纸。都什么时代了，还在用驴拉磨。

经过仔细观察，我们发现，其实每张小图都差不多，除了数据点的类型和图例发生变化外，基本是在重复。这种情况，就可以考虑用循环了。

## 5.2　循环是个救世主

当循环降临到这个世界上，我们的好日子就来了，上面那段臃肿的代码可以瘦身写成这样：

```
par(mfrow = c(3, 3), cex = 1.2, mar = c(3, 2, 0.5, 1))
for(i in c('p', 'l', 'b', 'c', 'o', 'h', 's', 'S', 'n')) {
  plot(x = mydata2$year, y = mydata2$Sep, type = i)
  legend('topleft', legend = i, cex = 0.8,
         bty = 'n', text.col = 'blue')
}
```

这三句跟上面的十几行语句效果完全等同。`i` 在 `c('p', 'l', 'b', 'c', 'o', 'h', 's', 'S', 'n')` 这些字符中按顺序取值走一遭；每走一个值，就执行一次后面方括号里的所有操作。这里出现的 `for()` 就是循环语句。

循环就是周而复始，每次的操作都类似，除了 `for()` 函数里的 `i` 值会发生变化，就像诗里写的：

> 究竟
> 这一头有几个人能够等你
> 下一个轮回翩然来归
> 至少我已经不能够
> 我的白发，纵有叁千丈怎跟你比长
> 下次你路过，人间已无我
>
> — 余光中《欢呼哈雷》

哈雷彗星的运行就是循环，76 年一个循环周期。他兜一个圈子回来，我（i）的值已经变了……

虽然写循环语句的时候要多费点脑子，但代码看上去简洁很多，最重要的是修改起来容易，比如我们写完代码后悔了，想把所有的线都改成点，那么循环语句里改一次就行了，不用逐条修改，而且不容易遗漏出错。循环次数越多，写循环语句就越划算。高效人士的又一大福音啊！

i 走的那个圈，可以是字符，也可以是别的，比如数字：

```
for(i in 1:20) print(i)
```

由于每走一圈只需执行一条语句，所以花括号就省略了。

到目前为止，R 中所有的括号都闪亮登场了一遍，小贴士 5.1 总结了三种括号在 R 里扮演的角色。

---

**小贴士 5.1.** *R 语言里三种括号的用法*

- 圆括号 *()*。用作函数和数学表达式，如 *mean(x), (1 + 2) * 3*。
- 方括号 *[]*。用作下标，也就是：*(1)* 调用向量中的某个元素，如 *x[3], pm[2,6]*，以及 *(2)* 公式 *expression()* 里的下标。
- 花括号 *{}*。用作组合，把一组指令组合在一起，如 *for() {}*。

---

**练习 5.1.** 用 `print()` 指令打印 1 到 100 之间的奇数。

R 中常用的循环语句，除了 `for()` 之外，还有 `while()` 和 `repeat()`，从逻辑上来说相互之间可以换用，我们这里就不做介绍了。

## 5.3 人口增长模型

下面我们用循环语句来做一个指数增长模型，再体会一下循环的意义。

指数增长模型，也叫马尔萨斯增长模型，一个函数的增长率与其函数值成比例。举个例子吧[1]，我们用 N1 表示 2008 年世界总人口 66.8 亿，r 表示人口自然增长率。为简化起见，假定 r 是个常数 0.011，那么在下一年的总人口就是 N2 = N1 + r * N1。我们想看看今后一百年的人口总数。

第一种方法可以是逐年计算，把各年的人口分别存储到不同的变量中：

```
r <- 0.011
N1 <- 66.8
N2 <- N1 + r * N1
N3 <- N2 + r * N2
# ..... 如此写 99 行，就可以写到 N100。
N100 <- N99 + r * N99
```

第二种方法，逐年计算，用一个向量来存储各年人口总数。

```
r <- 0.011
N <- numeric(100)
N[1] <- 66.8
N[2] <- N[1] + r * N[1]
N[3] <- N[2] + r * N[2]
# ..... 如此写 99 行，一直写到 N[100]。
N[100] <- N[99] + r * N[99]
```

第三种方法，用循环语句计算，用一个向量来存储各年人口数。

```
r <- 0.011
N <- numeric(100)
N[1] <- 66.8
for(t in 1:99) N[t + 1] <- N[t] + r * N[t]
```

对比这三种方法，循环的好处一目了然。

由于数据储存在一个向量里，我们很容易把人口增长曲线画出来。

[1] 本例来自 Björn Reineking 教授 Introduction to R 课程讲义。

```
Y <- seq(2008, length.out = 100)
plot(Y, N)
```

根据这个模型的预测，哪一年世界人口超过 100 亿？当然，可以用肉眼在数据里看。这里，我们画一条 y = 100 的横线，然后运行鼠标定位函数 `locator()`，接着在横线与指数曲线的交点点一下鼠标，按一下 Esc 键就可以了：

```
abline(h = 100)
locator(1)
```

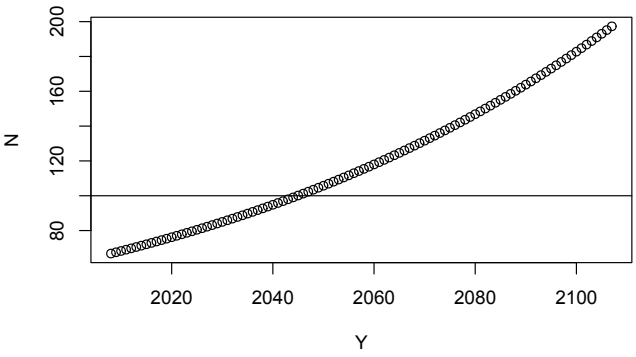



图 5.1: 人口增长模型

---

**练习 5.2.** 生成一个矩阵 $m$ （请 F1 小助理调出 `matrix()` 的帮助信息），令 `m[i, j] = x[i] * y[j]`，其中 `x <- c(2, 3, 5)`，`y <- c(1, 2, 3, 4)`。

## 5.4　制作动画

动画的原理，其实就是把多幅静态的图片连续播放。根据这个原理，我们在循环中使用 `plot()` 画图，相邻两图之间有少许变化，连起来就是一幅动画, 例如：

```
for(i in 1:360) {
    plot(1, ann = F, type = "n", axes = F)
    text(1, 1, "Ninja, go!", srt = i, col =
        rainbow(360)[i], cex = 7 * i/360)
}
```

下面，我们用类似的方法，来玩一个科学游戏：康威生命游戏。

游戏的设定是这样的：每个玩家有一个国际象棋棋盘，除了边缘外，每个方格被 8 个方格包围。每个方格里可以居住一个细胞，假定每个细胞周边养活细胞的资源是有限的。

游戏开始时，由玩家选任意多个方格，每个方格放进去一个活细胞。在下一个时刻，细胞的生死由相邻 8 个方格中的活细胞数量决定：

- 当相邻的活细胞多于 3 时，那么由于僧多粥少，中央的这个细胞就会饿死；
- 当相邻的活细胞少于 2 时，中央这个细胞会因太孤单而死；
- 只有在相邻活细胞数量刚好是 2 或 3 时，中央的细胞才会活下去；这种情况下，即使中央原本没有细胞，也会诞生一个新细胞。

所以，这个游戏有趣之处在于，游戏开始时该如何放置细胞，放多少个，才能让尽可能多的细胞存活尽可能长的时间？

我们用 R 代码来实现这个游戏[2]。用到的绘图函数是 `image()`，在第 3.2 节已经牛刀小试了一把，现在我们请 F1 和 `example()` 两位助理来帮个忙深入了解：

[2]本例来自 Björn Reineking 教授 Introduction to R 课程讲义。

```
example(image)
```

可见，image() 绘制的是颜色网格图。

跟上面的例子类似，我们这回用 image() 函数做出静态图片，用循环指令来连续播放。

```
install.packages('simecol')
require(simecol)
# 40 * 40 的棋盘：
m <- matrix(0, 40, 40)
# 玩家放置细胞的初始条件。0 表示该位置没有细胞，1 表示有细胞：
m[5:35, 19:21] <- 1
# 白色表示没有细胞，绿色有细胞：
image(m, col = c("white", "darkgreen"), axes = FALSE)
for(i in 1:200) {
  nn <- eightneighbours(m)
  m.old <- m
  # 当周围有三个细胞时该位置产生细胞：
  m[m.old == 0 & nn == 3] <- 1
  # 当周围细胞少于 2 个（太孤单）或大于 3 个（太拥挤）时，
  # 该位置细胞死亡。
  m[m.old == 1 & (nn < 2 | nn > 3)] <- 0
  image(m, col = c("white", "darkgreen"), axes = FALSE)
  Sys.sleep(0.1)
}
```

这里，我们使用了 simecol 扩展包 (Petzoldt and Rinke, 2007)，是为了使用其中的 eightneighbours() 函数。关于扩展包详见第九章。这个函数有什么作用呢？就像旧版 Windows 里的扫雷游戏，每个方格的数字表示周围 8 个方格里地雷的数量之和，而这个函数就用来计算矩阵的每个元素相邻周围 8 个元素之和。

游戏很好玩，如何把动画录制下来保存呢？只要在作图前后增加打开

和关闭图片的函数就可以了，比如存成 png 格式的图片：

```
png(paste("c:/R/data/conway_",
          formatC(i, width = 2, flag = "0"), ".png",
          sep = ""),
    width = 300, height = 300)
image(m, col=c("white", "darkgreen"), axes = FALSE)
dev.off()
```

进一步用其他软件（如 ffmpeg[3]）把这些图片连接之后保存在一个文件里，就可以得到一个动画文件。

---

**思考 5.1.** 所谓失之毫厘，谬以千里。如何用康威生命游戏来模拟蝴蝶效应？当初始条件只改变了一点点，结果会有多大的不同？

R 还可以制作 3D 立体动画。比如，我们可以使用 rgl 扩展包 (Adler et al., 2017)，绘制一个能旋转的带数据点的三维坐标系。

```
install.packages('rgl')
require(rgl)
example(movie3d)
```

## 5.5　超越循环

循环曾经是我们的救世主，但是当我们觉醒之后，却将会试图超越他，因为我们拥有了两种更强大的力量：向量计算和因子计算。

R 语言的精华思想之一，是用向量计算来替代循环，简化代码，提高效率。向量计算的例子早在第一章就出现过了。如果你还记得，我们曾经给向量 $x$ 里每个元素加了 100：

[3] ffmpeg：https://ffmpeg.org/

```
x <- c(61, 45, 55, 46, 56, 79, 86, 57, 56, 56, 57, 71)
x + 100
```

其中隐藏的逻辑等同于对 x 的所有元素循环逐个操作一遍：

```
for(i in 1:12) print(x[i] + 100)
```

又如第二章遇到过的 `rowMeans()`，`rowSums()`，`diff()` 等函数，都可以在形式上转换成循环计算。

---

**练习 5.3.** 第二章里，我们计算 mydata2 的各列平均值用的是向量计算函数：`colMeans(mydata2[, 2:13])`。请改写成循环语句，每次求一列的平均值，循环 12 次。体会循环与向量计算的联系和区别。

再如，第三章作出的图 3.18 和图 3.19，每幅图里边都有多条曲线，如果不使用扩展包 ggplot2 和 lattice，那么就得像本章开头的例子那样逐个画出来，或者用循环语句。而使用了这两个扩展包后，一条语句就画出来了。这得益于作图之前对数据分了组，每组分别作图。而用来分组分类的变量，就是因子（factor）。因子的存在，使得很多计算变得简单轻松了。

这里，我们继续用 R 自带的数据 WorldPhone，来简单介绍一下因子。

这个数据原来的格式中不含有因子，我们调整一下格式，生成 mydata3，内容跟原来没有差别，只是变成了列数为 3 的数据框，三列依次是电话数量、年份、洲名：

```
wp <- as.data.frame(WorldPhones) # 转化为数据框类型
wp$year <- as.numeric(rownames(wp)) # 将行名称转化为数值类型
mydata3 <- data.frame( # 生成一个新数据框
  nphone = unlist(wp[, 1:7]), year = rep(wp$year, 7),
  conti = rep(names(wp)[1:7], each = nrow(wp)))
```

`mydata3$nphone` 有 49 个数值，可以按年份分成 7 组，也可以按洲名分成 7 组，这个分组信息就是因子：

```
summary(mydata3)
```

```
##      nphone            year           conti
##  Min.   :   89   Min.   :1951   Africa  :7
##  1st Qu.: 1769   1st Qu.:1956   Asia    :7
##  Median : 3000   Median :1958   Europe  :7
##  Mean   :16435   Mean   :1957   Mid.Amer:7
##  3rd Qu.:29990   3rd Qu.:1960   N.Amer  :7
##  Max.   :79831   Max.   :1961   Oceania :7
##                                 S.Amer  :7
```

```
str(mydata3)
```

```
## 'data.frame':    49 obs. of  3 variables:
##  $ nphone: num  45939 60423 64721 68484 71799 ...
##  $ year  : num  1951 1956 1957 1958 1959 ...
##  $ conti : Factor w/ 7 levels "Africa","Asia",..: 5 5 5 5
5 5 5 3 3 3 ...
```

str() 函数用来查看数据的结构，返回的结果告诉我们，mydata3 前两列是数值型，第三列是因子型。由于我们打算把年份这一列也作为对 nphone 这一列的分组信息，那么就需要用 as.factor() 函数把 year 这一列转换成因子型：

```
mydata3$year <- as.factor(mydata3$year)
str(mydata3)
```

```
## 'data.frame':    49 obs. of  3 variables:
##  $ nphone: num  45939 60423 64721 68484 71799 ...
##  $ year  : Factor w/ 7 levels "1951","1956",..: 1 2 3 4 5
6 7 1 2 3 ...
##  $ conti : Factor w/ 7 levels "Africa","Asia",..: 5 5 5 5
5 5 5 3 3 3 ...
```

好了，现在，mydata3$year 也是个因子了。因子的取值叫做“水平”

(level)。看看因子有几个水平，水平分别是什么：

```
nlevels(mydata3$year)
```

```
## [1] 7
```

```
levels(mydata3$year)
```

```
## [1] "1951" "1956" "1957" "1958" "1959" "1960" "1961"
```

```
nlevels(mydata3$conti)
```

```
## [1] 7
```

```
levels(mydata3$conti)
```

```
## [1] "Africa"   "Asia"     "Europe"   "Mid.Amer"
## [5] "N.Amer"   "Oceania"  "S.Amer"
```

---

**练习 5.4.** 以年份为因子，将电话数量做出箱型图 `boxplot()`。如果以洲为因子呢？

因子有什么用呢？既然因子就是分类变量，那么用处当然就是对数据分类了。请看下面的例子：

如果我们要计算 mydata3 每年这几大洲的电话总和，可以利用循环函数，对每个年份因子依次计算。

```
for(i in levels(mydata3$year)) {
  print(sum(mydata3$nphone[mydata3$year == i]))
}
```

```
## [1] 74494
## [1] 102199
## [1] 110001
## [1] 118399
## [1] 124801
```

```
## [1] 133709
## [1] 141700
```

虽然能算出结果，但逐个计算的效率很低。幸好，我们有 tapply() 函数（table apply 的缩写）：

```
tapply(mydata3$nphone, mydata3$year, sum)
```

```
##   1951   1956   1957   1958   1959   1960   1961
##  74494 102199 110001 118399 124801 133709 141700
```

这样，就很方便地得到了整齐的表格。

tapply() 是 apply() 家族的函数之一。这个家族成员长得很像，脾气秉性却各有不同。我们再挑两个来举例。

前面用过的二氧化碳数据，如果想得到每一列的几个常见统计量，以前我们是这样操作的：

```
mydata2 <- read.csv(file = "c:/r4r/co2.csv")
smr1 <- summary(mydata2)
smr1
```

虽然可以把统计量保存在表格里，看起来也算美观，但想取出来里边的数据却比较麻烦，因为全是字符，不是数字。比如我们想计算这几十年的所有 1 月份数据里最大值（smr1[6, 2]）与最小值（smr1[1, 2]）的差，也就是极差，那么下面的计算是不行的：

```
smr1[6, 2] - smr1[1, 2]
```

```
## Error in smr1[6, 2] - smr1[1, 2]:
##  non-numeric argument to binary operator
```

我们将来在第十章会学到，字符型变量是不能做数学运算的。那么，如果想调用 summary() 得到的数据做后续计算该怎么办？用 lapply() 或 sapply() 函数：

```
smr2 <- lapply(mydata2, summary)
smr2[[2]][6] - smr2[[2]][1]

smr3 <- sapply(mydata2, summary)
smr3[6, 2] - smr3[1, 2]
```

`lapply()` 返回的值是个列表（list），我们将来会择吉日在第 8.1 节介绍；而 `sapply()` 返回的是个我们熟悉的数据框。这样，就可以进行后续计算了。

除了 `apply()`, `tapply()`, `lapply()`, `sapply()`, 这个家族还包括其他成员，如 `rapply()`, `vapply()`，`mapply()` 等。他们让很多原本需要循环才能完成的计算都变成了轻松爽快的向量计算。这里我们略过不表，感兴趣的话就找 F1 和 `example()` 两位助理吧。

很多函数对因子青睐有加，遇见因子就会做特殊考虑。比如第 3.5 节提到的 ggplot2 和 lattice 扩展包。再如，`plot()` 函数如果遇到了因子，就会呈现第五种化身（图 5.2）：

```
plot(x = mydata3$year, y = mydata3$nphone)
```

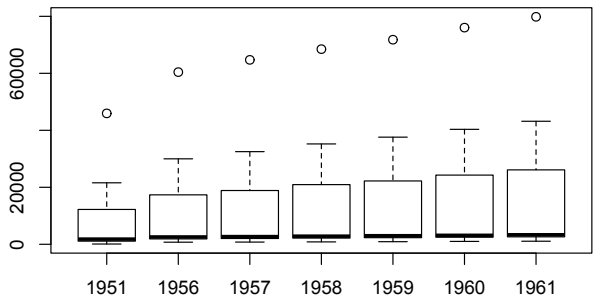



图 5.2: 当 `plot()` 遇见因子

由于 $x$ 是用来分组的因子，plot() 认为我们是要查看每组数据的分布情况，就自动化身为箱型图。等同于

```
boxplot(mydata3$nphone ~ mydata3$year)
```

利用向量计算和因子计算来代替循环，虽然效率高，但对新手来说，脑子需要多绕两个弯。如果不习惯，那么，继续使用循环也没有什么不妥。还记得我们的入门第二秘诀吗？

能用就行！

---

**练习 5.5.** 用 tapply() 和示例数据，分别计算各年几大洲的电话数量的最大值、最小值、中值。

## 5.6　课外活动：信息提示

有些循环比较耗时。比如，在对大量数据作成百上千张复杂的图形时，可能每张图都要耗时几秒钟，那么完成所有工作就需要几十分钟。在等待的过程中，让 R 适当地给个进度提示，会比较有助于估算剩余的时间。

如果想提示进度，最简单的是在代码里加入 print() 函数，正如我们在人口曲线中所用的代码一样。我们也可以让操作系统弹出对话框来提示。下面这个例子，我们用 winDialog() 函数，在每计算完一个年份的人口数量时，就将循环暂停，弹出个对话框，直到你关闭之后才计算下一个年份。

```
N <- numeric(100)
N[1] <- 66.8
r <- 0.011
for(t in 1:3)
{
  N[t+1] <- N[t] + r * N[t]
  winDialog(type = c("ok"),
            message = paste('population in', t + 2007,
```

```
                                'will be', N[t + 1])) # 信息提示框
}
```

当然，我们也可以把 winDialog() 放在一段代码的结尾，运行之后就可以放手去做别的事情，直到听到信息提示音，就知道 R 把工作做完了（我们将在第 13.2 节演示这样一个例子）。

winDialog() 连同我们前边见过的 file.choose() 函数一样，都属于 R 与用户交互对话的函数。这样的函数常见的还有 winDialogString()，用来让用户在弹出的窗口输入文字并返回 R：

```
n <- winDialogString(
  message = "which year's population would you like to see",
  default = '2050')
winDialog(type = c("ok"), message = paste(
  'population in', n, 'will be', N[as.numeric(n) - 2007]))
```

上面这个例子里，第一行用来让用户输入个年份，第二行把该年的预测人口数以对话框的形式弹出来。当然，这两行可以合成一句：

```
winDialog(type = c("ok"), message = paste(
  'population in', winDialogString(
    message = "which year's population would you like to see",
    default = '2050'), 'will be', N[as.numeric(n) - 2007]))
```

采用这样的方法，就实现了用户和 R 的交互对话。

# 第六章　分支

日出日落，月圆月缺，年尾年头，这是“循环”；

上学还是就业，单身还是结婚，丁克还是生娃，这是“分支”；

不管是循环还是分支，都嵌入在生老病死的时间轴上，这是“顺序”。

所谓尽人事听天命，想来就是心平气和地接受顺序结构，小心翼翼地制订循环结构，在关键时刻控制好分支结构，就这样度过一生罢。

—大鹏志[1]

顺序、循环和分支是编程的三大结构。顺序最简单，就是运行完第一行接着运行第二行；上一章我们学习了循环；这一章说说分支。

什么是分支？分支就是你到了一个路口，向左向右向前看，要选择到底往哪个方向走。每时每刻我们都要做出选择：渴了，是喝咖啡还是茶？见了人，是上去打招呼还是悄悄躲开？这本书读到这里，是扔掉还是继续读下去？这都是分支。

简单来说，分支就是先判断再选择。而所谓判断，就是逻辑运算。

## 6.1　判断：逻辑运算

关于逻辑运算，让我们先来做一套健脑操热热身：

[1]大鹏志：http://dapengde.com

```
3 > 2  # 3 比 2 大？是真（true）的。
```

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

```
1 > 2  # 1 比 2 大？是假（false）的。
```

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

```
!(1 > 2)  # 1 不比 2 大？呃……
```

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

```
3 > 2 & 1 > 2  # 3 比 2 大，并且 1 比 2 大？假的。
```

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

```
3 > 2 | 1 > 2  # 3 比 2 大，或者 1 比 2 大吗？真的。
```

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

```
3 > 2 & !(1 > 2)  # 3 比 2 大，并且 1 不比 2 大？好像是真的吧...
```

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

好了，热身完毕。包括上面出现的，常用的逻辑运算符有：大于 >，小于 <，等于 ==，不等于!=，小于或等于 <=，大于或等于 >=，与 &，非!，或 |。

逻辑运算符用在向量上，得到的结果也是向量：

```
x <- 1:3
x == 2
```

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

```
x == 2 | x == 3
```

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

```
y <- c(4, 1, 2)
x > y
```

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

```
x > y & y > 1
```

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

逻辑运算的结果，也就是 TRUE 和 FALSE，相当于数字的 1 和 0，可以做数学运算：

```
TRUE * FALSE
```

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

```
TRUE + TRUE
```

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

这意味着，既然上面最后一个例子 x > y 的返回值是逻辑值 FALSE, TRUE, TRUE，那么我们可以对这个逻辑值用 sum() 函数求总和，得到的就是 “x 里总共有多少个元素大于 y”：

```
sum(x > y)
```

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

---

**思考 6.1.** 除了加减法，逻辑值 *TRUE* 和 *FALSE* 还能参与哪些运算？逻辑值能否参与 “比大小” 和 “与非或” 的逻辑运算？

**练习 6.1.** 设 m <- 6:1，n <- c(3, 5, 6)，判断 m 中的每个元素是否在 n 中出现。

**练习 6.2.** 从 1:100 中挑出既能被 2 整除，又能被 3 整除的数。

**练习 6.3.** 从 1 到 10000 中，挑出所有的素数。

上面这些练习，可以用循环来完成。不过，由于 R 很照顾追求高效的人士，对于像练习 6.1 这样的任务，R 提供了更方便的方法：

```
x %in% y
```

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

得到的结果依次是 x 向量里的三个元素是否在 y 中出现。

如果逻辑值出现在下标里边，那么 R 就会把 TRUE 对应的下标挑出来。结合逻辑判断和下标系统，就可以挑出 x 中哪些元素出现在 y 中，以及出现的位置：

```
x[x %in% y]
```

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

```
which(x %in% y)
```

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

第五章提到了马尔萨斯增长模型，我们用鼠标取点的方式，找出了哪一年世界人口超过 100 亿。现在有了逻辑判断，完成这个任务就更容易更准确了：

```
Y[N > 100][1]
```

---

**练习 6.4.** 在练习 1.2 中，我们曾经提出个问题：$PM_{2.5}$ 一天中的最大值出现在几点钟？请用逻辑判断运算来回答。

上面讲的是分支两部曲的第一步：判断。下面说说第二步：选择。

## 6.2 选择：如果，那么，否则

分支语句的最简单结构，用人类语言表达就是：

```
如果饿了，那么就吃。
```

将人类语言翻译成 R 语言就是：

```
if(饿了) {吃}
```

跟循环语句类似，如果只有“吃”这一个动作，那么花括号 {} 可以不写。当发生一系列动作时，就需要全部放进花括号里，并且为了看上去清楚，一般会添加一些换行符。下面是几个实例。

```
x <- 60
# 如果 x 小于 75，那么在显示区打印出来一句话：
if(x < 75) print("x is less than 75")
```

```
## [1] "x is less than 75"
```

```
if(x < 75) {
    print("x is less than 75")
    y <- x + 10
    print(y)
    }
```

```
## [1] "x is less than 75"
## [1] 70
```

复杂一点的判断，就是后面加上了“否则”：

```
如果饿了，那么就吃；否则，就干活儿。
```

翻译成 R 语言，就是：

```
if(饿了) {吃} else {干活儿}
```

具体用起来是这样的：

```
x <- 60
if(x < 75) {
    print("x is less than 75")
```

```
} else {
    print("x is larger than 75")
}  # 如果 (), 那么 {}, 否则 {}。
```

```
## [1] "x is less than 75"
```

无论是循环语句还是条件语句，都可以像俄罗斯套娃那样，一层套一层，大壳套小壳，比如说：

```
如果饿了，那么就吃和喝；否则，如果困了，那么就睡；否则，就干活儿。
```

翻译成 R 语言就是：

```
if(饿了) {吃} elseif(困了) {睡} else {干活儿}
```

继续上面的例子：

```
x <- 60
if(x < 75) {
    print("small")
} else if(x > 90){
    print("large")
} else {
    print("good")
}
```

```
## [1] "small"
```

这样写起来显得啰嗦，于是有了瘦身版 `ifelse()` 函数。

```
ifelse(x < 75, "small", "large")
```

效果等同于上一个例子。不同的是，这里 x 的向量长度可以大于 1。比如，试试 `x <- 60:100`。

`ifelse()` 函数也可以嵌套，像这样：

```
ifelse(x < 75, "small", ifelse(x > 90, "large", "good"))
```

```
## [1] "small"
```

如果嵌套太多，ifelse() 函数也会显得啰嗦，那么可以适当考虑进一步的瘦身版——switch() 函数，详见 F1 小助理。

---

**思考 6.2.** *if() {} else {}*、*ifelse()*、*switch()*，这三个函数之间，何时能相互替换，何时不能？

有了分支语句，就可以在很多地方派上用场了。比如，继续以马尔萨斯人口增长模型的数据作图，把超过 100 亿的数据点用红色区分出来：

```
N <- numeric(100)
N[1] <- 66.8
for(t in 1:99) N[t + 1] <- N[t] + r * N[t]
Y <- seq(2008, length.out = 100)
plot(x = Y + 2007, y = N, xlab = "Year", ylab = "Population",
     cex = ifelse(N >= 100, 2, 1), pch = 16, type = "b",
     col = ifelse(N >= 100, "red", "darkgreen"))
```

## 6.3　课外活动：复活节

我们都知道，“计算机”一词在英文称为 computer，而 compute 和 computation 分别是动词和名词的“计算”，他们均来自拉丁语的 Computus，而这个词的本意是“计算复活节的方法”。作为欧美国家最重要的节日之一，复活节，竟然难以计算是日历上的哪一天；复活节的计算，自古就是在考验人类的计算水平。今天，就让我们用 R 来试试计算每年的复活节是在日历的哪一天。

复活节用来纪念耶稣基督被钉死后复活。根据定义，复活节是每年春分月圆之后第一个星期日。为什么这样规定呢？据说，春分后北半球开始

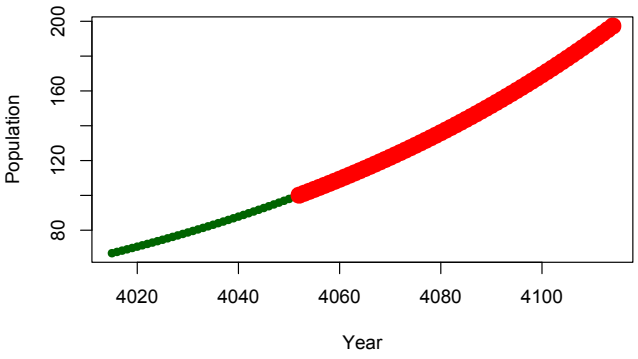



图 6.1: 用分支函数给数据点涂上不同颜色

日长夜短，所以春分意味着光明的到来；月圆时夜晚也洒满光辉，同样意味着光明的到来。耶稣在光明到来的这一天复活，那么耶稣便是光明的使者。而星期日，是因为上帝星期天休息，我们也不上班，在家过节。

由于基督教有三大主要派别，其中天主教和基督新教（华语圈常简称基督教）属于西方罗马教会，使用公历；东正教属于东方教会，使用儒略历。每种历法中，又存在不同的计算方法[2]。

第一种是数学王子高斯提出的高氏算法：

用 Y 表示年份，mod 表示整除的余数（例如 13 mod 5 = 3）。那么：

- a = Y mod 19
- b = Y mod 4
- c = Y mod 7
- d = (19a + M) mod 30
- e = (2b + 4c + 6d + N) mod 7

其中，M 和 N 的取值，在东正教会的儒略历 M = 15，N = 6，而西方教会所用的公历的取法参见下表：

[2]算法来自维基百科。

| 年份 | M | N |
|---|---|---|
| 1583-1699 | 22 | 2 |
| 1700-1799 | 23 | 3 |
| 1800-1899 | 23 | 4 |
| 1900-2099 | 24 | 5 |
| 2100-2199 | 24 | 6 |
| 2200-2299 | 25 | 0 |

- 若 d + e < 10，则复活节在 3 月 (d + e + 22) 日，反则在 4 月 (d + e - 9) 日，除了两个特殊情况：

1. 若公式算出的日期是 4 月 26 日，复活节在 4 月 19 日；
2. 若公式算出的日期是 4 月 25 日，同时 d = 28、e = 6 和 a > 10，复活节应在 4 月 18 日。

第二种是米氏（Meeus）算法。在公历中，

- a = Y mod 19
- b = Y / 100
- c = Y mod 100
- d = b / 4
- e = b mod 4
- f = (b + 8) / 25
- g = (b - f + 1) / 3
- h = (19a + b - d - g + 15) mod 30
- i = c / 4
- k = c mod 4
- l = (32 + 2 * e + 2 * i - h - k) mod 7
- m = (a + 11 * h + 22 * l) / 451
- month = (h + l - 7 * m + 114) / 31
- day = ((h + l - 7 * m + 114) mod 31) + 1

在儒略历中，

- a = Y mod 4
- b = Y mod 7
- c = Y mod 19
- d = (19c + 15) mod 30
- e = (2a + 4b - d + 34) mod 7
- 月 = (d+e+114) / 31
- 日 = ((d+e+114) mod 31) + 1

请把上面的算法用 R 代码写出来，并计算最近十年的复活节在几月几日。高氏算法和米氏算法的结果有差别吗？

# 第七章　办公

我们不要欺骗自己了：使用最广的统计软件其实是 `Excel`。

— Brian D. Ripley, September 2002

本章，我们停下学习 R 函数的匆匆脚步，稍事休整，来让 R 做一件狗拿耗子的事情：把 R 当办公软件来用。

一说办公软件，我们首先想到的是微软 Office 办公三剑客 Word，Excel，Powerpoint。这套软件售价不菲，安装还颇受限制，但并不影响对我们多年的吸引力。可惜，我们大部分人平时只用到 Office 不到 10% 的功能。即便如此，这 10% 的功能用起来也并不爽快，而要想使用另外 90% 的功能，并不比学 R 语言容易。

其实，我们的 R 就可以当作办公套件来用，计算数据、写文档、做幻灯片样样拿手。而且，作为办公套件，R 用起来更方便、安全、高效；用得久了就会感觉到，这是一种顶级享受。若是跟 Office 搭配使用，相映成趣，相得益彰。

## 7.1　从 Excel 到 R 代码

在 RStudio 的界面点击菜单 File – New File – R Script，就会新建一个扩展名为.r 的文件。我们已经对此很熟悉了，是从第一章以来一直接触的文件形式，相信你到现在已对 R 丰富的函数有了体会。将来，我们会学到 R 的扩展包，让我们可以对数据随心所欲地处理，在计算上完全可以取

代 Excel，功能上有过之而无不及，在编程上更是让 Excel 望尘莫及。Excel 当然也可以用宏命令和 VBA 实现很多强大功能，但门槛比较高。写代码是 R 初级用户上来就会的基本功，但 Excel 用户里会用 VBA 的比例寥寥无几。

---

**思考 7.1.** 审视一下你平时在 *Excel* 做的工作，哪些可以完全移植到 *R* 里？哪些只能部分移植？哪些工作 *R* 无法胜任？反过来呢？*R* 哪些工作是 *Excel* 不能胜任的？

从形式上来说，Excel 的输入数据、计算方法、计算结果这三个层面的内容是混合在一起的，这就带来很多不便。比如，有时候我们只想把计算方法分享给同事，但并不想分享原始数据，这在 Excel 很难办，而 R 很容易，只把代码分享就行了。数据量大的时候，Excel 文件的大小可能达到几百 MB，别说发送给别人，就是在本地打开都会非常吃力，有时候说不定会死机，而 R 的灵活性可以让代码与数据分离，也可以合在一起，Excel 的难题在 R 中迎刃而解。

> 有些熟悉 `R` 的人把 `R` 描述为微软 `Excel` 的增强版。
>
> — Ashlee Vance, January 2009

> 如果当他是朋友，就别让他用 `Excel` 做统计。
>
> — Jonathan D. Cryer, August 2001

## 7.2 从 Word 到 R 文档

我们不仅可以用 R 来取代 Excel，还可以取代 Word。

在 RStudio 的界面点击菜单 File – New File – R Markdown，在对话框选择 Document，将默认输出格式勾选 Word，填上（也可以不填）文档标题和作者，点 OK 按钮，一个新文件窗口就打开了。这是个模板，已经填了一些内容，目的是引领我们照葫芦画瓢创建自己的文档。

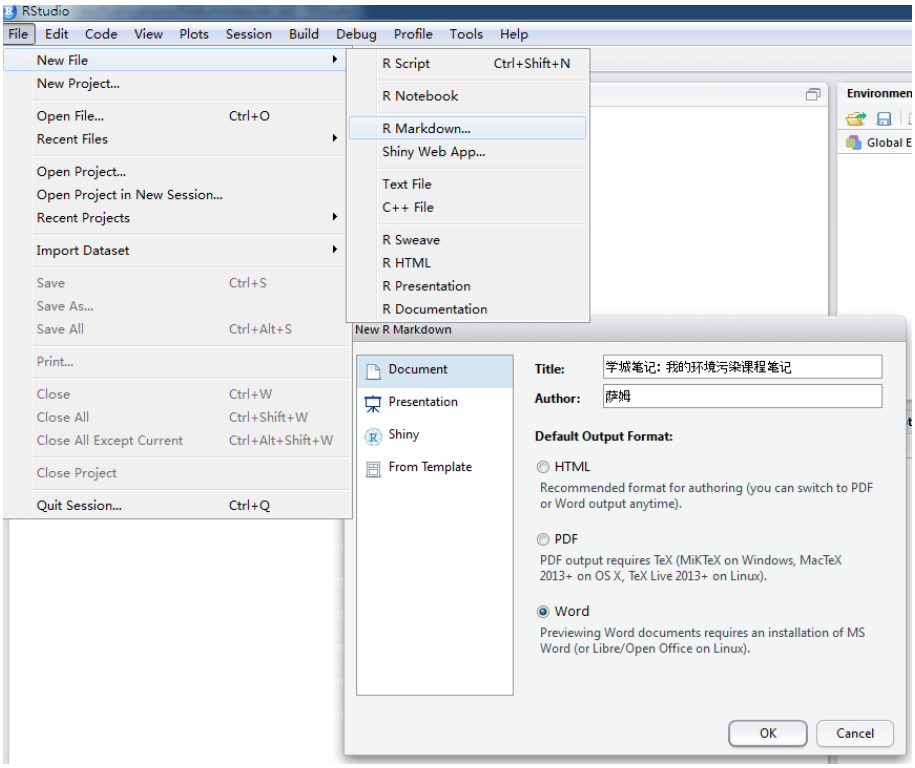



图 7.1: 创建 R Markdown 文档

下面，我们把它改成一份自己的文档。前边的 11 行不用改，从第 12 行的两个 # 号开始改：

```
## 大气颗粒物 pm2.5
```

pm2.5 最近几年在中国是个研究热点。R 自带的英国因肺部疾病而死亡的数据显示：

```
```{r}
summary(fdeaths)
```
```

```
## 作图

```{r}
plot(fdeaths)
```

## 其他污染

但是，跟大气污染相比，其他环境污染问题虽然不容易察觉，但同样不容忽视。

- 土壤污染
- 水污染
- 垃圾污染
- 其他
```

在这篇文档中，## 表示后面的文字是二级标题（# 越多，标题级别越低）；```{r} 和 ``` 两行表示两者之间夹着的是 R 代码；-表示后面是列表格式。这就是标记语言，是把格式用一些字符来表示，而不是通过点鼠标来选择格式。

其实，标记语言在前面的章节已经出现过了。我们在 expression() 函数里标明斜体或下标的时候，用的就是标记语言。

现在，把这个文件保存到硬盘里，我们权且起名叫 R-sam，再点击工具栏的 Knit 按钮。立刻，一份图文并茂的 Word 文档就华丽丽展开在面前了。这个 Word 文档，就是 R 生成的。

用 R 来写 Word 文档有什么好处呢？

应出版社要求，本书起初是用 Word 写的。然而，在多次修改过程中，需要在 R 中反复修改示例代码和导出图片，复制粘贴到 Word 文档中，极易出错，后续还要调整格式。一旦发现代码和图片需要修改，就得重来一次; 如此多次，无聊的过程让人心生绝望，最后索性用 R 来写书，整个世界顿时清静了。

# 学城笔记：我的环境污染课笔记

萨姆

March 2, 2017

## 大气颗粒物 $PM_{2.5}$

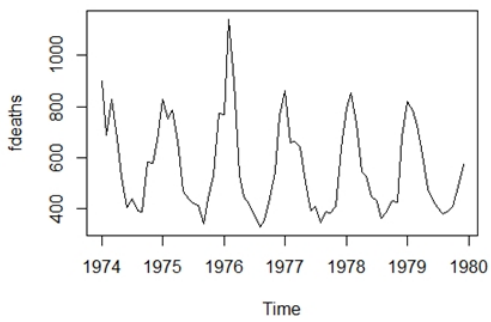
$PM_{2.5}$最近几年在中国是个研究热点。R自带的英国因肺部疾病而死亡的数据显示：

## 作图

```
summary(fdeaths)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   330.0   411.0   512.0   560.7   681.5  1141.0
```

```
plot(fdeaths)
```




## 其他污染

但是，跟大气污染相比，其他环境污染问题虽然不容易察觉，但同样不容忽视。

- 气态污染物

图 7.2: R 创建的 Word 文档

仔细研究一下这份 Word 文档，你就会更明白我们的意思。如果是在 Word 里写一模一样的这样一份文档，一般的步骤是这样的：

1. 写好文字部分；
2. 用鼠标逐个调格式：文字的颜色，大小，对齐方式，列表格式，代码的字体、颜色、浅灰色背景等等。
3. 用数据处理软件来计算数据，作图，再把计算结果和画好的图拷贝粘贴进文档。

对比一下 RStudio 里的这份文档，一步步对应的话，是这样操作的：

1. 写好文字，同时用一些符号把格式标记出来，格式会自动转换：
2. 在文字中嵌入 R 代码，计算结果和作图会自动展示。

这至少意味着两点：

1. 省却了鼠标调整格式的繁琐步骤；
2. 不用到别的软件界面拷贝图片了。

在这样的环境里写作，基本上可以避免被其他事情打扰分神，可以保持思维的连贯，只需关注内容，写起来一气呵成。

然而这仅仅是表面的好处。仔细思考，这背后有更丰富的意义。

我们发现，pm2.5 的正确写法应该是 $PM_{2.5}$，字母应当大写，数字应当用下标格式。除此之外，我们想把文档里所有的“污染”二字加粗强调。在我们的 R-sam 文件里，由于是标记语言，我们只需按 ctrl+f 快捷键，查找“pm2.5”，全部替换成 `PM~2.5~`；查找“污染”，全部替换成 `** 污染 **` 就行了。然后，保存文件，再按 Knit 按钮。Bingo！奇迹就这样发生了。看看生成的 Word 文档里，格式是不是按我们的心意已经调好了？

波浪线表示下标，双星号表示加粗。这就是标记语言的好处，格式容易批量修改。如果文档里有千百处 pm2.5 要改，在 R 的文档里改起来太简单了，而在 Word 里改起来就是个灾难。就算使用 Word 的高级查找替换功能，也比 R 麻烦很多。

这里用的这种标记语言叫做 Markdown。有了它，我们书写时只管放心写成 pm2.5，将来批量改格式就行了。Markdown 的标记方法十分简单，花个几分钟学习，就能满足绝大多数日常需求；如果使用它的扩展语法，那么几乎所有需求都可以得到满足（见附录）。

用 R 写文档的好处，当然不只是方便改格式。由于 R 代码是嵌入在文档中的，如果发现计算方法需要补充、修改，那么在这个文档里改就是了，输出的 Word 文档里数据和图片自动更新，不必考来考去，免去了麻烦的操作，大大减少出错的几率。

而且，跟别人交流时，只需这一份文档，就包含了数据处理过程和结果，别人很容易重复验证，只要有同样的输入数据，就会得到跟你一模一样的结果。这就是“可重复性研究报告”的概念。当然，如果你想对计算方法保密，或者对一部分文字保密，只需把文档中的代码设定为隐藏，或者用标记符号将文字隐藏，那么输出的文档里就不会出现代码。总之，这是一份订制性很高、个性十足的文档。关于可重复性研究报告，我们将在第十四章和第十五章进一步介绍。

---

**练习 7.1.** 将你的学习笔记做成一个 R-Markdown 文档，并导出为 Word 文档。

此外，这份文档还可以直接生成幻灯片，其中的数据和图片跟 Word 里保持高度一致。这意味着，我们只维护这一份文档就够了。

那么，如何把这份 R-word 文档直接变成幻灯片呢？

## 7.3　从 Powerpoint 到 R 幻灯片

在 RStudio 的界面点击菜单 File - New File - R Markdown，在对话框选择 Presentation，将默认输出格式勾选 HTML（ioslides），点 OK 按钮。跟上面一样，新打开的是个用作示例的模板文件。我们把它保存文件到硬盘里某个文件夹，权且起名叫 R-slides。点击 Knit 按钮，一个幻灯片文件

就做好了，保存在同一个文件夹下，名叫 R-slides.html，是个网页文件。双击打开看看，是不是跟 PPT 差不多？

如果把我们这个 R-slides 文档跟上一节 R-sam 文档比较一下，就会发现，文档的结构完全相同，只有文件开头 `output` 后面的内容不同。现在，让我们把 R-slides 文件开头的 `output: ioslides_presentation` 拷贝粘贴到 R-word 文档里，将原来的 `output: word_document` 替换掉，再点 Knit 按钮，见证一下奇迹的发生。

这份同样内容的 Word 文档就变成了幻灯片的模样！

这是件激动人心的事情。这意味着，我们以后只需专注这一份 R 文档就行了。需要 Word 报告就可以出报告，需要幻灯片就出幻灯片，并且内容都高度一致。

办公世界的纷乱就此平息。

当然，尺有所短，寸有所长，每个软件都有自己的优势。我们暂时无法用 R 像 Excel 那样展示绚丽多姿的表格，也无法做出 PPT 复杂的动画效果。所以，我们没必要厚此薄彼，只需认清他们的长处，为我所用便是了。

事实上，等我们学了第九章之后，这些看似不足的问题也会迎刃而解。我们甚至在将来学会字符和文件操作后，利用循环函数和分支函数，自动批量生成多个 Word 文档和 PPT 幻灯片。

---

**思考 7.2.** 绚丽多姿的表格和复杂的动画效果，真的不能用 *R* 实现吗？如果不能，那么这些功能我们真的需要吗？如果能，那么该如何实现？

**练习 7.2.** 将你的学习笔记文档转换成幻灯片。

## 7.4　课外活动：丰富多彩的幻灯片

做报告（presentation）时使用幻灯片（slides）早已成为标配[1] 。幻灯片不等于 Powerpoint，当然也不等于 Keynote。很多软件都可以用来做幻灯片。例如，跟 Powerpoint 最接近的有免费的 WPS 和 Openoffice，模样接近但原理迥异的有 Beamer 和 deck.js ，形式和风格大相径庭的有 Prezi 和 ipresst，也有人用思维导图来演示，甚至有人仅用 ‘‘记事本’’ 就可以把自己的想法说清楚。

对于幻灯片的制作，R 语言有多种解决方案。上网搜搜看，都有哪些有趣的方法，有哪些具有独特的优势。

不要被困在 Powerpoint 的牢笼里。打开 R 的窗户，尽是自由天地。

[1] 当然也有例外，拜罗伊特大学微气象学系答辩要求使用的是打印海报，大气化学系则是要求黑板 + 粉笔。

# 第八章　习题

> 就算现在不用 R，将来迟早你会用。
>
> — David Kane, November 2004

## 8.1　向量、矩阵、数据框和列表

纸上谈兵没有用，实战是练兵最有效的方法。本章是习题集。做习题之前，我们对前面的某些内容作个小结和补充，顺便热热身。

到目前为止，我们遇到过向量、矩阵、数据框和列表，这些都叫做对象的类型。他们的区别和联系在哪里？

向量，vector，是最简单的对象。向量由一个或多个同类变量组成。可以使用 `c()` 函数来新建一个向量。

```
x <- c(1, 1, 2, 2, 3)  # 生成一个向量。
is.character(x)  # x 是字符型吗？不是。
```

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

```
is.numeric(x)  # x 是数值型吗？是的。
```

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

```
mode(x)  # 确实是数值型。
```

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

```
y <- c(3, 4, 4, 5, 5)
z <- c(x, y)  # 多个向量并在一起。
z
```

```
##  [1] 1 1 2 2 3 3 4 4 5 5
```

```
z[4]  # 向量的下标。
```

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

矩阵，matrix，与向量差不多，不同的是分成了行和列，也就是行列式。可以使用矩阵函数 `matrix()` 来新建一个矩阵变量：

```
# 生成一个矩阵，指定行数和列数：
m <- matrix(c(2, 3, 1, 5), nrow = 2, ncol = 2)
m
```

```
##      [,1] [,2]
## [1,]    2    1
## [2,]    3    5
```

```
# 生成一个矩阵，指定行数：
m <- matrix(c(2, 3, 1, 5), nrow = 2)
m
```

```
##      [,1] [,2]
## [1,]    2    1
## [2,]    3    5
```

```
# 生成一个矩阵，指定行数，并按行排列：
m <- matrix(seq(1, 20, 1), nrow = 5, byrow = TRUE)
m
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
## [4,]   13   14   15   16
## [5,]   17   18   19   20
```

```
# 生成一个矩阵，指定行数，并按列排列：
m <- matrix(seq(1, 20, 1), nrow = 5, byrow = FALSE)
m[2, 2]  # 矩阵的下标。
```

```
## [1] 7
```

我们在第 3.1 节见过的 `image()` 函数，就是专门针对矩阵作图的。矩阵的任何一行或一列，都是一个向量。

数据框，dataframe，与矩阵差不多，区别在于，不用的列可以是不同类型。相当于 Excel 的表格。我们前面处理过的 mydata2 就是数据框。可以使用 `data.frame()` 函数来新建一个数据框变量。

```
a <- c(1, 2, 3, 4)
b <- seq(5, 8, by = 1)
d <- data.frame(a, b)  # 生成一个数据框。
d
```

```
##   a b
## 1 1 5
## 2 2 6
## 3 3 7
## 4 4 8
```

```
is.data.frame(d)  # 是数据框吗？
```

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

```
str(d)  # 数据框的结构。
```

```
## 'data.frame':    4 obs. of  2 variables:
##  $ a: num  1 2 3 4
##  $ b: num  5 6 7 8
```

```
class(d)
```

```
## [1] "data.frame"
```

```
nrow(d) # 数据框的行数。
```

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

```
ncol(d) # 数据框的列数。
```

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

```
e <- c(9, 10)
f <- rbind(d, e)  # 给数据框增加一行。
f
```

```
##   a  b
## 1 1  5
## 2 2  6
## 3 3  7
## 4 4  8
## 5 9 10
```

```
g <- c("one", "two", "three", "four", "five")
class(g)
```

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

```
h <- cbind(f, g)  # 给数据框增加一列。
h
```

```
##   a  b     g
```

```
## 1 1  5   one
## 2 2  6   two
## 3 3  7 three
## 4 4  8  four
## 5 9 10  five
```

```
class(h)
```

```
## [1] "data.frame"
```

```
ncol(h)
```

```
## [1] 3
```

```
colnames(h) <- c("one", "two", "three")  # 更改列名称。
h
```

```
##   one two three
## 1   1   5   one
## 2   2   6   two
## 3   3   7 three
## 4   4   8  four
## 5   9  10  five
```

数据框的任何一列都是一个向量，而任何一行仍然是个数据框，这就是思考题 2.3 的答案：`mean()` 函数的自变量可以是向量或矩阵，但不能是数据框。若想把数据框的行转化成向量，需要把该行用 `unlist()` 函数打散。这个操作在第 3.1 节出现过。

```
is.numeric(d[, 1])
```

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

```
is.data.frame(d[1, ])
```

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

```
is.numeric(d[1, ])
```

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

```
is.numeric(unlist(d[1, ]))
```

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

列表，list，跟数据框差不多，区别在于，列表里的不同项目可以有不同的长度。我们在第五章见过的 `lapply()` 函数，返回的结果就是列表。如果要新建一个列表，可以用 `list()` 函数：

```
mylist <- list(x = 1:4, y = letters[3:10])
mylist
```

```
## $x
## [1] 1 2 3 4
##
## $y
## [1] "c" "d" "e" "f" "g" "h" "i" "j"
```

列表里元素的调用方式比较特别，需要用两层方括号，或用美元符号。体会一下：

```
mylist[[1]]
mylist[[1]][1]
mylist[[2]]
mylist[[2]][5]
mylist$x
mylist$y[3]
```

---

**思考 8.1.** 向量、矩阵、数据框、列表四者之间如何转换？

下面开始习题。练习的内容主要是复习已经学过的，并提升一点点。如果遇到问题，请善用小助理。菜鸟学 R，不必面面俱到，多练练习题，上手

之后就会喜欢上 R，就有兴趣深入了解下去。

## 8.2　A 卷：照猫画虎题

请先运行下面的代码，生成一个示例数据文件：

```
write.csv(WorldPhones, file = 'c:/r4r/wp.csv')
```

打开这个文件浏览一下。这是 20 世纪五六十年代世界各大洲的电话数量。下面的练习，均以此文件为出发点。

---

**练习 8.1.** 读入数据

- 将 wp.csv 读入 R 中，保存到一个叫 wp 的对象中，并查看文件内容、总结报告和作图。
- 将 wp 的年份列的列名称改为 ‘‘year’’。
- 将 wp 的行名称改为对应年份。

**练习 8.2.** 数据类型。

- 给 wp 新增年代列，取值为对应行所属的年代，即 ‘‘1950s’’ 和 ‘‘1960s’’ 字符，列名称为 ‘‘decade’’。
- 查看 wp 各列的数据类型。

**练习 8.3.** 矩阵。

- 生成一个 5 行 6 列的矩阵，取值为整数数列 1:30。

**练习 8.4.** 矩阵与数据框。

- wp 是个对话框对象。请将 wp 转换成矩阵对象 wp_mt，并比较 wp 与 wp_mt 的区别。
- 从这两个对象中选取 1956 年欧洲的电话数量。
- 从这两个对象中选择亚洲和欧洲两列。

- 从这两个对象中选择第 2, 4, 5 行。选择除了 2, 4, 5 行之外的其他行。

**练习 8.5.** 计算。

- 计算 wp 数据中全世界每年的电话总数。
- 计算 wp 中任意相邻年份全世界以及各洲的电话增长数量以及增长百分比。

**练习 8.6.** 作图。

请在同一张图上以合理的布局、颜色和线型，做出亚洲 1951 到 1961 年之间：

- 电话数量逐年变化的散点图，以及
- 电话数量增长率的散点图。

**练习 8.7.** 循环：

请以合理的布局，用循环函数做出所有大洲在 1951 年到 1961 年之间各自的电话数量增长曲线。

**练习 8.8.** 拟合。

将各洲电话数量对年份进行线性拟合，并在散点图上添加拟合直线和拟合方程。

**练习 8.9.** 办公套件。

请把上述题目的解答过程、代码、注释以及结果均以 R Markdown 的形式写在同一个文档中。

## 8.3 B 卷：自由发挥题

还记得小学时做过的数学题吗？记得那时候做过的什么 ‘‘相遇问题’’、‘‘追及问题’’、‘‘鸡兔同笼问题’’，解决方法是四则运算。后来学会了方程，就把从前的解法全忘了。有一次收拾书柜，翻出一张小学试卷，有一道题目，

我只用了两步的四则运算就做出来了，换现在我只会列二元一次方程组来求解。我看着当时的算式发呆，因为怎么也看不懂每一步是什么意思。

下面这两道小学数学题，请用 R 语言解答。

---

**练习 8.10.** 在下面的竖式里，A, B, C, D, E, F 均为 0 到 9 的整数，那么他们分别是多少？

$$
\begin{array}{r}
\mathsf{ABC} \\
\times\ \mathsf{DC} \\
\hline
\mathsf{DEAC} \\
\mathsf{7ED} \\
\hline
\mathsf{FDBC}
\end{array}
$$

**练习 8.11.** 从前，山上有一棵魔法苹果树。树上长了 125 个苹果，有一天掉下来 1 个苹果。从第二天起，每天掉的苹果数量比前一天多 1 个，但如果某天树上的苹果数量少于这一天本应该掉的数量时，那么从这一天起又重新从掉 1 个苹果开始，按原来的规律进行新的一轮。那么，第几天树上的苹果会掉光？

# 第九章　函数

罗杰：对我来说，99% 的事情都能由 R 完成。但是让人郁闷的是，当我需要订个披萨饼时，我还是得抓起电话。

道格：美国很多披萨饼连锁店提供网上预订服务啊。R 有互联网模块，所以，披萨预订函数的出现只是个早晚问题。

— June 2004

终于轮到介绍 R 的精髓了，那就是包包。“包治百病”，R 里面有成千上万个漂亮包包，可以帮我们解决所有问题。——女同胞请先收敛一下两眼放出的光芒。R 的包是扩展包[1]，package。

欲知扩展包，必先知函数，function。

## 9.1　内置函数：全自动厨师机

到现在为止，我们已经遇到过很多函数了。一个函数就像一台全自动厨师机。我们输入一些原材料，然后按个按钮让它运转，叮！它就输出来一盘好菜。

比如，假定世界上有一台专门做番茄炒蛋的自动厨师机，那么我们用 R 语言来写使用说明书，就是这样的：

[1]严格来讲应该称为“程序包”，往下再细分为“基础包”和“扩展包”。但由于基础包是默认安装加载的，常常不必介绍，所以我们经常讲的 R 包就是扩展包。

```
x <- 番茄炒蛋机（番茄 = 3 个，鸡蛋 = 2 个，盐 = 2 勺，油 = 2 勺）
```

运行一下，一份美味的番茄炒蛋就放在名叫 x 的这个盘子里，送到了我们的眼前。圆括号里是我们指定的配方；如果我们没有指定，那么 R 就按他自己认为的默认配方来做。

对比一下前面我们学过的函数就会发现，他们确实都是这样的厨师机：

```
x1 <- mean(x = 1:3)
x2 <- read.csv(file = "c:/r4r/co2.csv")
```

R 其实就是由不同功能的自动厨师机拼成的一个大厨房，有的厨师机独立工作，但大部分厨师机是相互协作的，一台厨师机输出的半成品或成品，会接着输入到另一台厨师机继续加工，直到把我们点的菜做好。比如我们运行了一个叫做“可乐鸡翅”的函数，输入的是水、香精、鸡翅等，那么 R 就会先召唤“可乐饮料机”函数做出可乐，再和鸡翅等原材料一起做可乐鸡翅。这就是函数之间的调用。

我们以前用过的，都是 R 基础安装包里一些预先设置好的函数。比如：

```
y <- sd(x = 1:5)  # sd 是函数名，x 是自变量。
y
```

```
## [1] 1.581139
```

sd() 这台厨师机，输入的原料是个数值型向量，输出的菜是标准差。当我们点了 sd() 这道菜后，R 在厨房里到底忙活个啥呢？输入函数名并回车，或者光标移到函数名上按 F2 快捷键，就能看到烹饪方法了：

```
sd
```

```
## function (x, na.rm = FALSE)
## sqrt(var(if (is.vector(x) || is.factor(x)) x else
as.double(x),
##     na.rm = na.rm))
## <bytecode: 0x000000000eafe450>
```

```
## <environment: namespace:stats>
```

从这个烹饪方法的第二行可以看到，计算方法是先对输入的原材料 x 进行初步加工（检查和转换格式），再用 var() 函数求方差，最后开平方。

甚至像 x <- 1:5 这一句，其实背后运行的也是个函数，等同于：

```
assign("x", 1:5)
```

---

**思考 9.1.** *R 语言里，有没有哪些功能不是由函数实现的？*

## 9.2　自定义函数：自制厨师机

由内置函数组成的这个厨房，虽然功能强，但是毕竟有限，只能做那么千百道菜，不一定称心如意。你想吃的菜，厨房做不了怎么办？

这一点 R 贴心地考虑到了。你可以自己制作一台厨师机，做你独特的个性菜。

比如说，当年有一回全班考得都很惨，老师心软了，说为了提高及格率，把卷面分数开方乘十作为新分数吧。为了以后经常用来提高及格率，我们可以专门定义这样一个函数。继续以厨师机打比方的话，这道菜的菜名是“及格神器”，输入材料是“卷面分数”，输出的菜是“新分数”，那么我们可以这样制作这台厨师机：

```
及格神器 <- function(卷面分数) {
  新分数 <- 卷面分数开方乘十
  return(新分数)
}
```

那么，调用这台厨师机做一份及格神器菜的方法就是：

```
及格神器 (卷面分数 = 40)
```

翻译成 R 语言，就是这样的：

```
newscore <- function(x) {
  y <- sqrt(x) * 10
  return(y)
}
```

以后当考了 40 分的时候，可以这样调用你的新函数：

```
newscore(x = 40)
```

```
## [1] 63.24555
```

想一次可以用很久喔！有人说，学电脑的人，动脑筋就是为了偷懒。

—《大家来学 VIM》

上面这个例子中，自变量 x 只是用来在函数内部传递信息用的，不会影响函数之外的对象。看看这个例子：

```
x <- 36
y <- 81
newscore(x = y)  # 函数内部的 x 把 81 的值接过来，而不是 36。
```

```
## [1] 90
```

```
x # 函数外部的 x 仍然是 36。
```

```
## [1] 36
```

很多厨师机需要输入多种原材料，也就是说，函数可以有多个自变量，例如：

```
news <- function(x, n) {
  sqrt(x) * 10 + n
}
news(x = 36, n = 10)
```

```
## [1] 70
```

为了省事儿，可以按照函数要求的顺序来列出自变量的取值，省略自变量的名称。下面这条命令跟上一条完全等同：

```
news(36, 10)
```

但是，如果打乱了顺序，就必须指定谁是谁。

```
news(n = 10, x = 36)
```

每次调用自定义函数 newscore() 的时候，必须提供所有自变量的取值，否则就会报错：

```
newscore()
```

```
## Error in newscore(): 缺少参数"x", 也没有缺省值
```

为避免这个问题，我们可以给 x 指定个默认值，这样以后调用时，如果没有说 x 是多少的话，就按默认值计算：

```
newscore <- function(x = 36) # 将 x 默认值设为 36。
{
  sqrt(x) * 10
}
newscore()
```

```
## [1] 60
```

给函数定义时，我们用了花括号 {}，小贴士 5.1 说过，这意味着可以把一组操作都放进去，哪怕这一组操作有千万行，以后只用一行就可以调用一遍了！比如第 5.3 节提到过的人口指数增长，可以定义一个函数，名叫 exponentialGrowth()：

```
exponentialGrowth <- function(N0, r = 0.01, tmax = 10)
  # 三个自变量：初始值，增长率（默认为 0.01），时间（默认为 10）。
{
  N <- N0
  for(t in 1 : (tmax - 1)) {
    N[t + 1] <- N[t] + r * N[t]
```

```
  }
  return(N)
}
```

函数 return() 是返回值，意思是告诉 R，烹饪过程里其他的东西比如 t 都可以扔进垃圾箱了，只把 N 作为成品端出来当菜就行了。

以后再用到这个模型，比如计算初始值为 80，增长率为每年 2%，时间为 100 年，则用一条语句就行了：

```
plot(exponentialGrowth(80, 0.02, 100))
```

---

**练习 9.1.** 自定义一个名为 kaifang 的函数，用来开平方。

**练习 9.2.** 自定义一个名为 cv 的函数，用来计算变异系数，即标准差除以平均值的商。

**练习 9.3.** 将第 6.3 节中计算复活节的方法写成一个自定义函数，允许用户指定年份、计算方法（高斯法或 Meeus 法）、日历（公历或儒略历），返回值为给定条件的复活节日期，形如：

```
computus(year, method, calendar)
```

如果你有多个自定义函数，经常需要在不同场合调用的话，为了方便，可以把这些函数写在一个文件里，例如叫做 c:/r4r/myf.r，每次在 R 中使用 source() 函数运行这个文件就行了：

```
source('c:/r4r/myf.r')
```

source() 函数的作用就是运行指定文件里的所有代码。

图 1.4 就是用自定义函数作出的。只不过，那个厨师机不是我自己造的，而是我把别人制作的厨师机稍作了些改动，为己所用。

那么，别人是怎么把他自己的厨师机分享给我的呢？

## 9.3　扩展包：美食王国

上面我们自定义了几个函数，订制了几台私人专属厨师机。世界上很多角落都有想把 36 分变成 60 分的苦命同学，为了让他们也能方便地调用上文我们自定义的 `newscore()` 等函数，我们可以把它们打包上传到服务器上，这样别人下载了就可以直接用。我们打的这个包，就是扩展包。R 包常用的服务器是 CRAN 和 GitHub。当然你也可以用自己的服务器，只要别人能找得到下载地址就可以了。

一个扩展包，就是一套别人事先搭建的厨房。这样的厨房规模有大有小。小的厨房，可能只有一台厨师机（一个自定义函数）；大的厨房，能有成百上千台厨师机。有时候，厨房（扩展包）还自带了一些食材（数据）让你试着品尝，比如我们前边使用的 co2 数据就是。

扩展包是 R 的生命力所在。找到一个合适的扩展包，能起到事半功倍的效果。甚至可以说，会用扩展包，比本书前半部分介绍的所有内容都重要！很多人用 R 就是奔着扩展包来的。

我们来举个例子。

北京的天安门广场，常年根据预测的日出日落时刻来确定升降国旗仪式的时刻。日出日落时刻的计算涉及复杂的天文学知识、三角函数知识、立体几何知识、天文学知识等，最要命的是还得有足够的耐心。我一直望而却步，直到有一天，我需要把某个气象站半年的气温数据（每半小时一条）分为白天和黑夜两组，那么就要判断当地每天的日出和日落时刻，不得不设法揭开这个神秘面纱了。花了大概一天的工夫，硬着头皮算出了个数，却跟实际对不上号。

后来，我惊喜地发现了 maptools (Bivand and Lewin-Koh, 2017) 这个扩展包。安装这个包之后，用其中的 `sunriset()` 函数一条指令轻松搞定。我们用它计算一下 2017 年国庆节天安门广场的日出日落时间，也就是升降国旗时刻。

```
install.packages("maptools") # 第一次使用某个扩展包时要先安装。
```

```
require(maptools)  # 调用包，让 R 把其中的函数读进 R 的脑子里。
position <- c(116.39, 39.91)  # 天安门广场的经纬度。
mydate <- "2017-10-01"  # 要计算的日期。
# 日出时刻：
sunriset(matrix(position, nrow = 1),
         as.POSIXct(mydate, tz = "Asia/Shanghai"),
         direction = c("sunrise"), POSIXct.out = TRUE)$time
```

```
## [1] "2017-10-01 06:10:25 CST"
```

```
# 日落时刻：
sunriset(matrix(position, nrow = 1),
         as.POSIXct(mydate, tz = "Asia/Shanghai"),
         direction = c("sunset"), POSIXct.out = TRUE)$time
```

```
## [1] "2017-10-01 17:57:14 CST"
```

跟官方公布的日出日落时刻比一比，看看是不是相同。

一个完整的扩展包包括了帮助信息，所以我们的 F1 小助理仍然管用。自己试试把光标移到 `sunriset` 按 F1。

若要了解整个扩展包中所有的函数，可以用搜索引擎搜索 'cran maptools'，也可以在本地计算机 R 的安装路径下面 library 文件夹中找到。

虽然计算出了结果，但结果的展示方式我并不喜欢。这道菜我想再撒点胡椒粉，改成自己喜欢的口味。于是我更进一步，在扩展包已有函数的基础上自定义一个函数，计算任意一段时期的升降旗时刻：

```
# 函数名为 flag，默认是计算 2017 年情人节开始一周内升降国旗时刻。
flag <- function(date.start = "2017-02-14", date.length = 7)
{
  mydate <- seq(as.POSIXct(date.start, tz="Asia/Shanghai"),
                by = 3600 * 24, length.out = date.length)
  data.frame(
```

```
    sunrise = sunriset(
      matrix(c(116.39, 39.91), nrow = 1),
      as.POSIXct(mydate, tz="Asia/Shanghai"),
      direction=c("sunrise"), POSIXct.out = TRUE)$time,
    sunset = sunriset(
      matrix(c(116.39, 39.91), nrow = 1),
      as.POSIXct(mydate, tz="Asia/Shanghai"),
      direction=c("sunset"), POSIXct.out = TRUE)$time)
}

flag("2017-10-01") # 好了，以后调用这个函数就能很方便计算。
```

```
##                sunrise              sunset
## 1 2017-10-01 06:10:25 2017-10-01 17:57:14
## 2 2017-10-02 06:11:24 2017-10-02 17:55:37
## 3 2017-10-03 06:12:23 2017-10-03 17:54:01
## 4 2017-10-04 06:13:22 2017-10-04 17:52:24
## 5 2017-10-05 06:14:21 2017-10-05 17:50:48
## 6 2017-10-06 06:15:21 2017-10-06 17:49:13
## 7 2017-10-07 06:16:21 2017-10-07 17:47:38
```

---

**练习 9.4.** 利用 GPS 工具或在线地图查出你所在地点的经纬度，然后利用 maptools 扩展包，计算你所在地点 2017 – 2116 年 100 年的日出日落时刻。

再举个例子。CRAN 服务器上有几个专门为初学者写的扩展包，'beginr' (Zhao, 2017a) 就是其中一个。这个包可以帮助初学者解决一些常见问题。而且，包里的函数结构都比较简单，初学者可以用 F2 调出函数的源代码修改练手。我们在这里做一详细介绍。

beginr 的安装和加载的方法与其他包是类似的:

```
install.packages("beginr")
```

下面，我们分类简要介绍一下其中的函数.

### 备忘函数

初学者用 `plot()` 作图时, 常常会忘记不同形状数据点 (`pch`) 对应的编号, 实线虚线 (`lty`) 的编号, 散点图类型 (`type`) 的代码, 以及最难选择的颜色代码。虽然可以上网搜，或者查看本书的小贴士 3.2 和 3.3，但仍然不够快捷。现在, beginr 里提供了 `plotpch()`, `plotlty()`, `plottype()`, `plotcolors()` 等函数, 想不起来的时候运行一下就行了。其实, 这几个函数就是上面两个小贴士的来历。

```
beginr::plotpch()
beginr::plotlty()
beginr::plottype()
beginr::plotcolors()
```

“包名称:: 函数” 这种格式可以在不加载扩展包 (`library()` 或 `require()`)的情况下直接运行包里的函数。上面这几个函数，有的不含自变量，有的采用的是自变量的默认值。

### 快速作图函数

线性拟合是数据处理的常见操作, 每次又是作图又是添加拟合方程, 步骤繁琐。现在, 用 beginr 里的一个 `plotlm()` 函数就可以完成，畅快淋漓（图 9.1）。

```
x <- 1:10
y <- 1:10 + rnorm(10)
beginr::plotlm(x, y, refline = TRUE)
```

```
## [[1]]
##               Estimate Std. Error   t value
## (Intercept) 0.5254674  0.6672766 0.7874806
## x           0.9180288  0.1075414 8.5365180
```

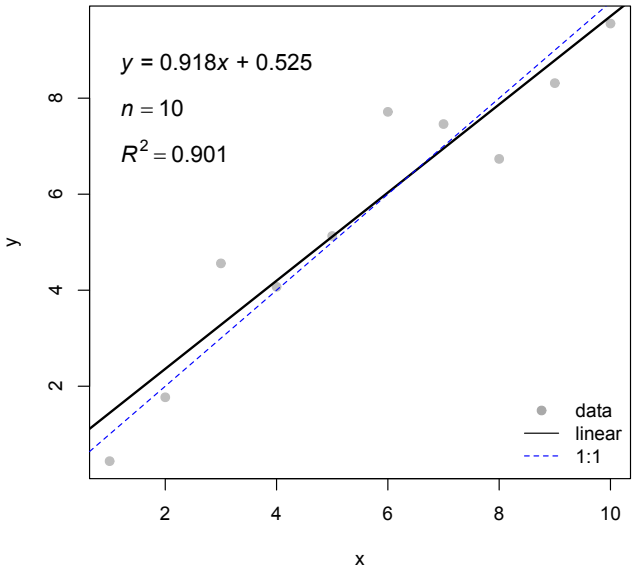



图 9.1: `beginr::plotlm()` 函数示例

```
##                   Pr(>|t|)
## (Intercept) 4.536973e-01
## x           2.728816e-05
##
## [[2]]
## [1] 0.9010784
```

我们经常需要了解一组数据的分布, 看看是不是属于正态分布. beginr 里的 `plothist()` 函数不仅一步就做出直方图, 而且标出中值、中位数、四分位数、标准偏差、样本数（图 9.2）.

```
x <- rnorm(10000)
beginr::plothist(x)
```

系统自带的成对儿散点图函数 `pairs()` 人见人爱, 只是功能再强大一

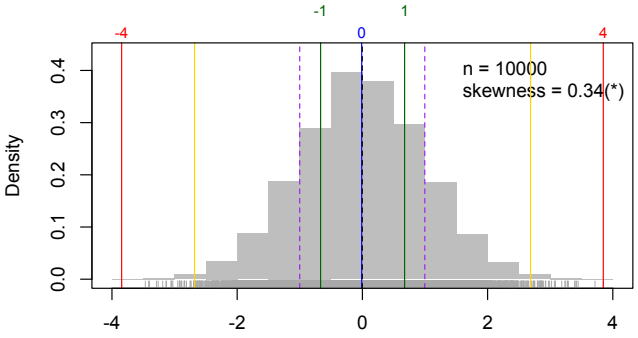



图 9.2: `beginr::plothist()` 函数示例

点就好了。beginr 里的 `plotpairs()` 和 `plotpairs2()` 就是对其进行的扩展，也是图 1.4 的来历。

```
df <- data.frame(a = 1:10,
                 b = 1:10 + rnorm(10),
                 c = 1:10 + rnorm(10))
beginr::plotpairs(df)
beginr::plotpairs2(df)
```

我们经常需要将一组自变量 $(x)$ 和多组因变量 $(y_1, y_2, ..., y_n)$ 在同一个坐标系作散点图图, 或者一组因变量 $(y)$ 对多组自变量 $(x_1, x_2, ..., x_n)$ 作图, 并且画上各自的误差线。beginr 里一条 `dfplot()` 或 `dfplot2()` 函数就能完成。

```
par(mfrow = c(1,2), mar = c(0.1, 0.1, 0.1, 0.1))
x <- seq(0, 2 * pi, length.out = 100)
y <- data.frame(sin(x), cos(x))
# 假定 yerror 是 y 的误差范围
yerror <- data.frame(abs(rnorm(100, sd = 0.3)),
                     abs(rnorm(100, sd = 0.1)))
```

```
beginr::dfplot(x, y, yerror = yerror)
beginr::dfplot2(y, x, xerror = yerror, xlab = '', ylab = '')
```

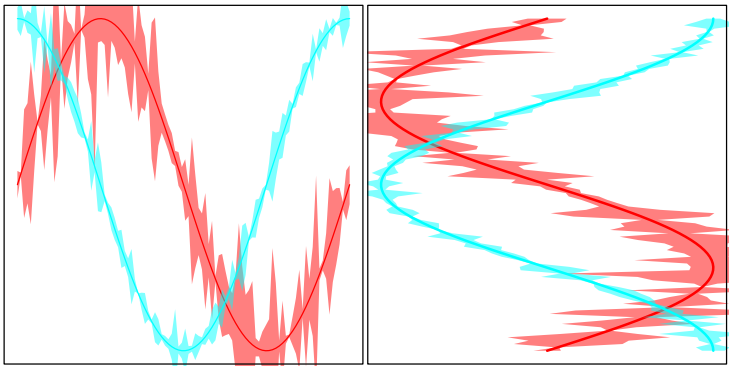


图 9.3: `beginr::dfplot()` 和 `beginr::dfplot2()` 函数示例

图 9.3 不同曲线的颜色默认状态下是函数自动选择的，或许不够美观，但对于初步查看数据来说已经足够了。当然也可以用指定的颜色作图。请查看该函数的帮助信息。

如果只是想简单地给散点图添加误差线，一般会用 R 基础包自带的 `arrow()` 函数逐步添加，而 beginr 里的 `errorbar()` 函数把多个步骤打了个包，一次性完成（图 9.4）。

```
x <- seq(0, 2 * pi, length.out = 100)
y <- sin(x)
plot(x, y, type = "l")
beginr::errorbar(x, y, yupper = 0.1, ylower = 0.1)
```

### 数据框操作

beginr 提供了一些对数据框计算的函数，例如计算多列数据里任意两列之间的相关性和线性拟合结果，可以用 `lmdf()` 函数，可以方便地得到拟合直线的斜率、截距及其标准误差以及 $R^2$.

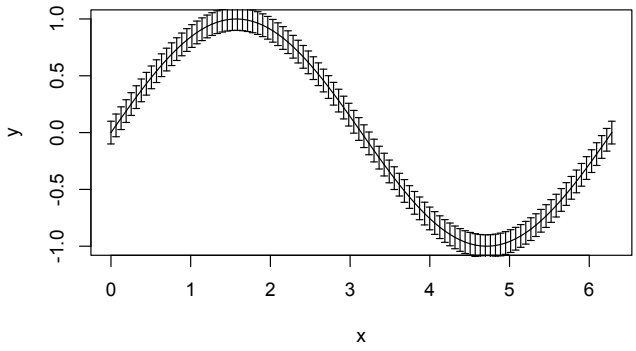



图 9.4: beginr::errorbar() 函数示例

```
df <- data.frame(a = 1:10,
                 b = 1:10 + rnorm(10),
                 c = 1:10 + rnorm(10))
beginr::lmdf(df)
```

```
##   x y r.squared adj.r.squared  intercept     slope
## 1 a b 0.9637183     0.9591831 -0.2884811 1.1171571
## 2 a c 0.8897951     0.8760195  1.6441984 0.7156784
## 3 b a 0.9637183     0.9591831  0.4484083 0.8626525
## 4 b c 0.8358135     0.8152902  2.0111482 0.6095207
## 5 c a 0.8897951     0.8760195 -1.4380868 1.2432890
## 6 c b 0.8358135     0.8152902 -1.7963576 1.3712636
##   Std.Error.intercept Std.Error.slope t.intercept
## 1           0.4755196      0.07663692   -0.606665
## 2           0.5525336      0.08904886    2.975743
## 3           0.3968687      0.05917792    1.129866
## 4           0.6405375      0.09551189    3.139782
## 5           0.9267664      0.15469722   -1.551725
## 6           1.2872939      0.21487701   -1.395453
```

```
##     t.slope Pr.intercept     Pr.slope
## 1 14.577270   0.56089072 4.808514e-07
## 2  8.036919   0.01771512 4.224404e-05
## 3 14.577270   0.29126867 4.808514e-07
## 4  6.381621   0.01380978 2.132732e-04
## 5  8.036919   0.15932818 4.224404e-05
## 6  6.381621   0.20039507 2.132732e-04
```

### 文件读写函数

如果需要同时对多个数据文件进行处理，逐个读入 R 里太麻烦。beginr 提供了个 `readdir()` 函数，可以一次性把指定文件夹里所有数据读入，保存在一个列表 (list) 里。

R 自带的 write 系列文件保存函数，一不小心就把原有同名文件给覆盖了。beginr 提供了个安全的函数 `writefile()`，避免一失足成千古恨。

`list2ascii()` 函数可以把一个列表原样保存成文本文件。

有些用户引用了别人的工作却不列出参考文献，这对于开源社区的生态圈是不利的。beginr 为初学者提供了 `bib()` 函数, 可以为指定的 R 扩展包生成文献引用信息。这个函数既可以打印出结果, 也可以直接保存为 '.bib' 文件方便为'bookdown' (Xie, 2016a) 或 'blogdown' (Xie, 2017) 调用，还可以方便地导入到其他文件管理软件里（如 Endnote）。

```
beginr::bib(pkg = c("mindr", "bookdownplus", "pinyin"))
```

```
## @Manual{R-bookdownplus,
##   title = {bookdownplus: Generate Varied Types of Books
and Documents with R 'bookdown'
## Package},
##   author = {Peng Zhao},
##   year = {2017},
##   note = {R package version 1.3.2},
##   url = {https://github.com/pzhaonet/bookdownplus},
```

```
## }
## @Manual{R-mindr,
##   title = {mindr: Convert Files Between Markdown or
Rmarkdown Files and Mindmaps},
##   author = {Peng Zhao},
##   year = {2017},
##   note = {R package version 1.1.0},
##   url = {https://CRAN.R-project.org/package=mindr},
## }
## @Manual{R-pinyin,
##   title = {pinyin: Convert Chinese Characters into
Pinyin},
##   author = {Peng Zhao},
##   year = {2017},
##   note = {R package version 1.1.0},
##   url = {https://CRAN.R-project.org/package=pinyin},
## }
```

事实上，`bib()` 函数是受 knitr (Xie, 2014, 2015, 2016c) 包里的 write_bib() 函数启发而写成的。请比较一下两者的区别。

此外，beginr 还有个 `rpkg()` 函数，可以帮助初学者开发自己的 R 扩展包，其用法将在下一节介绍。未来，beginr 包里还会增添新成员。

其实在本书前几章，早就有扩展包露出尖尖角了。在第七章，我们使用的 R markdown (Allaire et al., 2016) 就是个扩展包。当时生成的 Word 文档和幻灯片还比较简陋，现在有了更多的扩展包，我们就有了更多的选择。例如，xlsx (Dragulescu, 2014) 和 ReporteRs (Gohel, 2017) 扩展包在 R 与 Excel、Word 和 Powerpoint 之间搭起了友好的桥梁，可以将他们结合起来，高效地生成漂亮的办公产品。

现在出个抢答题：按章节先后顺序，你猜，本书里最先用到的是什么扩展包？

赶紧往前翻！

找到了吗？

我们在第三章介绍了 ggplot2 (Wickham, 2009) 和 lattice (Sarkar, 2008)，他们是诸多绘图厨房里配备的最为华美的餐厅。

但是，很可惜，本书最先用到的不是他们。

本书文字里最先用到、并且最经常用到的扩展包，其实是一个叫做 fortunes (Zeileis et al., 2016) 的扩展包。这是个神奇包包。我们来安装调用运行一下，下面是见证奇迹的时刻：

```
install.packages("fortunes")
```

```
require(fortunes)
fortune('Actually, I see it as part of my job')
```

```
##
## Actually, I see it as part of my job to inflict R on
## people who are perfectly happy to have never heard of
## it. Happiness doesn't equal proficient and efficient.
## In some cases the proficiency of a person serves a
## greater good than their momentary happiness.
##    -- Patrick Burns
##       R-help (April 2005)
```

这就是本书第一章开篇引用的那句话。别的章节出现的语录，很多来自这个扩展包，里面搜集的是 R 语言社区一些“自吹自擂”和“自省自黑”的语录，原文是英文，本书引用时被我翻译成了中文。老实说，第一次听说这个包的时候，我被雷到了，真不知道开发者是怎么想的，这个包平时没一点用处。直到写这本书的时候，发现里边的语录插在本书里倒是恰到好处。请呼唤小助理 `vignette('fortunes')` 来看看这个包的内容。

实际上，在本书用到的众多扩展包背后，隐藏着一个终极大 boss，从本书的第一个字就开始用了：本书的写作环境和排版工具，不是 Word，也不是 LaTeX，而是 bookdown (Xie, 2016a,b) 扩展包[2]。书中漂亮的格式、准

[2]bookdown 幕后仍然调用了 LaTeX，但在使用上大为简化，极大地优化了用户体验。

确方便的交叉引用、清晰的脚注和参考文献，全部是它完成的。

R 语言走到今天，是一个聚沙成塔、集腋成裘的过程，其中的“沙”和“腋”，正是众多热心人花心血写成并奉献出来的扩展包。每个人献出一滴水，终于创造出如今的汪洋大海任你畅游。所以，使用别人慷慨贡献出来的扩展包，别忘了对开发者致谢。就像本书列出的参考文献那样，引用信息可以用 `citation()` 函数，例如：

```
citation("bookdown")
```

```
##
## To cite the 'bookdown' package in publications
## use:
##
##   Yihui Xie (2017). bookdown: Authoring Books
##   and Technical Documents with R Markdown. R
##   package version 0.5.
##
##   Yihui Xie (2016). bookdown: Authoring Books
##   and Technical Documents with R Markdown.
##   Chapman and Hall/CRC. ISBN 978-1138700109
##
## To see these entries in BibTeX format, use
## 'print(<citation>, bibtex=TRUE)', 'toBibtex(.)',
## or set 'options(citation.bibtex.max=999)'.
```

R 的扩展包每年都在增加。现在到底有多少个扩展包呢？用这条命令：

```
length(unique(rownames(available.packages())))
```

这是四个函数的嵌套，等同于：

```
a <- available.packages() # 获取所有扩展包的信息
b <- rownames(a) # 挑出扩展包的名称
c <- unique(b) # 去掉重复的名称
```

```
d <- length(c) # 数数有几个
```

beginr 扩展包里有个 `plotpkg()` 函数，可以绘制指定扩展包被下载的情况。例如，我们看一下 'rmarkdown' 扩展包的下载趋势：

```
beginr::plotpkg('rmarkdown', from = '2014-01-01')
```

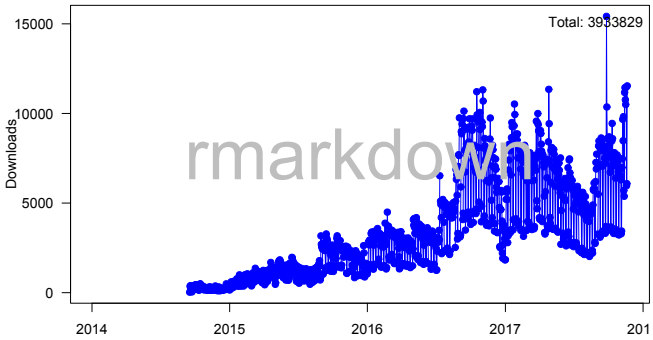



图 9.5: rmarkdown 扩展包日下载量趋势图

这上万个厨房，被那些好心的作者们免费放在网上，让我们随心所欲安装回家; 加上 R 的基础内置厨房，让 R 成为了一个超级美食王国。来尽情享受这饕餮盛宴吧！

---

**练习 9.5.** 办公扩展包

请使用 xlsx 和 ReporteRs 扩展包，制作一个 Excel、Word 和Powerpoint文档。

**练习 9.6.** 制作动画

我们在学习循环的时候制作过动画。实际上，扩展包里就有专门制作动画的。请自己找一个扩展包，制作一段动画。列出该扩展包的参考文献。

**练习 9.7.** 绘制风玫瑰图

风玫瑰图（windrose）是气象学常用的图形，显示的是一段时期内各个

风向所占的比例。请找一个能够绘制风玫瑰图的扩展包，绘制一张风玫瑰图，并列出该包的参考文献。

**思考 9.2.** 这么多有趣有用的扩展包，你想用来做哪些事情？

然而，“福兮祸之所伏”，在这包含了一万个扩展包的宝藏里，普通人很容易被贪欲吞噬而迷失自我，忘了自己的初衷。如何在这个大宝藏里找到适合自己的，以及自己想要的那一份？

继续请出法宝帮忙吧，网上搜，论坛问。CRAN 的网页上有个“任务视图”[3]，针对这些包进行了粗略分类，例如空间数据分析、时间序列分析、生态和环境数据分析等。按你的目的去找就是了。

看着那些琳琅满目的扩展包，有种逛苹果商场看应用商店的感觉。有了这些五花八门三教九流的扩展包，我们就：

> 可上九天揽月，可下五洋捉鳖，谈笑凯歌还。
> 世上无难事，R 包助通关。
>
> — 改自毛泽东《水调歌头 • 重上井冈山》

---

**小贴士 9.1.** 关于扩展包的常用函数

| 作用 | 函数 |
|---|---|
| 自定义函数 | `function()` |
| 查看可用扩展包 | `available.packages()` |
| 安装扩展包 | `install.packages()` |
| 调用扩展包 | `require()`, `library()` |
| 引用扩展包 | `citation()` |

[3]CRAN 任务视图：https://cran.r-project.org/web/views/

## 9.4　包的开发：开疆拓土

开发 R 扩展包，听起来像是个浩大的工程，但是实际上并不难。学到这里，你完全可以开发自己的扩展包，在美食王国里开辟一片自己的天地。

扩展包的开发，最好的参考资料之一是 RStudio 公司首席科学家 Hadley Wickham 写的 *R packages* 一书，国内有中文译本，英文版可以在网上免费获取[4]。这里，我们用最简短的篇幅，带你一起创建一个可用的扩展包，实际动手的时间前后不会超过十分钟。

**第一步：准备**

R 包开发需要一系列辅助工具。除了下载安装 Rtools[5]外，运行 R 安装几个扩展包：

```
install.packages(c("devtools", "roxygen2", "knitr", "beginr"))
```

**第二步：创建新包**

如果你已经安装了上一节介绍的扩展包 "beginr"，那么只需运行下面的代码，就可以在工作目录得到一个名为 rpkg 的文件夹。这就是我们的新包。

```
beginr::rpkg()
```

下面，我们只需修改其中的两个文件就可以了。

第一个文件是 DESCRIPTION 文件。双击 rpkg 文件夹里的 rpkg.Rproj，就会用 RStudio 打开该项目，然后点击右下面板 Files 标签下的 DESCRIPTION 文件。这个文件里是包的描述信息，包括包名、版本、标题、作者、维护者、描述等信息。把其中每一条冒号后面的信息改成我们自己的信息，不必要的行可以删除，例如可以改为：

```
Package: mypkg
Type: Package
```

[4] *R packages*：http://r-pkgs.had.co.nz/

[5] Rtools：http://cran.r-project.org/bin/Windows/Rtools/

```
Title: Calculate New Score
Version: 0.0.0
Author: Peng Zhao
Maintainer: Peng Zhao <pzhao@pzhao.net>
Description: Get a better score.
License: GPL
Encoding: UTF-8
LazyData: true
```

按 ctrl+s 保存这个文件。第一个文件 DESCRIPTION 就修改完毕了。

第二个文件是右下面板 Files 标签下的 R/foo.R 文件。这是存放我们自定义函数的地方。打开这个文件，把其中的内容全部删掉，换成自己的函数，例如前面提到的及格函数 newscore。

```
newscore <- function(x) {
  y <- sqrt(x) * 10
  return(y)
}
```

然后，将光标移到函数名称的位置，按快捷键 ctrl+shift+alt+r，函数前面就自动添加了几行信息。我们需要将信息填写完整：

- 首行是函数的标题，一般用来简要描述函数的作用。这里我们将 'Title' 改成 'Calculate new score'；
- `#' @param x` 一行用来介绍自变量。我们在后面添加文字 `old score`；
- `#' @return` 一行用来介绍函数的返回值。在后面添加 `new score`；
- `#' @examples` 一行用来举例子。在后面添加文字 `newscore(49)`。

最后得到的文件内容是这样的：

```
#' Calculate new score
#'
#' @param x old score
#'
#' @return new score
```

```
#' @export
#'
#' @examples newscore(49)
newscore <- function(x) {
  y <- sqrt(x) * 10
  return(y)
}
```

我们上面填写的信息当然是不够详细的，但是不要紧，作为一个测试包已经够用了。

好了，一个新包已经写完了。

**第三步：编译和本地安装**

按 ctrl+shift+b，或者点击 RStudio 右上模板 `Build` 标签下的 `Build & Reload` 按钮，包就自动编译、安装和加载好了！祝贺！

现在，我们可以像使用别人的包一样使用自己的包了。例如，运行一下刚才自定义的 `newscore()` 函数:

```
library(mypkg)
newscore(36)
```

```
## [1] 60
```

还记得小助理 `example()` 吗？

```
example(newscore)
```

```
##
## newscr> newscore(49)
## [1] 70
```

把光标移到函数名 `newscore` 上，按 F1 键，帮助信息就在右下面板的 Help 标签下出现了：

**Calculate new score**

**Description**

Calculate new score

**Usage**

newscore(x)

**Arguments**

x old score

**Value**

new score

**Examples**

newscore(49)

将这个帮助信息跟我们在 foo.R 里填写的说明信息对比，就会明白帮助信息里每处的说明是怎么来的了。你可以在 foo.R 里如法炮制，添加更多的自定义函数，重复上面的编译过程。

我们新建的这个扩展包，就是我们在 R 扩展包美食王国里开拓的属于我们自己的疆土。

简单来说，运行 `beginr::rpkg()` – 修改 DESCRIPTION 和 R/foo.R – 编译成包，这就是我们提出的 ‘‘R 包开发极简三步曲’’。

当然，也可以在 RStudio 里通过点击菜单栏来创建新包，但我们仍然建议从 beginr 出发来创建新包，更为简易，也更容易理解开发 R 包的工作流程。

---

**练习 9.8.** 将练习 9.1、练习 9.2、练习 9.3里的三个自定义函数写成一个名为 firstpkg 的扩展包。

欲穷千里目，更上一层楼。用极简三部曲写出的 R 包，已经足够自用。在这个高度，看到的风景已经够美。你可以在这个水平的观景台休整休整，再考虑是否继续往上攀登。孔子登东山而小鲁，登泰山而小天下。如果想看到更高处的风景，那么就继续下一步。

**第四步：发布和分享**

只有把扩展包发布和分享出去，包才会得以充分利用，得到更多用户的建议和改进，才真正有了生命力。正如史书分正史和野史，R 包可以根据发布和分享方式分为“正包”和“野包”。

正包，就是通过 R 官方认可并且发布在 CRAN 网站上的，等于领到了认证和许可，可以用 `install.package()` 直接安装，我们前面已经接触了很多；而野包就是自己私下写的，随便放在自己喜欢的地方小范围使用。很多包转正之前都是野包，经过反复测试和升级后才转的正，而有的野包会永远野下去；很多包转正之后仍然保留野包作为开发版，将野包升级到一定程度后再次转正。这是个自由的世界。

这里，我们主要介绍野包的发布和分享方法，而对于转正的方法仅作简略介绍。

野包的发布方法首推 GitHub 法。熟悉 GitHub 的话，建议用这种方法。既然熟悉，我们这里就不赘述 GitHub 的用法了。简单来说：

- 你要做的，就是在 github.com 上申请个账号，电脑里安装个客户端，在客户端新建个项目，并起个名字，例如叫 mymickey ，把你新的 R 包文件夹同步上去，就可以通知用户们来使用你的 mymickey 包了。
- 用户要做的，就是在 R 里运行

```
devtools::install_github(" 你的账号/你的项目名称")
```

就把你的包装到他们的 R 里了。

举例来说，beginr 这个扩展包的开发版就是这样在 GitHub 发布的，从 GitHub 安装的方法是：

```
devtools::install_github("pzhaonet/beginr")
```

如果你不喜欢 GitHub 的用法，那么可以用离线法发布：让用户把包的压缩文件先下载到他们电脑上，再用 R 包的离线安装方式。包的压缩文件其实在编译时已经自动生成了。去看看你电脑上的 R 包文件夹，跟它并列存放的，有个与包同名的.tar.gz 压缩文件。这就是离线安装包。下面：

- 你要做的，就是随便以任何方式，不管是 email、qq、网盘，还是优盘、手机、移动硬盘，只要让用户能得到你这个压缩文件就行了。
- 用户要做的，就是拿到文件后在他们的电脑上安装，方法是在 RStudio 菜单栏点 Tools – Install Packages – Install from Package Archive File，选上你的包，就行了。

上面讲的是野包的发布分享方法。如果想提交到 CRAN 转正，其实也不难，这里给出主要步骤：

1. 自查。包发布之前，先自查有没有错误。只需用 RStudio 打开待发布的 R 包的.Rproj 文件，然后点击右上方面板的 Build – Check 按钮，就可以看到自查结果。确保其中没有任何错误和警告，否则就要修改。
2. 提交。把包的压缩文件.tar.gz 在 CRAN 的提交网页[6]上传即可。成功上传后会收到 email 通知，需要点击里面的链接进行确认。确认后进入 CRAN 的自动检查状态。如果包的毛病太多的话，系统会自动发拒信，并给出拒绝理由。按理由去修改，重修提交即可。
3. 修订。如果通过了 CRAN 自动检查，就进入人工检查阶段。运气好的话，几乎在提交当天就会收到管理员的邮件，指出包的不足和修改意见。只需按意见修改，并且在 DESCRIPTION 文件里把版本修改成新的号码，重新编译再次上传就可以了。对于新手来说，这个过程可能会反复几个回合，有时候得到的修改意见可能比较苛刻，需要一些耐心。管理员不计酬劳义务做这项工作，请对他们保持足够的礼貌和尊重。

[6]R 包提交：https://cran.r-project.org/submit.html

从上面介绍的步骤可以看出，R 包的开发就像往学术期刊投稿一样繁琐耗时。那么，让我们回过头来谈一个问题：**我们为什么要开发 R 包？**

这个问题见仁见智，我说说从自己开发 R 包的体会。

**首先是方便自用。**

使用 R 语言这几年里，我逐渐积累的自定义函数越来越多，我把他们通通存到一个 'myfunction.R' 文件里。现在，这个文件已经有 3000 多行，将近 100 个自定义函数，越多越记不住，用时查找越来越困难。有个相对简单的方法，就是每次用时都运行 `source('myfunction.R')`，稍微方便一些，虽然会导致 RStudio 右上面板 Environment 标签下特别冗长，而且经常忘记这些函数的用法，需要查看函数源代码，但总好过每次把函数重写或重拷贝一遍。没写成包之前，这几年都是这么过来的。

如今，我把这些函数写成了一个野包，取名 mf，并分享到了 GitHub[7]上。这个包纯属自用，帮助信息写得不够齐全和详细，但自己懂就行了，用起来的爽快程度，当然比 `source()` 强多了，不仅 Environment 区清爽了，更关键的是，忘了的话就用 tab 和 F1 大法，几年积攒的不快一扫而空。就算换个工作单位，换台电脑，花几秒钟安装后照旧调用，处处无家处处家。

**其次是改善代码。**

写代码对我来说一直是件封闭的事情：自己写，自己用，一旦出错，只有自己知道。计算方法没有经过同行评议，出错的风险很大。但是我又不好意思请别人花时间帮我改代码。两全的办法，就是开源，写成包，放在 GitHub 上，接受志愿者挑错，有则改之无则加勉。只有这样，才能把代码错误越来越少，改得越来越好，最终还是自己用起来更放心。我有几个包在发布之后，很快得到了别人的修改意见，这才知道哪里可以改进。与此同时，代码也可以方便别人使用，利人利己，何乐不为？

**最后是审视自我。**

我的第一个 R 包 postr[8]是用 R markdown 做海报，兴致勃勃地放到

[7] mf 包：`https://github.com/pzhaonet/mf`

[8] `https://github.com/pzhaonet/postr`

GitHub 第一天，就有人留言汇报问题。我知道，这只是个开始，以后会有更多的问题，里面会有赞美，可能也会有谩骂。写个 R 包真的不算什么，等待在前面的一个又一个的坑才是让人真正头疼的事情。

这个阶段就要接收考验了。你会重新审视自己：我这么干到底是图什么？我自用已经够了，却要花大量的时间解决别的用户层出不穷的问题，为什么？

开源软件是面镜子，从中可以看见开发者的影子；开发 R 扩展包，可以从中看见自己。唐太宗说过：

> 以铜为鉴，可以正衣冠；
> 以人为鉴，可以知得失；
> 以史为鉴，可以知兴替；
> 以 R 包为鉴，可以知本我、自我和超我。

## 9.5 课外活动：餐后甜点

上大学的时候，在食堂吃完午饭，我们会玩几分钟扫雷游戏，时间短，乐趣多，健脑益智，而且随时可以停下来。

扫雷是旧版 Windows 自带的经典小游戏，可惜 Windows 7 之后，扫雷就不再是默认安装了。不要紧，我们可以在 R 中玩扫雷，因为有这样的扩展包在。

请用搜索引擎找到有扫雷游戏的扩展包，安装，并且试玩。这个扩展包里，包含了第一章对 R 表白时绘制那颗中国心的秘密。你能找到这个秘密吗？

看看你的扫雷记录吧。比比看，谁能登上扫雷英雄榜！

# 第十章　字符

你会发现一个奇特的现象：很多人使用 R 多年，已经能够用 R 做很多事情，但是他们却从不自诩懂得了这门语言。

— Luis Argerich

前边很多章节都用到了字符串类型的变量，例如作图的图例、表达式、坐标轴的标签，读写文件时文件的路径，数据框的行列名称等。显而易见，字符串的运算不是加减乘除，而是连接、分割、截取、查找、替换。

为什么要学习字符串的处理呢？对因子操作时，保存文件时，或是作图要添加文字时，如果懂得如何处理字符串，这些任务就会很方便完成。放眼望去，字符串充满了我们的世界，有了 R 这个利器，我们就可以做很多想做的事。比如前几年“韩方”大战时，有人对韩寒作品进行的文本分析，等学完本章之后，我们也可以做。

## 10.1　狐狸从懒狗身上跳过

让我们从这样一句话开始：

```
The quick brown fox jumps over the lazy dog.
```

不知你有没有注意到，这句话经常在电脑软件看到。敏捷的褐色狐狸从懒狗身上跳过？这是什么意思？

据说，在很久以前的打字机时代，打字机需要测试每个键都能打出字来。为了省纸省墨，打字员就去打这样一句话，英文里所有的 26 个字母都包含在这个简短的句子里，所有字母的字体效果一目了然。这个文字游戏叫做“全字母句”（pangram）。字母重复出现的次数越少越好（我表示不懂：把键盘上的键挨个儿敲一遍是不是更省脑子……）。

真有这么神奇？上面这句话里，真的包含了全部的 26 个字母吗？有哪些字母重复出现过，出现过几次？

我们用 R 来试试，边玩边学字符处理函数。

这个游戏里，我们不区分字母的大小写，T 和 t 算是同一个字母。所以，我们第一步先用 `tolower()` 函数或者 `toupper()` 函数，把全部字母转换成小写或大写。

```
x <- 'The quick brown fox jumps over the lazy dog'
xlower <- tolower(x)
class(xlower)
```

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

`class()` 函数返回的结果显示，这确实是个字符变量。下面看看字符的长度：

```
length(x)
```

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

```
nchar(x)
```

```
## [1] 43
```

`nchar()` 用来查看字符串的长度。注意它跟 `length()` 的区别。后者是查看向量中元素的个数。向量 x 里只有 1 个元素，这个元素包含的字符数是 43。

R 用成对儿的单引号或双引号来表示字符和字符串。其实前面我们已经接触过了，比如读入或存储文件时需要用到文件名，就是个字符串：

```
myfile2 <- "c:/r4r/co2.csv"
```

---

**思考 10.1.** 单引号和双引号的用法完全相同吗？什么情况下不相同？

第二步，我们用字符串分割函数 `strsplit()`，将 x 这个字符串拆分成单个字符。

```
xsingle <- strsplit(xlower, '')[[1]]
nchar(xsingle)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [26] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
length(xsingle)
```

```
## [1] 43
```

请在这里再次体会一下 `nchar()` 函数与 `length()` 函数的区别。

---

**思考 10.2.** 为什么 *strsplit()* 后面跟了个 *[[1]]*？去掉 *[[1]]* 行吗？

第三步，我们看看第一个字符是否在别的地方出现。可以用 `grep()` 函数（global regular expression print 的缩写）：

```
grep(xsingle[1], xsingle)
```

```
## [1]  1 32
```

返回的结果显示了第一个字符 t 在整个句子中出现的位置。

后面的就是重复操作了，逐个检查每一个字符就行。这提醒我们，应该用循环语句：

```
for(i in 1:length(xsingle)) print(grep(xsingle[i], xsingle))
```

```
## [1]  1 32
## [1]  2 33
## [1]  3 29 34
## [1]  4 10 16 20 26 31 35 40
## [1]  5
## [1]  6 22
## ......
```

print() 函数的作用是将结果打印在操作台。这样就得到了每个字符在句子中重复出现的位置。在多个位置出现的，自然就是重复的字母了。

这个输出结果不够明显，要是能同时显示是哪个字符就好了。这个好办，我们用字符串连接函数 paste()，将字符名称跟出现位置连起来后再打印就行了：

```
for(i in 1:length(xsingle))
  print(paste(xsingle[i], grep(xsingle[i], xsingle)))
```

```
## [1] "t 1"  "t 32"
## [1] "h 2"  "h 33"
## [1] "e 3"  "e 29" "e 34"
## [1] "  4"  "  10" "  16" "  20" "  26" "  31" "  35" "
40"
## [1] "q 5"
## [1] "u 6"  "u 22"
## ......
```

除了 paste() 函数，另外还有猫函数 cat() 可以用——当然，cat 在这里不是猫，而是 concatenate 的缩写。

由于学过了第 5.5 节，一看是循环，我们就考虑一下能不能超越循环，改为向量操作：

```
sapply(xsingle, function(x) grep(x, xsingle))
```

```
## $t
## [1]  1 32
##
## $h
## [1]  2 33
##
## $e
## [1]  3 29 34
## ......
```

得到的结果跟上一条指令是一致的。

我们已经得到了期望的结果，虽然输出的格式难看点。好看一点的输入结果，可以用 table(), duplicated() 和 unique() 三姐妹。

表格函数 table() 计算的是每个元素出现的个数：

```
table(xsingle)
```

```
## xsingle
##  a b c d e f g h i j k l m n o p q r s t u v w x y z
## 8 1 1 1 1 3 1 1 2 1 1 1 1 1 1 4 1 1 2 1 2 2 1 1 1 1 1
```

“重复函数”duplicated() 可以查找是否有重复元素。

```
duplicated(xsingle)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [9] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE
## [17] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE
## [25] FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
## [33]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE
## [41] FALSE  TRUE FALSE
```

哪些字符第二次出现，哪些就返回 TRUE。

结合逻辑运算符和元素的下标系统，就可以显示哪些字母没有重复：

```
xsingle[!duplicated(xsingle)]
```

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

这跟“不重复函数”unique() 返回的结果是一样的：

```
unique(xsingle)
```

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

总共有多少个不同的字符呢？

```
length(unique(xsingle))
```

```
## [1] 27
```

结果里包含了空格字符。由此我们可以确定，26 个字母的确全在这个狐狸句子里了。

---

**练习 10.1.** 全字母句有很多，比如下面是另外一个：Pack my box with five dozen liquor jugs. 请用 R 代码找出其中重复的字母以及重复的次数。

## 10.2 千字文的重复字

英文的全字母句很短，找重复的字母，就算不编程，瞪大眼睛数也能数得出来。但是，如果字数太多，眼睛就不好用了，比如我国的传统启蒙经典《千字文》。

十五个世纪前，南朝梁武帝令大臣周兴嗣把王羲之写过的 1000 个汉字连成一篇文章，增加临帖练字的便利和趣味性。周兴嗣冥思苦想，一夜未眠，东方既白，大作已成，镜子一照，黑发尽如雪。

这篇奇文大概是最接近“全字母句”要求的汉语文章。《千字文》最大的亮点是“字不重复”，但是，却有人提出质疑，认为有的字出现了重复，重复的字数众说纷纭，有的说 1 个，有的说 6 个，还有人说有 7 个。

现在，我们来用 R 语言来找找千字文里到底有几个重复字。

首先从网上找到千字文的全文。我们选择的是“古诗文网”[1]提供的版本，事先已经保存成了一个文本文件。下面的这条命令，将从我们的服务器上把这个文本文件下载到你的本地硬盘。

```
download.file(url =
                "http://dapengde.com/r4rookies/qianziwen.txt",
              destfile = "c:/r4r/qianziwen.txt")
```

这个文件是个纯文本文件，可以用记事本打开，不像我们以前常用的表格数据。这样的文本，我们用 `readLines()` 函数读取，得到的是个字符串向量。

```
qzw <- readLines('c:/r4r/qianziwen.txt', encoding = 'UTF-8')
class(qzw)
```

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

```
length(qzw) # 向量长度
```

```
## [1] 249
```

```
nchar(qzw) # 向量里每个元素包含的字符数
```

```
##   [1] 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9
##  [26] 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0
##  [51] 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9
```

[1]古诗文网：http://so.gushiwen.org/guwen/bookv_2745.aspx

```
##  [76] 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0
## [101] 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9
## [126] 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0
## [151] 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9
## [176] 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0
## [201] 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9
## [226] 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9
```

如果想借用前面处理英文句子方法，那么我们先得把这个向量打散后合并成一个元素：

```
qzwmerged <- paste(qzw, collapse = '')
```

上次我们宽容地把空格留下了，这回我们把它踢出去，免得干扰计数。这可以用替换字符函数 gsub() 来完成：

```
qzwmerged <- gsub(' ', '', qzwmerged)
nchar(qzwmerged)
```

```
## [1] 1000
```

得到的是整整 1000 个字了。字符替换函数另外还有 sub() 函数和 chartr() 函数。有什么区别？问你的小助理。

现在我们得到的千字文，跟狐狸懒狗一样，成了一句话，就可以用完全相同的方法来检验了：

```
qzwsingle <- strsplit(qzwmerged, '')[[1]]
chardup <- qzwsingle[duplicated(qzwsingle)]
for(i in chardup) print(paste(i, grep(i, qzw, value = TRUE)))
```

```
## [1] " 发 吊民伐罪 周发殷汤" " 发 盖此身发 四大五常"
## [1] " 义 节义廉退 颠沛匪亏" " 义 俊义密勿 多士实宁"
## [1] " 实 策功茂实 勒碑刻铭" " 实 俊义密勿 多士实宁"
## [1] " 云 云腾致雨 露结为霜" " 云 岳宗泰岱 禅主云亭"
## [1] " 昆 金生丽水 玉出昆冈" " 昆 昆池碣石 钜野洞庭"
```

```
## [1] " 戚 欣奏累遣 戚谢欢招" " 戚 亲戚故旧 老少异粮"
## [1] " 洁 女慕贞洁 男效才良" " 洁 纨扇圆洁 银烛炜煌"
## [1] " 并 九州禹迹 百郡秦并" " 并 释纷利俗 并皆佳妙"
```

果然，有 8 个字重复了！那么，千字文号称没有一字重复是学术造假么？

这就牵扯出简体字和繁体字的恩怨了。例如，表面上 ‘‘发’’ 字重复出现在 ‘‘周发殷汤’’ 句和 ‘‘盖此身发’’ 句，但原文其实为 ‘‘周發殷汤’’‘‘盖此身髮’’。‘‘發’’ 读第一声，是武王姬发的名字；‘‘髮’’ 读第四声，是头发。这大概是最常见的混淆了。有些理发店把店名写作 ‘‘理發店’’，卡拉 OK 把 ‘‘发如雪’’ 写成 ‘‘發如雪’’，都弄错了。周潤發以前给百年潤髮的洗发水做广告，更加深了这种误会。

再如 ‘‘云’’ 出现在 ‘‘云腾致雨’’ 句和 ‘‘禅主云亭’’ 句，原文为 ‘‘雲腾致雨’’‘‘禪主云亭’’，‘‘雲’’ 是云彩，‘‘云’’ 是泰山脚下的一座山名为 ‘‘云山’’，封禅之地。繁体字 ‘‘雲’’ 和 ‘‘云’’ 完全是两个不同的字。

此外，古诗文网提供的这个版本有错别字，比如错把 ‘‘俊乂密勿’’ 写成了 ‘‘俊义密勿’’ 而导致跟 ‘‘节义廉退’’ 的 ‘‘义’’ 字重复。

---

**思考 10.3.** 除去这三个字，另外重复的 *5* 个字是这么回事？

## 10.3　整理读书笔记

不管是全字母句还是千字文，找重复的字只是个趣味游戏，没什么实际用处。下面，我们用字符串的处理方法来做一件有用的事：整理文本。

我有很多朋友爱读书，或早或晚入手了 Kindle 电子书阅读器，从此陷入了读书的大坑里不能自拔。而我在使用了一年之后，把 Kindle 上的常规操作都熟悉了，但只有一个问题没有好好解决：整理笔记和高亮文本。

Kindle 的笔记和高亮文本都保存在 Kindle 的根目录下 documents 文

件夹里名为 “My Clippings.txt” 的文本文件里，在电脑中可以用记事本中直接打开，看上去是图 10.1的效果。文本是按做笔记的时间顺序逐条列下去的，非常凌乱。如果能整理成表格，按照自己需要的顺序来排列就好了。

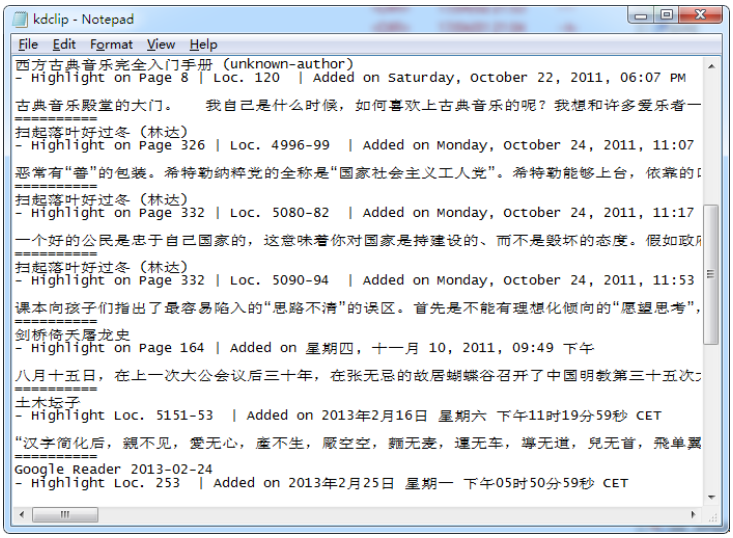



图 10.1: 电子阅读器的读书笔记

对此，网上倒是有现成的方法对笔记进行整理，比如下载 Word 的宏，或有人写了专门的软件，或在线转换，但要么是有安全隐患，要么是要收费，要么是不支持中文。最不爽的，是不能随心所欲。

R 用久了，就会习惯了一双自由的翅膀，和一颗任性的心。现在，我们用 R 把笔记整理到表格里，并把来自同一本书的内容排在一起。

这里，我们提供一个示例文件，仍然是来我们服务器上下载，并读入 R：

```
download.file(url =
                "http://dapengde.com/r4rookies/kdclip.txt",
              destfile = "c:/r4r/kdclip.txt")
```

```
aa <- readLines("c:/r4r/kdclip.txt", encoding = "UTF-8")
```

我们先用记事本或别的软件打开这个文本，观察一下它的结构。每条笔记占 4 行，依次是书名，书中的位置和做笔记的时间，空行，笔记内容，两条笔记之间的间隔符。我们要做的事情是，生成一个表格，第一列是书名，第二列是笔记内容。

我们先把所有笔记的标题找出来，保存在一个叫 title 的向量里：

```
length.aa <- length(aa)
title <- aa[c(seq(1, length.aa, by = 5))]
```

然后，用排序函数 order()，对所有的标题按字母排序：

```
title.o <- order(title)
title <- title[title.o]
```

接着，摘取笔记的高亮内容，存在叫 highlight 的向量里：

```
highlight <- aa[c(seq(4, length.aa, by = 5))][title.o]
```

最后，把这两个向量保存在一个数据框里，大功告成，数据框保存到硬盘里。

```
kn <- data.frame(Title = title, Highlight = highlight)
write.table(kn, "c:/r4r/kn.txt",
            sep = "\t", row.names = FALSE)
```

保存文件时我们用得分隔符 sep 参数为\t，表示用制表符 tab 分隔。用 Excel 打开我们整理的表格看看吧！

也许你还不够满意：嗯，要是，笔记位置和做笔记的时间信息也放进表格里备案就好了。这好办，我们先把包含信息的行都提取出来。

```
location <- aa[c(seq(2, length.aa, by = 5))][title.o]
location[1]
```

我们发现，笔记位置的信息的格式是统一这样的：每行从第 13 个字符开始，到 Added on 倒着数到第 5 个字符之间，就是笔记位置的信息；而 Added on 里的第一个字符往后数第 10 个字符起，到这一行的末尾就是做

笔记的时间。那么，我们用字符查找函数 regexpre() 来找到 Added on 出现的位置，然后用字符截取函数 substr() 截取信息就可以了：

```
time.aa <- substr(location,
                  (regexpr(" Added on ", location) + 10) ,
                  nchar(location))[title.o]
loc <- substr(
  location, 13,
  regexpr(" Added on ", location) - 5)[order(title.o)]
kn <- data.frame(Title = title, Highlight = highlight,
                 Loc = loc, Time = time.aa)
write.table(kn, "c:/r4r/kn2.txt",
            sep = "\t", row.names = FALSE)
```

看看新表格，是不是信息齐全了？

用这样的方法，不仅可以把笔记整理成表格，还可以对笔记进行再加工，例如计算笔记中某个关键词出现的频数，每本书做笔记的条数；还可以根据时间信息，来看看自己哪段时间读书多，看看自己的读书兴趣如何随时间在转移。随心所欲，想怎么玩就怎么玩。如果这里介绍的函数仍然不能满足你的需求，还可以使用 stringr 扩展包 (Wickham, 2017)，里面提供了更加强大的工具。

---

**练习 10.2.** 中文是博大精深的，除了 1000 字不重复的千字文以外，我们还有一部《中华字经》。

字经的官方网站[2]介绍说：

世界上最奇妙的汉字学习课本。……全文 4000 汉字没有一个字是重复使用的。

字经包括了常用汉字 3500 个。

该网站是国际网站，有繁体字版，还有英文介绍：

[2]中华字经：http://www.miktamchinese.com/

The book does not repeat any characters…… ‘‘Chinese Character Canon’’ is a poem composed of 4,000 characters…

人民日报对此报导说：

中华字经共收录无重复汉字 4000 个。

请用 R 语言看看中华字经是不是真的没有一字重复。如果有，是哪些字，各重复出现了几次，是什么原因造成了重复？

此外，中华字经是否真正包括了常用汉字 3500 个？

**小贴士 10.1.** 字符处理函数

| 函数 | 作用 |
|---|---|
| `readLines()`, `writeLines()` | 按字符读取和保存文本文件 |
| `tolower()`, `toupper()` | 大小写转换 |
| `nchar()` | 字符个数 |
| `strsplit()`, `substr()`, `substring()` | 字符串分割和截取 |
| `paste()`, `cat()` | 字符连接 |
| `grep()`, `gsub()`, `sub()`, `chartr()` | 查找和替换 |
| `table()`, `unique()`, `duplicated()` | 计数，查重（不限于字符变量） |

## 10.4　课外活动：张无忌的困惑

金庸的小说《倚天屠龙记》后记中写到：

张无忌却始终拖泥带水，对于周芷若、赵敏、殷离、小昭这四个姑娘，似乎他对赵敏爱得最深，最后对周芷若也这般说了，但在他内心深处，到底爱哪一个姑娘更加多些？恐怕他自己也不知道。

张无忌到底爱谁？对于这个问题，已经有人用 R 语言的字符处理方法进行研究了，研究的大体思路是，看看哪个姑娘的名字在书中出现的次数

最多，尤其是跟张无忌的名字同时出现的次数最多。最后，R 代码得出了结论，张无忌最爱的其实是……请上网搜吧。

发挥你的想象力，看看利用类似的方法，能否对你喜欢的作品进行分析，得到一些有趣的观点。例如，

- 贾宝玉最常跟哪个女孩在一起；
- 诸葛亮和周瑜在三国演义里的对手戏出现过几次；
- 西天取经路上前后有多少妖精要吃唐僧肉；
- 晁盖是什么时候起开始被宋江架空的……

# 第十一章　地图

学习 R 花的精力就是代价，就像买个商业软件所支付的美元或欧元一样。

— Felix Grant, November 2004

地图数据，经常用到的是地理信息系统 GIS 数据，很多人都用昂贵的 ArcGIS 软件来处理。其实，免费的 R 配上强大的扩展包，也能够处理很多 GIS 问题，有时甚至更灵活。这里, 我们举几个最简单的例子，来看看 R 是如何处理地图数据的。

## 11.1　绘制点阵地图

最简单的地图数据，莫过于用我们熟悉的数据框形式存储的经纬度坐标了。下面，我们用 R 来处理这样的点阵地图数据。

示例数据文件可以来我们的服务器下载。我们先用 `download.file()` 函数，将文件下载到本地电脑硬盘里。

```
download.file(url = "http://dapengde.com/r4rookies/us.csv",
              destfile = "c:/r4r/us.csv")
```

好了，已经下载到硬盘上了。请用 Excel 或记事本打开本地的这个文件，看看内容是什么。

这个数据框有 3 列，依次是经度、纬度、州名。州名包含了美国除夏威夷和阿拉斯加之外各州。

首先，我们读入数据，并看看总结报告：

```
us <- read.csv("c:/r4r/us.csv")
summary(us)
```

第三列是州名，我们可以用因子水平函数来看看有几个州：

```
nlevels(us$state)
```

```
## [1] 49
```

这样的数据，我们只要像前面接触的普通数据一样，绘制以经度为 $x$，纬度为 $y$ 的散点图，那么就会画出一张美国点阵地图。

```
plot(us$lon, us$lat)
```

用第三章学到的方法，我们可以给某个州指定个颜色，比如德克萨斯州：

```
plot(us$lon, us$lat)
points(us$lon[us$state == "Texas"],
       us$lat[us$state == "Texas"], col="blue")
```

类似地，可以给每个州涂上不同的彩虹色。

```
cols <- rainbow(nlevels(us$state))
plot(us$lon, us$lat, col = cols[us$state], pch = 20)
```

可想而知，如果我们有各州的某个统计数据，例如人口密度、气温、选票支持率、$PM_{2.5}$ 浓度等，以不同的颜色表示其大小，那么就都可以用类似的方法绘制在地图上。

利用某个城市的经纬度，就可以把城市在图上标出来。例如，我们用添加数据点的方法，把纽约 (经纬度 40.73, -74.02) 用三角形标出来。我们还可以用添加文字的方法，给每个州标上州名，标注的位置是各州图形的

中点：

```
lon.median <- tapply(us$lon, us$state, median)
lat.median <- tapply(us$lat, us$state, median)
text(labels = levels(us$state), x = lon.median, y = lat.median,
     cex = 0.5, col = 'White')
points(-74.02, 40.73, pch = 17, cex = 2)
```

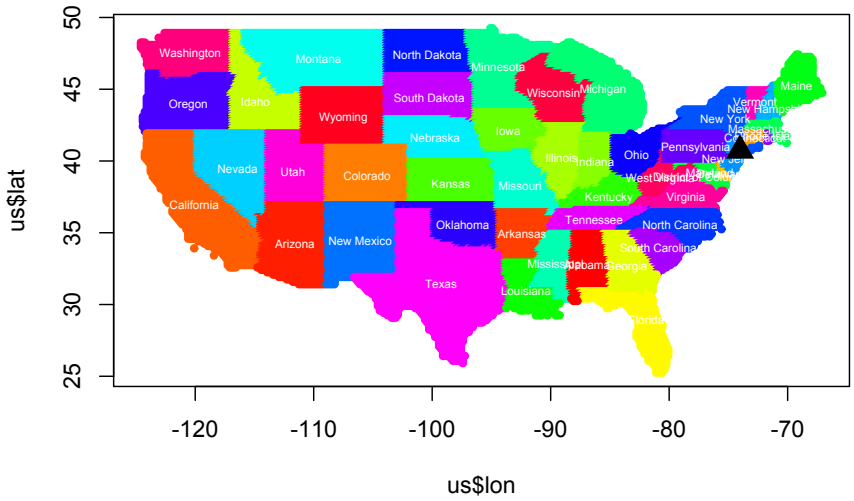



图 11.1: R 绘制点阵地图

看得出来，这些都是我们前面学过的基本作图方法，没什么新鲜的。

---

**练习 11.1.** 在美国地图上标出三个你感兴趣的城市的位置。

**练习 11.2.** 在美国地图上画出纬度线，间隔为 5 度。并找出北纬 35 度线穿过美国哪些州。

**练习 11.3.** 重画美国地图，用不同颜色来区分时区。美国跨过几个时区？

## 11.2 绘制矢量地图

矢量地图是平时更常见的地图文件。shape 格式的地图文件，是美国环境系统研究所公司（ESRI）开发的一种空间数据开放格式，是地理信息软件界的一个开放标准。

下面，我们用矢量地图文件，结合网上找到的各省空气质量数据，来绘制一张中国空气质量地图。

中国地图文件可以在网上下载[1]。为了方便，我们已经事先把找到的文件放到了我们自己的服务器上。所以，下面的代码，先从我们服务器下载一个名为 cm.zip 的压缩文件，然后用 `unzip()` 函数，将这个压缩包文件解压缩。

```
download.file(url = "http://dapengde.com/r4rookies/cm.zip",
              destfile = "c:/r4r/cm.zip")
unzip(zipfile = "c:/r4r/cm.zip", exdir = "c:/r4r")
```

得到的文件夹下应该有三个文件，即 bou2_4p.dbf，bou2_4p.shp，bou2_4p.shx。这就是我们的地图文件。读取这套文件，需要先安装和调用 rgdal 扩展包 (Bivand et al., 2016)。

```
install.packages('rgdal')
```

```
require(rgdal)
cm <- readOGR(dsn = "c:/r4r/cm", layer = "bou2_4p")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "c:/r4r/cm", layer: "bou2_4p"
## with 925 features
## It has 7 fields
## Integer64 fields read as strings:  BOU2_4M_ BOU2_4M_ID
```

`readOGR()` 函数用来读取地图文件。这个函数的自变量格式比较特别，

[1]矢量地图下载：http://www.diva-gis.org/gdata

dsn 是数据源名称 data source name 的缩写，需要指定为地图文件所在文件夹路径，并且末尾不要斜线符号；layer 需要指定为地图文件名，但不要扩展名。

读入数据后，就可以直接画地图了：

```
plot(cm)
```

---

**思考 11.1.** 矢量地图跟点阵地图相比，有什么优势？

下面，我们按照空气污染程度，给各省图上不同的颜色。先看看有哪些省。

```
summary(cm)
is.factor(cm$NAME)
levels(cm$NAME)  # 省市名称。
```

cm$NAME 里存放的是各省名称，是个因子。我们只需把各省的空气质量以不同颜色或灰度表示，然后跟省名称的因子联系起来就可以了。

各省的空气质量，我们可以从中国空气质量在线监测分析平台[2]得到，这里以 2016 年 12 月 30 日 14 时的空气质量指数 (AQI) 为例，当时的 AQI 数据可以来我们的服务器下载：

```
download.file(url =
                "http://dapengde.com/r4rookies/aqi.csv",
              destfile = "c:/r4r/aqi.csv")
```

```
aqi <- read.csv("c:/r4r/aqi.csv")
aqi
```

```
##    aqi         province
## 1   77           福建省
## 2   90           甘肃省
```

[2]中国空气质量在线监测分析平台：https://www.aqistudy.cn/

```
## 3  190              北京市
## 4  157              安徽省
## 5   79              吉林省
## 6   88              江苏省
## 7   90              江西省
## 8   61              广东省
## 9   81    广西壮族自治区
## 10  72              贵州省
## 11  98              辽宁省
## 12 129    宁夏回族自治区
## 13  71      内蒙古自治区
## 14  54              青海省
## 15 163 新疆维吾尔自治区
## 16 112              山东省
## 17 172              山西省
## 18 112              四川省
## 19 160              陕西省
## 20  39              上海市
## 21  -1              台湾省
## 22  45        西藏自治区
## 23  -1    香港特别行政区
## 24 227              天津市
## 25  46              海南省
## 26  93              浙江省
## 27  45              云南省
## 28 244              河北省
## 29 226              河南省
## 30  80              重庆市
## 31 158              湖北省
## 32 119              湖南省
## 33  47            黑龙江省
```

-1 表示无数据。我们按照空气质量的分级标准，用 cut() 函数来计算

各省空气属于哪一级（0 – 50 优，50 –100 良，100 – 150 轻污，150 – 200 中污，200 – 300 重污，300 – 500 严重）：

```
aqstandard <- c(0, 50, 100, 150, 200, 300, 500, Inf)
aqilevel <- cut(aqi$aqi[match(levels(cm$NAME),aqi$province)],
                aqstandard)
```

我们用 match() 函数调整了一下顺序，以便跟每个省区一一对应。

根据级别的多少，来确定分几个灰度显示，然后把灰度设为各省名称的因子，就可以作图了：

```
mycol <- grey(seq(1, 0,
                  length.out = nlevels(aqilevel)))[aqilevel]
col <- cm$NAME
levels(col) <- mycol
```

```
plot(cm, col = as.character(factor(col)), axes = TRUE)
```

最后，我们用 plotrix 扩展包 (Lemon, 2006) 的 color.legend() 函数，给我们的地图添加个漂亮的图例：

```
install.packages('plotrix')
```

```
library(plotrix)
legendn <- character((length(aqstandard) - 2) * 2 + 1)
legendn[seq(2, length(legendn), by = 2)] <-
  aqstandard[2:(length(aqstandard)-1) ]
color.legend(
  xl = 135, yb = 10, xr = 137, yt = 30,legend = legendn,
  rect.col = grey(seq(1, 0, length.out = nlevels(aqilevel))),
  align = "lb", gradient = "y")  # 添加图例。
color.legend(
  xl = 135, yb = 10, xr = 137, yt = 30,
  legend = c('Excellent','Good','Lightly', 'Moderately',
             'Heavily', 'Severely', 'OMG'),
```

```
  rect.col = grey(seq(1, 0, length.out = nlevels(aqilevel))),
  align = "rb", gradient = "y")  # 添加图例。
```

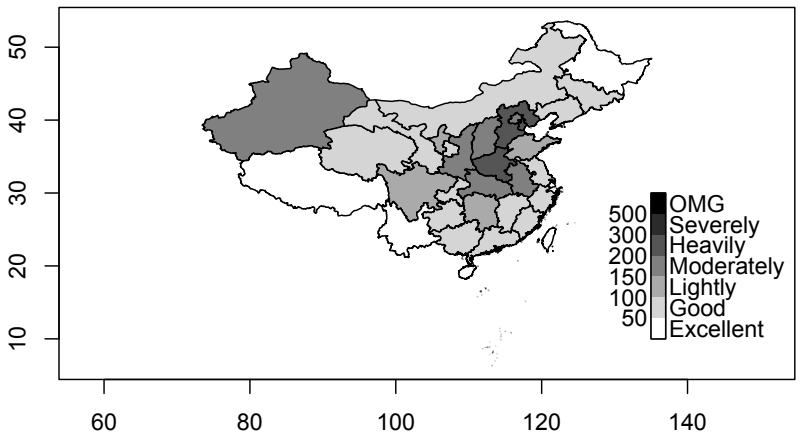



图 11.2: R 绘制矢量地图

---

**练习 11.4.** 示例的地图用的是灰度，有的省份的大气污染级别从图例上不太容易区分，请将上图的颜色改为容易区分的彩色，重新绘制。

**练习 11.5.** 从网上下载世界地图并画出来，给不同的国家地区涂上颜色。

## 11.3 绘制交互地图

交互地图，就是像谷歌、百度、高德那种鼠标点击可以放大缩小的地图，看起来很炫，料想大约会更难画出来吧？其实不然，因为我们有现成的扩展包——leafletCN (Lang, 2017)。如果事先安装好这个扩展包，并且按

要求准备好数据，那么做出交互式地图只需一条指令，简单得让人不画张地图都不好意思。

我们先来安装和调用 leafletCN 扩展包，然后将上一节使用的数据框 aqi 稍微转化一下格式：

```
install.packages('leafletCN')
require(leafletCN)
aqi$aqi[aqi$aqi == -1] <- NA
pvs <- regionNames("china")
loc <- match(pvs, aqi$province)
aqi2 <- data.frame(name = pvs, value = aqi$aqi[loc])
```

aqi2 数据框就是作图要用的数据，含有两列：名为 name 的一列是全国省区的名称，取值来自 leafletCN 扩展包的 `regionNames()` 函数；名为 value 的一列是各省区的 AQI 数值，来自先前用用过的 aqi 数据框，我们用 `match()` 函数调整了一下顺序，以便跟 name 的每个省区一一对应。`regionNames()` 的具体用法，请咨询 F1 小助理。

好了，万事俱备，只差一句话：

```
geojsonMap(dat = aqi2, mapName = "china",
           popup =  paste(aqi2$name, aqi2$value),
           legendTitle = "AQI")
```

`geojsonMap()` 函数里，我们设置了 `popup` 参数，这样，鼠标放在地图上点击省区，就会弹出 AQI 数值。

怎么样，是不是很炫？

---

**练习 11.6.** 从网上下载全国各市的空气质量数据，并在交互地图上用不同颜色标示。

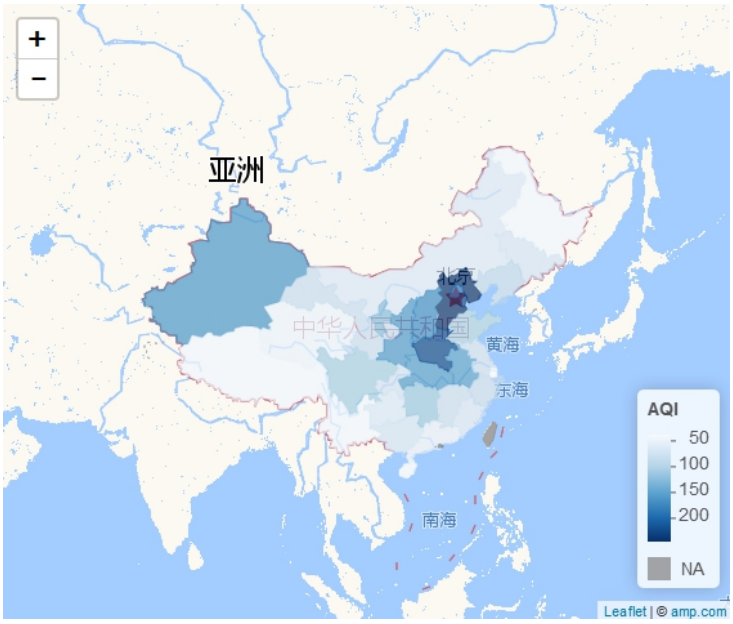



图 11.3: 交互地图示例

## 11.4 课外活动：绘制动画地图

在本章，我们绘制了一张中国大气质量地图，输入的数据是一组 AQI 数值。如果我们有多组数据，每个小时的数据做一张地图，有一年的数据，那么该怎么做？如果能做出来的话，连续播放，不就是动态显示我国空气质量的时空分布了吗？试试看。其中涉及的函数，我们都已经学过了。

提示：

1. 将各省的 AQI 数据保存到一个数据框里，用循环语句每次读取其中的一组来作图。
2. 图片保存为文件时，用字符串的处理方法为不同的文件取不同的名称。
3. 可以用自定义函数的方法，将绘图过程写成一个函数，方便每次调用。

# 第十二章　时间

我看到了我的爱恋 我飞到她的身边
我捧出给她的礼物 那是一小块凝固的时间
时间上有美丽的条纹 摸出来像浅海的泥一样柔软
她把时间涂满全身 然后拉起我飞向存在的边缘

—刘慈欣《三体 3：死神永生》

科幻小说《三体》三部曲，描绘了时间这个物理量的奥妙。在第六章对复活节的计算里，我们已经领教了时间数据的厉害。时间数据非常特殊又十分有趣，主要是因为时间单位不是十进制的：满 60 秒才进 1 分钟，满 24 小时进 1 日，满 7 日进 1 周，而考虑到闰年和闰秒等因素，日、周、月、年之间的进位就更加复杂，再加上时区和夏时制，处理起时间数据来花费的精力折算成金钱的话，真是一寸光阴一寸金。

举个例子就能切身体会到时间的复杂：你一定知道自己的生日几月几日，但大概并不知道自己出生那天是星期几吧？从出生那一刻到今天为止，已经过去了多少天？这辈子能遇见几次在星期天过生日？

回答这些问题，我们的 R 语言可以轻松办到。R 对时间数据的复杂性有充分的准备，让这件工作变得不是那么困难。

为了不引起歧义，本书中，我们将时间点（比如 ‘‘现在几点几分了’’）称为 ‘‘时刻’’，而时间长度（比如 ‘‘还差几分钟下班’’）称为 ‘‘时间’’。

## 12.1 时刻数据的获取

假定你出生的时刻是 1994 年 9 月 22 日 20 时 30 分 0 秒。我们先把你的生日告诉 R，让 R 听得懂。

一般来说，我们从数据文件中读取数据表时，其中的时刻数据列往往是“1994-09-22 20:30:00”这样的格式，读入之后，这一列自动按照字符格式存储。

```
bd <- '1994-09-22 20:30:00'
```

我们知道，字符串是不可以做数学的加减法的，比如我们想把这一列时刻都加 20 分钟，字符串是无法做到的，这就需要先把字符串转换成 R 能识别的时刻格式。

```
bdtime <- strptime(x = bd, format = '%Y-%m-%d %H:%M:%S',
                   tz = "Asia/Shanghai")
bdtime
```

```
## [1] "1994-09-22 20:30:00 CST"
```

为什么 `strptime()` 函数里需要个 `format` 和 `tz` 参数呢？因为如果不说一声的话，R 就不知道这个字符串里哪个是年哪个是月，也不知道是哪个时区。

我们发现，bdtime 的值依然带有引号，除了多了个时区代码外，看上去跟 bd 一模一样。这难道不是跟 bd 一样的字符串吗？不是的。

```
class(bd)
```

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

```
class(bdtime)
```

```
## [1] "POSIXlt" "POSIXt"
```

他们属于不同的类。bd 属于字符串（character），而 bdtime 属于一种称为 POSIXct 的类。

除了从外界数据文件读取时刻数据外，R 还可以获取计算机时钟的时刻。下面三个函数，都可以获得 R 运行环境中的当前时刻或日期。

```
t1 <- Sys.time() # 返回当前时刻
t2 <- date() # 返回当前时刻
t3 <- Sys.Date() # 返回当前日期
t1
```

```
## [1] "2017-11-22 10:25:36 CET"
```

```
t2
```

```
## [1] "Wed Nov 22 10:25:36 2017"
```

```
t3
```

```
## [1] "2017-11-22"
```

如果查看一下这三个变量的类，会发现各有不同：

```
class(t1)
```

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

```
class(t2)
```

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

```
class(t3)
```

```
## [1] "Date"
```

这些类有什么区别和联系？请自行放狗去搜。

一切准备工作就绪，R 现在知道了你的生日，就可以进行后续计算了。

## 12.2 时刻数据的格式

女孩和男孩陷入爱河。后来战争即将爆发，兵荒马乱之中，男孩把写有日期的纸条塞进女孩手心，约定在战争结束那一年的那一天老地方见。如果谁没有赴约，就说明已遭遇不幸。

战争结束后，女孩拿着纸条如约到达，苦苦等了一个月，却不见恋人踪影，悲愤之下自尽身亡。不久男孩赶到，看到的却是昔日恋人的坟墓，悲痛之下，也殉情而死。

纸条上的日期是“12.10”。女孩是德国人，以为是 10 月 12 日；男孩来自美国，以为是 12 月 10 日。

这个爱情悲剧告诉我们：处理时间数据，格式很重要。

— 我听来的故事

先看看你出生那天是星期几。很简单，只需这样操作：

```
bdtime$wday
```

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

是个星期四，对吗？

`$wday` 表示从时刻数据中提取星期几。类似的，`$hour`, `$min`, `$sec` 分别用来提取时分秒，`$year`, `$mon`, `$mday` 提取年月日，`$yday` 提取一年内的第几天。

---

**练习 12.1.** 今天是今年的第几天？

**练习 12.2.** 算算你出生那天是当年的第几天。

时刻数据的全部组成信息，可以用 `unclass()` 函数来查询：

```
unlist(unclass(bdtime))
```

```
##    sec    min   hour   mday    mon   year   wday
##    "0"   "30"   "20"   "22"    "8"   "94"    "4"
```

```
##   yday   isdst    zone gmtoff
##  "264"     "0"   "CST"      NA
```

为了方便显示，我们调用了 unlist() 函数。

由于世界各地风俗习惯不同，采取的时刻格式也各种各样。比如在表述日期时，放在最前面的，我们中国习惯先说年，美国习惯先说月，而德国习惯先说日。年月日的分隔符也有不同。R 提供了 format() 函数，来对这些格式进行随意转换。

```
format(bdtime, format = '%d.%m.%Y')
```

```
## [1] "22.09.1994"
```

百分号用来提示后面的格式是什么：%d 表示日，%m 表示月，%Y 表示四位年份。他们用圆点 . 分隔。小贴士 12.1 列出了更多格式代表的含义。有了他们，就可以对时刻数据进行任意转换了。

---

**练习 12.3.** 请把你的生日格式转换成包含日期时刻星期几的全称。

## 12.3　时刻数据的计算

时刻数据是可以参加加减法计算和逻辑运算的。看看你的生日加上 1 是什么：

```
bdtime + 1
```

```
## [1] "1994-09-22 20:30:01 CST"
```

bdtime + 1 得到的是 bdtime 之后 1 秒的时刻。可以预料，+ 60 就是 1 分钟后，+ 3600 就是 1 小时后，而 + 86400 就是 1 天后。

```
bdtime + 60
```

```
## [1] "1994-09-22 20:31:00 CST"
```

**小贴士 12.1.** 常用时刻格式说明。

| X | 中文示例 | 英文示例 | 解释 |
|---|---|---|---|
| %a | 周日 | Sun | 星期几的缩写 |
| %A | 星期日 | Sunday | 星期几的全拼 |
| %b | 十月 | Oct | 月份缩写 |
| %B | 十月 | October | 月份全称 |
| %c | 周日十月 1 | Sun Oct 1 | 日期时刻全称 |
| | 10:30:59 2017 | 10:30:59 2017 | |
| %C | 20 | 20 | 世纪 |
| %d | 1 | 1 | 日（0 补位） |
| %D | 10/1/2017 | 10/1/2017 | 日/月/年 |
| %F | 2017-1-10 | 2017-1-10 | 年-月-日 |
| %H | 10 | 10 | 时（00-23） |
| %I | 10 | 10 | 时（00-12） |
| %j | 274 | 274 | 一年的第几天 |
| %m | 10 | 10 | 月 |
| %M | 30 | 30 | 分 |
| %p | 上午 | AM | 上下午 |
| %r | 10:30:59 上午 | 10:30:59 AM | 12 小时制时刻 |
| %R | 10:30 | 10:30 | 24 小时制时刻 |
| %S | 59 | 59 | 秒 |
| %T | 10:30:59 | 10:30:59 | 时刻 |
| %u | 7 | 7 | 星期几（周日为 7） |
| %U | 40 | 40 | 年中第几周 |
| %V | 39 | 39 | 年中第几周（ISO8601） |
| %w | 0 | 0 | 星期几（周日为 0） |
| %y | 17 | 17 | 年份后两位 |
| %Y | 2017 | 2017 | 年份四位 |
| %Z | GMT | GMT | 时区 |

```
bdtime + 3600
```

```
## [1] "1994-09-22 21:30:00 CST"
```

```
t3 <- bdtime + 86400
t3
```

```
## [1] "1994-09-23 20:30:00 CST"
```

果然如此。R 把我们加上的秒数按照复杂的时间进位方式自动处理了。当然，时刻也可以减去一个数，让时光倒流。

两个时刻之间可以相减，得到两个时刻之间的时间长度。假如我的生日是 1995 年 9 月 1 日 7 时 30 分，那么你我生日之间相差的时间就是

```
bdtime2 <- strptime(
  '1995-09-01 7:30', format = '%Y-%m-%d %H:%M',
  tz = 'Asia/Shanghai')
bdtime2 - bdtime
```

```
## Time difference of 343.4583 days
```

还记得我们有个 `diff()` 函数来计算两者之差么？类似地，时刻数据处理中有个 `difftime()` 函数，效果跟上面的减法大体相同，不同之处是可以指定输出数值的单位：

```
difftime(time1 = bdtime2, time2 = bdtime, units = 'secs')
```

```
## Time difference of 29674800 secs
```

我们还可以计算距离你我生日相等长度的那一天：

```
mean(c(bdtime, bdtime2))
```

```
## [1] "1995-03-13 07:00:00 CET"
```

除了本文介绍的函数外，我们还可以根据需要选用扩展包，例如 timeDate (Team et al., 2015)，可以让我们对时间数据的处理能力如虎添翼。

**思考 12.1.** 既然两个时刻可以相减，那么两个时刻能否相加？为什么？

**练习 12.4.** 算算到今天为止，你已经来到这个世界多少天？多少小时？多少秒？

**练习 12.5.** 算算你从 0 岁到 100 岁，每年的生日都是星期几。作图展示，横轴是年份或你的年龄，纵轴是星期几。数数有几个生日是在星期天。

## 12.4 课外活动：夏令时

如果你只处理中国的时刻数据，那么是比较幸运的。如果遇到欧美国家的时刻数据，就不得不考虑一个特殊的问题：夏令时。

夏令时，或称夏时制，正规的名称是“日光节约时制”。欧洲和美国很多地区，夏季天亮得太早，于是就硬生生把时钟调快一小时，大家早起早睡，据说可以充分利用白天的光照而省电。但是，夏令时的实际收效至今仍有争议，很多人认为靠照明省下那点电微不足道，而粗暴地调时钟把大家的作息时间弄得很乱，尤其对老人和孩子影响很大，列车、航班等统统要做调整。我国其实在 1986 年到 1991 年实行了六年夏时制，实在得不偿失，就取消了。

图 12.1 是用 R 做出的德国拜罗伊特市日出日落时刻图，虚线和实线分别显示的是不使用和使用夏时制的日出日落时刻。很容易看出夏时制的好处：夏天把一小时的时间往后挪了之后，不至于天亮之后过很久才需要上班，并且下班后距离天黑还有很长时间，可以让人去泡吧，泡网，泡……不挪到下午的话，这一个小时在早上只够泡个澡。

R 在处理时刻数据的时候充分考虑了夏时制。时刻数据变量名，后面接上 `$isdst` 就表示“操作中是否是光节约时制”（逻辑值，1 表示 `TRUE`，0 表示 `FALSE`）

我学习 R 语言时正生活在德国，遇到了一件趣事，至今记忆犹新。德国的夏令时是这样调整时钟的：每年 3 月的最后一个星期天，凌晨 2 点钟

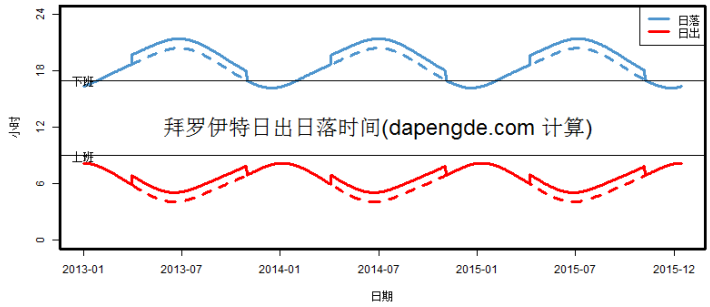



图 12.1: 德国拜罗伊特日出日落时刻

把时钟拨到 3 点。夏令时结束时，在 10 月的最后一个星期天凌晨再拨回来。那么，在 R 里是怎么处理这种复杂的情况呢？

我们以 2018 年为例，3 月最后一个星期天是 25 日，由于凌晨 2 点后会直接跳至 3 点，我们看看 R 的世界里这段时间发生了什么：

```
(a <- strptime("2018-03-25 03:00:00",
               format = "%Y-%m-%d %H:%M:%S", tz = "CET"))
```

```
## [1] "2018-03-25 03:00:00 CEST"
```

注意，我们明明设定的时区是 CET 中欧时间（Central European Time），但 R 自动转换成了 CEST 中欧夏季时间。

照理说，3 点钟的前一秒应该是 2:59:59，我们来试试：

```
a - 1
```

```
## [1] "2018-03-25 01:59:59 CET"
```

竟然是 1:59:59！2 点钟就这么凭空消失了么？

我不甘心，于是试了试另外一个函数，同样可以给时刻变量赋值：

```
(a <- as.POSIXlt("2018-03-25 03:00:00", tz = "CET"))
```

```
## [1] "2018-03-25 03:00:00 CEST"
```

```
a - 1
```

```
## [1] "2018-03-25 01:59:59 CET"
```

时区符号仍然自动改变了。

如果我硬来，非要找回 2 点钟呢？

```
(a <- as.POSIXlt("2018-03-25 02:00:00", tz = "CET"))
```

```
## [1] "2018-03-25 CET"
```

R 索性把时刻给丢了，仅保留了日期。这是在惩罚我的任性么？

换回 `strptime()` 函数试试？

```
(a <- strptime("2018-03-25 02:00:00",
               format = "%Y-%m-%d %H:%M:%S", tz = "CET"))
```

```
## [1] "2018-03-25 02:00:00"
```

欧叶！但是……

```
a - 1
```

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

啊哦……

接受这个现实吧：在中欧，2018 年 3 月 25 日 2:00:00 到 2:59:59 这段时间是不存在的！若要在数据处理中避免这种情况发生，还是强行把时区都设置成 GMT 保险。为了保险起见，我每次从别人那里拿到数据都会问：你这数据里的时间，用没用夏令时？

然而，这也暗示男孩女孩们可以用一种新方法对别人的表白说 no：那我们就在中欧时间 2018 年 3 月 25 日凌晨 2:30 见吧……

# 第十三章　批量处理文件

这就是 R。没有做不到，只有想不到。

— Simon Blomberg, April 2005

相信你和我一样，电脑里保存了大量的文件吧。比如数据，有时候需要从多个文件里提取有用的信息，手动查找和摘选十分耗时耗力；比如照片，动辄成百上千张，虽然有些软件可以自动整理，但毕竟难以随心所欲。

我们前边学了编程的结构和字符串处理，如果把这些结合起来，再学几个文件操作函数，就可以高效地对电脑文件为所欲为了。

## 13.1　批量整理照片文件

事实上，我们从第二章起就已经悄悄开始接触文件操作函数了。当时我们用 `dir.create()` 创建了个文件夹，还用 `file.choose()` 来选取文件。后者有几个双胞胎姐妹：`choose.files()` 函数可以用来选取多个文件，而 `choose.dir()` 可以用来选取文件夹。请复习一下这几个函数，热热身。

下面，我们运行下面的代码，会从我们服务器下载一个压缩文件，并解压缩到指定文件夹。打开这个文件夹，你会看到 3 个图片文件。我们假定这是我们手机里拍的照片，我们的目标是修改这 3 个文件的名称为拍摄的时刻。一旦能实现这个目标，那么对成千上万个文件重新命名都将不费吹灰之力。

```
download.file(url= "http://dapengde.com/r4rookies/figren.zip",
              destfile = "c:/r4r/figren.zip")
unzip(zipfile = "c:/r4r/figren.zip", exdir = "c:/r4r")
```

首先，我们使用 dir() 函数，可以获取文件夹里的文件列表。如果将参数指定为完整文件名，那么得到的是文件的完整路径。

```
fotodir <- 'c:/r4r/figren'
fotofilefull <- dir(fotodir, full.names = TRUE)
fotofile <- dir(fotodir)
```

然后，使用 file.info() 函数，可以获取文件的完整信息：

```
fotoinfo <- file.info(fotofilefull)
fotoinfo
```

```
##                             size isdir mode
## c:/r4r/figren/IMG_689.jpg  1716 FALSE  666
## c:/r4r/figren/IMG_690.jpg 56321 FALSE  666
## c:/r4r/figren/IMG_691.jpg   307 FALSE  666
##                                         mtime
## c:/r4r/figren/IMG_689.jpg 2017-06-29 22:02:44
## c:/r4r/figren/IMG_690.jpg 2017-06-29 22:02:44
## c:/r4r/figren/IMG_691.jpg 2017-06-29 22:02:44
##                                         ctime
## c:/r4r/figren/IMG_689.jpg 2017-06-29 22:02:44
## c:/r4r/figren/IMG_690.jpg 2017-06-29 22:02:44
## c:/r4r/figren/IMG_691.jpg 2017-06-29 22:02:44
##                                         atime exe
## c:/r4r/figren/IMG_689.jpg 2017-06-29 22:02:44  no
## c:/r4r/figren/IMG_690.jpg 2017-06-29 22:02:44  no
## c:/r4r/figren/IMG_691.jpg 2017-06-29 22:02:44  no
```

其中的 mtime 列，是我们需要的文件创建时刻，小时分秒之间使用冒号分隔。由于操作系统不允许文件名中含有冒号，我们利用前面学过的时

刻格式处理函数，稍微调整一下格式，设定新的文件名：

```
fototime <- format(fotoinfo$mtime, '%Y-%m-%d-%H%M%S')
newname <- paste(
  fotodir, '/', fototime, '_', fotofile, sep = '')
```

最后，我们用文件重命名函数 `file.rename()` 函数，就可以对文件批量重命名了：

```
file.rename(fotofilefull, newname)
```

如果有成千上万张照片需要这样整理，R 是极佳的选择。

我们还可以将照片按年份或月份归类整理到文件夹里。该怎么操作呢？下面就会讲到。

---

**思考 13.1.** 除了照片文件，平时还有什么其他文件需要类似的方法整理吗？

**练习 13.1.** 将你自己的一批照片文件重新命名，在文件名里插入日期信息，并按默认排序在文件名里增加从 1 开始的编号。

## 13.2　从网页批量下载和整理图片

我所在单位有个网站，有个页面展示着研究组野外观测照片缩略图，并按站点分了类[1]。新来到这里时，我想把本组的工作了解一番，在看图的时候，需要逐个点开才能看大图上的细节，将来需要某张图时，找起来很不方便，我就萌生了把图片全部下载到本地并按站点分类保存的想法。但是这几百张图，一一点开下载也太累了。如何批量下载网页上的图片呢？

方案有很多，列举如下：

- 可以通过浏览器的“保存网页全部内容”来实现，本地生成一个文件夹，包含了网页上的图片。这个我试了，但只保存下来了缩略图，没

[1] 因斯布鲁克大学生态气象研究组：http://www.biomet.co.at/pictures/

有大图。

- 可以安装迅雷、快车之类的软件，但是我不想装。有些软件臃肿庞大也就算了，关键是不知道他们在背后悄悄做了些什么。另外，他们无法解决图片分类保存的问题。
- 傲游、360 等浏览器有批量下载功能，或者 firefox+BatchDownload 插件也行，但我不想装，并且他们也无法解决图片分类的问题。
- chrome 浏览器有个 fatkun 插件，专门用来批量下载图片，能下载大图，是我最满意的方案了，但也并非完美。下载的图片文件名要么是原始文件名，要么只能简单编号。这样一来，所有观测站点的图片都混在了一起，这仍然不是我想要的。我希望下载到本地的图片能自动按观测站分类保存。

其实，查看一下网页的源代码（chrome 浏览器里按快捷键 ctrl+u），发现每张图片所属站点的信息，包含在了图片的链接里。比如 Neustift 观测站某图的链接是:

```
http://...gallery/neustift/img_8260_59_58_....jpg
```

这个链接里是含有站名信息的（neustift）。这就好办了，可以自己动手用 R 代码实现。

首先，我们把网页的源代码读进 R，跟学习字符函数时读取千字文文件的方法一样：

```
urlink <- 'http://www.biomet.co.at/pictures/'
aa <- readLines(urlink, encoding = 'UTF-8') # 读取网页
```

然后，我们找到有图片链接的行，也就是含有下面字符串的行：

```
src="http://www.biomet.co.at/wp/wp-content/gallery
```

获取这些行的方法是我们学过的 `grep()` 函数：

```
linkformat <-
  'src="http://www.biomet.co.at/wp/wp-content/gallery'
```

```
bb <- aa[grep(linkformat, aa)]
```

---

**思考 13.2.** 有没有别的更方便的办法，来获取一个网页上的超级链接呢？

接着，我们用循环函数，对得到的每一行进行处理，把图片的链接提取出来，并且去掉重复的图片链接：

```
for(i in 1:length(bb))
  bb[i] <- substring(
    bb[i],
    regexpr("http", bb[i])[1],
    regexpr(".jpg\"", bb[i])[1]+3) # 获取链接
bb <- unique(bb)
length(bb)
writeLines(bb, 'c:/r4r/links.txt')
```

每个链接里，从第 47 个字符开始就是观测站名了。为了简便，我们截取站名的前 4 个字母，并以此为名称，用 `dir.create()` 函数新建一批文件夹：

```
stname <- substring(bb, 47, 50)
stname <- stname[-which(stname == '')]
for(i in unique(stname))
  dir.create(paste('c:/r4r/', i, sep = ''))
```

快看看新的空文件夹是不是已经在那里了？

一切准备妥当了。下面，我们就用 `download.file()` 将图片下载保存到对应的文件夹里。由于文件多，可以预料，整个过程可能会比较耗时，所以我们用第 5.6 节的方法，在循环中添加了一个 `print()` 函数来提醒我们下载进度，并在循环结束后用 `winDialog()` 函数来弹出一个任务完成的提示框。

```
for(i in 1:length(bb)) {
  download.file(
    url = bb[i],
    destfile = paste(
      'c:/r4r/', stname[i],'/', stname[i], i, '.jpg',
      sep = ""),
    method = 'curl', quiet = TRUE)
  print(paste(i, 'of', length(bb), 'downloaded.'))
}
winDialog(type = c("ok"), message = ' 下载完毕！ ')
```

## 13.3 从大量文件里提取汇总信息

有时候，我们需要从大量的数据文件中，提取感兴趣的条目并汇总到一起进行分析。这里我们举个例子。

我国有数千个气象观测站。每隔一段时间，各气象站就把当地的气象要素观测数据上传到国家一级的服务器，在信息中心汇总成一个文件，每个观测站的数据占一行，这个汇总的文件就有几千行。如果要从中获取某个观测站气象要素的时间序列，就需要从每个这样的文件里找到来自该观测站的那一行，附加上时刻信息，合并为该观测站的一个数据文件进行后续分析。

这样的文件可以从我们的服务器下载。为了方便，我们把原来的几千行删减为几十行，并且仅给出 6 个文件供示范。

我们先把这样文件的一个压缩包下载到本地电脑，并解压缩为文件夹。

```
download.file(url = "http://dapengde.com/r4rookies/obs.zip",
              destfile = "c:/r4r/obs.zip")
unzip(zipfile = "c:/r4r/obs.zip", exdir = "c:/r4r")
```

请用记事本打开任意一个文件。可以看出，每个文件的前两行是文件

头。从第三行起是个数据表，第一列是观测站的编号，从第二列起是各种观测要素。时刻信息既包含在文件头里，也包含在文件头里。现在，假定我们要从这 6 个文件里提取编号为 54527 观测站的数据。

像前面的例子那样，我们先获取文件列表，把文件名作为时刻信息存储。

```
stn <- 54527
obsdir <- 'c:/r4r/obs'
obsfilefull <- dir(obsdir, full.names = TRUE)
obstime <- as.numeric(dir(obsdir))
```

然后，我们先准备好一个名为 output 的空变量，用来存放输出结果。我们使用循环语句，逐个读取每个文件，从中找到目标行，并将其作为新行追加到 output 后面，将时刻信息作为这个新行的行名称。

```
output <- NULL
for(k in 1:length(obsfilefull))
{
  input <- read.table(obsfilefull[k], header = FALSE,
                      skip = 2, sep="")
  output_new <- input[which(input[, 1] == stn),]
   if(nrow(output_new) != 0)
     rownames(output_new) <- obstime[k]
  output <- rbind(output, output_new)
}
output$time <- rownames(output)
```

得到的 output 数据框，就是目标观测站的时间序列。

---

**练习 13.2.** 从上述气象观测站的数据中，筛选出所有编号以 50 开头的观测站，并提取这些观测站的第 4 列气象要素，连同时刻信息一起合并在一个文件里。时刻信息从文件头获取。

**练习 13.3.** 将练习 13.2 得到的数据，对每个观测站该项气象要素做时间序

列图，并做出每个观测站该气象要素的箱式图（boxplot）。

**小贴士 13.1.** 常用文件操作函数

| 名称 | 作用 |
|---|---|
| `file.show()`, `file.info()` | 查看文件内容或信息 |
| `file.exists()` | 检查文件是否存在 |
| `file.create()`, `dir.create()` | 新建文件或目录 |
| `file.copy()` | 文件复制 |
| `file.remove()` | 文件删除（注意！不进入回收站!） |
| `file.rename()` | 文件重命名 |
| `download.file()` | 下载文件 |
| `unzip()` | 解压缩文件 |
| `dir()` | 查看目录下文件清单 |
| `file.choose()`, `choose.files()` | 选取文件或目录 |
| `choose.dir()` | 选取文件夹 |

## 13.4 课外活动：打通任督二脉

如果你熟悉 Windows 操作系统，那么应该知道 cmd，也就是命令行。同时按下键盘的 Windows 键和 r 字母键，在弹出的小窗口里输入 cmd，回车，出现的那个简陋的黑色小窗口就是 cmd。在开始菜单里搜索 cmd 也能搜到。别看它又黑又丑，它的强大会让你惊叹。如果 R 和 cmd 强强联手，你的电脑就被打通了任督二脉，离练成绝世神功已经不远了。下面我们举几个例子。

让我们先打开任脉。请在 cmd 小黑窗里输入：

```
notepad
```

并回车，记事本就被打开了。这就是用 cmd 打开记事本的指令。

如果不使用 cmd 小黑窗，在 R 里也可以进行等同的操作，只需使用 `shell()` 函数：

```
shell('notepad')
```

shell() 就是 R 用来呼唤 cmd 的方式。R 可以呼唤电脑里已经安装的软件，例如网页浏览器：

```
shell('start iexplore http://xuer.pzhao.net')
```

或者打开 qq：

```
shell('cmd /c "D:/Program Files/Tencent/QQProtect.exe"')
```

当然，你得把上面这条代码里 QQProtect.exe 的完整路径改成你自己电脑上的路径才行。

如果一段 R 代码在办公室处理大量数据尚未完成，而我又着急下班，那么我可以事先在 R 代码的最后加上一条打开 qq 的指令，就可以回家了。当我在手机上看到办公室电脑的 qq 上线了，就意味着 R 已经把数据处理完了。如果配合 AutoHotKey 这样的软件，我甚至可以让办公室的 qq 自动发一条“搞定！”的信息给自己的手机[2]。

cmd 支持的命令非常丰富，可以完成很多原本复杂的工作。R 调用 cmd，就是强强联手。如果感兴趣，可以去学一下批处理脚本，把 cmd 要执行的多条命令写在.bat 文件里，让 R 来调用，那必将是一片繁华。

现在，R 呼唤 cmd 的任脉已经打通了，下面我们打通督脉——让 cmd 呼唤 R。

假定你有一批打算要自动运行的 R 代码，保存在文件`c:/r4r/timer.r`，并且假定你的 R 安装路径是 `D:/Program Files/R/bin/R.exe`，那么，请在 cmd 小黑窗里运行：

```
"D:/Program Files/R/bin/R.exe" CMD BATCH --vanilla --slave
  "c:/r4r/timer.r"
```

[2]事实上，R 有专门的扩展包，可以自动发送电子邮件，比调用 qq 更爽快。

这条命令的含义是由 cmd 呼唤 R 来运行 `timer.r` 里的代码。一声召唤后，`timer.r` 里的代码全部自动运行，该画的图自动画，该写出的数据自动写，根本不用打开 RStudio。

这有什么用呢？说说我是怎么用的吧。

有段时间，我参加了连续一个多月的野外科研观测，需要把这一个月的天气状况记录在案，每个小时是阴是晴，是雨是雪。天气状况其实在很多天气网站上都有，但是只能实时浏览，网页每天更新，并且不能下载最新一个月的历史记录。这种事当然可以安排个人每天手动把信息摘抄整理下来，但是我有 R 在手啊！于是，我写了一段 R 代码，可以获取天气网站当天显示的天气现象并保存到一个文件里，并做出需要的图表。然后，我又写了一句 cmd 代码保存在脚本文件.bat 里，用来呼唤上述 R 代码文件。最后，在 Windows 的 ‘‘计划任务’’ 里设置每天运行一次.bat 文件。好了，一切每天都自动完成了。

我们在本章学了用 R 整理照片、下载图片和处理大量数据文件；在下一章，我们还将学会批量制作 Word 文档和 PPT 幻灯片。如果这些任务需要定期大量重复操作，那么，cmd+R+ 计划任务的组合，将节省大量的人工劳动。

这只是我作为一个菜鸟的想法。反正，任督二脉已经打通。在这个神奇世界里，发挥你的想象力，想怎么折腾就怎么折腾咯！能不能成为绝顶高手，就看你自己咯！

# 第十四章　论文书稿写作

用嘴说出的话随风而散，用笔写出的话永不磨灭。

— 莫言

自古皆死，不朽者文。

— 宋之问

如果你是个科研工作者，那么多半每天都要跟期刊论文、学位论文打交道；如果你不是科研工作者，有时也需要处理一些文字，例如写日记、写信、做个日历、画个思维导图等；喜欢文艺的朋友可能会写个诗集、小说、散文集，记录个乐谱等等。R 语言提供的 bookdown 和 bookdownplus 等扩展包，为文字工作者和爱好者提供了一个友好的平台，可以将这些各类文字轻松地转化成文档，格式丰富，样式美观。

## 14.1　魅力不可挡

bookdown (Xie, 2016a) 是 R 语言的一个扩展包，一个用来写书、写文章的平台。bookdown 初版发布于 2016 年，原本为撰写 R 语言类书籍而生，非常适合撰写学位论文、学术期刊论文，也可以用来写文学类作品。使用 bookdown，可以方便地在文章里插入目录、图表、交叉引用、脚注、索引，以及公式、参考文献、R 代码，是撰写可重复性报告的不二之选。它可以同时生成漂亮的 pdf、Word 和网页文件，并免费发布到官方网站上，得到

的文档比 Word 更美观，比传统的 LaTeX 更易用。bookdown 发音为 “不可挡”，包如其名，bookdown 的魅力的确不可阻挡。有了它，我再也不愿意用 Word 和 LaTeX 了。

bookdown 并非完美无瑕。对于 R 初学者来说，如果身边没有人指导，bookdown 上手并不是很容易。酒香巷深，虽然官方手册[1]提供了完整而详细的资料可以参考，但使用者可能仍然需要花精力应付 LaTeX，YAML，Pandoc，而很多初学者可能从来没听说过这些，站在 bookdown 的门口往里张望了一下，说不定就被吓跑了，从而错过了门里的精彩世界。所以，我们推荐 bookdownplus 扩展包 (Zhao, 2017b)，可以让新手对 bookdown 快速上手。bookdownplus 已在 CRAN 正式发布。

bookdownplus 是对 bookdown 的增强和简化, 是进入 bookdown 世界的捷径。它提供了很多有用的模板，可以很方便地在 bookdown 平台写期刊论文、学位论文、学术海报、化学分子式、信件、日记、日历、诗集、吉他谱等各种常用文档和书籍（图 14.1）。这是功能上的 “+”。bookdownplus 使用时只需指定一个模板，给定作者和书名，就可以一键生成模板文件，用户在模板文件里照猫画虎写自己的文字就可以了，不必再花力气上网找模板、设置 YAML 和 LaTeX。这是操作上的 “−”。正是这 “一加一减”，大大降低了 bookdown 的使用门槛，让每个初学者都可以方便地使用。

bookdownplus 各个模板的使用方法详见官方文档 R bookdownplus Textbook[2]一书。这本电子书本身就是用 bookdownplus 生成的，尤其是它的 pdf 版本很美观。此书的源码开放，可以作为使用 bookdown 写书的示例（图 14.2）。

有了 bookdownplus，从此，bookdown 这坛好酒就不必去深巷苦寻，它就放在你手边。

[1]bookdown 官方手册：https://bookdown.org/yihui/bookdown/

[2]bookdownplus 官方手册：https://bookdown.org/baydap/bookdownplus

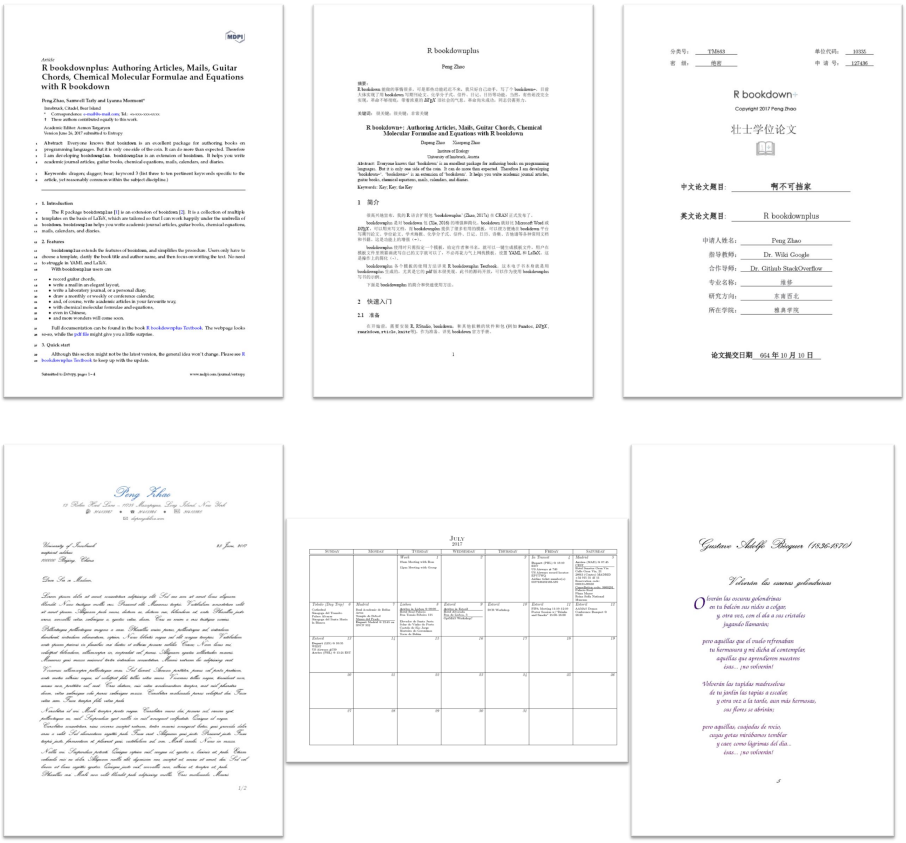



图 14.1: bookdownplus 生成的英文、中文学术论文、学位论文、信件、日历、诗集文档

## 14.2　准备工作

bookdown 环境的准备工作略显繁琐，这是因为它调用了目前最优秀的几个现成软件。首先，需要下载和安装免费软件 LaTeX[3]和 Pandoc[4]。有的安装文件比较庞大，下载和安装时需要一点耐心，但是绝对值得，磨刀不误砍柴工嘛。

[3] $CTEX$: `http://www.ctex.org/CTeXDownload`

[4] Pandoc: `http://pandoc.org/installing.html`

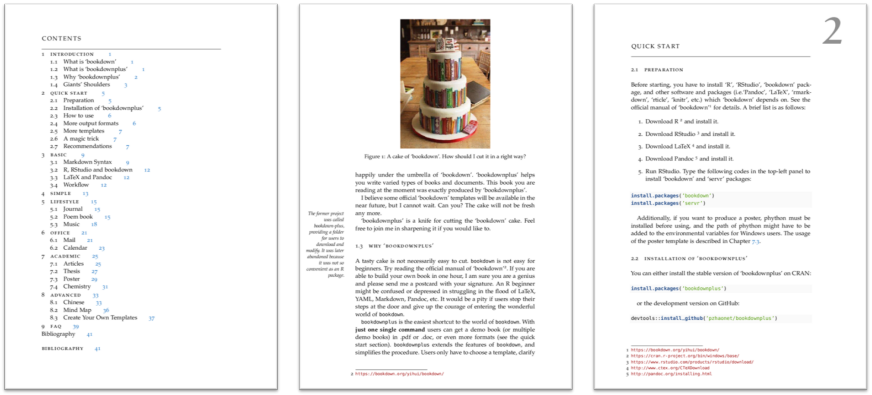


图 14.2: R bookdownplus Textbook 示例书页

其次，bookdown 的背景平台是 R 语言，用户界面首选 RStudio，R 和 RStudio 的下载安装方法见第 1.1 节。

运行 RStudio，在左上面板的代码窗口输入并运行以下代码，来安装和调用 bookdownplus 扩展包 (bookdown 扩展包会随带自动安装)：

```
install.packages("bookdownplus")
```

```
require(bookdownplus)
```

好了，需要的工具已经安装完毕。现在，检查你的 R 语言工作目录 (`getwd()`)，确保这个文件夹是空的。`bookdownplus` 将在目录里产生很多临时文件，所以我们建议使用空文件夹作为工作目录。如果懒得更改工作目录，也可以在 RStudio 里创建一个新项目 (File – New Project - New Directory – Empty Project)，以后每次总是在这个项目里工作就可以了。

## 14.3 见证奇迹的时刻

准备工作只需做一次，一劳永逸。一切准备就绪后，接下来就是见证奇迹的时刻。

从版本 V1.2.0 开始，bookdownplus 提供了两种魔法。第一种是用一个循环条指令来自动生成若干格式不同、风格迥异的文档，让初学者迅速体验 bookdown 的工作方式和成果。运行下面的代码，然后去喝个茶：

```
for(i in template()[1:19])
  bookdownplus(template = i,
               more_output = more_output()[1:3])
```

回来以后，你会在工作目录里看到很多文件和文件夹。打开那个叫做 _book 的文件夹，Duang！19 个示例文档，每个都以 pdf、Word、网页和电子书格式乖乖地等在那里，等你打开了。其中，电子书 epub 格式可以在手机上阅读，也可以在电脑里用免费的 Calibre 软件打开阅读。这些示例文档，从学术论文到诗歌乐谱，从中文到英文，展示了 bookdown 能胜任的工作。是不是动心了？

第二个魔法，是使用嵌套循环，来生成其中一个模板 “mail” 不同样式的示例文档。运行下面的代码，然后看两眼窗外的风景，bookdownplus 就把事情做完了：

```
for(mf in mail_font()) {
  for(ms in mail_style()) {
    for(mt in mail_theme()) {
      outputname <- paste('mail', ms, mf, mt, sep = '_')
      bookdownplus(template = 'mail',
                   mail_style = ms,
                   mail_font = mf,
                   mail_theme = mt,
                   output_name = outputname)
    }
  }
}
```

仍然打开那个叫做 _book 的文件夹，Duang！不同字体、不同主题、不同布局的信件示例文档就全部在那里了。我们在申请国外的学习和工作岗

位时经常要求提供 cover letter，这就是 “mail” 模板的用武之地。

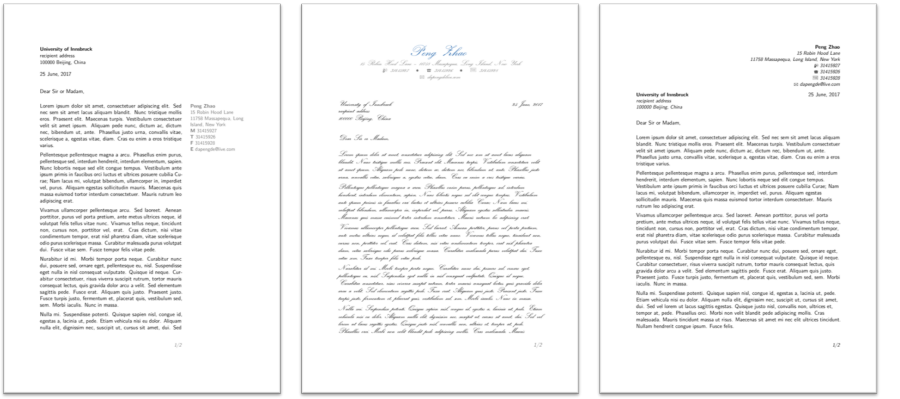


图 14.3: bookdownplus 的 mail 模板文档示例

下面我们用几个例子，来解释一下 bookdownplus 的基本用法，同时理解一下上面两个魔法是如何实现的。

## 14.4 撰写学术论文

### 示范文档

写学术期刊论文是 bookdown 的强项，也是 bookdownlus 扩展包开发的初衷。事实上，学术论文是 bookdown 里最为动人的精华所在。只有在写学术论文的过程里，遇到输入上下标、脚注、图表、方程、交叉引用、参考文献等任务时，bookdown 的魅力才会充分展示出来。本节介绍的学术论文生成、修改、导出、发布方法同样适用于后续章节介绍的其他模板，不再赘述。

图 14.1 里已经展示了两份期刊论文的示例文档。下面的代码，可以生成一个双栏的英文期刊论文文档（图 14.4左）：

```
bookdownplus(template = 'article')
```

打开 _book 文件夹，示范文档 article.pdf 已经在那里了。随带附送的 article.tex 可以直接投稿给学术期刊，一般的国际期刊都接收这个格式的文稿。

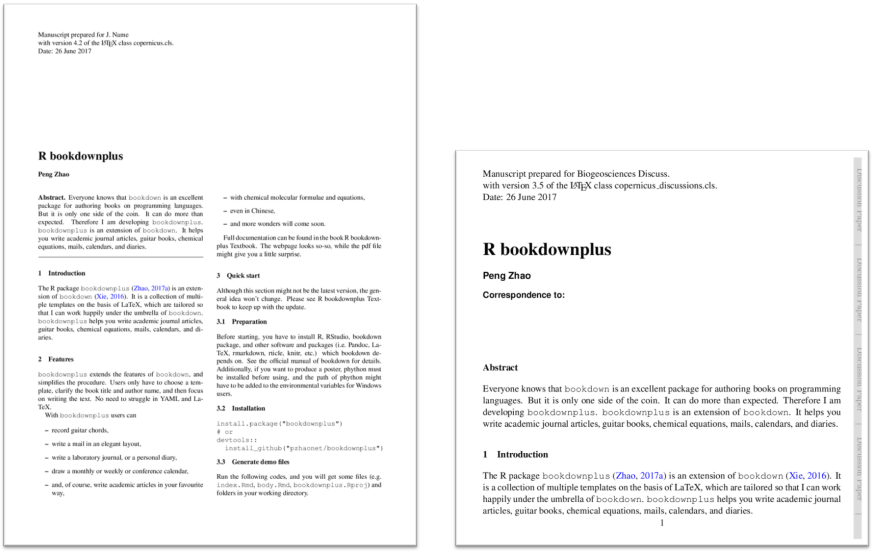



图 14.4: bookdownplus 生成的双栏论文（article，左）和讨论论文（discussion，右）示范文件

下面，我们进一步解释一下上面这条代码的含义和用法。

### 选择模板

参数 `template` 用来指定模板名称。可用的模板名称可以用 F1 魔法调出 `bookdownplus()` 函数的帮助文件来查看。如果你懒得用键盘输入模板名称 "article"，那么可以换用下面的代码：

```
bookdownplus(template = template()[1])
```

`template()[1]` 就是 `template()` 里的第一个元素，而这里出现的函数 `template()` 是用来列出所有的可用模板名称的，以便选择：

```
template()
```

```
##  [1] "article"          "article_mdpi"
##  [3] "article_zh"       "calendar"
##  [5] "chemistry"        "chemistry_zh"
##  [7] "discussion"       "guitar"
##  [9] "journal"          "mail"
## [11] "nte_zh"           "poem"
## [13] "thesis_classic"   "thesis_mypku_zh"
## [15] "thesis_ubt"       "thesis_zju_zh"
## [17] "yihui_demo"       "yihui_mini"
## [19] "yihui_zh"         "poster"
```

如果换成 “discussion” 模板，就得到图 14.4 右图的格式。

#### 标题和作者

文章的标题和作者，有两种方式来设定。第一种是利用 `author` 和 `title` 参数来指定：

```
bookdownplus(template = 'article',
             author = 'John Smith',
             title = 'My article')
```

第二种是打开 index.Rmd 文件，将开头的 `title` 和 `author` 字段改成你文章的信息即可。

#### 正文内容

文章的正文内容在 body.Rmd 里修改即可。将 body.Rmd 与 pdf 示范文档对照一下，就可以快速掌握 markdown 格式标记的用法。本书的附录给出了格式标记的速查表。如果这些还不够，那么就去读 bookdown 的官方手册[5]吧，里面介绍得最为齐全详尽。

修改成你自己的文字后，就可以生成自己的文档了。只需双击

[5]bookdown 官方手册：`https://bookdown.org/yihui/bookdown/`

bookdownplus.Rproj 用 RStudio 打开这个项目，按快捷键 'ctrl+shift+b'，Duang！你的书就在'_book' 里出现了。

你自己的 body.Rmd 文件可以备份到别处。每次运行 `bookdownplus()` 创建新示范文件后，用你的'body.Rmd' 文件将模板的同名文件覆盖，就可以直接生成新书稿了。

### 导出与发布

bookdownplus 可以很方便地将你写好的文字导出为多种格式。除了默认的.pdf 和.tex 格式外，支持导出的格式有：

```
more_output()
```

```
## [1] "word_document2" "html_document2" "epub_book"
## [4] "gitbook"
```

我们在生成示例文档时，可以用 `more_output` 参数指定导出的格式：

```
bookdownplus(template = 'article',
             more_output = more_output())
```

这样，在你完成 body.Rmd 的编写后，按 ctrl+shift+b 就会生成这些格式的书稿。Word 是我国广大人民群众喜闻乐见的文档格式。epub 格式意味着你可以在手机上阅读论文了。gitbook 文档可以用网页浏览器打开。这些文档都可以免费发布到网上，只需在 bookdown.org 注册并登录后，在 RStudio 运行命令：

```
bookdown::publish_book()
```

就可以把当前项目的作品发布在 bookdown 的官网上了。浏览器会自动打开你的专属网页。快把地址分享给亲朋好友来围观吧。

### 更多模板

这里示范的 article 和 discussion 模板，来自 Copernicus Publications 出版社的 LaTeX 模板[6]，经修改后收进了 bookdownplus 的模板库里。事

[6]Copernicus Publications 的论文模板：`http://publications.copernicus.org/for_authors`

实上，很多国际期刊都提供了各自的 LaTeX 模板，按照 R bookdownplus Textbook 给出的步骤，稍微修改一下就可以纳入这个模板库。比如图 14.1 中展示的论文模板就是来自 MDPI[7]。如果你创建了新的 bookdownplus 模板，或者发现了喜欢的 LaTeX 期刊论文模板，可以提交到 bookdownplus 的项目主页[8]，以便纳入模板库里，让 bookdownplus 越来越强大。下面介绍的几个模板，就是这样得到的。

## 14.5 撰写学位论文

bookdownplus 的模板库里有几个专门用来写学位论文的模板。从这些模板出发，可以非常方便地撰写学位论文。由于数据计算过程的 R 代码可以嵌入到文档中，一旦发现错误，只需在文档中修改即可。我在写科研论文时曾遇到一件事：论文有个计算公式，其中一个数值不小心敲错了。后续计算都是建立在这个计算的基础上，所以后续计算全错。虽然不影响结论，但严格来讲，论文里所有的数据都得改。这种灾难往往让人郁闷和沮丧，幸亏我用了 bookdown：由于正文里的数据和图表大多数是调用 R 代码计算得到的，走的是“可重复性研究”的思路，只要把 R 代码里那个敲错的数据改过来，重新编译论文文档，后续计算就全部自动更新，导出的 pdf 或 Word 文档也相应自动更新，节省了大量时间，还极大减少了敲数字或拷贝粘贴出错的可能。

运行下面的代码，可以生成一个英文的博士论文示例文档：

```
bookdownplus(template = 'thesis_ubt')
```

这个模板来自德国拜罗伊特大学博士论文。模板名称里的 ubt 是拜罗伊特大学（University of Bayreuth）的缩写。thesis_ubt 模板适合使用英语或德语撰写的学位论文。

此外，bookdownplus 为中文用户提供了支持中文的学位论文模板。运行下面的代码：

[7]MDPI: http://www.mdpi.com/

[8]bookdownplus 项目主页：https://github.com/pzhaonet/bookdownplus

```
bookdownplus(template = 'thesis_mypku')
```

将得到一个简单的支持中文环境的示范论文文档。

另一个支持中文的学位论文模板是 'thesis_zju'，来自浙江大学（Zhejiang University）的 LaTeX 模板。网上也有其他大学的 LaTeX 论文模板，例如北京大学[9] 和清华大学[10]，如果你能把任何一个改成 bookdownplus 模板，那么可以提交到 bookdownplus 的模板库里。

## 14.6　生成思维导图

由于 bookdown 生成的文档基于 markdown 语法，采取的是标记语言，那么就很容易从文档中提取指定标记的字段，来进行进一步加工。比如说，将 bookdown 或 markdown 文稿自动生成思维导图。

思维导图又称为脑图、头脑风暴图、心智导图，是时下流行的一种记忆知识、交流想法的工具。一般情况下，我们在 bookdown 文档里用 # 标识来标记章节标题，# 的数量表示标题的级别。因此，只需把文稿里 # 后面的内容提取出来，就可以自动生成文稿的大纲；格式稍作转换，就可以自动生成思维导图了。

图 14.5 是用 mindr (Zhao, 2017c) 扩展包从本书的原始书稿自动生成的思维导图。这里简要介绍一下 mindr 的用法。

mindr 包已在 CRAN 正式发布，安装和载入方法跟别的扩展包一样：

```
install.packages('mindr')
library('mindr')
```

在 R 的工作目录（getwd()）创建名叫 mm/ 的文件夹，丢进去一个或多个 markdown 文件 (.md, .Rmd)，比如可以把 bookdownplus 生成的 body.Rmd 示例文件拷贝进去，然后运行：

[9] 北京大学论文模板：https://github.com/CasperVector/pkuthss

[10] 清华大学论文模板：https://github.com/xueruini/thuthesis

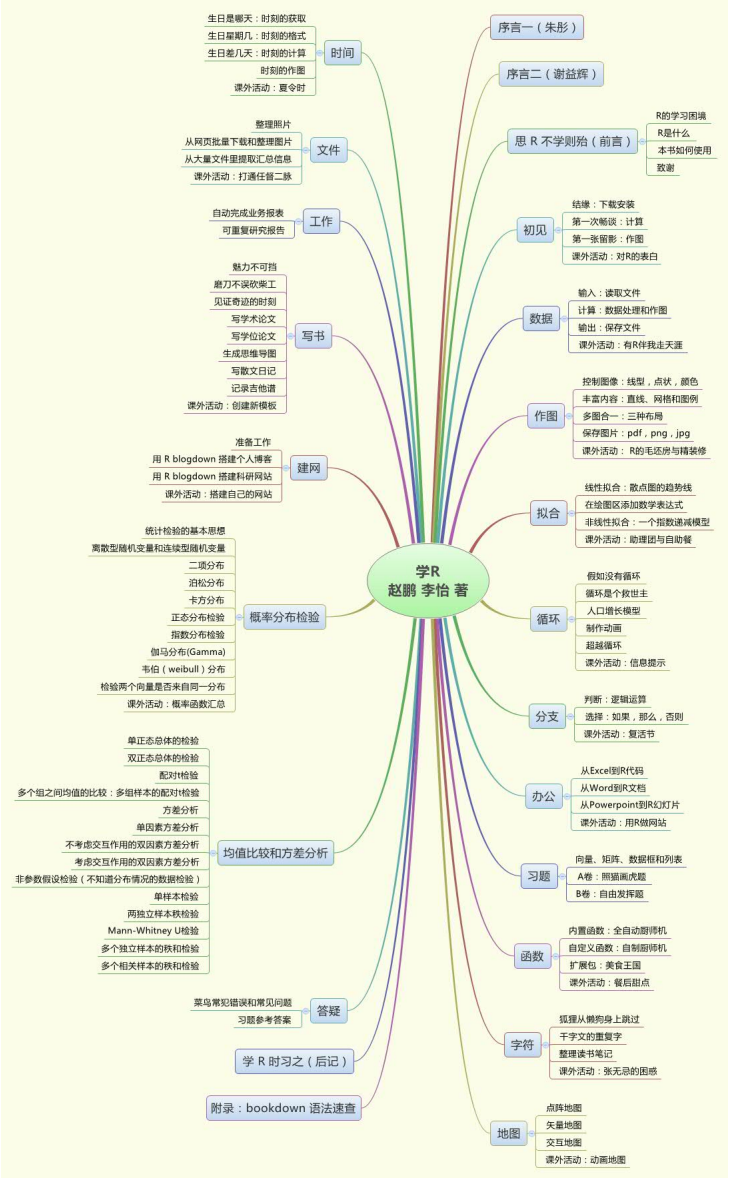



图 14.5: 《学 R》一书的思维导图, 由 mindr 扩展包制作

```
md2mm()
```

就会在工作目录下得到一个名为 mm.mm 的思维导图文件。这个文件可以用大多数主流的思维导图软件打开。这里我们推荐 FreeMind[11]，因为.mm 格式是 FreeMind 可以直接打开的默认格式。FreeMind 的界面比较简陋，所以我们另外推荐 Xmind[12]，安装运行后在菜单栏选择导入 freemind 导图即可。漂亮的思维导图就自动出现了。

如果你并不想从 bookdown 或 markdown 文件里生成思维导图，而只想获取文档的大纲，那么可以用 mindr 的 outline() 函数；如果你是思维导图的老用户，想从你现有的思维导图作为出发点，进一步展开写文档，那么可以用 mm2md() 函数，会将.mm 格式的思维导图转换成 markdown，然后就可以用 bookdown 进一步生成各种需要的文档了。

---

**练习 14.1.** 利用 bookdownplus 的 article 模板，写一篇学术论文，并利用 mindr 生成该论文的思维导图。

**练习 14.2.** 利用 bookdownplus 的 mail 模板，写一封发给某国外大学的留学申请信（cover letter）。

## 14.7　撰写散文和日记

用 bookdown 来写散文、日记、小说和诗集等文学作品，是一种特别的享受。文学作品跟学术论文的结构在逻辑上是不同的，常常不需要标题分级编号，常常不需要强制每章第一页出现在奇数页，有时候需要特殊字体来点缀。bookdownplus 里的几个模板，把这些问题都考虑进去了，用户不再需要为此烦心。

中文的散文和小说，可以用 'nte_zh' 模板：

[11] FreeMind：http://freemind.sourceforge.net/wiki/index.php/Download

[12] Xmind http://www.xmind.net/download/win/

```
bookdownplus(template = `nte_zh`)
```

这个模板特别为中文订制，自动生成双栏的目录，并选择了适合中文的字体。如果你喜欢记录一些生活里的小故事，小想法，不妨用这个模板发布到 bookdown.org 上。我就是这样把博客里的育儿帖子整理成了一本书（图 14.6），发布在网上跟亲朋好友分享[13]。


图 14.6: bookdownplus 的散文小说模板

‘nte_zh’ 模板当然也可以用来写日记。不过，bookdownplus 另外提供了 ‘journal’ 模板：

```
bookdownplus(template = 'journal')
```

‘journal’ 模板适合外语日记。该模板最初是设计用来做实验室日志记录的。由于 bookdown 允许很方便地在文档里插入 R 代码来生成可重复性的数据图表，因此 ‘journal’ 模板先天具有写科研日记的优势。而且，该模

[13]爸爸三定律：https://bookdown.org/baydap/papasdiary/

板为生成的 pdf 文档留有较宽的左右页边距，方便将来打印装订后添加一些注释或说明。

不要小看这个页边距。四百年前，法国数学家费马在阅读丢番图《算术》一书时，突然有了小灵感，但一时找不到本子做记录，于是就在书的页边写道:

> 将一个立方数分成两个立方数之和，或一个四次幂分成两个四次幂之和，或者一般地将一个高于二次的幂分成两个同次幂之和，这是不可能的。关于此，我确信我发现了一种美妙的证法，可惜这里的空白处太小，写不下。

此后，全人类花了三个世纪的努力，才把费马猜想证明为费马定理。要是丢番图的书用了 bookdownplus 的 'journal' 模板，为费马留出足够的页边距空白，那么会省下人类多少宝贵的时间啊！

## 14.8　记录吉他谱

bookdown 不仅可以用于学习、工作和生活记录，还可以用来记录音乐，例如吉他谱。

我国的民谣吉他伴奏谱常见的一般是六线和弦全谱，然而录入太麻烦，经常需要专业软件。为了省事儿，经常只保留和弦名称和歌词，简化为文字谱：

```
  C     Em        F        G       C     Em
让我掉下眼泪的不只昨夜的酒，让我依依不舍的不……
```

文字谱的好处是用不着任何专业软件，录入方便。但是，这比较坑吉他初学者。想不起来 `Em` 和弦的指法该怎么办？`Em` 还好办，看见个 `G#7sus4`，我崩溃了，自认水平低，乖乖翻和弦字典去。

能不能像录入文字谱那样简单地录入和弦谱呢？

这不是异想天开。在 RStudio 里输入以下代码，就会生成一本吉他和弦示例书：

```
bookdownplus(template = 'guitar')
```

这是怎么实现的呢？

打开 'body.Rmd' 文件，输入：

```
让\CM 我掉下\Em 眼泪的　不\F 只昨夜的\GM 酒　让\CM 我依依\Em 不
……
```

得到的 pdf 文档里呈现的就是图 14.7。

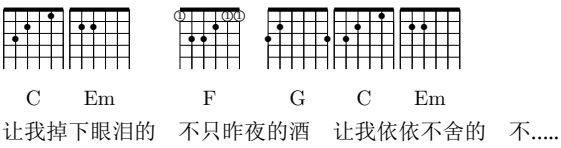



图 14.7: bookdownplus 的 guitar 模板生成的和弦谱

这里，'CM' 和 'GM' 两个符号分别表示 C 和弦和 G 和弦，跟习惯稍有不同。这两个符号当然是可以更改的，只需在 `tex/template_guitar.tex` 定义即可。例如 C 和弦，定义方式是：

```
\newcommand{\CM}{\upchord{\chord{t}{n,p3,p2,n,p1,n}{C}}}
```

这样，在正文里，bookdown 看到`\CM`，就会转换成 C 和弦指法图。限于篇幅，关于上面这条指令的具体含义我们就不详细解释了，请参阅 LaTeX 的 gchords 扩展包[14]。

吉他和弦谱有很多，我们没必要自己费力气逐个创建对应的指令，只需去互联网找现成的就行了，有来自开源社区贡献的常见指法设置代码，我们只需拷贝粘贴到这里即可。

利用 bookdownplus 的 'guitar' 模板，我把喜欢的歌制作成了一本电子书放在网上作为示例[15]，并且把用到的和弦指法表汇总在了书里（图 14.8）。

[14] gchords 包：`http://kasper.phi-sci.com/gchords/`

[15] GuITaR bookdown：`https://bookdown.org/baydap/bdguitar/bdguitar.pdf`

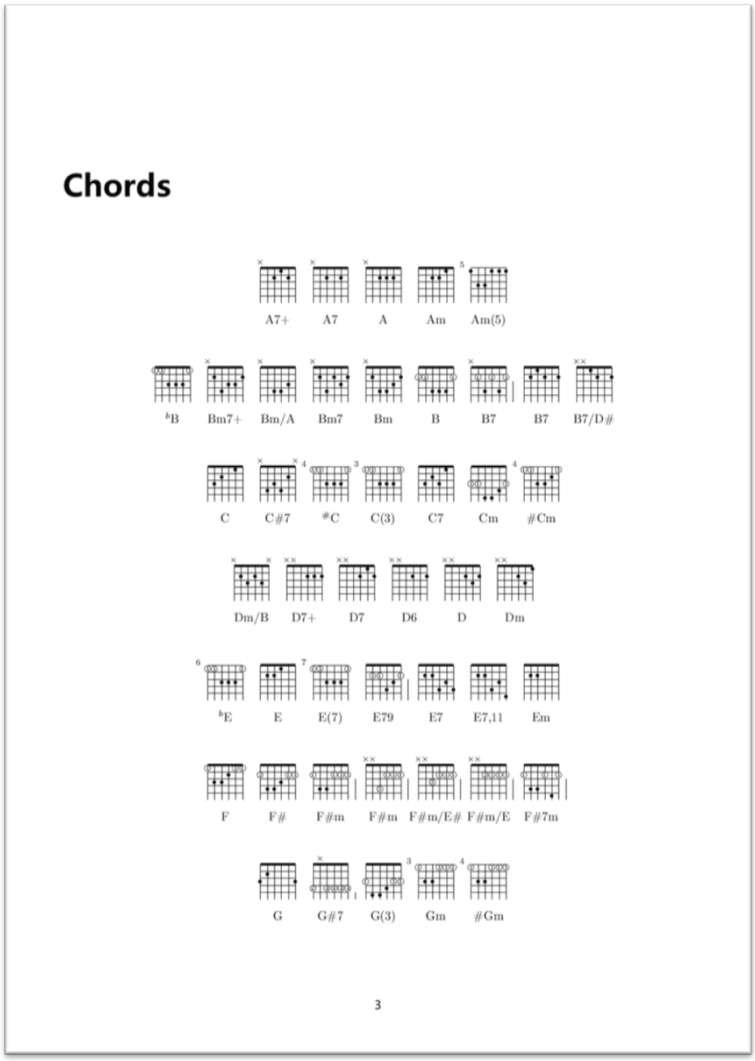



图 14.8: Guitar bookdown 一书汇总的吉他和弦指法图

开一下脑洞，我们可以利用 bookdown 把吉他和弦谱插入到文学作品中。我们甚至可以在某本书里同时呈现 R 代码、R 图表、诗词散文和吉他谱，开创出一种全新的文体。

## 14.9 课外活动：创建新模板

在本章中，我们鼓励大家创建自己喜欢的 `bookdownplus` 模板，那么，一个 bookdownplus 模板是如何创建的呢？

截至本书成稿时，bookdown 1.2.0 的模板库包含大约 20 个用于生成 pdf 文档的模板。创建 bookdownplus 模板库并不需要太多的 LaTeX 知识，只需了解 LaTeX 工作的逻辑和基本流程即可。仔细研究一下现有的模板，认真研读 bookdown 官方文档[16]的第四章，那么你就知道如何创建自己的模板了。下面是基本流程和我的一些体会。

1. 首先，找到一个新模板。有很多网站提供免费的 LaTeX 模板。下载一个你最喜欢的，尤其推荐那些有详细的说明文档和示例文档的。

2. 在本地电脑上对模板的示例文档用 LaTeX 进行编译，确保顺利生成一个你想要的 pdf 文档。关于编译工具，我们推荐免费好用的 TeXStudio[17]。如果编译成功，并且你懒得进行下一步操作的话，你可以把相关的文档通过邮件打包提交到 bookdownplus 的项目主页[18]，并说明这个模板有何优点和特色。然而，靠人不如靠自己，我们建议你勇敢地进行下一步。

3. 将你的 LaTeX 模板拆成两个文件：template_yours.tex 和 index.Rmd。拆分原则如下：

- 将 LaTeX 模板的文章主题部分，也就是位于 `\begin{document}` 和 `\end{document}` 之间的部分，用符号 `$body$` 替代，保存为

[16]bookdown 官方文档：`https://bookdown.org/yihui/bookdown/`

[17]TeXStudio：`http://www.texstudio.org/`

[18]bookdownplus 项目主页：`https://github.com/pzhao/bookdownplus`

template_yours.tex。`$body$` 所在的部分将会用 body.Rmd 里的内容来填充。

- 创建一个新的 index.Rmd。可以使用 bookdownplus 生成的任何一个 index.Rmd，将其中的模板字段换成 template_yours.tex。

4. 如果你的 LaTeX 模板足够简单，那么现在就可以试着用 bookdown 来编译了。运气好的话你会得到想要的 pdf 文档。祝贺！

5. 但是，多数情况下并不能编译成功。一个漂亮的 LaTeX 模板经常比较复杂，尤其是中文模板。你可能需要修改导言区，挑出一部分来另存为其他文件，然后在 index.Rmd 里标明。详见 bookdown 官方手册[19]。

6. 努力不懈，坚持到底，相信你最终必然会得到你想要的 pdf 文档。

你的新模板，包括 index.Rmd, body.Rmd, template_yours.tex 和其他相关文件和说明，可以提交到 bookdownplus 项目的模板库。

让我们联合起来，一起建立全人类第一个 bookdown 模板库吧！

[19]bookdown 官方手册：`https://bookdown.org/yihui/bookdown/yaml-options.html`

# 第十五章　搭建小型网站

你见，或者不见我
我就在那里
不悲 不喜
你念，或者不念我
情就在那里
不来 不去

—扎西拉姆 • 多多

互联网时代有个自己的网站，就好像真实世界里有了自己的房子（服务器）和住址（域名）。如果把自己的简历放在网站上，不仅方便在求学求职的过程中展示个人风采，同时也是结交朋友的好方式。不管我本人身在何处，你见还是不见，我的网站就在那里，不悲不喜。

以前，建立一个网站并不容易，涉及诸多领域的知识和技能，门槛比较高。好在，我们有万能的 R 语言，做个网站就像印个名片一样稀松平常，搭建网站只需几分钟，而并不需要多少网络知识。你甚至不用懂 R 语言是什么，照本章去做就行了。

我们已经知道，R 可以用来生成 html 的网页文件（见第七章和第十四章）。那么，只要有服务器空间，把 R 制作的单个静态网页上传到空间即可。服务器空间有免费的也有收费的，后者很便宜，每年只需百十块钱。这是最简单的方案。如果你的网页不止一个文件，那么本章介绍的就是更强大的网站搭建方案。

本章提供两个示范，演示如何使用 R 语言的 blogdown 扩展包 (Xie, 2017)，用来搭建一个个人博客[1] 和一个科研小组网站[2]。截至本文成稿之日，blogdown 含苞待放，尚未正式发布，但是已经能够正常安装使用了。我们来提前一睹她的芳容。

## 15.1 准备工作

R blogdown 是 R 语言的一个扩展包，是个用来制作网站的工具，可以用非常简洁的方式快速搭建静态网页构成的网站。虽然包名里有“博客”（blog）字样，但并不仅限于博客。由于 blogdown 基于 markdown 扩展语法，可以在网页中方便地插入图表、脚注、数学公式、R 代码、公式、参考文献等等元素，所以，非常适合搭建一个科研小组的展示窗口。blogdown 搭建的网站页面元素跟第十四章介绍的 bookdown 是共通的，因此，可以非常方便地把 bookdown 书写的论文、书籍、信件、散文等转化成网页展示，易于维护、迁移和备份。

本书的官方网站[3] 就是用 blogdown 搭建的。网站里给大家提供的 R 示例代码、试读章节、勘误表等，都是直接从书稿中拷贝粘贴到文本文件里，然后运行一下 blogdown 的指令，网站就自动更新了。

### 安装 blogdown

blogdown 的背景平台是 R 语言，用户界面首选 RStudio ，产生的网站框架是 Hugo ，掰手指头数一数，要安装 4 样东西，略显繁琐，但并不难，一劳永逸。相信你已经安装好了 R 和 RStudio。运行 RStudio，在左上面板的代码窗口输入并运行以下代码：

```
if(!require(devtools)) install.packages('devtools')
devtools::install_github('rstudio/blogdown')
```

由于 blogdown 目前只发布到了 GitHub 服务器上，尚未发布到 CRA

[1] blogdown 个人博客示例：http://dapengde.com/blogdown_demo_default

[2] blogdown 科研网站示例：http://dapengde.com/blogdown_demo_academic

[3] 本书官网：http://xuer.pzhao.org

N，所以上面第一条代码先安装开发工具 devtools 包，第二行用这个包的 install_github() 函数从 GitHub 安装。

安装好 blogdown 之后，就可以安装 Hugo 框架了。在 RStudio 左上面板的代码窗口输入并运行以下代码：

```
blogdown::install_hugo()
```

安装完毕。

## 15.2　搭建个人博客

### 创建示例网站

上文介绍的 blogdown 个人博客的示例[4]，我们看看这个博客是如何创建的。

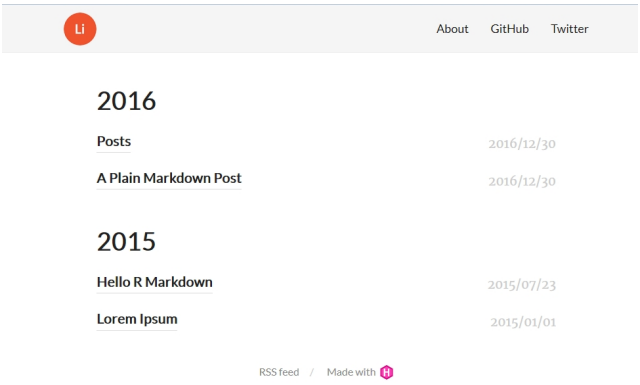



图 15.1: blogdown 搭建的 Hugo 个人网站示例

首先，在你的电脑里新建一个文件夹，准备存放网站文件，假定是 c:/blogdown_default，那么创建网站只需在 RStudio 里运行两行代码：

[4]个人博客示例：http://dapengde.com/blogdown_demo_default

```
setwd('c:/blogdown_default')
blogdown::new_site()
```

示例网站搭建完毕。对，就这么简单！

去 c:/blogdown_default 看看吧，出现了好多新文件夹。别的不用管也不用动，只注意 content 和 public 这两个就行了。打开 public/index.html，这就是网站的主页。

要想在本地电脑上浏览你的新网站，可以运行:

```
blogdown::serve_site()
```

这样，就可以在 RStudio 的右下面板实时看效果了。

### 发布到网上

上面搭建的网站，目前只存在于本地电脑里，下面我们发布到网上。

如果你像我一样，已经有了自己租用的服务器，上面有了 WordPress 博客，那最简单，在服务器的 public_html 文件夹下面创建一个新文件夹，例如叫做 test，那么只需将刚刚在本地新建的博客或科研网站的 public 文件夹里所有文件上传到服务器的 test 文件夹即可。访问你的域名`/test`，就可以看到新网站了！

如果你没有自己的服务器，那么最简单的是在 netlify[5] 免费注册个账号，然后按提示将你本地的 public 文件夹拖进去，会自动布署，等几分钟，就可以访问了。你可以免费申请一个 netlify 子域名，也可以买个自己的域名绑定上去。我的个人科研网站[6]就是这样做的。

那么，如何将自动生成的网站更新成自己的，并添加新内容呢？

### 网站更新

正如前文所述，新建的网站文件夹里有个 content 文件夹。这里就是更

[5]Netlify：`https://www.netlify.com`

[6]个人科研网站：`http://pzhao.net`

新网站内容的地方。你只需用记事本或 RStudio，打开其中的.md 或 .Rmd 文件，修改成自己的内容后保存，然后在 R 里运行建站函数：

```
blogdown::build_site()
```

这样，public/ 就自动更新了，再发布到网上即可。

.md 或 .Rmd 文件使用扩展的 markdown 语法，见本书附录。

如果要发表新帖子，那么有两种方式最简单：

- 方法一：将原有的.md 或 .Rmd 拷贝粘贴，改一下文件名和文件内容即可；
- 方法二：在 RSudio 代码窗口点击 Addins – New Post，按提示填写即可。

写完保存，运行建站函数，上传。完毕。

#### 自动同步

也许你嫌每次更新上传太麻烦。我们当然可以选择自动同步。

如果网站搭建在 Netlify，那么只需做三件事：

1. 下载并安装 GitHub 客户端，申请个免费账号，并创建一个项目，例如叫做 myweb，将前面我们举例的 c:/blogdown_default 文件夹设为项目的文件夹，并同步到云端。
2. 在你的 Netlify 设置里，设为自动跟你的 GitHub 里的 myweb 项目同步，同步的 Branch 设为 “master”，Build command 设为 “hugo_0.19”，Publish directory 设为 “public”。
3. 以后每次更新并运行完建站函数后，在 GitHub 客户端同步一下即可。

## 15.3　搭建科研网站

上文说到，我申请了一个 Netlify 免费账号，并且创建了自己的科研网站。这个科研网站的构架比个人博客复杂，有简介、新闻、发表论文、科研

项目等基本模块，还可以根据需要新增，并且可以设置多种语言，非常适合作为科研小组的展示窗口。

这个网站的搭建也很简单，跟个人博客类似。在你的电脑里新建文件夹，比如 `c:/blogdown_academic`，在 RStudio 里运行两行代码：

```
setwd('c:/blogdown_default')
blogdown::new_site(theme='gcushen/hugo-academic')
```

去 `c:/blogdown_academic` 看看吧，科研网站搭建完毕！

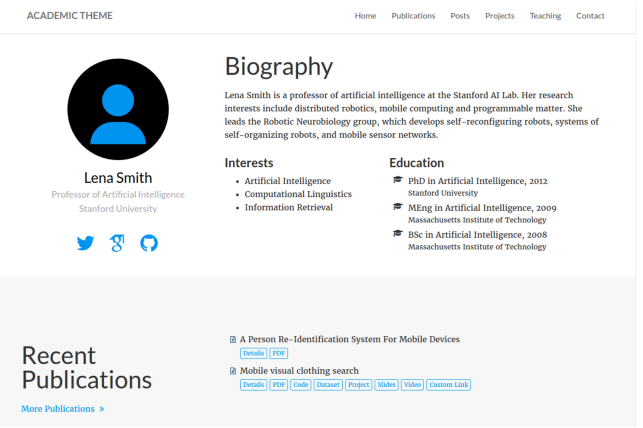



图 15.2: blogdown 搭建的 Hugo 科研网站示例

其他的发布、更新、同步，跟前文讲述的一致，这里不再赘述。需要特别指出的是，这个科研网站首页上的个人信息，需要在前面提到的 `config.toml` 里的 `[params]` 项目里修改。

这里的科研网站示例，使用的是 Hugo-academic 主题。Hugo 还有很多其他主题可以使用。我们相信，只要深入研究一下，一定可以搭建让人更合心意的网站。同时，blogdown 也在进一步完善中，让我们期待她的盛装绽放吧！

## 15.4　课外活动：搭建自己的网站

互联网技术日益成熟完善，现在搭建一个个人网站，有很多简单易行的方案，并不需要多少网络知识。本章介绍的 blogdown 是搭建静态个人网站最优秀的方案之一，快捷易用成本低。此外，还可以用 WordPress，搭建动态网站。我以前用的是后者，后来用前者创建了新网站。动态网站和静态网站各有所长，初学者不必在选择上过于纠结，从任何一种开始都可以。用得久了，才会实际体会到哪种更好，到时候再更改也不迟，因为有很多工具都可以在两者之间迁移转换。

快快动手，选一个你喜欢的方式，为你自己或者你的工作小组、单位搭建一个网站吧！

# 第十六章　在工作中应用

如果说我看得比别人远一些，那是因为我站在巨人的肩膀上。

— Isaac Newton, February 1676

## 16.1　自动完成业务报表

进入职场之后，有时需要定期完成一些业务汇报。比如，曾经有一份工作摆在我的面前，就是每个月制作一份 Word 格式的报告，内容是当月全国几十个观测站数据上传到服务器的情况。这份报告的格式是固定的，每次只需把里面的数据和图表更新即可。说来容易，可是每次都要有几十次的拷贝粘贴，非常容易出错，并且耗时耗力。

只恨当年未遇 R。

在前几章，我们已经学会了用循环来解决重复工作，学会了对文件进行批量操作，并且学会了用 R Markdown 来制作 Word 文档。我们只需把这些本领结合起来，就完全可以让 R 来完成那些格式固定、只需更新内嵌的数据和图表的业务报告了。

这里，我们来举个例子。

我们仍然使用前面用过的世界各大洲 7 年的电话数量数据。假定我们需要把每年的情况写一份报告，内容是当年的全球电话数量总和，电话数

量最多的洲及其所占比例，并用柱状图来展示。我们看看如何用 R 自动生成 7 份格式相同、内容不同的报告。

首先，我们用第七章的方法，建立一个 R Markdown 文档，作为 7 份报告的模板。方法是，在 RStudio 的界面点击菜单 File – New File – R Markdown，在对话框选择 Document，将默认输出格式勾选 Word 即可。我们的文档内容如下：

```
---
title: " 学城报告：全球电话数量"
author: " 萨姆"
date: "`r Sys.time()`"
output: word_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
```{r preprocess, echo=FALSE}
wp <- read.csv('c:/r4r/wp.csv')
i <- 1
year <- wp[i, 1]
wpi <- unlist(wp[i, 2:8])
conti <- c(" 北美", " 欧", " 亚", " 南美", " 大洋", " 非", " 中美")
```

## 数据

在 `r year`年，全球电话总数为 `r sum(wpi)`。拥有电话数量最多的是 `r conti[which.max(wpi)] `洲，电话数量为 `r max(wpi)`，占全球总数的 `r paste(round(max(wpi)/sum(wpi), 4) * 100, "%", sep =="")`。
```

```
## 作图

下图是 `r year`年全球各洲电话数量的柱状图。

```{r, echo=FALSE}
barplot(wpi, names.arg = conti)
```
```

我们来解释一下。这个模板文件可以分为三部分。第一部分是文件头，是新建这个文件时自动生成的，我们只是改了标题和作者等信息。第二部分是数据预处理（r preprocess），将数据读入，并把即将在文档中调用的数据（年份，电话数量，洲名等）准备好。这部分代码的参数 `echo` 设置为 `FALSE`，意思是代码不在文档中显示。第三部分是文档的正文，凡是需要根据年份来更新的数据，包括全球电话总数、电话最多的洲名等，都用 R Markdown 调用 R 代码的方式来计算。

现在，我们按 ctrl+s 键，将这个模板文档保存为 `c:/r4r/rdoc.Rmd`。再点击工具栏的 Knit 按钮，测试一下能否成功生成 Word 文档。

如果测试成功，我们就可以批量生产了。注意观察这个模板文件里，`i <- 1`，后面的数据就从 wp 数据框的第 i 行选取，那么不同的文档我们只需更改 `i <- 1` 这一行即可。所以，我们新建一个.r 代码文件，敲入下面的代码：

```
rdoc <- readLines('c:/r4r/rdoc.Rmd', encoding = 'UTF-8')
for(k in 1:7) {
  newrmd <- paste('c:/r4r/', k, '.Rmd', sep = '')
  rdoc[13] <- paste('i <-', k)
  writeLines(rdoc, newrmd, useBytes = TRUE)
  rmarkdown::render(newrmd, encoding = 'UTF-8')
}
```

第一行是用读入文本文件的方法读入我们刚刚创建的 rdoc.Rmd 模板文档。从第二行起，开始循环，依次取不同的 k 值，将 rdoc 的第 13 行即 i

赋值为 1 的代码，改为赋值为 k，然后将改变之后的文档新存为 newrmd，最后用 rmarkdown 的 `render()` 函数将 newrmd 转换为同名的 Word 文档。

现在，我们来见证一下奇迹的诞生。运行上面的代码，一两秒钟后请来 `c:/r4r` 文件夹看看，7 个 Word 报告是不是已经乖乖地等在那里了？

有了这种方法，我们就可以轻松地从无聊的重复劳动中解脱出来，把时间花在更有意义的事情上。那我们就用省出来的时间在 R 中玩玩扫雷吧！

---

**练习 16.1.** 仿照批量制作 Word 报告的方法，将各大洲电话数量情况批量制作成幻灯片，每年一个文件，格式可以是 ppt、html 或 pdf 均可。

## 16.2 可重复性研究报告

如果你是刚刚来到某个科研小组的新人，大概已经遭遇或将会遭遇以下情节：

- 从导师那里得到师兄留下来的参考文献库。可是，你用的文献管理软件跟师兄的不同，光转格式就花了大半天时间。
- 师兄已经毕业了，留下的数据处理方法保存在一些专门的商业软件才能打开的文件里。如今，这些软件都升级多次，但导师却对初来乍到的你千叮咛万嘱咐：千万要用旧版软件，不然，师兄的文件就没法正确打开了。
- 终于想方设法装上了旧版软件，打开师兄的文件一看，是个跟 Excel 类似的数据表，里边各列数据之间的计算错综复杂，花了两三天时间也没理出个头绪。
- 那就看看师兄的学位论文里的公式吧，说不定有线索。论文是个 Word 文档，打开一看，不仅里面的公式都成了乱码，而且对应的数据也不知在哪里找得到。

这时，你大概会暗自嘀咕：这巨人的肩膀咋就这么难爬上去呢？是不是我在科研能力上有问题？

这不是你一个人的问题。虽然科研成果都写在发表的论文里，但很多情况下，实验数据、数据的处理方法、处理的结果这三个层面的内容在传播过程中是分离的。比如，我们在数据处理软件里计算和作图，然后把计算结果和图片拷贝粘贴到论文中发表。如果别人想重复我们发表的工作，就先得花大量时间弄清楚论文中每个公式的含义、公式适用的条件、各种前提假设等等，再回到数据处理软件中实现，这个过程给科研工作的交流带了障碍。

最近几年，国际学术界发生了几件涉及学术道德的事件，例如日本的小保方晴子事件，他们的实验难以被同行重复，因此备受质疑，有人为此自杀，有人引咎辞职，无论圈里圈外的人都被惊动了。科研成果被他人使用，人类的科学才会承前启后发展。如果重复别人的方法存在诸多障碍，那么巨人的肩膀就无从谈起。

为了消除这种障碍，“可重复性研究报告”这个概念出现了，并且在 R 语言里实现得相当顺利。R 的 knitr (Xie, 2014, 2015, 2016c)、rmarkdown (Allaire et al., 2016) 等扩展包，可以将数据处理方法和详细说明、论文结果、结论很方便的有机融合在一篇 R 文档里，而 bookdown (Xie, 2016a,b)、bookdownplus (Zhao, 2017b)、blogdown (Xie, 2017)、mindr (Zhao, 2017c)、xlsx (Dragulescu, 2014)、ReporteRs (Gohel, 2017) 等扩展包，可以将研究成果轻松输出为美观易用的期刊论文、技术档案、学术论文、书籍、网页、办公文档、思维导图等格式。只要有这样一份报告，同研究组的成员就很容易接手前人的工作；只要换上自己的数据，就会得到自己的新结果，从而来证实前人方法的可重复性。

这样，R 语言就给我们提供了“科研一条龙”的打包解决方案。在这个方案里，除了原始数据外，其他的所有资料，包括数据处理方法、公式、参考文献等，全部是以纯文本的形式保存的，不对任何软件产生依赖。即使学生毕业、教授退休，即使软件升级，哪怕将来 R 语言不存在了，哪怕 RStudio 公司倒闭了，但我们的一切资料都可以用最简单的记事本软件打开阅读，为科研工作的前后相继提供了最大的便利。

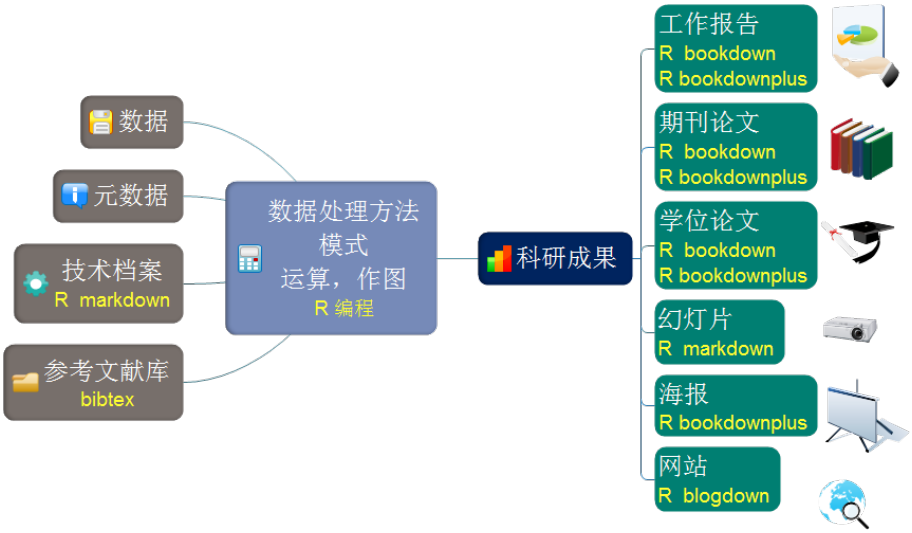



图 16.1: R 的科研一条龙整体解决方案

R 语言，把巨人的肩膀改造成了容易攀登的阶梯。

## 16.3 课外活动：学以致用

学以致用效果好，以赛代练效率高。

对你自己在学习或工作里用到的数据进行分析，用 R markdown 写一份可重复性研究报告。

然后，用 rmarkdown 或 ReporteRs 扩展包，将上述研究报告以幻灯片的形式展示。

# 第十七章　用 R 进行基础统计（一）：概率分布检验

罗宾：我想说，要是没有个像 R 这样的工具，你休想学好统计学。
大卫：费舍尔他们没有 R 也搞定了。
彼得：如果有 R 的话，想想他们该能走多远！

— December 2004

数据分析离不开统计检验。比如你可能需要回答这样的问题：

- 不同生产线的次品率是否有显著差异？
- 长时间序列的观测数据是否具有显著的趋势？
- 实验组和对照组是否有显著差异？等等。

我们经常需要对观测数据或者实验数据进行统计分析和比较。经过了统计检验，你就可以理直气壮地说，我的方案/设计有了显著效果，不是碰巧撞上的！

统计学发展到现在，已经给我们提供了许多已知的分布。比如一个城市每天的死亡人数，服从泊松（Poisson）分布。比如对一个水体样本某种成分的重复测试得到的数据集属于正态分布，其中心值就是浓度的真值（不可知）。比如对于结局只有“全或无”的事件，服从于二项分布（扔硬币，某一面朝上的概率为 0.5；产品合格与否；击球命中与否，等等），这种二项分布的参数往往是确定的。

大多数直接由观测/实验得到的数据都是属于某一种分布，比如汇率的波动，大气污染的数据，产品的合格率等等。如果是确定的某种分布，我们就可以说得到的数据服从于某个参数分布；实际数据分析中，如果我们要进行不同样本的比较，往往需要先知道得到的数据属于什么分布。R 提供了各种常用的分布检验函数。

在本章和下一章里，我们学习基础统计检验。没有搞懂统计原理没关系！只要确定地知道应该用什么统计方法，就可以在 R 里找到相应的命令或者包。

## 17.1 统计检验的基本思想

在统计推断中，若给定或者假定了总体分布的具体形式（如正态分布），但是其参数（parameter）未知，要基于来自总体的样本对未知参数作出估计或者进行某种形式的假设检验，这类推断方法就称为参数方法（也有的书称为参数检验），检验的目标是某个参数落在特定的范围内的假设。

另外一类检验，不是针对具体的参数，而是针对分布的类型进行检验，比如假设总体分布具有正态性，则需要基于样本来检验这个假设是否可以接受。

因此，统计检验的真正意思其实是检验已知数据集是否服从于特定参数的某种已知分布，或者检验已知数据集服从于某种已知分布（不指定参数）。什么是参数呢？参数就是描述总体样本分布的特征的数据，比如中心值、方差等等，总体的分布由参数决定。参数是确定的，但是往往不可知，因为在一般情况下，我们无法得到全部样本，只能得到一部分（从总体中抽取部分样本）。对我们得到的这部分样本数据进行统计描述得到的数据，就相应地叫做统计量。那么参数估计就是根据统计量来推断总体样本（是否服从于某一已知分布？两个样本是否来自同一分布？等等）的过程：对参数提出一定的假设，然后根据已知的统计量对提出的假设进行假设检验。我们通过一个例子来简单说明一下假设检验的基本思想和概念。

假设有一个工厂生产某种产品，有一定的次品率，用 $w$ 表示。行业规

定这个次品率必须要小于 0.02（$w < 0.02$），否则这批产品要被退货。这里 "$w < 0.02$" 就是我们需要做的假设，记为 H0（null hypethesis，称为原假设或者零假设）。与 H0 对立的假设则记为 H1，称为备择假设或者对立假设（alternative hypothesis）。

现在我们假设从某天的产品中抽样 250 个，发现了 5 个次品。这个抽样结果就成为了判定原假设 H0（次品率低于 0.02）成立的依据。如果我们用 $n$ 表示抽到的次品个数，显然 $n$ 越大，对 H0 成立就越不利。那么 $n$ 大于多少时就可以判断假设不成立，即，这批产品应该被退货呢？

统计检验的过程是：先假定 H0 成立，然后计算在这种假设下，$n$ 大于 5 的概率有多大。因为我们知道抽样个数为 250 个，次品率服从于二项分布（这二项就是：合格 or 不合格，只有两种结果。不合格的概率为 0.02。只有两种结局的事件服从二项分布，下面我们会详细说），0.02 是我们已知的参数。那么从这样一个总体样本中抽出 250 个样品，"次品数量 $n = 5$" 这个事件发生的概率大约是 0.62（这个计算过程将由 R 完成，接下来我们会说到），算是一个大概率事件。也就是说，当产品生产的次品率 <0.02 时，一次抽样 250 个，有 5 个次品的事件是个大概率事件，我们应该接受原假设，即次品率符合行业标准，这批产品合格。

那如果抽样 250 个里面，有 12 个次品呢？再次计算出 $n = 12$ 的概率是 0.005，这是一个小概率事件。而在一次观察中小概率事件是不可能发生的，即假设 H0 成立，那么一次抽样中发现 12/250 的次品率是不可能的，因此要拒绝 H0，接受 H1，即产品不符合要求，次品率高于 0.02。

通过尝试我们可以知道，在这样的次品率要求下，一次抽样 250 个，次品率少于 10 个才是合格的。

统计思想解释了，我们还需要回到统计表达来看一下。本次产品检验的结果，如果用 "统计分布" 的角度来表达，该怎么说呢？

- 250 个抽样样品，5 个次品：这 250 个样品来自于概率为 0.02 的二项分布总体，对次品率合格与否的检验其实是对这样一个样本服从于一个已知分布的检验。

- 250 个抽样样品，12 个次品：这 250 个样品不来自于概率为 0.02 的

二项分布总体（它们来自于一个概率高于 0.02 的总体），是对这个样本是否服从于一个已知分布的检验。

最后来总结一下。无论是什么假设检验，其思想是一致的，就是所谓概率性质的反证。其根据是实际推断原理：小概率事件在一次观察中是几乎不可能发生的。我们先给出原假设 H0，然后在这个前提下去验证我们观察到的事件，如果这个事件发生的概率很小（$p$ 值小于 0.05），说明小概率事件居然在一次观察中发生了，这于刚才我们说的 ‘‘实际推断原理’’ 相悖，这表明 ‘‘假设 H0 成立’’ 是错误的，我们应该拒绝原假设。反之，如果这个事件发生的概率很大，那么在一次观察中发生的可能性很大，我们就应该接受原假设。

需要注意以下三点：

1. ‘‘小概率事件在一次观察中发生’’ 与实际推断原理相矛盾，这种矛盾并不是形式逻辑中的绝对矛盾，因为 ‘‘小概率事件在一次观察中几乎不会发生’’ 并不等同于 ‘‘小概率时间在一次观察中绝对不会发生’’。因此，根据概率性质的反证法得出的接受或拒绝 H0 的决定，并不等同于我们就证明了原假设 H0 正确或错误，而是我们根据样本所提供的信息，在一定的可靠程度上认为 H0 正确或错误。

2. 原假设与备择假设不对称，不可以交换，在假设检验中的地位不同。原假设和备择假设的设定要根据具体问题来决定，通常我们把没有把握、不能轻易肯定的命题作为备择假设，而把没有充分理由不能轻易否定的命题作为原假设，只有理由充分时才能拒绝它，否则应该接受。

3. 既然小概率事件在一次观察中 ‘‘几乎不可能’’ 发生，那么小概率事件也有可能发生，不是绝不可能发生。在这种情况下我们拒绝了 H0，就犯错了，这个犯错的概率为 $P_1$（拒绝 H0|H0 为真）。还有一种错误，就是 H0 不正确而我们却接受了 H0，犯这个错误的概率为 $P_2$（接受 H0|H0 为假）。通常样本量都是有限的，因此这两类错误都有可能发生。通常我们把解决这类问题简化为只针对第一种错误（正如前面所述，在选择 H0 和 H1 有所区别），限制犯错的概率大小，而对第二类错误不管（统计学家这么决定，应该有他们的道理吧）。

这样，我们将犯第一类错误的最大概率称为假设检验的显著性水平。

定义：在假设检验中，拒绝原假设 H0 的最小显著性水平称为检验的 $p$ 值（$p$-value）。$p$ 值表示的是对原假设的怀疑程度，或者说，首次拒绝原假设的概率 $p$ 值越小，原假设越不可靠。$p$ 值的计算依赖于原假设和所得样本。在接下来介绍的 R 执行的各种检验中，都会给出 $p$ 值。

## 17.2　离散型随机变量和连续型随机变量

我们前面介绍了参数分布的概念以及检验思想。接下来我们就要面对得到的数据，也就是变量了。我们需要根据变量的性质来选择 R 中提供的检验方法。R 提供针对离散型随机变量和连续型随机变量的检验方法，先来了解一下相关概念：

我们生活中遇到的事件可以分为定性事件和定量事件，因此，这两类事件发生的概率也就相应地是定性概率和定量概率。比如我们前面提到的产品检验，结果就只有 ‘‘合格’’ 和 ‘‘不合格’’ 两种，这是定性的（但是如果们具体到某一条生产线，数值化不合格的比例，就是定量了）。再比如交通事故，基本上分为 ‘‘轻微事故’’、‘‘一般事故’’、‘‘重大事故’’ 等，也是定性的。这些事件结局之间，只有类型不同，而不是测量数值上的区别。我们不能预测某条特定道路上的交通事故会属于哪一种，因此这类变量就是定性随机变量。这样的例子还有所属党派、社会经济地位、苹果叶脉的分布图案、女士偏爱的服装品牌等。与定性变量相关的可能事件结局是有限的，除了计算发生概率，貌似我们能做的事情也不多。

我们更感兴趣的往往是与定量随机变量相关的事件。顾名思义，定量随机变量，就是量化的观测值，源于数值型观察值或测量值。对于定量随机变量，我们就可以求平均值，标准偏差，并且估计随机变量可能误差等等。定量随机变量又可分为两种，接下来我们要介绍的分布检验都是针对这两种定量随机变量的。

- 离散型随机变量：当一个变量的观察值只能取可数个数值时，该变量称为离散型随机变量。离散型随机变量可能的取值是孤立和分散的。

比如：安装信号灯后的十字路口交通事故的变化；总统选举中的无效选票数等等，这些随机变量能够取到的值都有限，可以数出来。

二项分布和泊松分布是最常见的离散型随机变量分布。

- 连续型随机变量：当一个变量的观察值可以取到某一个区间上的任何一个数值时，该变量称为连续型随机变量。连续型随机变量能取到的数值是无限、不可数的。比如气温、降水量、股票价格、大气污染物浓度等等。

正态分布、均匀分布、指数分布等都是常见的连续型随机变量分布。

因此，我们在得到数据之后，首先要区分变量的类型，才能进一步地做概率和分布的分析。本节我们介绍常见的离散型随机变量分布检验和连续型随机变量分布检验。

在介绍离散型随机变量分布检验之前，我们先来认识一下伯努利（Bernoulli）试验（独立重复试验），因为常用的几种离散型随机变量分布都与它相关。

若实验 E 满足条件：

1. 每次试验独立进行；

2. 每次试验只有两种结果：事件 A 发生或者不发生。这两种结果是互逆的，完全不相容。

设 $\mathrm{P(A)} = p$，$0 < p < 1$, (这个基本概率表示方式还记得吧)，将试验 E 重复 $n$ 次，就是 $n$ 重伯努利试验。例如击球命中与否；抛硬币的某一面朝上；产品检验抽到次品等等，都可以看作伯努利试验。离散型随机变量分布都和伯努利试验相关，包括二项分布、负二项分布、几何分布、超几何分布和多项分布。我们经常接触到的是二项分布和泊松分布。

## 17.3　二项分布

二项分布描述的是 $n$ 重伯努利试验中出现 $x$ 次成功的概率分布。在 R 中产生随机样本的命令是：

```
binom(x, size, prob)
```

x 是试验成功次数，size 是总实验次数，prob 是成功的概率。

最经典的就是扔硬币，硬币落地时某一面朝上的次数（概率）服从二项分布。如果想验证一下这个 "0.5" 的概率，那么只需要把硬币扔 50 次（或者更多），记录这一面朝上的次数，然后进行检验就可以。

假设我们扔了 60 次，其中有 27 次朝上。那么检验函数是：

```
binom.test(x = 27, n = 60, p = 0.5, alternative = "less")
# n 是总实验次数，x 是“试验成功”的次数，p 是已知的二项分布概率。
# 分布已知，参数已知，典型的参数方法（参数检验）


##
## Exact binomial test
##
## data:  27 and 60
## number of successes = 27, number of trials = 60,
## p-value = 0.2595
## alternative hypothesis: true probability of success is
## less than 0.5
## 95 percent confidence interval:
## 0.0000000 0.5639903
## sample estimates:
## probability of success
## 0.45
```

可见，$p$ 大于 0.05，无法拒绝原假设，即扔硬币结果是符合二项分布的。0.45 是这一次实验的实际比率，即某一面朝上的次数与总实验次数的

比例。

需要注意的是，二项分布，并不意味着概率一定是 0.5。比如我们一开始举的产品次品率，概率是 0.02。再来看一个例子：假设有一种比赛，内容就是一个运动员击球，只比成功率高低。M 是一个成功率较高的击球手，根据他前十年的赛事成绩，大家评价他的成功率为 0.4，即 40%。新的赛季以来，他一共击球 558 次，击中了 203 次，命中率为 0.36。那么，如果保持这个水平，本赛事结束后，他的成功率低于 0.40 的概率是多少？

```
binom.test(x = 203, n = 558, p = 0.4, alternative = "less")
```

```
##
## Exact binomial test
##
## data:  203 and 558
## number of successes = 203, number of trials = 558,
## p-value = 0.04378
## alternative hypothesis: true probability of success is
## less than 0.4
## 95 percent confidence interval:
## 0.0000000 0.3986764
## sample estimates:
## probability of success
## 0.3637993
```

注意：这里的备择假设设置为 “less”，那么得出的 $p$-value 就是成功率高于原成功率 0.40 的概率，小于 0.05，因此我们应该拒绝原假设，接受备择假设，即这个击球手退步了，他本赛季的成功率会低于 0.40. 如果我们把备择假设设为 “greater”，那么：

```
binom.test(x = 203, n = 558, p = 0.4, alternative ="greater")
```

```
##
## Exact binomial test
```

```
##
## data:  203 and 558
## number of successes = 203, number of trials = 558,
## p-value = 0.9637
## alternative hypothesis: true probability of success is
## greater than 0.4
## 95 percent confidence interval:
## 0.3299983 1.0000000
## sample estimates:
## probability of success
## 0.3637993
```

得出的 $p$-value 就是成功率低于原成功率 0.40 的概率，$p$ 值为 0.96，因此我们应该接受原假设，即这个击球手退步了，他本赛季的成功率会低于 0.40. 总而言之这名击球手就是退步了。

因此，R 语句的完整表达为：

```
binom.test(x, n, p,
           alternative = c("two sided","less","greater"))
```

默认设置为 two sided。

## 17.4　泊松分布

泊松分布是 1837 年由数学家泊松提出，作为二项分布的近似引入的。如果我们把伯努利试验中发生概率很小的事件称为“稀有事件”，那么根据泊松定理，$n$ 重伯努利试验中稀有事件发生的概率近似服从泊松分布。也可以大概理解为，泊松分布是二项分布的极限（总实验次数 $n$ 很大，发生概率非常小）。

比如：

- 细胞培养板里的细菌个数。

- 一个大城市每天因心血管疾病死亡的人数。
- 热线电话每 10 分钟的呼入数。
- 本书各章的错别字数。

泊松分布产生的一般条件：

在自然界和现实生活中，常遇到在随机时刻出现的某种事件。因此，我们把在随机时刻相继出现的事件形成的序列称为随机事件流。若随机事件流具有平稳性、无后效性、普通性，则称该事件为泊松事件流（泊松流）。

- 平稳性——在任意时间区间内，事件发生 $k$ 次（$k \geq 0$）的概率只依赖于区间长度而与区间端点无关。
- 无后效性——在不相重叠的时间段内，事件的发生相互独立。
- 普通性——如果时间区间充分小，事件出现两次或两次以上的概率可忽略不计。

可见，泊松分布往往和时间有关联，其分布参数的概率意义是单位时间出现的随机质点的平均个数.

因此在下面的执行语句中可以看到时间参数。

R 中执行泊松分布检验的语句是：

```
poisson.test(x, T = 1, r = 1, conf.level = 0.95,
             alternative = c("two.sided", "less", "greater"))
```

其中，

- `x`：预期事件发生的次数。
- `T`：time base for event count. 用于比较的数据 (基线数据)。
- `r`：假设或已知的发生概率
- `alternative`：选择 ‘‘two.sided’’, ‘‘greater’’ or ‘‘less’’
- `conf.level`：置信区间

还是先通过一个例子来感受一下。某公司为了宣传产品，开设了一条购物热线，投资方期望这条热线电话每 10 分钟能有 15 次呼入。开通后计数表明，10 分钟内呼入次数为 13 次。那么这个呼入次数达到了投资方的

预期值了吗？

```
poisson.test(13, 15)
# 13 是实际计数得到的值，15 是预期值。这个格式是不是很简单？
```

```
##
##  Exact Poisson test
##
## data:  13 time base: 15
## number of events = 13, time base = 15, p-value =
## 0.6991
## alternative hypothesis: true event rate is not equal to 1
## 95 percent confidence interval:
##  0.4614635 1.4820264
## sample estimates:
## event rate
##  0.8666667
```

$p$ 大于 0.05，应该接受原假设，即呼入次数是符合预期的。用统计学术语来说，就是要对样本计数 $x = 13$ 和总体均数 $\mu = 15$ 的差别作统计学检验。$p = 0.6991$，意思是说从 $\mu = 15$ 的总体中随机抽样获得的 x 大于等于 13 的可能性达到 0.69，是一个大概率事件，因此不能拒绝原假设。

我们来把这个问题扩展一下。某个热线电话 10 分钟内呼入的电话数为 13，而另一热线电话 10 分钟内呼入的电话数为 24。这两个热线电话的受欢迎程度一致否？这个检验的实质是比较两个数据集是否来自同一个泊松分布。

```
poisson.test(13, 24, alternative = "less")
```

```
##
##  Exact Poisson test
##
## data:  13 time base: 24
## number of events = 13, time base = 24, p-value =
```

```
## 0.01072
## alternative hypothesis: true event rate is less than 1
## 95 percent confidence interval:
##  0.0000000 0.8611904
## sample estimates:
## event rate
##  0.5416667
```

```
# 要与第二个热线电话打呼入数持平，
# 临界值（呼入电话数）最多是多少？
qpois(0.95, lambda = 24)
```

```
## [1] 32
```

```
# 要与第二个热线电话呼入次数持平，
# 临界值（呼入电话数）最少是多少？
qpois(0.05, lambda = 24)
```

```
## [1] 16
```

$p$ 小于 0.05，拒绝原假设，两个热线受欢迎程度不一样。如果要和第二条热线一样受欢迎，第一条热线的临界呼入次数最多 32，最少 16。

---

扩展阅读：利用 Pearson 卡方检验来检验是否符合正态分布：（来自：薛毅、陈立萍《统计建模与 R 软件》清华大学出版社 2006）

```
# 现有 42 个数据，分别表示某一时间段内电话总机呼入的次数，
# 呼入的次数 0  1  2  3  4  5  6
# 出现的频率 7 10 12  8  3  2  0
# 问：某个时间段内呼入次数是否符合 Possion 分布？


x <- 0:6
y <- c(7, 10, 12, 8, 3, 2, 0)
mean <- mean(rep(x, y))
```

```
q <- ppois(x, mean)
n <- length(y)
p <- q
p[1] <- q[1]
p[n] <- 1 - q[n - 1]
for(i in 2:(n - 1))
  p[i] <- 1 - q[i - 1]
chisq.test(y, p = rep(1/length(y), length(y)))
```

```
##
##  Chi-squared test for given probabilities
##
## data:  y
## X-squared = 19.667, df = 6, p-value = 0.003174
```

$p$ 小于 0.05，拒绝原假设，电话呼入次数不符合泊松分布。

## 17.5　卡方分布

简单地说，如果一个数据集（包含 $n$ 个数据）服从于正态分布，那么这个数据集里每个数据的平方服从于自由度为 $n$ 的卡方分布。因此卡方分布的形状取决于 $n$。

卡方拟合优度检验（Chi-squared goodness of fit tests) 用来检验样本是否来自于特定类型卡方分布，在 R 中的执行语句是 `chisq.test()`:

```
chisq.test(x, y = NULL, correct = TRUE,
           p = rep(1/length(x), length(x)),
           rescale.p = FALSE,
           simulate.p.value = FALSE, B = 2000)
```

- `x` 是数据集，可以是数值型向量、矩阵或者因子。

- `y` 是数值型向量。如果 `x` 是数值型矩阵，则 `y` 可以忽略。如果 `x` 是因子，`y` 必须是同样长度的因子。
- `correct` 是逻辑型变量，指示是否需要做连续型校正。
- `p` 是指定的概率分布。
- `rescale.p` 是逻辑型变量，指示是否各项 $p$ 的和为 1。
- `simulate.p.value` 是逻辑型变量，指示 $p$ 的计算过程是否要进行蒙特卡罗模拟。
- `B` 是整数，蒙特卡罗检验里的重复次数。

还是来看一个例子。

如果连续掷骰子 200 次，得到如下点数分布。那么这个骰子是均匀的吗？

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| 30 | 21 | 24 | 35 | 42 | 48 |

如果骰子是均匀的，那么每一面出现的期望概率应该相等，都是 1/6。但是我们注意到点数 5 和 6 竟然出现分别出现了 42 和 48 次。这是纯属巧合，还是骰子不均匀？

```
myfreq <- c(30, 21, 24, 35, 42, 48)  # 得到的频数
myprobs <- c(1, 1, 1, 1, 1, 1)/6  # 指定的概率分布
chisq.test(myfreq, p = myprobs)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  myfreq
## X-squared = 16.3, df = 5, p-value = 0.006038
```

$p$ 值远小于 0.05，拒绝原假设，这个骰子不均匀。

接下来我们来看看如何检验一个已知连续变量数据集是否服从于某种

假定的分布。

## 17.6　正态分布

正态分布是我们日常生活学习中最常见的连续型随机变量概率分布，很多随机变量的概率分布都可以近似地用正态分布来描述。例如，在生产条件不变的情况下，产品的强力、抗压强度、口径、长度等指标；相同生长环境下同种生物的身长、体重等指标；同一批种子的重量；同一实验测量结果；弹着点沿某一方向的偏差，等等。一般来说，如果一个量是由许多微小的独立随机因素影响的结果，那么就可以认为这个量具有正态分布。

该检验原假设 H0 为：数据集符合正态分布。R 计算出统计量 $W$，其最大值是 1。$W$ 越接近 1，表示样本越接近正态分布。如果 $p$-value 小于显著性水平 $\alpha$ (0.05)，则拒绝原假设 H0，该分布不属于正态分布。

R 中实现正态性检验的函数是 `shapiro.test()`，执行 Shapiro-Wilk 检验（要求样本量在 3 到 5000 之间）：

```
shapiro.test(x)
```

来看一个具体的例子：1949-1960 年乘坐飞机的乘客人数是否服从正态分布（参数未知）？首先我们来看看直方图或者 QQ 图，直观地感受一下。

```
par(mfcol = c(1, 2), ps = 6.5)
hist(AirPassengers, breaks = 20)
qqnorm(AirPassengers, pch = 1)
```

从直方图来看，不太像钟型分布。QQ 图也比较不太接近直线。感觉不像正态分布，那么我们来检验一下直觉是否正确。

```
shapiro.test(AirPassengers)
```

```
##
##  Shapiro-Wilk normality test
##
```

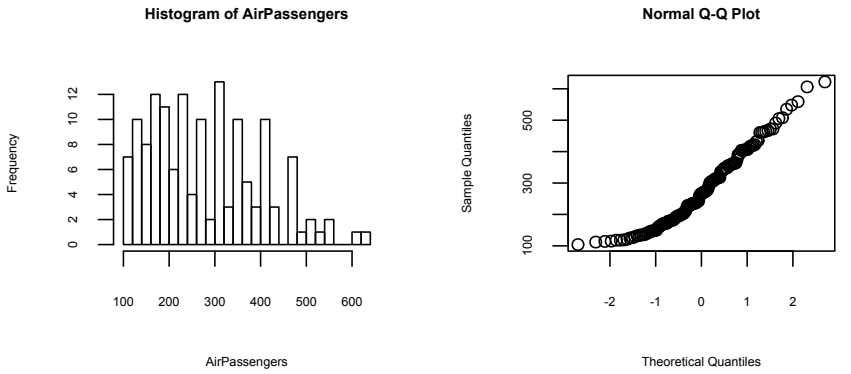



图 17.1: 1949-1960 年乘坐飞机的乘客人数分布的直方图和 QQ 图

```
## data:  AirPassengers
## W = 0.95196, p-value = 6.832e-05
```

$W$ 值不接近 1，$p$ 值远小于 0.01，可见直觉正确，我们应拒绝原假设，即 1949-1960 年的乘客人数不服从于正态分布。

再看看采集的人群血样中促黄体素（Luteinizing Hormone）浓度数据。

```
par(mfcol = c(1, 2), ps = 6.5)
hist(lh, breaks = 20)
qqnorm(lh, pch = 1)
```

看不出感觉来吧？还是不要猜了，检验一下吧。

```
shapiro.test(lh)  # 人群血样中促黄体浓度属于正态分布吗？
```

```
##
##  Shapiro-Wilk normality test
##
## data:  lh
## W = 0.97077, p-value = 0.2714
```

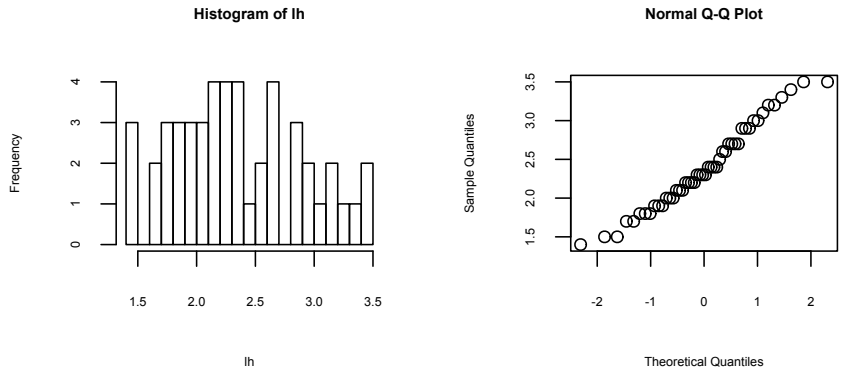



图 17.2:　人群血样中促黄体素（Luteinizing Hormone）浓度布的直方图和 QQ 图

$W$ 值接近 1，$p$ 值大于 0.05，接受原假设，即人群血样中的促黄体素浓度属于正态分布。

那么如何检验已知数据集是否服从特定参数的正态分布呢？R 中还有一个非常好用的检验分布的执行语句 `ks.test()`,它执行的是 Kolmogorov-Smirnov 连续分布检验，可以用于包括正态分布的多种连续型概率分布的检验。这个检验的原理是，以样本数据的累计频数分布与特定理论分布比较，若两者间的差距很小，则推论该样本取自某特定分布族。R 的执行结果会给出一个统计量 $D$，它的意义是假设的分布累计频数与受检验样本之间累计频数分布概率之间的差异（这句话看不懂不要紧）。记住：$D$ 值越小，越接近 0，表示样本数据越接近你预先假设的分布。如果 $p$-value 小于显著性水平 $\alpha$ (0.05)，则拒绝原假设（你的样本和假设的分布不一致）。`ks.test()` 可以通过对已知样本的分布分析，与已知的概率分布做差别检验，从而对样本的分布类型作出统计学判断。此外，`ks.test()` 还可以对两个样本分布进行差别检验，以判断它们是否服从于同一分布，且分布类型并不受限。

回到我们刚才的问题：如何检验已知数据集服从特定参数的正态分布？还是通过一个例子来说明。

```
# 生成参数为 5, 3 的正态分布随机数
A <- rnorm(100, 5, 3)
# 检验数据集 A 是否服从 mu = 5, sd = 3 的正态分布
ks.test(A, 5, 3)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  A and 5
## D = 0.52, p-value = 0.9703
## alternative hypothesis: two-sided
```

```
# 检验数据集 A 是否服从 mu = 15, sd = 1 的正态分布
ks.test(A, 15, 1)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  A and 15
## D = 1, p-value = 0.0198
## alternative hypothesis: two-sided
```

从 $p$ 值来看，显然，数据集 A 服从 $\mu = 5$，$sd = 3$ 的正态分布，而不服从 $\mu = 15$，$sd = 1$ 的正态分布。

Kolmogorov-Smirnov 常用的连续分布检验还有以下几种：

## 17.7 指数分布

来举例说明一下：

```
set.seed(1)
mydata <- rexp(950)
```

```
ks.test(mydata, "pexp")
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  mydata
## D = 0.038167, p-value = 0.1256
## alternative hypothesis: two-sided
```

结论: $D$ 值很小, $p$-value $> 0.05$，不能拒绝原假设，所以数据集 mydata 符合指数分布。

## 17.8　伽马分布

伽马分布 (Gamma)，也称为皮尔逊 III 型分布。它的曲线是左右不对称的一个峰。还是用 Kolmogorov-Smirnov 来检验

```
set.seed(1)
mydata <- rgamma(1500, 1)
ks.test(mydata, "pgamma", 1)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  mydata
## D = 0.020827, p-value = 0.5334
## alternative hypothesis: two-sided
```

$D$ 值很小, $p$-value $> 0.05$，不能拒绝原假设，所以数据集 `mydata` 符合 shape $= 1$ 的伽马分布，嗯，理解为 1 型伽马分布吧。跟在名字 "pgamma" 后面那个数字，就是表示对几型分布的假设 "1" 型或者 "2" 型，用参数 $\alpha$ 表示。根据参数 $\alpha$ 的不同，概率密度函数和累积分布函数的图形也不同。那么看看这个数据集是否符合 2 型伽马分布呢（废话，符合 1 型了怎么可

能还符合 2 型……但是我们就是要试一试）

```
ks.test(mydata, "pgamma", 2)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  mydata
## D = 0.36615, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

试一试，果然就死心了吧，$D$ 值大而 $p$ 值小于 0.05，拒绝原假设，不符合 2 型分布。是不是很简单容易上手？

## 17.9 韦伯分布

韦伯（weibull）分布，又称韦氏分布或威布尔分布，是可靠性分析和寿命检验的理论基础。韦伯分布通常用在故障分析领域（field of failure analysis）中；甚至可以模拟故障率随时间的变化分布。故障率为以下三种情况（$\alpha$ 也是形状参数）：

- 一直为常数，$\alpha = 1$，提示故障可能在随机事件中发生
- 一直减少，$\alpha < 1$，提示 “infant mortality”——先天不足的意思？
- 一直增加，$\alpha > 1$，提示 “wear out” – 随着时间的持续，故障的可能性增加

检验方法依然是 Kolmogorov-Smirnov：

```
set.seed(1)
mydata <- rweibull(8500, 1)
ks.test(mydata, "pweibull", 1)
```

```
##
##  One-sample Kolmogorov-Smirnov test
```

```
##
## data:  mydata
## D = 0.0096477, p-value = 0.4074
## alternative hypothesis: two-sided
```

可见 $D$ 值很小，而 $p$ 值大于 0.05，因此不能拒绝原假设，该样本服从于韦伯分布 1 型。那显然就不服从于 2 型了。

```
ks.test(mydata, "pweibull", 2)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  mydata
## D = 0.18703, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

Kolmogorov-Smirnov tests 是一个很有用的检验工具，可以用于常用的很多连续型概率分布检验。而且，这些检验的 R 命令都是相似的，它们都是这个格式：

```
ks.test(x, y, ...,
        alternative = c("two.sided", "less", "greater"),
        exact = NULL)
```

- x 就是你的数据集。
- y 就是你假设的连续型概率分布。用小贴士 17.1 中 “p” 开头的命令放到 y 的位置，就可以轻松地做一个分布检验了。
- ... 是你假设的分布中的参数描述，比如我们刚才用到的 1 型，2 型，也可以没有。
- alternative 是你的备择假设，注意在二项分布中的意义。
- exact 指示是否需要计算出精确的 $p$ 值。

好啦，只要套用相应的名词，你就可以做相应的分布检验啦！比如 `ks.test(mydata, "pweibull",2)`，意思就是检验数据集是否服从于韦氏 2 型

分布。“pweibull” 这个命令可以在小贴士 17.1 中 “韦伯分布” 一行找到。

**必须要注意！** Kolmogorov-Smirnov tests 确实好用，但是一定要记住，它只能用于连续型概率分布检验。因此，二项分布、泊松分布等就不能用它。

能够运用这个命令的连续型概率分布检验，除了上述的正态分布检验、指数分步检验等，还有 $t$ 分布检验、$F$ 分布检验，均匀分布检验。请自己去尝试一下吧。

## 17.10 检验两个向量是否来自同一分布

Kolmogorov-Smirnov 还可以用于检验两组数据是否来自同一分布。仍然以二氧化碳浓度数据为例，我们看看 1959 年和 1965 年浓度是否来自同一分布。

需要注意的是，在做单样本 K-S 检验或者正态检验时，有时会有错误提示 “Kolmogorov - Smirnov 检验有结节”（或者类似意思的生硬中文表达，R 的中文表达问题），这是因为 K-S 检验只对连续累积分布函数（CDF）有效，而连续累积分布函数中出现相同值的概率为 0，因此 R 会报错。这也提醒我们，在做正态性检验之前，要了解自己的数据，对数据进行描述性分析/作图，有个大致的认识，才能选择正确的检验方法并正确解读。

还是以夏威夷的二氧化碳浓度为例，我们看看 1959 年和 1965 年浓度是否来自同一分布。

```
ks.test(co2[1:12], co2[97:108], alternative = "two.sided")
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  co2[1:12] and co2[97:108]
## D = 1, p-value = 7.396e-07
## alternative hypothesis: two-sided
```

$p$ 值小于 0.05，可见这两个年份的二氧化碳浓度差异很大，不来自于同一分布。

**练习 17.1.** 生成两个不同的 $F$ 分布或者学生 $t$ 分布，然后通过检验证实它们确实不同。

**练习 17.2.** 检验数据集 nottem 属于哪种分布。这是一个叫做 Nottingham 的城市 1920-1939 的月平均气温数据，在基础安装包里，可直接调用。

## 17.11　课外活动：概率函数汇总

R 中几乎所有生成概率分布值的函数工作机制都是相同的（是相同，不是相似哦），且遵循同样的命名规则（“func” 表示你的分布）：

- 概率密度函数（PDF）以 `d` 开头，调用格式为 `dfunc(x, p1 , p2…)`，得到密度函数。`x` 为数值型量
- （累积）分布函数以 `p` 开头。调用格式为 `pfunc(q, p1 , p2…)`，得到分布函数。`q` 为数值型量。
- 分位数函数以 `q` 开头。调用格式为 `qfunc(p, p1 , p2…)`，得到分位数函数。`p` 为由概率构成的向量。
- 随机数生成函数以 `r` 开头。调用格式为 `rfunc(n, p1 , p2…)`，生成符合此分布的随机数，`n` 为生成随机数的个数。

其中 `p1`，`p2` 是分布的参数值。所有 `pfunc` 和 `qfunc` 的函数都具有逻辑参数，`lower.tail` 和 `log.p` ，所有的 `dfunc` 函数都有参数 `log`。此外，对于正态分布，具有学生化样本区间的分布还有 `ptukey` 和 `qtukey` 这样的函数。

来举例说明一下。

显著性水平为 5% 的正态分布的双侧临界值：

**小贴士 17.1.** 分布函数汇总。

| 分布 | R 函数 | 分布参数 |
|---|---|---|
| 贝塔分布 | dbeta, pbeta, qbeta, rbeta | shape1, shape2, ncp = 0 |
| 二项分布 | dbinom, pbinom, qbinom, rbinom | size, prob |
| 生日分布 | dibirthday, pbirthday | classes, coincident |
| 柯西分布 | dcauchy, pcauchy, qcauchy, rcauchy | location, scale |
| 卡方分布 | dchisq, pchisq, qchisq, rchisq | df, ncp = 0 |
| 指数分布 | dexp, pexp, qexp, rexp | rate |
| $F$ 分布 | df, pf, qf, rf | df1, df2, ncp |
| 伽马分布 | dgamma, pgamma, a qgamma, rgamma | shape, rate = 1, scale = 1/rate |
| 几何分布 | dgeom, pgeom, qgeom, rgeom | prob |
| 超几何分布 | dhyper, phyper, qhyper, rhyper | m, n, k(注意，普通变量 ‘‘n’’ 被命名为 ‘‘nn’’，‘‘因为’’n‘‘已经被占用) |
| 指数正态分布 | dlnorm, plnorm, qlnorm, rlnorm | meanlog, sdlog |
| 罗吉斯分布 | dlogis, plogis, qlogis, rlogis | location, scale |
| 多项式分布 | dmultinom, pmultinom | size, prob |
| 负二项分布 | dnbinom, pnbinom, qnbinom, rnbinom | size, prob, mu |
| 正态分布 | dnorm, pnorm, qnorm, rnorm | mean, sd |
| 泊松分布 | dpois, ppois, qpois, rpois | lambda |
| 学生 $t$ 分布 | dt, pt, qt, rt | df, ncp |
| 学生化极差分布 | dtukey, ptukey | nmeans, df, nranges |
| 均匀分布 | dunif, punif, qunif, runif | min, max |
| 韦伯分布 | dweibull, pweibull, qweibull, rweibull | shape, scale |
| Wilcoxon 秩和分布 | dwilcox, pwilcox, qwilcox, rwilcox | m, n |
| Wilcoxon 符号秩分布 | dsignrank, psignrank, signrank, rsignrank | n |

```
qnorm(0.025)
```

```
## [1] -1.959964
```

```
qnorm(0.975)
```

```
## [1] 1.959964
```

因此，根据小贴士 17.1 的列表，可以检验某种连续型随机变量的概率分布（用 p 开头的命令，套用到 `ks.test(x, y, ...,alternative = c("two.sided", "less", "greater"),exact = NULL)` 里去），或者生成某种连续型随机变量分布的随机数（用 r 开头的命令），等等。

# 第十八章　用 R 进行基础统计（二）：均值比较和方差分析

开放、绿色、功能强大、具有源源不断巨大资源的 R 不仅有必要而且一定能够在中国推广和发展。

—吴喜之

前面我们说了各种分布检验，那么，知道自己的数据属于什么分布后，再进行均值或者方差的比较就很容易了。

$t$ 检验是适用于正态分布数据集的检验，又分为单正态总体的检验和多个正态总体的检验。

## 18.1　单正态总体的检验

通俗地说，用一个已知的真值来检验一组数据是否服从于以这个真值为中心值的分布。举例：

为了测试一种新的检验大气中二氧化碳含量方法的可靠性，则用此方法来检验一定浓度的标准二氧化碳气体（此情况则为已知真值）。

我们依然以二氧化碳浓度数据举例。假设 1959 年的 12 个观测值为此方法检验了 12 次的结果，真值为 316.45:

```
myco2 <- co2[1:12]
myco2
```

```
##  [1] 315.42 316.31 316.50 317.56 318.13 318.00 316.39
##  [8] 314.65 313.68 313.18 314.66 315.43
```

现在来通过均值检验来看看观测方法方法是否可靠：

```
t.test(myco2, mu = 316.45)
```

```
##
##  One Sample t-test
##
## data:  myco2
## t = -1.3381, df = 11, p-value = 0.2079
## alternative hypothesis: true mean is not equal to 316.45
## 95 percent confidence interval:
##  314.7992 316.8525
## sample estimates:
## mean of x
##  315.8258
```

结果分析：均值 315.8258 看上去和给定的真值 316.45 有一定差异。因此我们需要利用 $t$ 检验的结果来判断这个差异是否在可接受的误差范围内。

第二行的 $p$ 值（$p$-value) 等于 0.2079，大于显著性水平 0.05，因此我们可以认为检测值 315.8258 在允许的误差范围内，这个方法可靠。

从统计学术语来解释是这样的：首先，检验统计量 $t$ 为 -1.3381，自由度为 11，检验的 $p$ 值为 0.2079。$p$ 值的意义是实际样本的均值大于 316.8525 或者小于 314.7992 的概率是 0.2079。

第三行给出备择假设：此方法得到的检验真值不等于 316.45。

根据上述结果，我们不足以拒绝原假设 ‘‘检验结果等于真值 316.45’’，即应该接受原假设，而拒绝备择假设 ‘‘检验值不等于 316.45’’。

接下来的数据是算出的检验值于 95% 显著性水平的置信区间：下限 314.7992，上限 316.8525，即检验值落在这个范围内都是可接受的。

## 18.2　双正态总体的检验

在实际生活中我们很少能用一个已知真值去检验一组数据，更多的是比较两组数据平均值。

依然以二氧化碳浓度为例子。1959 年和 1969 年的浓度，是差不多还是有统计意义的显著差别？

```
myco2.t1 <- co2[1:12] # 1959 年的观测数据
myco2.t2 <- co2[13:24] # 1960 年的观测数据
t.test(myco2.t1,myco2.t2)
```

```
##
##  Welch Two Sample t-test
##
## data:  myco2.t1 and myco2.t2
## t = -1.2262, df = 20.897, p-value = 0.2338
## alternative hypothesis: true difference in means is not
equal to 0
## 95 percent confidence interval:
##  -2.4852531  0.6419197
## sample estimates:
## mean of x mean of y
##  315.8258  316.7475
```

结果分析：两年的均值分别为 315.8258 和 316.7475，看上去有一定差异。因此我们需要利用 $t$ 检验的结果来判断这个差异是真的有统计意义的差异，还是由各种误差导致的可接受范围。

$p$ 值（$p$-value）等于 0.2338，大于显著性水平 0.05，因此我们可以两年

的浓度差值在允许的误差范围内，1959 和 1960 两年的二氧化碳浓度没有显著的差别。

来看看别的年份：

```
myco2.t1 <- co2[1:12] # 1959 年的观测数据
myco2.t3 <- co2[97:108] # 1965 年的观测数据
t.test(myco2.t1,myco2.t3)
```

```
##
##  Welch Two Sample t-test
##
## data:  myco2.t1 and myco2.t3
## t = -8.7432, df = 21.617, p-value = 1.516e-08
## alternative hypothesis: true difference in means is not
equal to 0
## 95 percent confidence interval:
##  -7.664928 -4.723405
## sample estimates:
## mean of x mean of y
##  315.8258  322.0200
```

结果分析：两年的均值分别为 315.8258 和 322.02，看上去差别挺大的。$p$ 值（$p$-value) 等于 1.516e-08，小于极显著性水平 0.01，因此我们可以得出结论：两年的浓度差值已经超过允许的误差范围，1959 和 1965 两年的二氧化碳浓度有极显著的差别。

## 18.3 配对 $t$ 检验

在比较两个数据集时，如果它们是彼此独立的，比如我们刚才用的两年的二氧化碳观测值，两个班的学生对相同样品的检测值，等等。这种情况下我们用普通 $t$ 检验，R 的默认设置是 `paired = FALSE`。但是有一些样本，它们彼此不独立（有共同的属性），有千丝万缕的联系，这时候我们就

不用能独立样本 $t$ 检验了，要在执行语句中设置一个配对参数 paired = TRUE。比如为了检验某种新的降血压药物的效果，服药一段时间后，医生检验病人在服药前后血压是否有显著差异以判断降压药的效果。因为服药前后的数据都是和病人自身体质密切相关的，因此这两组数据不完全独立，就不能使用前面说的独立样本 $t$ 检验，而必须使用配对 $t$ 检验。

类似的例子还有减肥效果的判断——某种减肥方式是否真的有效？

```
require(graphics) # 学生的睡眠数据
with(sleep, t.test(extra[group == 1], extra[group == 2]))
```

```
##
##  Welch Two Sample t-test
##
## data:  extra[group == 1] and extra[group == 2]
## t = -1.8608, df = 17.776, p-value = 0.07939
## alternative hypothesis: true difference in means is not
equal to 0
## 95 percent confidence interval:
##  -3.3654832  0.2054832
## sample estimates:
## mean of x mean of y
##      0.75      2.33
```

```
# 用于比较的两组睡眠数据，10 个学生，受试编号 1-10.
# 先来看一眼数据。ID 是受试学生编号，
# 假定 extra 就是睡眠质量的一个指标，
# group 的编号 1 和 2 分别代表某种治疗前和治疗后。
sleep
```

```
##    extra group ID
## 1    0.7     1  1
## 2   -1.6     1  2
## 3   -0.2     1  3
```

```
## 4   -1.2     1  4
## 5   -0.1     1  5
## 6    3.4     1  6
## 7    3.7     1  7
## 8    0.8     1  8
## 9    0.0     1  9
## 10   2.0     1 10
## 11   1.9     2  1
## 12   0.8     2  2
## 13   1.1     2  3
## 14   0.1     2  4
## 15  -0.1     2  5
## 16   4.4     2  6
## 17   5.5     2  7
## 18   1.6     2  8
## 19   4.6     2  9
## 20   3.4     2 10
```

```
# 先来看看，如果忽视样本之间的联系，用独立样本 t 检验会怎么样
t.test(extra ~ group, data = sleep)
```

```
##
##  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 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
```

看上去，虽然两组数据差别挺大的，但是因为 $p$ 值大于 0.05，因此这个差别还是认为统计不显著。原因就在于 10 个学生本身的个体差异很大，导致他们彼此的测量数据差别很大，因此每组数据的偏差也很大；这样即使两组数据均值看起来差别不小，但是仍然认为统计学意义上的差别不显著。

接下来我们看看如果用配对 $t$ 检验，会得到什么样的结果：

```
require(graphics)  ## 学生的睡眠数据
with(sleep, t.test(extra[group == 1], extra[group == 2]))
```

```
##
##  Welch Two Sample t-test
##
## data:  extra[group == 1] and extra[group == 2]
## t = -1.8608, df = 17.776, p-value = 0.07939
## alternative hypothesis: true difference in means is not
equal to 0
## 95 percent confidence interval:
##  -3.3654832  0.2054832
## sample estimates:
## mean of x mean of y
##      0.75      2.33
```

```
# 用于比较的两组睡眠数据，10 个学生，受试编号 1-10
t.test(extra ~ group, data = sleep, paired = TRUE)
```

```
##
##  Paired t-test
##
## data:  extra by group
## t = -4.0621, df = 9, p-value = 0.002833
## alternative hypothesis: true difference in means is not
equal to 0
```

```
## 95 percent confidence interval:
##  -2.4598858 -0.7001142
## sample estimates:
## mean of the differences
##                   -1.58
```

```
# 考虑到样本之间的联系，用配对 t 检验
```

殷勤的 R 一次把两种检验的结果都列出来了。对比之下可以发现，因为考虑了自身的属性，配对 $t$ 检验的结果证明，睡眠治疗方法是有效的（$p$ 远小于 0.05，治疗前后有显著差异）。

只有采用正确的检验方法，才能得到合理的结果。

统计学和 R 语言一样，都是工具，关键在使用它们的人。否则，拿一位统计学大牛的话来说，就是 “garbage in，garbage out”。所以，请一定要把握好对数据的理解和认识哦。

## 18.4 多组之间均值比较：多组样本的配对 $t$ 检验

如果想知道更多关于组间差异的信息，可以对任意两组数据进行 $t$ 检验。R 中实现这一命令的语句是：

```
pairwise.t.test(x, g, p.adjust.method = p.adjust.methods,
    pool.sd = !paired, paired = FALSE,
    alternative = c("two.sided", "less", "greater"), ...)
```

x 为数值型变量，g 为一个指定分组的因子型变量，pool.sd 指定是否计算所有组的统一标准差。

来看一个例子，以某个城市空气中臭氧含量为例，比较各个月份之间是否有显著差异。

```
attach(airquality)
Month <- factor(Month, labels = month.abb[5:9])
```

```
pairwise.t.test(Ozone, Month)
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  Ozone and Month
##
##     May     Jun     Jul     Aug
## Jun 1.00000 -       -       -
## Jul 0.00026 0.05113 -       -
## Aug 0.00019 0.04987 1.00000 -
## Sep 1.00000 1.00000 0.00488 0.00388
##
## P value adjustment method: holm
```

```
pairwise.t.test(Ozone, Month, p.adj = "bonf")
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  Ozone and Month
##
##     May     Jun     Jul     Aug
## Jun 1.00000 -       -       -
## Jul 0.00029 0.10225 -       -
## Aug 0.00019 0.08312 1.00000 -
## Sep 1.00000 1.00000 0.00697 0.00485
##
## P value adjustment method: bonferroni
```

```
pairwise.t.test(Ozone, Month, pool.sd = FALSE)
```

```
##
##  Pairwise comparisons using t tests with non-pooled SD
```

```
##
## data:  Ozone and Month
##
##     May      Jun      Jul      Aug
## Jun 1.00000 -        -        -
## Jul 0.00026 0.01527 -        -
## Aug 0.00195 0.02135 1.00000 -
## Sep 0.86321 1.00000 0.00589 0.01721
##
## P value adjustment method: holm
```

```
detach()
```

可见，有的月份之间有显著差别，比如 7 月和 9 月；而有的月份之间无显著差异，比如 5 月和 6 月。

多重 $t$ 检验的优点是使用方便，但是均值的多重检验中，如果因素的水平较多，检验又同时进行，多次重复使用会增大第一类错误的概率（前面我们说过的 H0 为真而被误伤，拒绝掉了），所以有时候结论不见得可靠。

为了克服多重 $t$ 检验的缺点，统计学家们提出了一些有效的方法来调整 $p$ 值。这些方法涉及到太多统计学知识，这里只是解释一下 R 中的调整参数设置。在刚才这个例子里我们看到了 `p.adj = "bonf"`，意思是调整方法是 Bonderroni。$p$ 值调整函数是 `p.adj = " "`，其使用方法如下：

```
p.adjust(p, method = p.adjust.methods, n = length(p))
```

`p.adjust.methods:c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none")`，选择其中一种。`"none"` 表示不作任何调整，默认设置是`"Holm"`，即按 Holm 方法调整。

## 18.5　方差分析

方差的意义就是总体样本的离散程度——即一个数据集里，所有数据的分散程度。这种分散程度，也就是观测值的波动。引起这种波动的原因主要有两类，一类是不可控的随机因素干扰或者观测误差；另一类是因为不同的处理方式或者实验设计引起的可控波动。方差分析的目的就是将数据的总波动分解为上述两类，并且做出数量分析，以比较这两类波动所占的比例，从而进一步分析数据或者改善实验设计。因此，方差分析的本质也是一种均值检验，通过对方差来源进行解析，推测某一个或多个因素下各水平的因变量均值是否有明显差异，从而判断那些因素有显著影响。

前面我们说过 $t$ 检验用于检验两个正态总体均值是否相等，比如对照组和实验组的差异。方差分析则可以检验多个总体（多个组）的均值是否存在差异。

### 18.5.1　单因素方差分析

单因素方差分析就是只考虑一个因素对结果的影响。在 R 中执行这个分析的函数是 `oneway.test()`, 它的格式为

```
oneway.test(x ~ group, var.equal = T)
```

x 为样本观察值，group 为描述分类情况的因子（当然也可以用别的名字）。以 R 自带的小鸡生长数据为例。

```
oneway.test(weight ~ feed, data = chickwts, var.equal = T)
```

```
##
##  One-way analysis of means
##
## data:  weight and feed
## F = 15.365, num df = 5, denom df = 65, p-value =
## 5.936e-10
```

$F$ 检验统计量为 15.365，而 $p$ 值小于 0.05，因此拒绝原假设，不同的小鸡喂养方式有极显著差异。

学生睡眠数据的例子：

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

```
##
##  One-way analysis of means (not assuming equal
##  variances)
##
## data:  extra and group
## F = 3.4626, num df = 1.000, denom df = 17.776,
## p-value = 0.07939
```

```
# var.equal 这个设置如果为 TRUE，则进行单因素 F 检验。
# 反之则用另一种计算方法 Welch method，属于双因素 Welch 检验的
# 一种近似算法
oneway.test(extra ~ group, data = sleep, var.equal = TRUE)
```

```
##
##  One-way analysis of means
##
## data:  extra and group
## F = 3.4626, num df = 1, denom df = 18, p-value =
## 0.07919
```

$p$ 值大于 0.05，两组没有显著差异（跟独立样本 $t$ 检验一样的结论）。

更多分析的详细信息可以通过函数 `anova()` 和 `aov()` 得到，注意，应用函数 `anova()` 需要在线性模拟函数 `lm()` 产生的结果基础上调用。仍然以小鸡喂养的例子：“feed” 这一行给出了组间离差平方和。$F$ 值和 $p$ 值，和 `oneway.test()` 给出的结果是一致的。此外，也可以使用 `aov()` 代替 `anova(lm)` 组合，但是，要得到输出结果需要加上 `summary()` 命令。

```
summary(aov(weight ~ feed, data = chickwts))
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## feed          5 231129   46226   15.37 5.94e-10 ***
## Residuals    65 195556    3009
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

得到的结果也是一样的。

### 18.5.2　不考虑交互作用的双因素方差分析

双因素方差分析这一节的例子，都来自于包 geepack (Højsgaard et al., 2016)。（注意，我们只是利用这个包的数据来学习 R 语句的使用，相应的方差分析结果是没有实际意义的。）

顾名思义，不考虑交互作用，就是两个因素各自作用于因变量，看看各自的变异对结果有无影响。

R 中的执行语句是

```
aov(formula, data = NULL, projections = FALSE, qr = TRUE,
    contrasts = NULL, ...)
```

我们以数据集 dietox 为例，看看两个因素 `Cu` 和 `Evit` 对猪体重的影响是否显著。

```
library(geepack)
data(dietox)
weight <- aov(Weight ~ Cu + Evit, data = dietox)
summary(weight)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## Cu            1    521   521.2   0.835  0.361
```

```
## Evit          1    253   252.6   0.405  0.525
## Residuals   858 535818   624.5
```

两项 $F$ 统计量的 $p$ 值都大于 0.05，可见这两项对体重并无显著影响。（再次注意这结果的意义并没有真实存在，意思就是那两项没影响。）

### 18.5.3 考虑交互作用的双因素方差分析

考虑交互作用，就说两种影响因素彼此不完全独立，有交互作用。还是以刚才的数据为例，注意 R 的执行语句变了：

```
data(dietox)
weight <- aov(Weight ~ Cu + Evit + Cu : Evit, data = dietox)
# 增加了一项，用“:”链接两因素，表示交互作用
summary(weight)
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## Cu             1    521   521.2   0.834  0.361
## Evit           1    253   252.6   0.404  0.525
## Cu:Evit        1    165   164.9   0.264  0.608
## Residuals    857 535654   625.0
```

从结果可见，不仅这两个因素都没有显著作用，而且它们的交互作用也不显著。这里需要想清楚，我们这个题目叫“需要考虑交互作用”，并不是说就一定有交互作用，而是可能有，因此需要你来检验是否真的有交互作用。再看一下有影响的因素。以 geepack 中的哮喘数据 seizure 为例。

```
data(seizure)
seiz.l <- reshape(seizure,
    varying = list(c("base","y1", "y2", "y3", "y4")),
    v.names="y", times = 0:4, direction = "long")
seiz.l <- seiz.l[order(seiz.l$id, seiz.l$time),]
seiz.l$t <- ifelse(seiz.l$time == 0, 8, 2)
seiz.l$x <- ifelse(seiz.l$time == 0, 0, 1)
```

```
result <- aov(y ~ x + trt + x : trt, data=seiz.l)
summary(result)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## x             1  24895   24895  93.181 <2e-16 ***
## trt           1      8       8   0.032  0.859
## x:trt         1     26      26   0.099  0.753
## Residuals   291  77746     267
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

结果表明 x 这一项是有影响的，但是 trt 这项没有影响，而且它与 x 也没有交互作用。

再看看小鸡数据中别的几个因素是否有交互作用。

```
data(dietox)
weight <- aov(Weight ~ Cu + Time + Cu : Time, data = dietox)
# 增加了一项，用“:”链接两因素，表示交互作用
summary(weight)
```

```
##              Df Sum Sq Mean Sq F value  Pr(>F)
## Cu            1    521     521   10.22 0.00144 **
## Time          1 492382  492382 9658.42 < 2e-16 ***
## Cu:Time       1      0       0    0.00 0.99629
## Residuals   857  43689      51
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

从这个结果看，两因素 Time 和 Cu 都有显著影响，但它们并无交互作用。

## 18.6 非参数假设检验

在前一节我们已经介绍了如何检验一个数据集是否服从某种分布。知道某一数据集服从正态分布，就可以进行相应的均值检验了（$t$ 检验，方差分析等）。但是，很多情况下，随机变量的分布类型是未知的。比如降雨的 pH 值属于某种参数分布，而降水中的硫酸根离子浓度，因为是经过了雨水体积的加权平均计算得到，所以我们就不知道它属于什么分布了。硫酸根离子的浓度往往能反映降水污染的程度。如果酸雨研究者需要比较两个城市的降水污染严重程度是否一致，或者需要比较某种调控措施后降水污染程度是否有显著改善。该如何来检验呢？

与前面我们说的参数假设检验相对应，我们把对分布一无所知的数据检验称为非参数假设检验。我们需要应用一种不必依赖某一专门的总体分布的统计分布，称之为 ‘‘不需要分布 (distribution free) 统计法’’。总体分布不确定，即与参数无关，这样的检验通常将数据转换为秩（rank）来进行分析。所谓秩，就是数据按照升幂排列后，每个观测值的位置。利用数据的秩来分析，就避免了不知道数据分布的困扰，这就是大多数非参数检验的优点。本节我们介绍 R 如何利用秩检验来比较：两个样本的数据是否来自同一总体？是否服从同样分布？是否有同样的中位数（类似于参数检验中的，两个样本均值是否相同）？

常用 Wilcoxon 检验，也就是不依赖分布的 $t$ 检验。

在进行检验之前我们往往通过图示先得到一个初步的了解。常用的绘图有：（前面都有介绍，所以就不举例了。只说作用）

- 直方图（histogram）：频数用矩形块标绘，通过直方图推测分布的密度函数对比，得到直观印象。
- 茎叶图（Stem-and-leaf Diagrams）：将数据集中的数据按位数进行比较，把数的大小基本不变或者变化不大的位作为主干（茎），把变化大的位的数作为分枝（叶），列在主干的后面，这样每个主干后面几个数，一目了然。
- Q-Q 图：仅用于检验正态性。

### 18.6.1　单样本检验

Wilcoxon 检验中的单样本检验叫 Wilcoxon signed rank test，有的翻译为“Wilcoxon 符号秩检验”。这个检验利用样本的观察值和假设的中心位置之差来进行比较检验，不同的符号代表在中心位置的哪一边，差的绝对值的秩大小代表和假设中心的距离远近。R 中的执行语句为：

```
wilcox.test(x, y = NULL,
    alternative = c("two.sided", "less", "greater"), mu = 0,
    paired = FALSE, exact = NULL, correct = TRUE,
    conf.int = FALSE, conf.level = 0.95, ...)
```

来看一个例子。还是以臭氧浓度数据为例，看看 5 月到 8 月的臭氧浓度数据是否来自于中心位置为 36 的总体，因此给定的 mu 值是 36。

```
wilcox.test(airquality$Ozone, mu = 36,
    subset = Month %in% c(5, 8), exact = NULL,
    correct = TRUE,conf.int = FALSE, conf.level = 0.95)
```

```
##
##  Wilcoxon signed rank test with continuity
##  correction
##
## data:  airquality$Ozone
## V = 3503.5, p-value = 0.5237
## alternative hypothesis: true location is not equal to 36
```

$p$ 值大于 0.05，因此接受原假设，即这些数据来自中心位置为 36 的总体。

### 18.6.2　两独立样本秩检验

两独立样本的秩检验，就是要检验两个样本的中位数是否相等。

```
require(graphics)
# 读者不用挨个输入这些数据。
# 这些数据和代码都来自在线帮助 wilcox.test
x <- c(1.83,  0.5,  1.62,  2.48, 1.68, 1.88, 1.55, 3.06, 1.3)
y <- c(0.878, 0.647,0.598, 2.05, 1.06, 1.29, 1.06, 3.14,1.29)
wilcox.test(x, y, paired = TRUE, alternative = "greater")
```

```
##
##  Wilcoxon signed rank test
##
## data:  x and y
## V = 40, p-value = 0.01953
## alternative hypothesis: true location shift is greater
than 0
```

```
# “alternative”设置的意义和前面的参数检验是一样的
wilcox.test(y - x, alternative = "less")
```

```
##
##  Wilcoxon signed rank test
##
## data:  y - x
## V = 5, p-value = 0.01953
## alternative hypothesis: true location is less than 0
```

```
# 和上一句命令的意义一样
wilcox.test(y - x, alternative = "less",
            exact = FALSE, correct = FALSE)
```

```
##
##  Wilcoxon signed rank test
##
## data:  y - x
## V = 5, p-value = 0.01908
```

```
## alternative hypothesis: true location is less than 0
```

```
# exact 的设置表示是否需要计算出精确的 p 值，
# correct 的设置意义是，样本量很大时是否需要做连续型修正
```

$p$ 小于 0.05，我们可以拒绝原假设，即两个样本的中位数不相等。

### 18.6.3　Mann-Whitney U 检验

各文献中对 Wilcoxon 秩检验和 Mann-Whitney U 检验的定义仍然不是完全统一。本节就从 R 在这方面的应用出发，根据 `wilcox.test()` 函数对这两种检验加以说明。

与 Wilcoxon 秩和统计量等价的有 Mann-Whitney U 统计量。简单地说，Wilcoxon 比较的是样本的大小排列秩序，而 Mann-Whitney U 比较的是样本 1 的观察值大于样本 2 的观察值的个数。这两个统计量之间可以通过一个数学公式换算（有兴趣的话可以去找统计学书来看看）

设 $X_1$，$X_2$,…,$X_m$ 来自于连续型总体 $X$（容量为 $m$），$Y_1$，$Y_2$,…,$Y_n$ 来自于连续型总体 $Y$（容量为 $n$），且 $X$ 和 $Y$ 相互独立。

- $M_X$：总体 $X$ 的中位数
- $M_Y$：总体 $Y$ 的中位数
- $W_{XY}$：把总体 $X$ 和 $Y$ 的观察值做比较之后，$Y$ 的观察值大于 $X$ 的个数。

$W_{XY}$ 就是 Mann-Whitney U 统计量，它和 wilcoxon 秩和统计量的关系如下：

$$W_Y = W_{XY} + \frac{n(n+1)}{2}, W_X = W_{YX} + \frac{m(m+1)}{2}.$$

在 R 中执行 u 检验的命令依然是 `wilcox.test()`，但是在参数设置上有所区分：

```
wilcox.test(x, y = NULL,
    alternative = c("two.sided", "less", "greater"),
    mu = 0, paired = FALSE, exact = NULL, correct = TRUE,
    conf.int = FALSE, conf.level = 0.95, ...)
```

paired = TRUE 表示 Wilcox 秩和检验，paired = FALSE 表示 Mann-Whitney U 检验。

```
## randu 是 R 自带的数据集
x <- randu$x
y <- randu$y
wilcox.test(x, y, paired = FALSE)  # Wilcox 秩和检验
```

```
##
##  Wilcoxon rank sum test with continuity
##  correction
##
## data:  x and y
## W = 86324, p-value = 0.05301
## alternative hypothesis: true location shift is not equal
to 0
```

```
wilcox.test(x, y, paired = TRUE)  # Mann-Whitney U 检验
```

```
##
##  Wilcoxon signed rank test with continuity
##  correction
##
## data:  x and y
## V = 44406, p-value = 0.06277
## alternative hypothesis: true location shift is not equal
to 0
```

可见因为 Wilcoxon 秩和统计量和 Mann-Whitney U 统计量有固定的

数学关系，因此，即使他们计算出来的统计值有所不同（w 和 v），但是检验的结果是一致的，原假设也是一致的。$p$ 值大于 0.05，无法拒绝原假设，即 x 和 y 来自同一中心位置的总体。

### 18.6.4　多个独立样本的秩和检验

多个独立样本的非参数检验方法是 Kruskal-Wallis 秩和检验，这个检验的目的是看多个总体的位置参数是否一样。

这些样本之间都是彼此独立的，也就是说，他们的分布函数有可能形状相同，只是位置参数不同。

这个检验的原假设是，这些样本的位置参数全部相等，因此相应的备择假设就是，不全都相等（至少有一个不一样）。

这个检验对应的参数检验就是 Anova 检验，在非参数检验中，则用秩代替具体数值，再用 Anova 方法做统计分析。

来看一个例子。

```
boxplot(weight ~ group, data = PlantGrowth)
```

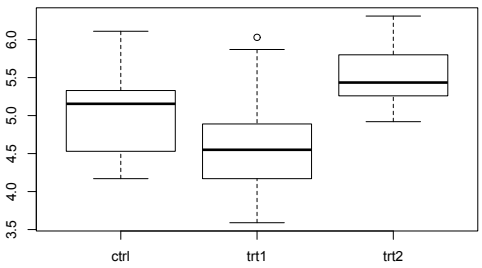



图 18.1: 不同组别的植物的重量箱体图

```
# 做比较前先来个直观了解
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
```

$p$ 值小于 0.05，拒绝原假设，即至少有一组的位置参数不同。

### 18.6.5 多个相关样本的秩和检验

多个总体分布、多个配对样本的差异检验方法是 `friedman.test()`。这个和配对 $t$ 检验的感觉相似，参与比较的样本之间不完全独立，往往是匹配的实验设计。`friedman.test()` 要求的数据是向量或矩阵，否则会报错。

来看一个例子。这是 `friedman.test` 在线帮助中使用的数据，比较三种方法击球的差异。

```
RoundingTimes <-  matrix(c(5.40, 5.50, 5.55,
                           5.85, 5.70, 5.75,
                           5.20, 5.60, 5.50,
                           5.55, 5.50, 5.40,
                           5.90, 5.85, 5.70,
                           5.45, 5.55, 5.60,
                           5.40, 5.40, 5.35,
                           5.45, 5.50, 5.35,
                           5.25, 5.15, 5.00,
                           5.85, 5.80, 5.70,
                           5.25, 5.20, 5.10,
```

```
                          5.65, 5.55, 5.45,
                          5.60, 5.35, 5.45,
                          5.05, 5.00, 4.95,
                          5.50, 5.50, 5.40,
                          5.45, 5.55, 5.50,
                          5.55, 5.55, 5.35,
                          5.45, 5.50, 5.55,
                          5.50, 5.45, 5.25,
                          5.65, 5.60, 5.40,
                          5.70, 5.65, 5.55,
                          6.30, 6.30, 6.25),
    nrow = 22,byrow = TRUE,
    dimnames = list(1:22,
        c("Round Out", "Narrow Angle", "Wide Angle")))
friedman.test(RoundingTimes)
```

```
##
##  Friedman rank sum test
##
## data:  RoundingTimes
## Friedman chi-squared = 11.143, df = 2, p-value =
## 0.003805
```

$p$ 值小于 0.05，拒绝原假设，可见三种方法有显著差异。

**练习 18.1.** 比较数据集 randu 里的 x，y，z 三组数据均值是否有明显差异，做 $t$ 检验和多组样本的 $t$ 检验以及配对 $t$ 检验。

# 第十九章　相关性分析和协方差

## 19.1　相关性检验及可视化

做数据分析时，往往需要知道两个变量是否相关。相关系数可以用来衡量两个随机变量之间的线性相关程度，其取值范围是 -1 到 1。通常用来衡量相关关系的统计量是 Pearson 相关系数，也正是 Excel 软件中用 CORREL 函数计算出来的结果。因为 Pearson 相关系数是基于正态分布计算的，因此对于服从正态分布的变量来说度量效果更好。

另外一个统计量是 Spearman 相关系数。正如我们前面所说，有很多数据是不服从正态分布的，甚至都不知道它是否服从于某一种已知的分布。因此，Spearman 相关系数是一个非参数统计量，它不对数据的分布情况做出假设。

此外，还有一个统计量叫做 Kendall T，它比较的是两个随机变量的秩，而不是比较随机变量的数值。显然，这种统计量对变量的分布情况更没有要求。我们以 R 自带的数据集 iris（鸢尾花数据集）为例来说明各种用途的相关检验。方法以及如何将相关性检验的结果漂亮地可视化。

```
data(iris)  # 鸢尾花数据集
head(iris)  # 查看下数据集
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          5.1         3.5          1.4         0.2
## 2          4.9         3.0          1.4         0.2
```

```
## 3          4.7         3.2          1.3         0.2
## 4          4.6         3.1          1.5         0.2
## 5          5.0         3.6          1.4         0.2
## 6          5.4         3.9          1.7         0.4
##   Species
## 1  setosa
## 2  setosa
## 3  setosa
## 4  setosa
## 5  setosa
## 6  setosa
```

```
iris.data <- iris[, -5]  # 去掉最后一列，非数字变量
```

首先，R 内置的 `cor()` 函数就可以计算两个数据集直接的相关系数。

```
cor(iris.data)
```

```
##              Sepal.Length Sepal.Width Petal.Length
## Sepal.Length    1.0000000  -0.1175698    0.8717538
## Sepal.Width    -0.1175698   1.0000000   -0.4284401
## Petal.Length    0.8717538  -0.4284401    1.0000000
## Petal.Width     0.8179411  -0.3661259    0.9628654
##              Petal.Width
## Sepal.Length   0.8179411
## Sepal.Width   -0.3661259
## Petal.Length   0.9628654
## Petal.Width    1.0000000
```

但是，这个分析只给出了相关系数，我们还需要知道显著性水平（$p$ 值）才能确定这种相关关系的意义。

用 R 自带的 `cor.test()` 函数可以得到我们想要的信息：

```
cor.test(x, y,
    alternative = c("two.sided", "less", "greater"),
    method = c("pearson", "kendall", "spearman"),
    exact = NULL, conf.level = 0.95, continuity = FALSE, ...)
```

- x, y：需要检测的数据集。
- alternative：备择假设，“two.sided”, “greater” 和 “less” 选一个。
- method：上面所说的三种统计量，根据数据实际情况选一种。
- exact：是否需要精确计算出 $p$ 值。
- conf.level：置信区间，只对 Pearson 相关系数有效（要求 x，y 各包含至少四个数据）
- continuity：是一个逻辑变量。如果选择 TRUE，则对 Kendall's 检验的 tau 值和 Spearman's 检验的 rho 值进行连续性矫正。

以 R 自带的数据 airquality 为例，我们来看一下检验的过程。

```
data(airquality)
head(airquality)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

```
cor.test(airquality$Wind, airquality$Temp)
```

```
##
##  Pearson's product-moment correlation
##
## data:  airquality$Wind and airquality$Temp
## t = -6.3308, df = 151, p-value = 2.642e-09
```

```
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.5748874 -0.3227660
## sample estimates:
##       cor
## -0.4579879
```

默认的是 Pearson 相关，$p$ 值远小于 0.01，可见风速与气温显著相关，相关系数为 -0.457，为负相关。

如果我们选择 kendall 统计量：

```
cor.test(airquality$Wind, airquality$Temp, method = "kendall")
```

```
##
##  Kendall's rank correlation tau
##
## data:  airquality$Wind and airquality$Temp
## z = -5.7059, p-value = 1.157e-08
## alternative hypothesis: true tau is not equal to 0
## sample estimates:
##       tau
## -0.3222418
```

结果相似，因此选择哪一种统计量，需要结合数据本身的特性。

这样两个两个地比较变量，显然是很笨拙的方法，尤其是面对一个变量很多的数据集的时候。因此，R 的 psych 包 (Revelle, 2017) 提供了一次给出相关系数矩阵的命令：

```
if (!require(psych)) install.packages("psych")
library(psych)
corr.test(iris.data)
```

```
## Call:corr.test(x = iris.data)
## Correlation matrix
```

```
##              Sepal.Length Sepal.Width Petal.Length
## Sepal.Length         1.00       -0.12         0.87
## Sepal.Width         -0.12        1.00        -0.43
## Petal.Length         0.87       -0.43         1.00
## Petal.Width          0.82       -0.37         0.96
##              Petal.Width
## Sepal.Length        0.82
## Sepal.Width        -0.37
## Petal.Length        0.96
## Petal.Width         1.00
## Sample Size
## [1] 150
## Probability values (Entries above the diagonal are
adjusted for multiple tests.)
##              Sepal.Length Sepal.Width Petal.Length
## Sepal.Length         0.00        0.15            0
## Sepal.Width          0.15        0.00            0
## Petal.Length         0.00        0.00            0
## Petal.Width          0.00        0.00            0
##              Petal.Width
## Sepal.Length           0
## Sepal.Width            0
## Petal.Length           0
## Petal.Width            0
##
##  To see confidence intervals of the correlations, print
with the short=FALSE option
```

相关系数和 $p$ 值一并给出来了。

可以只提取相关系数:

```
corr.test(iris.data)$r
```

```
##              Sepal.Length Sepal.Width Petal.Length
## Sepal.Length    1.0000000  -0.1175698    0.8717538
## Sepal.Width    -0.1175698   1.0000000   -0.4284401
## Petal.Length    0.8717538  -0.4284401    1.0000000
## Petal.Width     0.8179411  -0.3661259    0.9628654
##              Petal.Width
## Sepal.Length   0.8179411
## Sepal.Width   -0.3661259
## Petal.Length   0.9628654
## Petal.Width    1.0000000
```

或者只提取 $p$ 值：

```
corr.test(iris.data)$p
```

```
##              Sepal.Length  Sepal.Width Petal.Length
## Sepal.Length    0.0000000 1.518983e-01 0.000000e+00
## Sepal.Width     0.1518983 0.000000e+00 1.353994e-07
## Petal.Length    0.0000000 4.513314e-08 0.000000e+00
## Petal.Width     0.0000000 4.073229e-06 0.000000e+00
##               Petal.Width
## Sepal.Length 0.000000e+00
## Sepal.Width  8.146457e-06
## Petal.Length 0.000000e+00
## Petal.Width  0.000000e+00
```

注意，此函数默认输出结果只保留小数点后两位，并且不可以采用常用的 `option(digitals = 3)` 或者 `round(x, 3)` 来指定小数位数。只能通过使用 `print()` 函数来对输出结果中的小数位数加以指定：

```
print(corr.test(iris.data)$r, digits = 3)
```

```
##              Sepal.Length Sepal.Width Petal.Length
```

```
## Sepal.Length        1.000      -0.118        0.872
## Sepal.Width        -0.118       1.000       -0.428
## Petal.Length        0.872      -0.428        1.000
## Petal.Width         0.818      -0.366        0.963
##              Petal.Width
## Sepal.Length       0.818
## Sepal.Width       -0.366
## Petal.Length       0.963
## Petal.Width        1.000
```

有时候我们得到的相关系数矩阵很庞大，看上去眼花缭乱，难以快速地获取有效信息。这时候就需要将相关系数矩阵优雅漂亮地做出图来。R 中提供了好几个包可以达到这个目的，我们依然以鸢尾花数据为例来展示这些包的使用。首先看看数据两两之间的散点图（图 19.1）：

```
if (!require(graphics)) install.packages(graphics)
require(graphics)
pairs(iris)
```

先安装和载入需要的 corrplot 包 (Wei and Simko, 2016)：

```
if (!require(corrplot)) install.packages("corrplot")
library(psych)
library(corrplot)
```

corrplot() 命令的默认结果如下（图 19.2）：

```
corrplot(corr.test(iris.data)$r)
```

事实上，corrplot() 这个命令有多种选择可以让结果图更明确或者有个性 (看看函数使用格式就知道)：

```
corrplot(corr, method = c("circle", "square", "ellipse",
  "number", "shade", "color", "pie"), type = c("full",
  "lower", "upper"), add = FALSE, col = NULL, bg = "white",
  title = "", is.corr = TRUE, diag = TRUE, outline = FALSE,
```

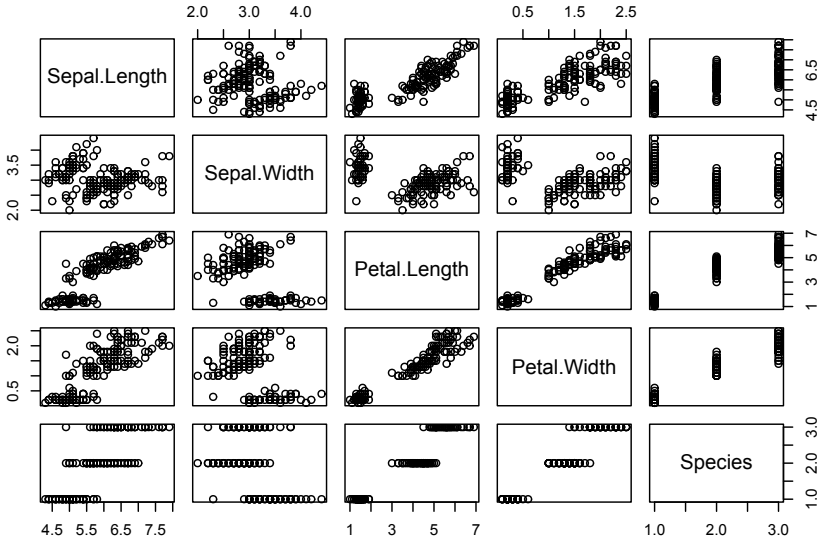



图 19.1: 鸢尾花数据集里的两两散点图

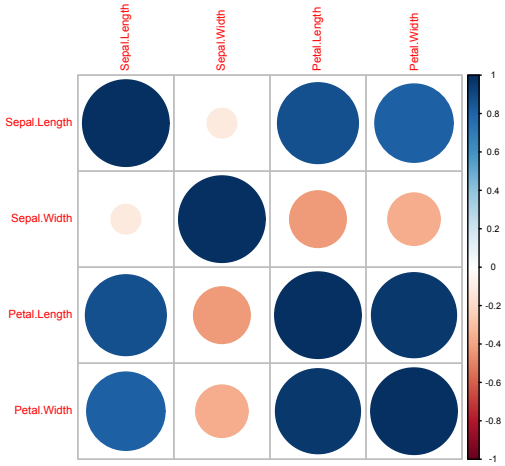



图 19.2: corrplot() 默认设置作图示例

```
mar = c(0,0,0,0), addgrid.col = NULL, addCoef.col = NULL,
addCoefasPercent = FALSE, order = c("original", "AOE",
"FPC", "hclust", "alphabet"), hclust.method = c("complete",
"ward", "ward.D", "ward.D2", "single", "average",
"mcquitty", "median", "centroid"), addrect = NULL,
rect.col = "black", rect.lwd = 2, tl.pos = NULL, tl.cex =1,
tl.col = "red", tl.offset = 0.4, tl.srt = 90, cl.pos =NULL,
cl.lim = NULL, cl.length = NULL, cl.cex = 0.8,
cl.ratio = 0.15, cl.align.text = "c", cl.offset = 0.5,
number.cex = 1, number.font = 2, number.digits = NULL,
addshade = c("negative", "positive", "all"), shade.lwd = 1,
shade.col = "white", p.mat = NULL, sig.level = 0.05,
insig = c("pch", "p-value", "blank", "n"), pch = 4,
pch.col = "black", pch.cex = 3,
plotCI = c("n", "square", "circle", "rect"),
lowCI.mat = NULL, uppCI.mat = NULL, na.label = "?",
na.label.col = "black", ...)
```

以及各种不厌其烦的参数设置：

- `corr`：相关系数矩阵。
- `method`：这是一个类型变量选择，可以选择为 “circle” (default), “square”, “ellipse”, “number”, “pie”, “shade” and “color”. 下面我们会举例来说明。方块或者圆圈的面积直接代表相关系数的数值。
- `type`：也是类型变量，可选的有 “full” (默认), “upper” or “lower”, 展示为全图，或者矩阵的上半部分或下半部分。add：逻辑型变量，如果选择 TRUE，则在已有图形上加上新图形，否则另开一个作图窗口。
- `col`：表示颜色的变量，如果不设置，默认为，颜色均一分布，为 colorRampPalette(col2)(200)。
- `bg`：背景颜色。
- `title`：图标题。
- `is.corr`：逻辑型变量，设置输入的矩阵是否是相关系数矩阵。如果不是相关系数矩阵，可以通过设置 `is.corr = FALSE` 来进行作图。

- diag：逻辑型变量，是否要在主对角线写出相关系数。
- outline：逻辑型变量，如果设置为 TRUE，则默认选择为黑色。
- mar：和绘图中 par 的意义一样。
- addgrid.col：格子颜色。
- addCoef.col：图上系数的颜色
- addCoefasPercent：逻辑型变量，是否要将系数转换为空间百分比。
- order：类型变量，如何在图中安排变量顺序。
  - "original"——默认按输入文件的变量顺序。
  - "AOE" —— 按照特征向量的夹角顺序排列.
  - "FPC" ——按主成分排序。
  - "hclust"——聚类顺序。
  - "alphabet" ——变量字母顺序。

这些是主要的参数设置，其他一些更细节的设置，大家可以通过 ??corrplot 来获得在线帮助。下面我们通过不同的参数设置来感受一下作图的效果。以 R 自带的汽车数据为例。

```
data(mtcars)
M <- cor(mtcars)
# different color series
col1 <- colorRampPalette(c("#7F0000","red","#FF7F00",
    "yellow","white", "cyan", "#007FFF", "blue","#00007F"))
col2 <- colorRampPalette(c("#67001F", "#B2182B", "#D6604D",
    "#F4A582", "#FDDBC7", "#FFFFFF", "#D1E5F0", "#92C5DE",
    "#4393C3", "#2166AC", "#053061"))
col3 <- colorRampPalette(c("red", "white", "blue"))
col4 <- colorRampPalette(c("#7F0000","red","#FF7F00",
    "yellow","#7FFF7F", "cyan", "#007FFF", "blue","#00007F"))
wb <- c("white","black")
```

```
# 不同设置的图例
library(corrplot)
par(mfrow = c(2, 1))
corrplot(M, method = "number")
```

```
corrplot(M)
```

```
# 不同颜色设置的效果
corrplot(M, order = "AOE", col = col1(20), cl.length = 21,
         addCoef.col = "grey")
```

```
# 不同色块和形状设置的效果
par(mfrow = c(3,1))
corrplot(M, method = "ellipse", col = col1(200),order ="AOE")
corrplot(M, method = "shade", col = col3(20),order = "AOE")
corrplot(M, method = "pie", order = "AOE")
```

```
# 中国围棋的感觉
par(mfrow = c(2,1))
corrplot(M, col = wb, order="AOE", outline=TRUE, cl.pos="n")
corrplot(M, col = wb, bg="gold2",  order="AOE", cl.pos="n")
```

```
# mixed methods: It's more efficient if using
# function "corrplot.mixed"
# circle + ellipse
corrplot(M,order = "AOE", type = "upper", tl.pos = "d")
corrplot(M,add = TRUE, type = "lower", method = "ell",
    order = "AOE", diag=FALSE, tl.pos= "n", cl.pos = "n")
```

```
par(mfrow = c(2, 1))
## visualize a matrix in [-100, 100]
ran <- round(matrix(runif(225, -100, 100), 15))
corrplot(ran, is.corr = FALSE)
corrplot(ran, is.corr = FALSE, cl.lim = c(-100, 100))
```

```
par(mfrow = c(3, 1))
## text-labels and plot type

corrplot(M, order = "AOE", tl.srt = 60)
```

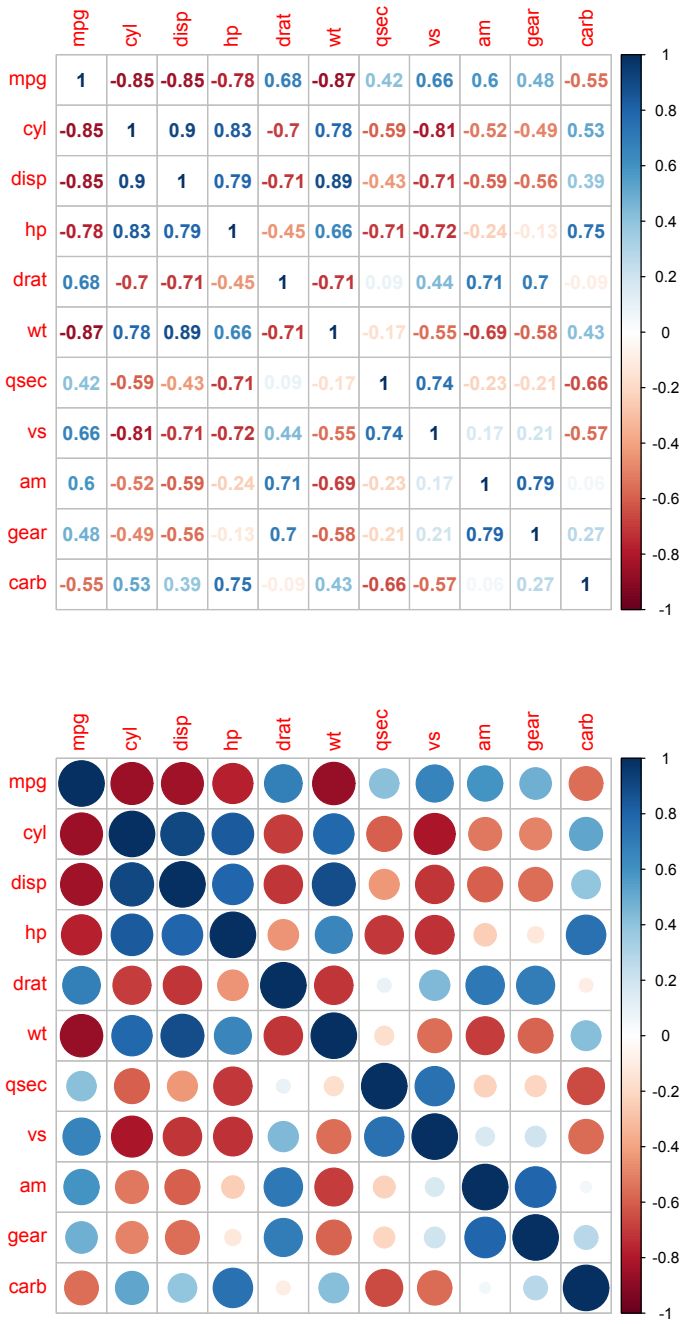





图 19.3：不同设置的图例

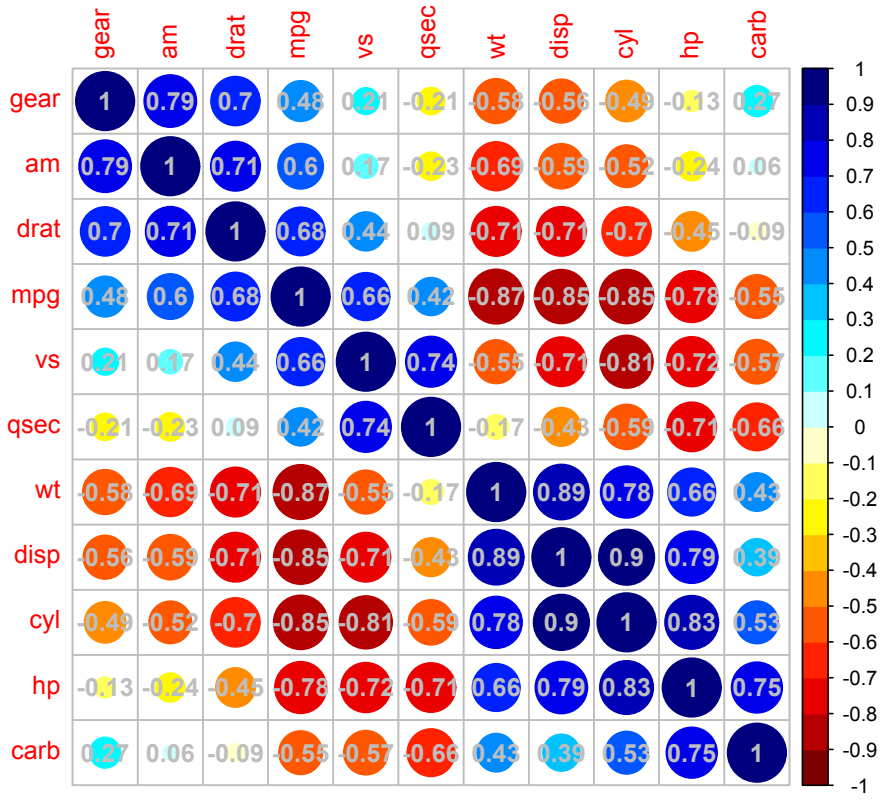



图 19.4: 不同颜色设置的效果

```
corrplot(M, order = "AOE", diag = FALSE, tl.pos = "d")
corrplot(M, order = "AOE", type = "lower", cl.pos = "b")
```

```
cor.mtest <- function(mat, conf.level = 0.95){
  mat <- as.matrix(mat)
  n <- ncol(mat)
  p.mat <- lowCI.mat <- uppCI.mat <- matrix(NA, n, n)
  diag(p.mat) <- 0
```

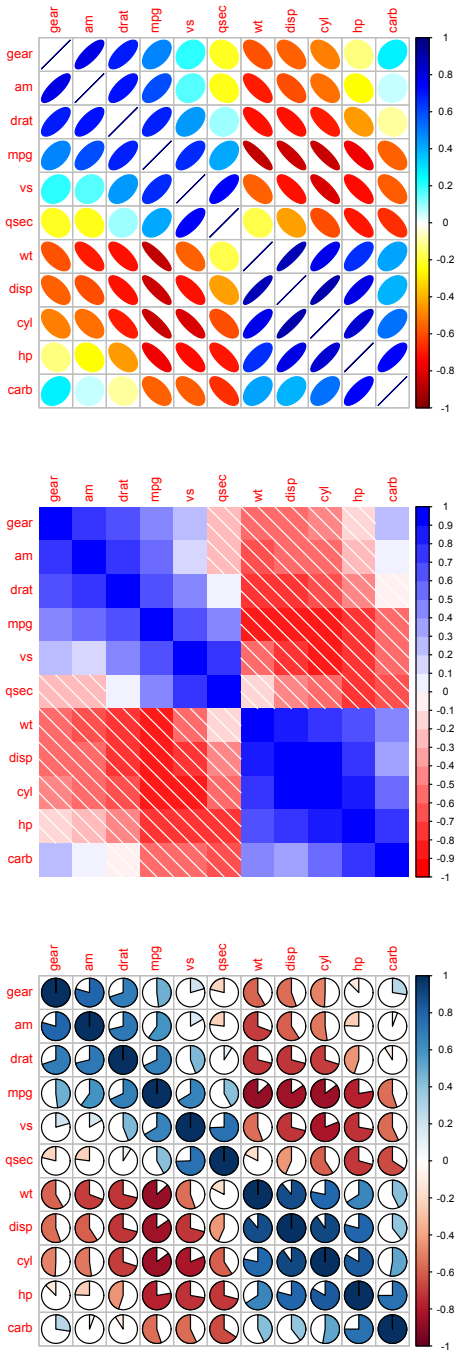





图 19.5：不同色块和形状设置的效果

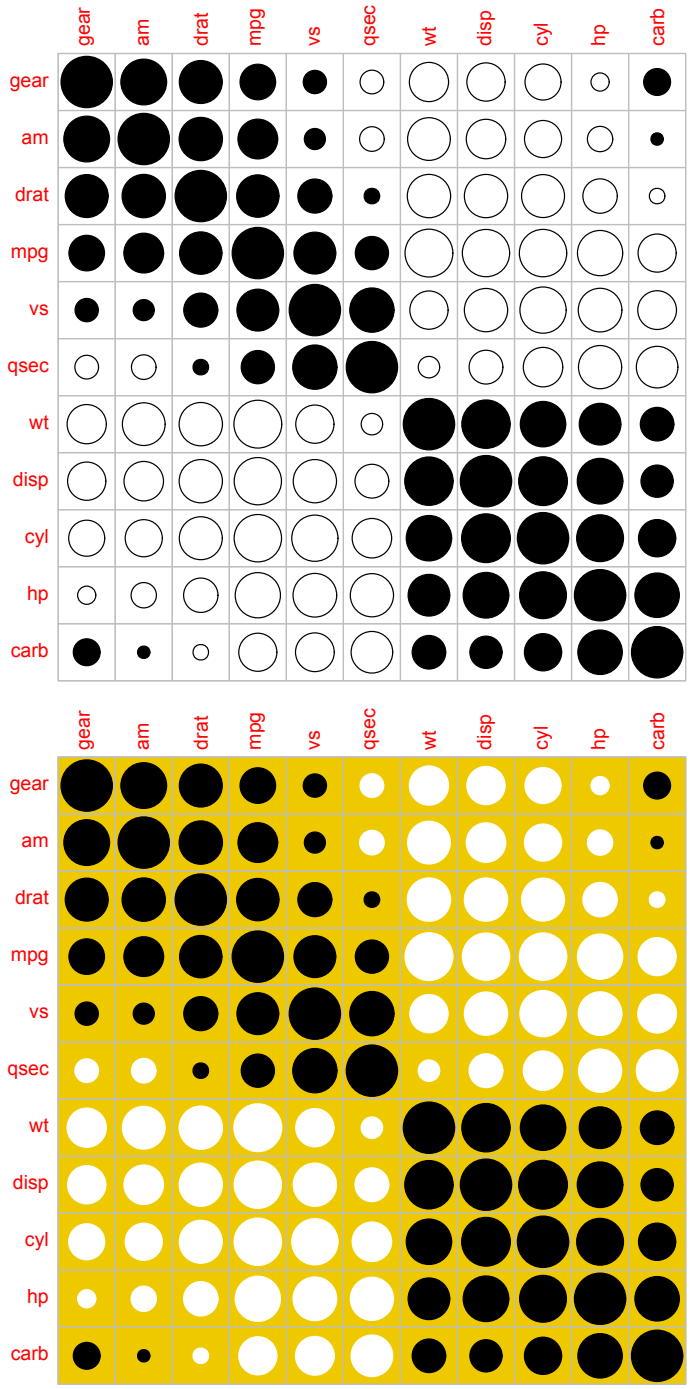



图 19.6: 设置为中国围棋感觉的效果

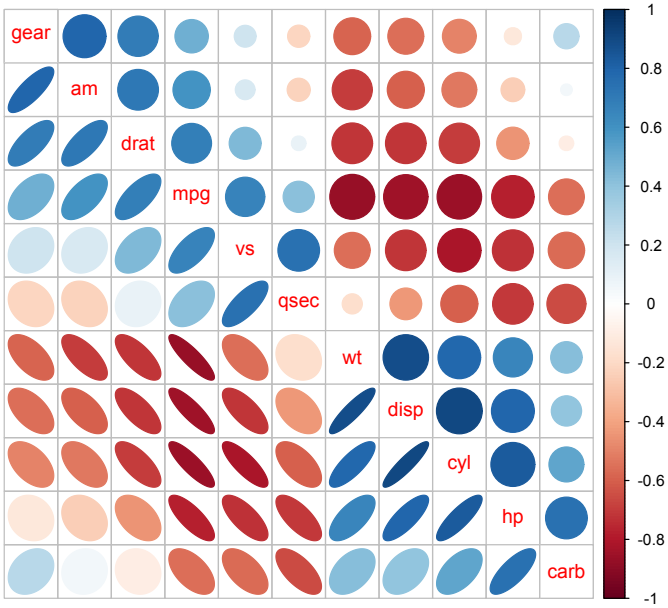

gear
am
drat
mpg
vs
qsec
wt
disp
cyl
hp
carb
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

图 19.7：不同形状组合

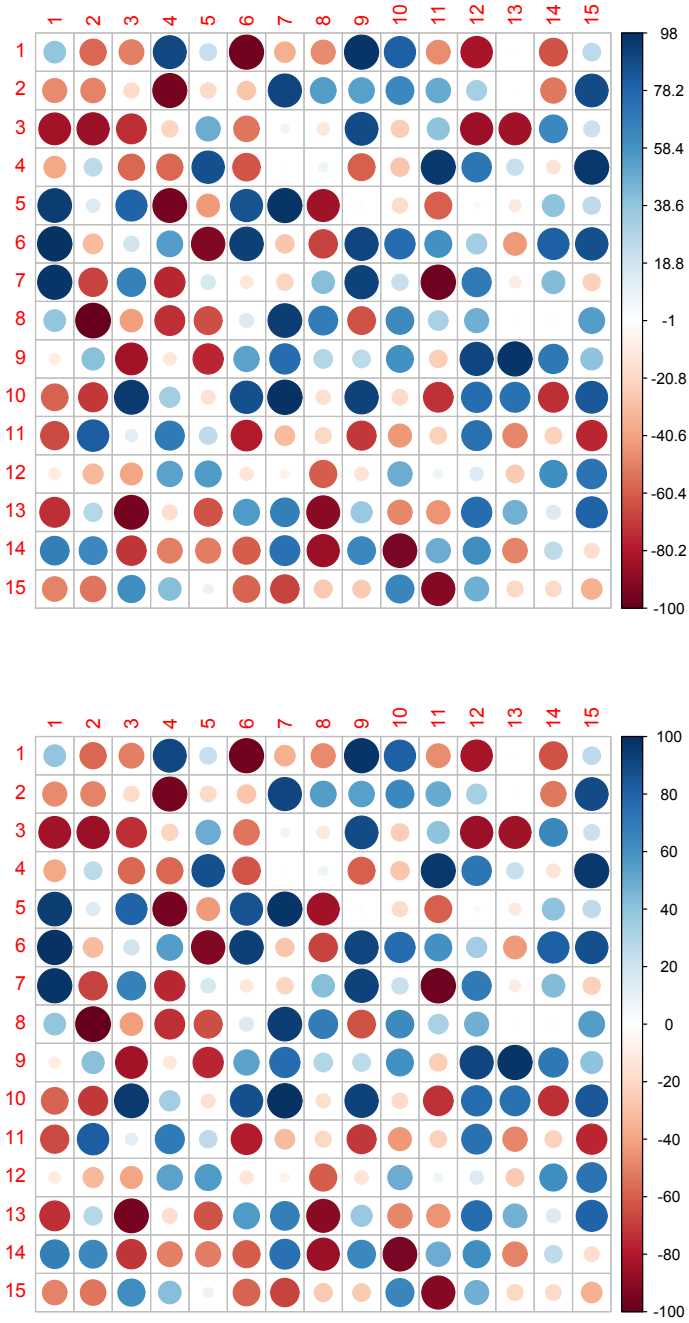



图 19.8:　设定范围的矩阵可视化

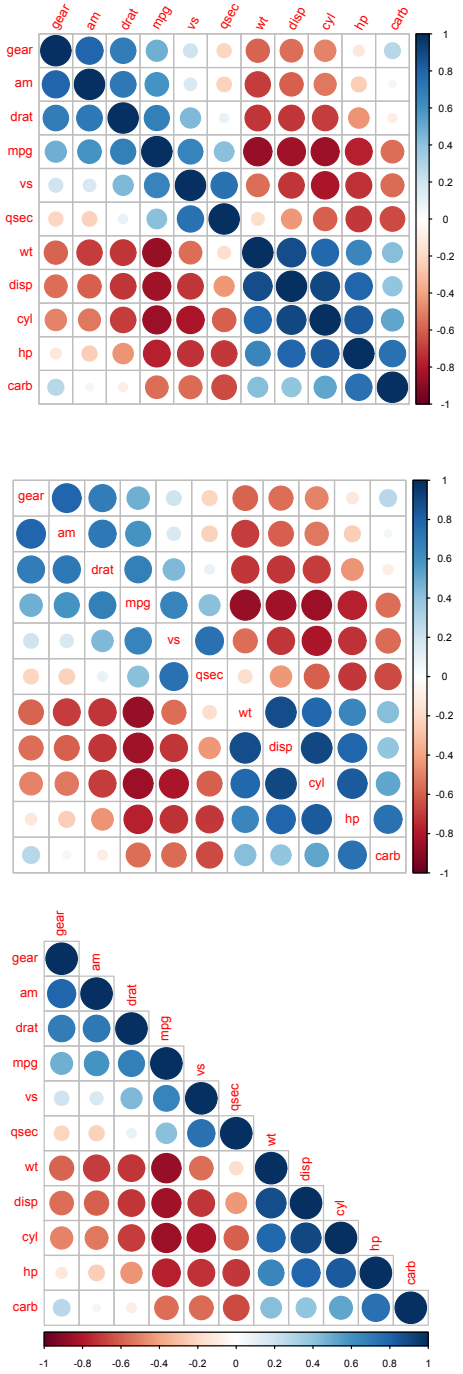





图 19.9: 不同页面、不同图例的设置

```
  diag(lowCI.mat) <- diag(uppCI.mat) <- 1
  for(i in 1:(n-1)){
    for(j in (i+1):n){
      tmp <- cor.test(mat[,i], mat[,j],
                      conf.level = conf.level)
      p.mat[i,j] <- p.mat[j,i] <- tmp$p.value
      lowCI.mat[i,j] <- lowCI.mat[j,i] <- tmp$conf.int[1]
      uppCI.mat[i,j] <- uppCI.mat[j,i] <- tmp$conf.int[2]
    }
  }
  return(list(p.mat, lowCI.mat, uppCI.mat))
}

res1 <- cor.mtest(mtcars,0.95)
res2 <- cor.mtest(mtcars,0.99)

par(mfrow = c(3,1))
# 特征标示出显著性水平
corrplot(M, p.mat = res1[[1]], sig.level = 0.2)
corrplot(M, p.mat = res1[[1]], insig = "p-value",
         sig.level= -1) # add all p-values
corrplot(M, p.mat = res1[[1]], order = "hclust",
         insig = "pch", addrect = 3)
```

corrplot 包的参数设置还有其他可选择的细节，读者可以通过在线帮助或者使用文档进一步了解。

现在我们介绍另外一个包：ellipse (Murdoch and Chow, 2013)，以鸢尾花数据为例。可以看到相关系数可视化之后，得到的信息非常明确而且直接。

```
if(!require(ellipse)) install.packages('ellipse')
library(ellipse)
```

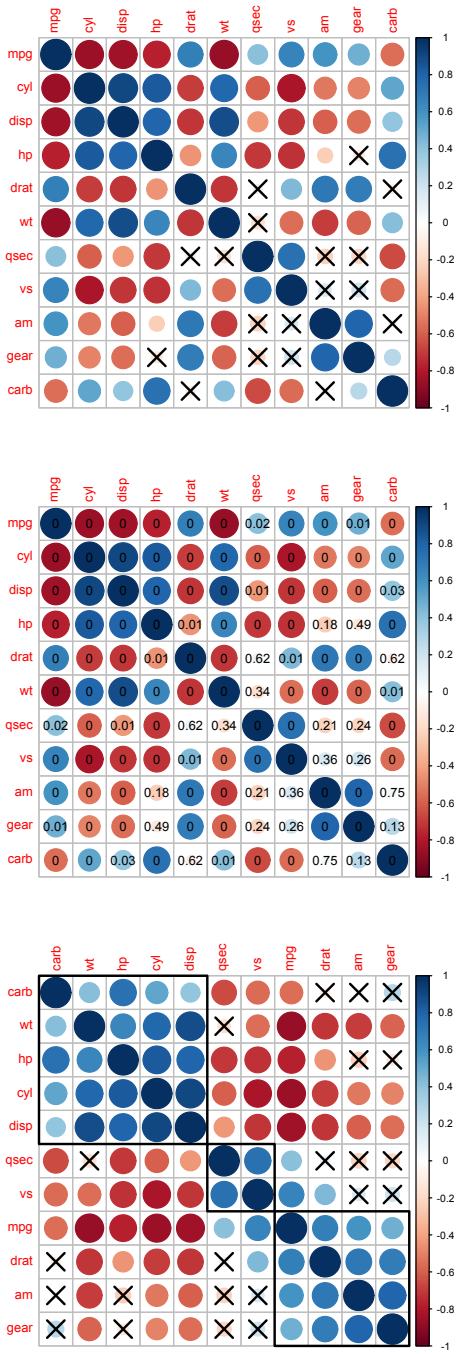



图 19.10: 标示出显著性水平的设置

```
library(psych)
mycol = colors()[as.vector(apply(corr.test(
  iris.data)$r, 2, rank))]
plotcorr(corr.test(iris.data)$r,
         col = mycol, mar = rep(0, 4))
```

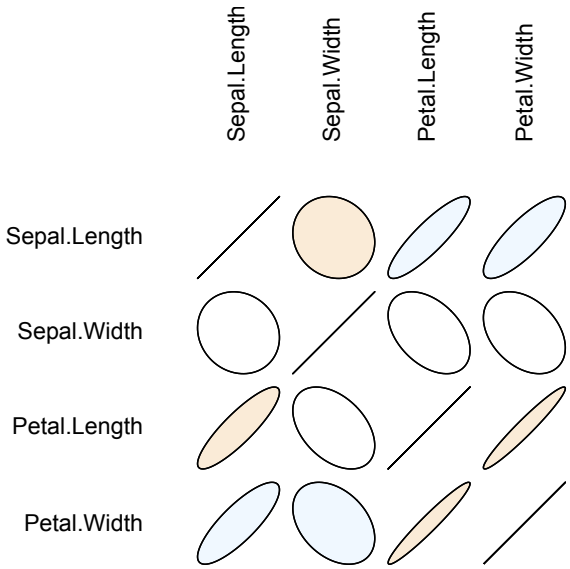



图 19.11:　鸢尾花数据展示的 ellipse 包作图

## 19.2　协方差

协方差是与相关系数紧密联系的一个统计量，它的定义正好是 Pearson 相关系数公式中的分子项。

用通俗易懂的话来说，协方差就是两个数据集的方差，体现两个数据集的变化趋势关系，如果协方差为正值，则表明两个数据集变化趋势一致，负值则表示变化趋势相反，0 则表示两个数据集变化趋势不相关。

这个意义是不是和相关系数相似？事实上，协方差和相关系数的关系是这样的：

相关系数＝数据集 x，y 的协方差/（x 的标准差 * y 的标准差）

因此，这两个统计量本身就是关联的，通过下面的例子读者可以得到更直观的印象。

在 R 中计算协方差的函数是 `cov()`，它的用法以及参数和 `cor()` 函数相同：

```
cov(x, y = NULL, use = "everything",
    method = c("pearson", "kendall", "spearman"))
```

我们来看一个例子：

```
x <- c(44.4, 45.9, 41.9, 53.3, 44.7, 44.1, 50.7, 45.2, 60.1)
y <- c(2.6, 3.1, 2.5, 5, 3.6, 4, 5.2, 2.8, 3.8)
par(mfrow = c(2, 1), mar = c(2, 4, 0.1, 0.1))
plot(x)
lines(x)
plot(y)
lines(y)
```

```
cor.test(x, y, method = "spearm", alternative = "g")
```

```
##
##  Spearman's rank correlation rho
##
## data:  x and y
## S = 48, p-value = 0.0484
## alternative hypothesis: true rho is greater than 0
## sample estimates:
## rho
## 0.6
```

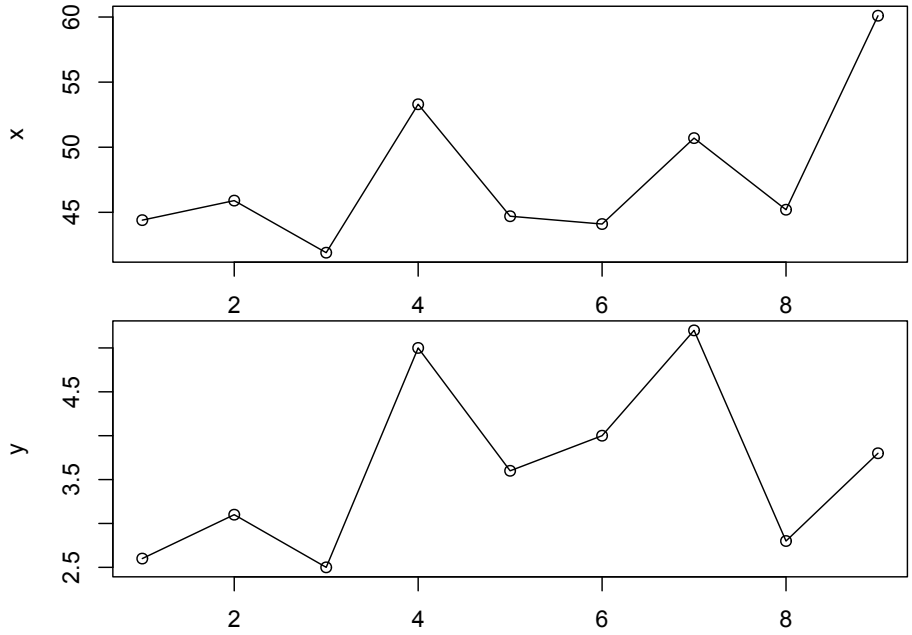



图 19.12:　协方差作图示例

```
cov(x, y)
```

```
## [1] 3.279722
```

从 x，y 的点线图可以很直观地看出，两个数据集的变化趋势一致。计算结果也表明它们协方差为正，显著相关。

```
cor.test(x, y, method = "spearm", alternative = "g")
```

```
##
##  Spearman's rank correlation rho
##
## data:  x and y
## S = 48, p-value = 0.0484
## alternative hypothesis: true rho is greater than 0
## sample estimates:
```

```
## rho
## 0.6
```

如果已经计算得到了一个协方差矩阵，就可以用 R 中的 cov2cor() 函数来计算相关系数矩阵，或者用 cov.wt() 命令来计算加权协方差矩阵：

```
x <- as.data.frame(x)  # 这个命令要求 x 是数据框或者矩阵
cov.wt(x, wt = rep(1/nrow(x), nrow(x)), cor = FALSE, center = TRUE,
    method = c("unbiased", "ML"))
```

- x：矩阵或者数据框。通常来说，行代表观测值，列代表变量。
- wt：每一个观测值权重组合的向量（必须为正），其长度必须和 x 的行数相同。
- cor：逻辑变量，设置是否返回相关系数矩阵。
- center：逻辑或者数值型变量，确定用于变量计算的中心。如果设置为 TRUE，将使用每一个变量的均值。如果设置为 FALSE，则使用 0。如果中心为数值，它的长度必须和 x 的列数相等。
- method：确定结果的规格，详见 “Details”。可以简化。
- Value：是关于下列变量的清单：
- cov：（加权）的协方差矩阵
- n.obs：观测值 x 的行数。
- wt：估值的权重。
- cor：估算的相关系数矩阵。

看一个例子：

```
xy <- cbind(x = 1:10, y = c(1:3, 8:5, 8:10))
w1 <- c(0, 0, 0, 1, 1, 1, 1, 1, 0, 0)
cov.wt(xy, wt = w1)  # i.e. method = 'unbiased'
```

```
## $cov
##      x    y
## x  2.5 -0.5
## y -0.5  1.7
##
```

```
## $center
##   x   y
## 6.0 6.8
##
## $n.obs
## [1] 10
##
## $wt
##  [1] 0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0
```

```
cov.wt(xy, wt = w1, method = "ML", cor = TRUE)
```

```
## $cov
##      x     y
## x  2.0 -0.40
## y -0.4  1.36
##
## $center
##   x   y
## 6.0 6.8
##
## $n.obs
## [1] 10
##
## $wt
##  [1] 0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0
##
## $cor
##            x          y
## x  1.0000000 -0.2425356
## y -0.2425356  1.0000000
```

# 参考文献


Adler, D., Murdoch, D., and others (2017). *rgl: 3D Visualization Using OpenGL*. R package version 0.97.0.

Allaire, J., Cheng, J., Xie, Y., McPherson, J., Chang, W., Allen, J., Wic kham, H., Atkins, A., and Hyndman, R. (2016). *rmarkdown: Dynamic Documents for R*. R package version 1.3.

Bivand, R., Keitt, T., and Rowlingson, B. (2016). *rgdal: Bindings for the Geospatial Data Abstraction Library*. R package version 1.2-5.

Bivand, R. and Lewin-Koh, N. (2017). *maptools: Tools for Reading and Handling Spatial Objects*. R package version 0.9-1.

Carslaw, D. C. and Ropkins, K. (2012). openair — an r package for air quality data analysis. *Environmental Modelling & Software*, 27–28(0): 52–61.

Dragulescu, A. A. (2014). *xlsx: Read, write, format Excel 2007 and Excel 97/2000/XP/2003 files*. R package version 0.5.7.

Gohel, D. (2017). *ReporteRs: Microsoft Word and PowerPoint Documents Generation*. R package version 0.8.8.

Højsgaard, S., Halekoh, U., Yan, J., and Højsgaard, M. S. (2016). Package 'geepack' .

Lang, D. (2017). *leafletCN: An R Gallery for China and Other Geojson Choropleth Map in Leaflet*. R package version 0.2.1.

Lemon, J. (2006). Plotrix: a package in the red light district of r. *R-News*, 6(4):8–12.

Murdoch, D. and Chow, E. D. (2013). *ellipse: Functions for drawing ellipses and ellipse-like confidence regions*. R package version 0.3-8.

Petzoldt, T. and Rinke, K. (2007). simecol: An object-oriented framework for ecological modeling in r. *Journal of Statistical Software*, 22(9):1–31.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Revelle, W. (2017). *psych: Procedures for Psychological, Psychometric, and Personality Research*. Northwestern University, Evanston, Illinois. R package version 1.7.8.

Sarkar, D. (2008). *Lattice: Multivariate Data Visualization with R*. Sprin ger, New York. ISBN 978-0-387-75968-5.

Team, R. C., Wuertz, D., Setz, T., Chalabi, Y., Maechler, M., and Byers, J. W. (2015). *timeDate: Rmetrics - Chronological and Calendar Objects*. R package version 3012.100.

Wei, T. and Simko, V. (2016). *corrplot: Visualization of a Correlation Matrix*. R package version 0.77.

Wickham, H. (2009). *ggplot2: Elegant Graphics for Data Analysis*. Sprin ger-Verlag New York.

Wickham, H. (2017). *stringr: Simple, Consistent Wrappers for Common String Operations*. R package version 1.2.0.

Xie, Y. (2013). animation: An R package for creating animations and demonstrating statistical methods. *Journal of Statistical Software*, 53 (1):1–27.

Xie, Y. (2014). knitr: A comprehensive tool for reproducible research in R. In Stodden, V., Leisch, F., and Peng, R. D., editors, *Implementing*

*Reproducible Computational Research*. Chapman and Hall/CRC. ISBN 978-1466561595.

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2016a). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.3.10.

Xie, Y. (2016b). *bookdown: Authoring Books and Technical Documents with R Markdown*. Chapman and Hall/CRC, Boca Raton, Florida. ISBN 978-1138700109.

Xie, Y. (2016c). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.15.1.

Xie, Y. (2017). *blogdown: Create Blogs and Websites with R Markdown*. R package version 0.0.24.

Xie, Y., Mueller, C., Yu, L., and Zhu, W. (2015). *animation: A Gallery of Animations in Statistics and Utilities to Create Animations*. R package version 2.4.

Xie, Y., Wei, T., and Qiu, Y. (2011). *fun: Use R for Fun*. R package version 0.1-0.

Zeileis, A., the R community. Contributions (fortunes and/or code) by Torsten Hothorn, Dalgaard, P., Ligges, U., Wright, K., Maechler, M., Halvorsen, K. B., Hornik, K., Murdoch, D., Bunn, A., Brownrigg, R., Bivand, R., Graves, S., Lemon, J., Kleiber, C., Reiner, D. L., Gunter, B., Koenker, R., Berry, C., Schwartz, M., Dewey, M., Bolker, B., Dunn, P., Goslee, S., Blomberg, S., Venables, B., Rau, R., Petzoldt, T., Tur ner, R., Leeds, M., Charpentier, E., Evans, C., Sonego, P., Ehlers, P., Steuer, D., Galili, T., Snow, G., Ripley, B. D., Sumner, M., Winsemius, D., Andronic, L., Diggs, B., Stigler, M., Friendly, M., Eddelbuettel, D., Heiberger, R. M., Burns, P., Menne, D., de Vries, A., Rowlingson, B., Lancelot, R., Weylandt, R. M., Skoien, J. O., Morneau, F., Unwin, A.,

Wiley, J., Therneau, T., Hanson, B., Singmann, H., Szoecs, E., Passolt, G., and Nash., J. C. (2016). *fortunes: R Fortunes*. R package version 1.5-4.

Zhao, P. (2017a). *beginr: Functions for R Beginners*. R package version 0.1.0.

Zhao, P. (2017b). *bookdownplus: Generate Varied Types of Books and Documents with R 'bookdown' Package*. R package version 1.3.0.

Zhao, P. (2017c). *mindr: Convert Files Between Markdown or Rmarkdown Files and Mindmaps*. R package version 1.1.0.

# 附录 A　Markdown 和 bookdown 语法速查

bookdown 的语法规则详见其官方文档 *bookdown: Authoring Books and Technical Documents with R Markdown*[1]，这里作一简要总结，以便查询。第一次使用时，建议对照着 'bookdownplus' 扩展包的模板文档来理解用法。

**Markdown 基本语法**

| 标记示例 | 输出 |
|---|---|
| `*italic*` | 斜体 *italic* |
| `** 粗体 **` | **粗体** |
| `CO~2~` | 下标（$CO_2$） |
| `R^2^` | 上标（$R^2$） |
| `$E = mc^2$` | 行内公式 $E = mc^2$ |
| | （双美元符号为行间公式） |
| `[网站](http://xuer.pzhao.net)` | 超级链接 |
| `<xuer@pzhao.net>` | 邮件链接 |
| `![](http 图片链接)` | 插入图片 |
| `> 引用文字` | 引用 |
| `` `plot()` `` | 行间代码 |
| 四个空格 | 整行代码 |

[1] https://bookdown.org/yihui/bookdown/

| 标记示例 | 输出 |
|---|---|
| 三个反引号 | 区块代码 |
| # 第一章 | 章节标题 |
| 1. 列表... | 带编号的列表 |
| - 列表... | 不带编号的列表 |
| ^[脚注] | 脚注 |

### 章节划分、编号和交叉引用

| 标记 | 输出 |
|---|---|
| # (PART) Part I {-} | 部分 |
| # chapter {#ID} | 带编号的章节 |
| # chapter {#ID .unnumbered} | 不带编号的章节 |
| # References {-} | 参考文献 |
| # (APPENDIX) Appendix {-} | 附录 |
| \@ref(ID) | 交叉引用 |

### 插图

第一种插图法 (不推荐)：用 markdown 基础语法。下面这条语句，将在输出的文档中以 caption 为标题插入 images 文件夹下名为 img1.png 的图片：

```
![caption](images/img1.png)
```

第二种插图法：用嵌入的 R 代码作图（推荐）：

````
```{r fig1, fig.cap='caption',
out.width='80%', fig.align='center', echo=FALSE}
plot(1:10)
```
````

交叉引用方法是`\@ref(fig:fig1)`。

第三种插图法：将现成的图片文件插进来（推荐）：

````
```{r img1, fig.cap='caption'}
knitr::include_graphics("images/img1.png")
```
````

交叉引用方法是`\@ref(fig:img1)`。

**表格**

第一种表格：直接用 markdown 语法画表格。输入：

```
第一列标题  第二列标题
---------- ----------
第一行 1     第一行 2
第二行 1     第二行 2
```

输出：

| 第一列标题 | 第二列标题 |
|---|---|
| 第一行 1 | 第一行 2 |
| 第二行 1 | 第二行 2 |

第二种表格：用 R 函数展示表格（推荐）：

````
```{r tab1, tidy=FALSE, echo=FALSE}
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```
````

交叉引用方法：`\@ref(tab:tab1)`。

**参考文献**

主流学术期刊和数据库都提供.bib 格式参考文献的下载。先将你下载的

参考文献信息合并在一个或几个.bib 文件里，然后在模板文档的 index.Rmd 开头部分 bibliography 条目里声明你自己的.bib 文件名称。.bib 文件里每条参考文献都有个条目名称。假定某篇文献的条目名称是 foo，那么文中引用的方法就是 `[@foo]`。

**定义、定理、示例等环境**

| 环境全称 | 引用简称 |
|---|---|
| theorems | thm |
| lemma | lem |
| definition | def |
| corollary | cor |
| proposition | prp |
| example | exm |
| exercise | exr |

````
```{环境全称, label='', name=""}
内容
```
````

定义、定理、示例等环境的交叉引用方法：`\@ref(引用简称:label)`，如描述某定律的格式是：

````
```{theorem, label='mythm1'}
物体内能的增加等于物体吸收的热量和对物体所作的功的总和。
```
````

交叉引用方法：`\@ref(thm:mythm1)`。

**输出 word 文档**

在 _output.yml 里添加一行：

```
bookdown::word_document2: default
```

**公式编号和引用**

pdf 文档里的公式编号和交叉引用，建议使用公式环境。例如，输入：

```
\begin{equation}
E = mc^2
  \label{eq:mc2}
\end{equation}
```

交叉引用方法：`\@ref(eq:mc2)`。

word 文档里的公式的编号和交叉引用，可以使用行内公式，公式前用圆括号里加公式标签。例如，输入：

```
(@eq-mc) $E = mc^2$
```

交叉引用方法：`@eq-mc`。

# 附录 B　答疑

## 菜鸟常犯错误和常见问题

**读取数据文件(read.table())时,显示无法打开连接(Error in file (file, "rt") : cannot open the connection)**

很有可能是路径的名称写错了。路径的名称应该是个字符串。相邻两级文件夹之间要用 \\ 或 / 分开，不能有汉字或特殊字符。最好不要有空格。如果有空格，要用 \ 进行转义。下面是一些路径示例。

```
read.csv(c:/r4r/co2.csv) # 错误
# 文件完整路径必须是个字符串，放在一对儿单引号或双引号里。
# 不要用中文里的引号。

read.csv('c:\r4r\\co2.csv') # 错误
# c: 后面跟两个反斜线或一个斜线。
# 相邻两级文件夹之间必须是两个反斜线\\或一个斜线/。

read.csv('c:/r4r/co2.scv') # 错误
# 文件名不要写错，应为 co2.csv。

read.csv('c:/r4r/co2.csv') # 正确。

read.csv('c:\\r4r\\co2.csv') # 正确。
```

### 找不到对象（object not found）

这并不意味着 R 在抱怨他自己的单身问题，而是说明某个变量并不存在。Windows 用户常见的错误是忘记了 R 对变量名的大小写是敏感的，$x$ 和 $X$ 是两个不同的变量，例如：

```
pm25 <- 40
PM25 + 1
```

```
## Error in eval(expr, envir, enclos): 找不到对象'PM25'
```

被赋值的是 `pm25`，而不是 `PM25`，做加法的时候当然找不到后者的值了。

### 括号用错或不完整导致各种意想不到的错误

R 中的圆括号、方括号、花括号各自有不同的含义（见小贴士 5.1）。如果用了不该用的括号，那么 R 运行时会出现难以预料的错误。例如:

```
x <- 1:6
x{1}
```

```
## Error: <text>:2:2: 意外的'{'
## 1: x <- 1:6
## 2: x{
##      ^
```

我们想调用 $x$ 里的第一个元素，应该用方括号 `x[1]`。错用花括号后，R 的错误提示是“意外的花括号”。这个提示是明显的，告诉你括号用错了。但大多数情况下，是下面这种提示：

```
x(1)
```

```
## Error in x(1): 没有"x" 这个函数
```

这里，我们错用了圆括号后，R 的错误提示是“没有 x 这个函数”。R 误以为你把 `x` 当方程用，不会提示你错用了括号。

另一个常见错误是括号不配对或配对不完整，也会导致无法预料的错

误。

要避免这种错误，方法很简单，RStudio 默认是在输入括号时自动补成对儿的，所以输入时一般不会出现问题，只要记得删除时也成对儿删掉就行了。另外，如果配对儿出现问题，RStudio 会用红色标记来提示该行的括号有问题。

### 求和、求平均用 sum() 和 mean() 得到的结果是 NA

实际数据里常有缺失值，R 默认以 NA 来表示（Not Available 的缩写）。这样的向量在很多函数里要特殊对待，例如：

```
x <- c(1, 3, NA, 4)
mean(x)
```

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

我们用 F1 小助理看一下 mean() 函数的帮助信息，会发现函数参数里有个 na.rm，默认值为 FALSE ，即“不删除缺失数据”。改为 na.rm = TRUE 就行了，

```
mean(x, na.rm = TRUE)
```

```
## [1] 2.666667
```

注意，这种情况下求平均值，向量的长度是以删除缺失值后计算的。

很多函数都有这个参数，用之前记得查看帮助。

### 别人发给我的.r 或.Rmd 文件，在 RStudio 里打开有乱码

问问是不是里面有英文之外的字符，例如中文的注释等。如果是的话，很有可能是别人在生成代码文件时用了不同的编码。

在 RStudio 的界面点击 File – Reopen with Encoding，换成别的编码试试看。

为了减少这种问题，建议大家生成代码文件时都选择 UTF-8 编码，点击 File – Save with Encoding 选中 UTF-8 即可。

**箭头赋值号 <- 与小于号 <**

如果不注意留空格，新手有时候会因为混淆箭头赋值号与小于号而得到意外错误，例如：

```
x <-1
x < -1
x < - 1
```

上面这三个例子里，哪个是把 $x$ 赋值为 1，哪个是判断 $x$ 是否小于 -1？试一下就知道了。

为了杜绝这种无聊错误，建议在 RStudio 使用 alt+_ 来输入箭头符号，会自动在箭头前后加空格。或者干脆用等号。

**用 read.table() 读一个几万行的数据文件，结果只读进去一千多行，提示我还有多少行没读入**

R 处理大数据文件时，受电脑的虚拟内存和地址限制，解决的方法有很多[1]，例如删除不再用的对象、尽量使用向量计算、用矩阵替代吃内存的数据框等。限于篇幅，本书对这个问题不展开了，请上网搜索 "R 语言大数据集"。

**运行 blogdown::install_hugo() 错误，显示 "无法与服务器建立连接"**

此类出错信息一般是：

```
Error in download.file(url, ...):
  cannot open URL
'https://github.com/spf13/hugo/release/....'
```

一般情况下，是你的电脑所处的网络环境造成的。试着换个地方联网。

**我在论坛发帖提问求助，为何没有人帮我解答？**

有以下几种可能：

[1] R 处理大数据：http://yanping.me/cn/blog/2012/01/01/working-with-large-datasets/

1. 你的问题太难，没人知道怎么解答。这种情况下建议你耐心等待，或者换个论坛求助。

2. 你问的方式不够礼貌。看看有没有用 “请” 字。回忆一下，上次别人帮你解答问题后，你有没有提供反馈，比如对别人说 “谢谢”。帮你的人如果得不到回应，下次就不会再帮你。

3. 你的标题没写清楚。比如发帖的标题是 “菜鸟求助！紧急！”。这种帖子基本没人点开看内容。用尽量简短的语句在标题里陈述问题，节省别人的时间，例如标题写为 “[求助] 运行 `blogdown::install_hugo()` 错误，显示 ‘无法与服务器建立连接’”。

4. 你的正文里没有把问题表述清楚，或者问得太笼统。别人无法快速看懂你的问题，自然没法解答。例如 “我的数据文件没法读进 R，怎么办？” 这样得到的回答只能是 “去看书”。尽量把问题的来龙去脉说清楚，例如附上那个没法读取的数据文件的下载链接。

5. 你没有给出示例代码。别人一般不会花时间从零开始为你写代码。如果没法快速重现你的故障，就不会回答你。尽量贴出你的代码，让别人只需拷贝粘贴过去，就能直接运行重现你的问题。

6. 你没有说清楚你的软件运行环境。有些问题是跟运行环境有关的。你的 R 是什么版本？操作系统是什么？加载了那些扩展包？如果你不确定是否跟运行环境有关，那么保险起见，把你的运行环境信息贴出来，让别人帮你判断。R 专门提供了两个函数来获取运行环境信息：

```
sessionInfo()
devtools::session_info()
```

与人方便，自己方便。在论坛上，没有任何人有义务帮你解决问题。所以，降低你的期望值，并且换位思考一下，站在别人角度来看你的求助帖，就知道应该如何提问了。只有先为别人节省时间，才能为自己节省时间，从而快速得到别人的帮助。

# 习题参考答案

这里给出了各章练习的参考答案以及详细注释。正如我们前文所说的，每道题的解答方式都不是唯一的，只要得到想要的答案就可以了。

参考答案里，我们使用了 timeDate (Team et al., 2015)、fun (Xie et al., 2011)、animation (Xie, 2013; Xie et al., 2015) 和 openair (Carslaw and Ropkins, 2012) 等扩展包。

**练习 1.1 参考答案**

```
# 方案 1
365 %% 7
365 %/% 7

# 方案 2
floor(365/7) # 向下取整函数
365 - floor(365/7)
# 与 floor() 属于同一家族的还有：
# ceiling(), trunc(), round(), signif()
```

**练习 1.2 参考答案**

```
x <- c(97, 80, 64, 91, 87, 100, 128, 144, 150, 150, 150, 106,
       78, 68, 62, 46, 55, 68, 84, 92, 95, 108, 128, 138)
plot(x)

# 方案 1
max(x)
min(x)
mean(x)

# 方案 2
summary(x)[c(1, 4, 6)]
```

练习 2.1 参考答案

```
mydata2 <- as.data.frame(t(matrix(
  co2, 12,
  dimnames = list(month.abb, unique(floor(time(co2)))))))
mydata2$year <- as.numeric(rownames(mydata2))
x1995 <- mydata2['1995', 2:13]

# 方案 1
max(x1995)
min(x1995)

# 方案 2
range(x1995) # range() 返回取值范围。
```

练习 3.1 参考答案

```
mydata2 <- as.data.frame(t(matrix(
  co2, 12,
  dimnames = list(month.abb, unique(floor(time(co2)))))))
mydata2$year <- as.numeric(rownames(mydata2))
mydata2$mean <- rowMeans(mydata2[, 2:13])

plot(x = mydata2$year, y = mydata2$mean,
     type = 'p', pch = 17, col = colors()[652])
```

练习 3.2 参考答案

```
x <- mydata2$year
y <- mydata2$Sep
plot(x = x, y = y)
abline(v = seq(min(x), max(x), 1),
       h = seq(min(y), max(y), 10), col = 'grey')
text(x, y, labels = y, col = 'red', pos = c(2, 4))
```

```
# pos 参数来指定文字位于给定坐标的哪一侧，
# pos 取值可以是 1,2,3,4。
```

### 练习 3.3 参考答案

```
x <- mydata2$year[1:6]
y <- mydata2$Sep[1:6]

# 方案 1：逐个画图
pdf('c:/r4r/9in1.pdf')
par(mfrow = c(3,3), cex = 1.2, mar = c(3, 2, 0.5, 1))
plot(x = x, y = y, type = 'p')
legend('topleft', legend = 'p',
       cex = 0.8, bty = "n", text.col = 'blue')
plot(x = x, y = y, type = 'l')
legend('topleft', legend = 'l',
       cex = 0.8, bty = "n", text.col = 'blue')
plot(x = x, y = y, type = 'b')
legend('topleft', legend = 'b',
       cex = 0.8, bty = "n", text.col = 'blue')
plot(x = x, y = y, type = 'c')
legend('topleft', legend = 'c',
       cex = 0.8, bty = "n", text.col = 'blue')
plot(x = x, y = y, type = 'o')
legend('topleft', legend = 'o',
       cex = 0.8, bty = "n", text.col = 'blue')
plot(x = x, y = y, type = 'h')
legend('topleft', legend = 'h',
       cex = 0.8, bty = "n", text.col = 'blue')
plot(x = x, y = y, type = 's')
legend('topleft', legend = 's',
       cex = 0.8, bty = "n", text.col = 'blue')
```

```
plot(x = x, y = y, type = 'S')
legend('topleft', legend = 'S',
       cex = 0.8, bty = "n", text.col = 'blue')
plot(x = x, y = y, type = 'n')
legend('topleft', legend = 'n',
       cex = 0.8, bty = "n", text.col = 'blue')
dev.off()

# 方案 2: 用循环结构以简化操作
par(mfrow = c(3,3), cex = 1.2, mar = c(3, 2, 0.5, 1))
for (i in c("p", 'l', "b", "c", "o", "h", "s", "S", "n")) {
  plot(x = x, y = y, type = i)
  legend('topleft', legend = i,
         cex = 0.8, bty = "n", text.col = 'blue')
}
```

### 练习 4.1 参考答案

```
wp <- as.data.frame(WorldPhones)
m0 <- lm(wp$Asia ~ wp$Europe + 0)
summary(m0)
plot(x = wp$Europe, y = wp$Asia, pch = 19)
abline(m0, col = "red")
legend("bottomright", pch = c(19, NA), lty = c(NA, 1),
       legend = c("Data", "Linear fit"),
       col = c("black", "red"), bty = 'n')
eqlm <- expression(italic(y) == 0.1854 * italic(x))
eqr2 <- expression(italic(R) ^ 2 == 0.9856)
eqn <- expression(italic(n) == 7)
text(x = 23000, y = 7000, labels = eqlm, adj = 0)
text(x = 23000, y = 6000, labels = eqr2, adj = 0)
text(x = 23000, y = 5000, labels = eqn, adj = 0)
```

**练习 4.2 参考答案**

```
# 只需将 plot() 函数的 ylab 参数改为：
ylab = expression(CO[2])
```

**练习 4.3 参考答案**

```
x <- seq(0, 50, 1)
y <- runif(1, 5, 15) * exp(-runif(1, 0.01, 0.05) * x) +
  rnorm(51, 0, 0.5)
a_start <- 14
b_start <- - log(6/a_start) / 50
m <- nls(y ~ a * exp(-b * x),
         start = list(a = a_start, b = b_start))
plot(x, y)
lines(x, predict(m), col = 'darkgreen')
# predict() 函数返回预测值
eqnls <- expression(italic(y) == 15.18 *
                      italic(e) ^ {-0.04786 * italic(x)})
text(10, 4, eqnls)
legend("topright", pch = c(1, NA), lty = c(NA, 1),
       legend = c("Data", "Non-linear fit"),
       col = c("black", "darkgreen"), bty = 'n')
```

**练习 5.1 参考答案**

```
# 方案 1
for (i in seq(from = 1, to = 100, by = 2)) print(i)

# 方案 2
print(seq(from = 1, to = 100, by = 2))
# seq() 函数本来内藏的逻辑就是循环
```

**练习 5.2 参考答案**

```
x <- c(2, 3, 5)
y <- c(1, 2, 3, 4)
m <- matrix(nrow = length(x), ncol = length(y))
for (i in 1:length(x)){
  for (j in 1:length(y)){
    m[i,j] = x[i] * y[j]
  }
}
m
```

### 练习 5.3 参考答案

```
for (i in 2:13) print(colMeans(mydata2[, i]))
```

### 练习 5.4 参考答案

```
wp <- as.data.frame(WorldPhones)
wp$year <- as.numeric(rownames(wp))
mydata3 <- data.frame(
  nphone = unlist(wp[, 1:7]), year = rep(wp$year, 7),
  conti = rep(names(wp)[1:7], each = nrow(wp)))
boxplot(mydata3$nphone ~ mydata3$year)
boxplot(mydata3$nphone ~ mydata3$conti)
```

### 练习 5.5 参考答案

```
tapply(mydata3$nphone, mydata3$year, max)
tapply(mydata3$nphone, mydata3$year, min)
tapply(mydata3$nphone, mydata3$year, median)
```

### 练习 6.1 参考答案

```
m <- 6:1
n <- c(3, 5, 6)
for (mi in m) print(mi == n)
```

### 练习 6.2 参考答案

```
# 方案 1
for (i in 1:100){
  if (i %% 2 == 0 & i %% 3 == 0) print(i)
}

# 方案 2
i <- 1:100
i[i %% 2 == 0 & i %% 3 == 0]

# 方案 3
seq(from = 6, to = 100, by = 6)
```

### 练习 6.3 参考答案

```
z <- 2
for (i in 3:10000) {
  j <- 2:(i-1)
  if (sum(i %% j == 0) == 0) z <- c(z, i)
}
z
# 这个方案完全按照素数的定义来逐个判断，虽管用，但效率低。
# 请试着用其他方案来提高运算效率。
```

### 练习 6.4 参考答案

```
x <- c(97, 80, 64, 91, 87, 100, 128, 144, 150, 150, 150, 106,
       78, 68, 62, 46, 55, 68, 84, 92, 95, 108, 128, 138)
hour <- 0:23
hour[x == max(x)]
```

### 第6.3节参考答案

```
# 方案 1
year = 2017: 2026
method = "Meeus"
calendar = "Gregorian"

Y <- year
if (method == "Gauss") # 以下为高斯算法，适用于 1583 -- 2299 年
{
  if (calendar == "Gregorian") # 公历（天主教）
  {
    k <- (Y - 1500) %/% 100 + 1
    M <- c(22,22,23,23,24,24,24,25)[k]
    N <- c(2,2,3,4,5,5,6,0)[k]

  } else if (calendar == "Julian") # 儒略历（东正教）
  {
    M <- 15; N <- 6
  } else
  {
    print("sorry, the calendar does not exist.
          use method = 'Gregorian' or 'Julian'")
  }
  a <- Y %% 19
  b <- Y %% 4
  c <- Y %% 7
  d <- (19 * a + M) %% 30
  e <- (2 * b + 4 * c + 6 * d + N) %% 7
  computus.date <- ifelse(d + e < 10,
                          paste("3-", d + e + 22, sep = ""),
                          paste("4-", d + e - 9, sep = ""))
  if (computus.date == "4-26") computus.date <- "4-19"
  if (computus.date == "4-25" & d == 28 & e == 6 & a > 10)
```

```
    computus.date <- "4-19" # 高斯，I 服了 U!
  computus <- paste(Y, computus.date, sep = "-")
} else if (method == "Meeus") # 以下为 Meeus 算法
{
  if (calendar == "Gregorian") # 公历（天主教）
  {
    a <- Y %% 19
    b <- Y %/% 100
    c <- Y %% 100
    d <- b %/%4
    e <- b %%4
    f <- (b + 8) %/% 25
    g <- (b - f + 1) %/%3
    h <- (19 * a + b - d - g + 15) %% 30
    i <- c %/% 4
    k <- c %% 4
    L <- (32 + 2 * e + 2 * i - h - k) %% 7
    m <- (a + 11 * h + 22 * L) %/% 451
    month <- (h + L - 7 * m + 114) %/% 31
    day <- ((h + L - 7 * m + 114) %% 31) + 1
  } else if (calendar == "Julian") # 儒略历（东正教）
  {
    a <- Y %% 4
    b <- Y %% 7
    c <- Y %% 19
    d <- (19 * c + 15) %% 30
    e <- (2 * a + 4 * b - d + 34) %% 7
    month <- (d + e + 114) %/% 31
    day <- ((d + e + 114) %% 31) + 1
  } else
  {
    print("sorry, the calendar does not exist.
```

```
          use method = 'Gregorian' or 'Julian'")
  }
}
computus <- paste(Y, month, day, sep = "-")
computus

# 方案 2: 使用别人写好的函数（扩展包）。
if(!require("timeDate")) install.packages("timeDate")
require("timeDate")
Easter(year = 2017:2026)
```

练习 8.1 参考答案

```
write.csv(WorldPhones, file = 'c:/r4r/wp.csv')
wp_path <- "C:/r4r/wp.csv"
file.show(wp_path)
wp <- read.csv(file = wp_path)
summary(wp)
plot(wp)
names(wp)[1] <- 'year'
rownames(wp) <- wp$year
```

练习 8.2 参考答案

```
wp$decade <- c('1950s', '1950s', '1950s', '1950s', '1950s',
               '1960s', '1960s')
str(wp)
```

练习 8.3 参考答案

```
m <- matrix(data = 1:30, nrow = 5)
```

练习 8.4 参考答案

```
wp_mt <- as.matrix(wp)
str(wp_mt)
# 由于矩阵里的元素要求全部是同一类型，
# 所以数字全部强制转换成了字符。
wp['1956', 'Europe']
wp_mt['1956', 'Europe']
wp[, c('Asia', 'Europe')]
wp_mt[, c('Asia', 'Europe')]
wp[c(2, 4, 5), ]
wp[-c(2, 4, 5), ]
```

**练习 8.5 参考答案**

```
annualsum <- rowSums(wp[, 2:7])
annualdiff <- diff(annualsum)
annualdiffeach <- apply(wp[, 2:7], MARGIN = 2, diff)
annualrate <- annualdiff/annualsum[1:6]
annualrateeach <- apply(wp[, 2:7], MARGIN = 2,
                        function(x) diff(x)/x[1:6])
# apply 用于自定义函数。
```

**练习 8.6 参考答案**

```
par(mfrow = c(2, 1), mar = c(0, 3, 3, 1))
plot(wp$year, wp$Asia, type = 'l', xlim = c(1951, 1961),
     axes = FALSE, xlab = '', ylab = 'Phone Asia')
axis(2)
box()
par(mar = c(3, 3, 0, 1))
plot(wp$year[1:6], annualrateeach[, 'Asia'], type = 'l',
     xlim = c(1951, 1961),
     xlab = 'year', ylab = 'Increase rate')
dev.off()
```

练习 8.7 参考答案

```
myylim <- range(wp[, 2:8])
mycol <- rainbow(7)
plot(wp$year, wp[, 2], col = mycol[1], type = 'l',
     xlab = 'year', ylab = 'Phone number', ylim = myylim)
for (i in 3:8)
  lines(wp$year, wp[, i], col = mycol[i-1])
legend('topleft', legend = names(wp)[2:8],
       lty = 1, col = mycol, bty = 'n')
```

练习 8.8 参考答案

```
par(mfrow = c(3, 3), mar = c(4, 4, 0.1, 0.1))
for (i in 2:8) {
  plot(wp$year, wp[, i], col = mycol[i-1],
       xlab = 'year', ylab = names(wp)[i])
  m <- lm(wp[, i] ~ wp$year)
  a1 <- m$coefficients[1]
  a2 <- m$coefficients[2]
  abline(m)
  legend('topleft', bty = 'n',
         as.expression(substitute(y == a + b * x,
                                  list(a = a1, b = a2))))
}
```

练习 8.10 参考答案

```
# 方案：穷举
for (Aa in 1:9){
  for (Bb in c(0:9)[!0:9 %in% Aa]){
    for (Cc in c(0:9)[!0:9 %in% c(Aa, Bb)]){
      for (Dd in c(1:9)[!1:9 %in% c(Aa, Bb, Cc)]){
        for (Ee in c(0:9)[!0:9 %in% c(Aa, Bb, Cc, Dd)]){
```

```
          for (Ff in c(1:9)[!1:9 %in% c(Aa, Bb, Cc, Dd)]){
            if ((100 * Aa + 10 * Bb + Cc) * (Dd * 10 + Cc) ==
                Ff * 1000 + Dd * 100 + Bb * 10 + Cc &
                (100 * Aa + 10 * Bb + Cc) * Cc ==
                Dd * 1000 + Ee * 100 + Aa * 10 + Cc &
                (100 * Aa + 10 * Bb + Cc) * Dd ==
                700 + Ee * 10 + Dd &
                Dd * 1000 + Ee * 100 + Aa * 10 + Cc + 7000 +
                Ee * 100 + Dd * 10 == Ff * 1000 + Dd * 100 +
                Bb * 10 + Cc)
              print(c(Aa, Bb, Cc, Dd, Ee, Ff))
          }
        }
      }
    }
  }
}
# 这里给出的方法是穷举，把 1001 到 9999 之间的所有数都试了一遍。
# 这是计算机擅长的方式。

# 方案 2
# 小学生的解题思路是这样的：
# ABC 乘以 C 等于 DEAC，C 只能是 5 或 6，
# 但根据 ABC 乘以 D 等于 7ED，所以 C 不可能是 5，C = 6。
# D + 7 = F，D 只能是 1 或 2，但 C 乘以 D 末尾是 C，
# 所以 D = 2，F = 9，E = 1。
# 由于 C = 6，D = 2，那么 A 只能是 3，B = 5。
```

**练习 8.11 参考答案**

```
d <- 1
left <- 125
leftc <- left
```

```
drop <- 1
dropc <- drop
drop <- drop + 1
while (left != drop) {
  d <- d + 1
  left <- left - drop
  leftc <- c(leftc, left)
  dropc <- c(dropc, drop)
  if (left < drop + 1) drop <- 1 else drop <- drop + 1
}
plot(leftc, ylim = c(0, 123))
points(dropc, col = 'red')
# 仍然是穷举法。
# 小学生的解题思路应该是：125 = 120 + 3 + 1 + 1
# 三项对应着 15 天，2 天，1 天，1 天，所以总共 19 天掉完。
```

**练习 9.1 参考答案**

```
kaipingfang <- function(x) return(sqrt(x))
```

**练习 9.2 参考答案**

```
cv <- function(x) sd(x)/mean(x) # 省略了 return()
```

**练习 9.3 参考答案**

```
# 只需将第 6.3 节参考答案代码的前三行删掉，
# 其他代码放进下面的花括号里即可。
computus <- function(year, method, calendar) {}
```

**练习 9.6 参考答案**

```
install.packages('animation')
library(animation)
demo("fireworks") # 会用网页浏览器打开一个动画。
```

```
citation("animation") # 看看作者。
```

### 练习 9.7 参考答案

```
install.packages('openair')
library(openair)
example(windRose)
citation('openair')
```

### 第9.5 参考答案

```
install.packages('fun')
library(fun)
if (.Platform$OS.type == "windows") x11() else
  x11(type = "Xlib")
mine_sweeper()
demo(package = 'fun')
```

### 练习 10.1 参考答案

```
p1 <- 'Pack my box with five dozen liquor jugs'
p1 <- tolower(p1)
dup1 <- table(strsplit(gsub(' ','', p1), ''))
dup1[dup1>1]
```

### 练习 10.2 参考答案

```
# 提示:
# 1. 跟千字文的处理方法类似
# 2. 中华字经的原文可以在网上搜
# 3. 常用汉字 3500 字可以在网上搜，由国务院颁布
```

### 练习 11.2 参考答案

```
us <- read.csv("C:/r4r/us.csv")
cols <- rainbow(nlevels(us$state)) # create 49 rainbow cols
```

```
plot(us$lon, us$lat, col = cols[us$state], pch = 20)
lon.median <- tapply(us$lon, us$state, median)
lat.median <- tapply(us$lat, us$state, median)
text(labels=levels(us$state),
     x=lon.median, y=lat.median, cex=0.5, col = 'White')
abline(h = seq(from = 25, to = 50, by = 5), col = 'grey')
unique(us$state[us$lat > 34.9 & us$lat < 35.1])
```

练习 11.3 参考答案

```
timez <- as.factor(us$lon %/% 15)
mycol <- rainbow(nlevels(timez))[timez]
plot(us$lon, us$lat, col = mycol, pch = 20)
```

练习 11.4 参考答案

```
# 只需把 mycol 一行改为:
mycol <- rev(rainbow(nlevels(aqilevel)))[aqilevel]
```

练习 12.1 参考答案

```
format(Sys.time(), '%j')
```

练习 12.2 参考答案

```
bd <- '1994-09-22 20:30:00'
bdtime <- strptime(x = bd, format = '%Y-%m-%d %H:%M:%S',
                   tz = "Asia/Shanghai")
bdtime$yday
```

练习 12.3 参考答案

```
format(bdtime, '%c')
```

练习 12.4 参考答案

```
difftime(Sys.time(), bdtime, units = 'days')
difftime(Sys.time(), bdtime, units = 'hours')
difftime(Sys.time(), bdtime, units = 'secs')
```

**练习 12.5 参考答案**

```
bdyears <- seq(from = 1994, by = 1, length.out = 101)
bd100 <- paste(bdyears, '-09-22 20:30:00', sep = '')
bd100 <- strptime(x = bd100, format = '%Y-%m-%d %H:%M:%S',
                  tz = "Asia/Shanghai")
bdwd <- bd100$wday
plot(x = bdyears, y = bdwd)
abline(v = seq(from = 1995, by = 5, to = 2100),
       col = 'grey')
sum(bdwd == 0)
```

**练习 13.2 参考答案**

```
download.file(url = "http://dapengde.com/r4rookies/obs.zip",
              destfile = "C:/r4r/obs.zip")
unzip(zipfile = "C:/r4r/obs.zip", exdir = "C:/r4r")

stn <- 50
obsdir <- 'C:/r4r/obs'
obsfilefull <- dir(obsdir, full.names = TRUE)
output <- NULL
for (k in 1:length(obsfilefull))
{
  input <- read.table(obsfilefull[k], header = FALSE,
                      skip = 2, sep = "")
  obstimestr <- strsplit(
    readLines(obsfilefull[k])[2], ' ')[[1]]
  obstime <- paste(
    '20', obstimestr[3], '-', obstimestr[5], '-',
```

```
    obstimestr[7], ' ', obstimestr[9], sep = '')
  output_new <- input[which(
    input[, 1] >= stn * 1000 &
      input[, 1] < (stn + 10) * 1000), c(1, 4)]
  output_new$time <- obstime
  output <- rbind(output, output_new)
}
output
```

**练习 13.3 参考答案**

```
output$stn <- as.factor(output$V1)
boxplot(output$V4 ~ output$stn)
```

**练习 17.1 参考答案**

```
set.seed(1)
mydata<-rf(1000,2,2,4)
ks.test(mydata, "pf", 2,2,4)# 按这个自由度生成的数集，检验一下
ks.test(mydata, "pf", 1,1,2)
# 用另一组自由度来检验生成的数集（另一个分布）
# 结果，mydata 显然服从于生成它的参数分布，
# 不服从于不生成它的参数分布。
```

**练习 17.2 参考答案**

```
summary(nottem)
par(mfcol=c(1,2),ps=6.5)
hist(nottem,breaks=20)
qqnorm(nottem,pch=1)
shapiro.test(nottem)
#p 远小于 0.05，数据集 nottem 不属于正态分布。
```

**练习 18.1 参考答案**

```
x<- rnorm(30,3,0.8)
y<- rnorm(30,5,1.2)
z<- rnorm(30,4,2.1)
t.test(x,y,paired=T)
t.test(x,y)
pairwise.t.test(x,y,z)
```

# 索引

# 后记：学 R 时习之

学 R 时习之，不亦说乎？
有朋自远方来，不亦乐乎？
人不知 R 不愠，不亦君子乎？

—《论语 • 学 R》

我们学 R 之旅到此告一段落。从此以后，你的身份不再是零基础的 R 菜鸟了。祝贺你！回忆一下在第一章我们曾表白过学习 R 的理由。学到这里，你是否还记得自己的初衷？如果本书让你对 R 产生了浓厚的兴趣，本书写作的目的就达到了。有了本书介绍的 R 助理团无处不在的帮助，以及上网搜索和提问的自学习惯，你将无往不胜。

那么，入门之后该做些什么？

“半部论语治天下”。既然入了门，我们只需按照圣人指示的道路，做三件事就可以了。

第一件事：常练（“学 R 时习之”）。

经常温习和使用 R，让 R 融入你的工作和生活，享受其中的乐趣。书籍资料可以帮助你温故知新，例如 Zuur 等著 *A Beginners' Guide to R*，是我见过的 R 书里最为浅显易懂的，书中举例的数据和代码可以在官方网站免费下载。再如刘思喆编写的《153 分钟学会 R (R 常见问题解答)》和陈钢翻译的《R 入门 25 招》等，适合常备案头。此外，还有代码学校[2]等很

[2]代码学校：http://tryr.codeschool.com/

实用的在线课程。一旦读完这些书和资料，你的 R 水平必然会有质的飞跃，并且自己会明白下一步该做什么。

然而，空读多少本关于 R 的书，也不如拿几个例子和代码来实际操作一下更有效。R 丰富的扩展包能胜任很多有趣的工作，怎么样玩得转，就看你的想象力。在 R 的世界里，只有想不到，没有做不到。让 R 成为一种生活方式，这个过程也许会很慢，不过，享受乐趣的过程恰恰是越慢越好，不是吗？更何况 R 是如此美好，一旦拥有，别无所求。

携 R 之手，与 R 偕老，终究你会爱上她的。

第二件事：交友（“有朋自远方来”）。

R 是个开源的自由软件，这从根本上带来一个结果，那就是大多数 R 用户有着开放的胸怀，乐意分享和帮忙。在这个世界里，你将结识一群志同道合的朋友，他们可能来自不同国家，有着不同的肤色，却跟你说着同样的语言—— R 语言。

在哪里交到这些朋友呢？

- 可以去统计之都[3]的论坛，看看大家在聊什么，R 有什么最新的有趣玩法;
- 可以参加一年一度的中国 R 语言会议[4]，看看三教九流的英雄豪杰们怎样通过 R 语言而从五湖四海走到了同一个聚义厅;
- 可以加入 R 社区的邮件列表[5]，跟世界各地的 R 使用者和爱好者交流，顺便练习英文。

第三件事：分享（“人不知 R 不愠”）。

很多新手会在论坛提一些菜鸟问题，你可以耐心地回答他们；很多人仍然不知道 R 的好处，你可以骄傲地展示给他们。这个过程中，说不定你能看见旧日自己的影子。向别人分享你学习 R 的心得，分享你有用的代码。赠人玫瑰，手有余香。分享的形式可以多种多样，比如：

[3]统计之都：https://cosx.org/

[4]中国 R 语言会议：http://china-r.org/

[5]R 社区邮件列表：https://www.r-project.org/mail.html

- 可以把你的代码分享到 GitHub[6]，与别人协作完成一个项目;
- 可以把你自己制作的扩展包发布到 CRAN[7]，让别人共享你的成果;
- 可以把你的代码写成网页形式的 ShinyApps[8]，让不懂 R 的人也可以使用。
- 可以把你的日志、随笔、散文等文字发布到 bookdown 官网[9]，与亲朋好友分享你生活的点点滴滴。

选个你喜欢的方式便好。

当年，拜罗伊特大学的同事将他们的 R 代码分享给我的时候，我未曾料到将来会以一本书的形式跟你分享。希望未来的某一天，在网上的某个角落，我会看到你的分享。说不定，在书店里，我会买到你写的书。

若干年后，在地球上某个角落，说不定我们会不期而遇，喝杯咖啡，共叙 R 与你我的故事，宛如故友久别重逢。就像丘处机致江南七怪的书信中所写：

江南花盛之日，当与诸公置酒高会醉仙楼头也。人生如露，大梦一十八年，天下豪杰岂不笑我辈痴绝耶？

—《射雕英雄传》

[6]GitHub：`https://github.com/`

[7]CRAN：`https://cran.r-project.org/`

[8]ShinyApps：`https://www.shinyapps.io/`

[9]bookdown 官网：`https://bookdown.org/`