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。

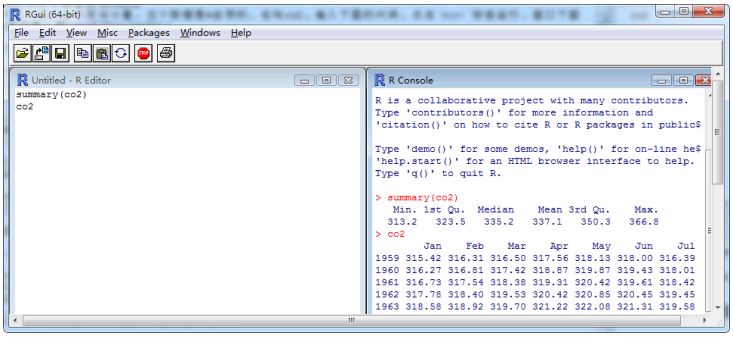


图 1.1: 裸奔的 R 默认界面

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。

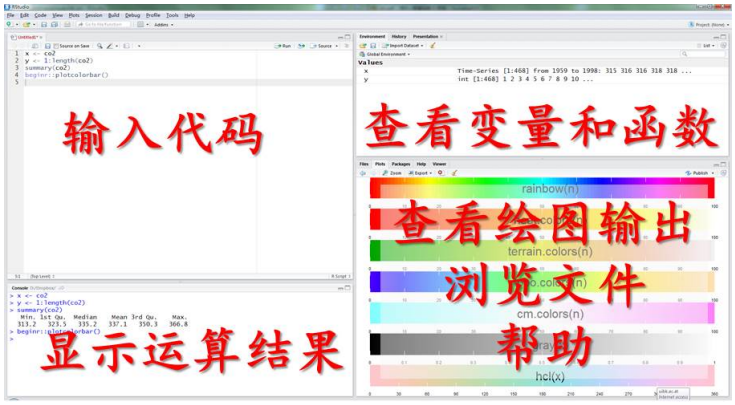


图 1.2: RStudio:R 的豪宅

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 π，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
```

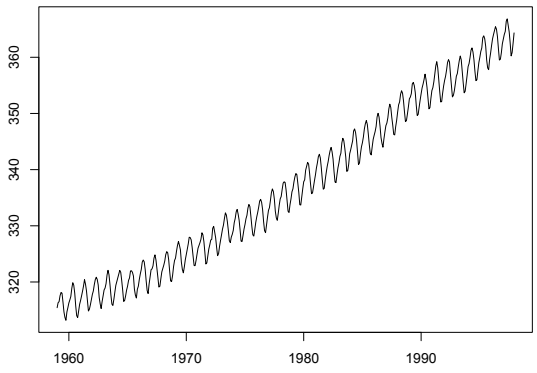


图 1.3: Mauna Loa 二氧化碳浓度

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？

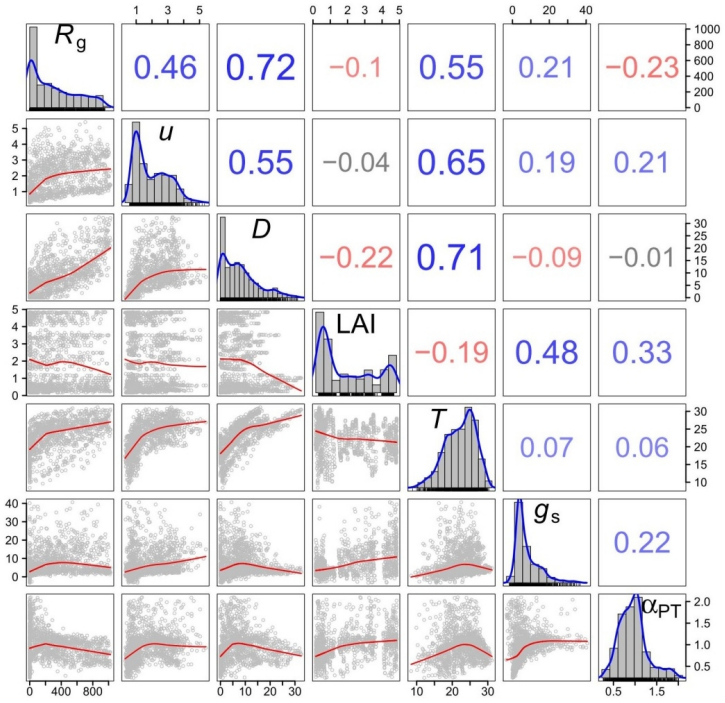


图 1.4: 增强版两两散点图

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

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)
```

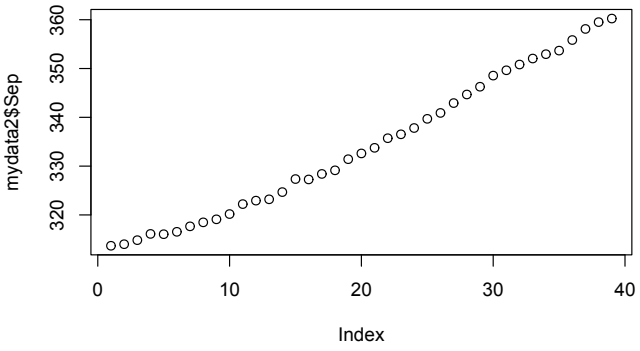


图 3.1: plot() 函数第一种用法示例：一维数据散点图

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)
```

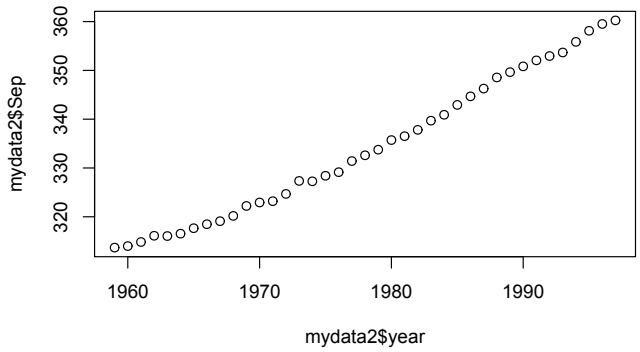


图 3.2: plot() 函数第二种用法示例：二维数据散点图

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")
```

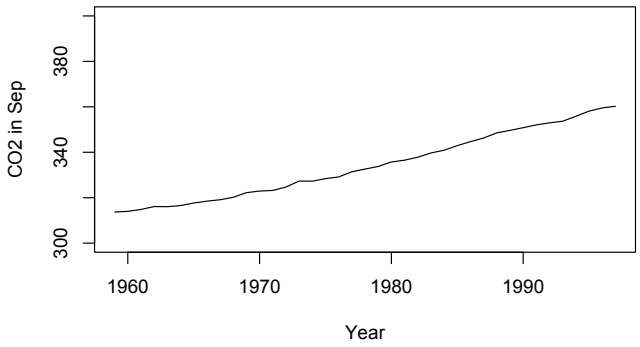


图 3.3: plot() 函数 type 参数的用法示例

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：

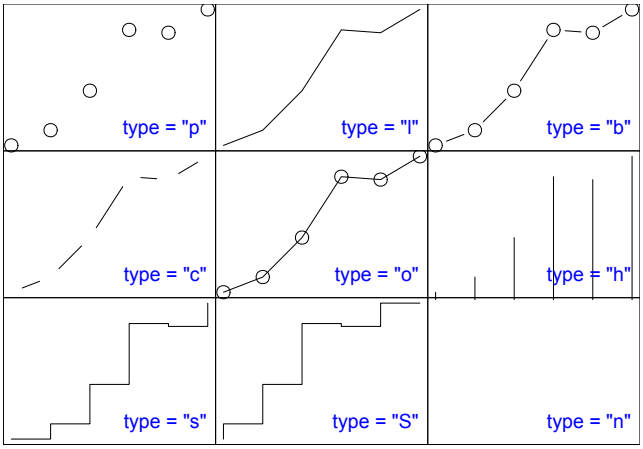
```
mydata2 <- read.table(file = myfile2,
                      header = TRUE, sep = ",")
```

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')
```

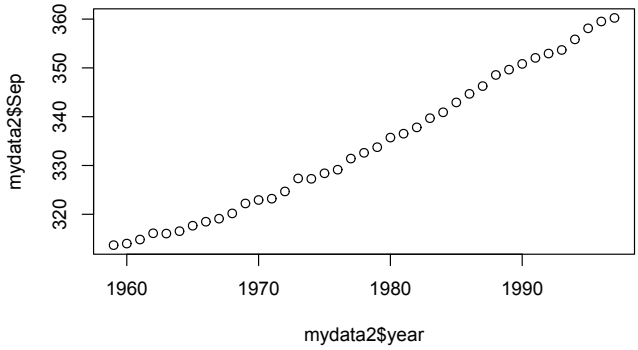


图 3.4: `plot()` 函数：`pch` 默认值

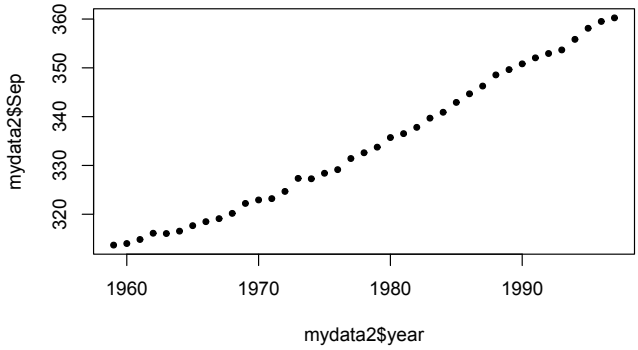


图 3.5: `plot()` 函数示例：`pch = 20`

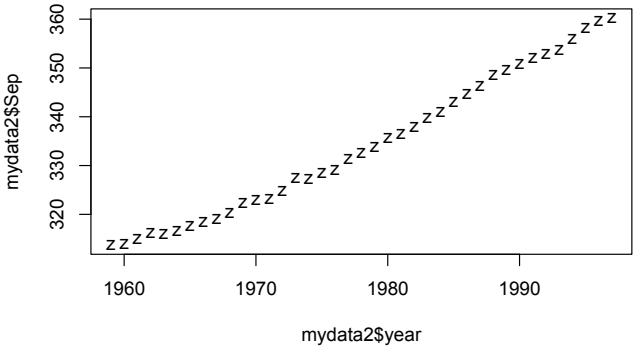


图 3.6: plot() 函数示例：pch = "z"

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)
```

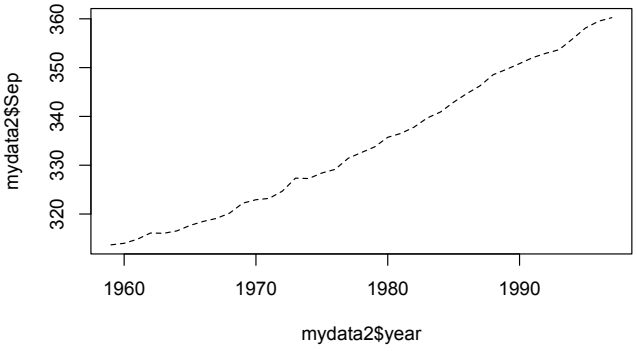
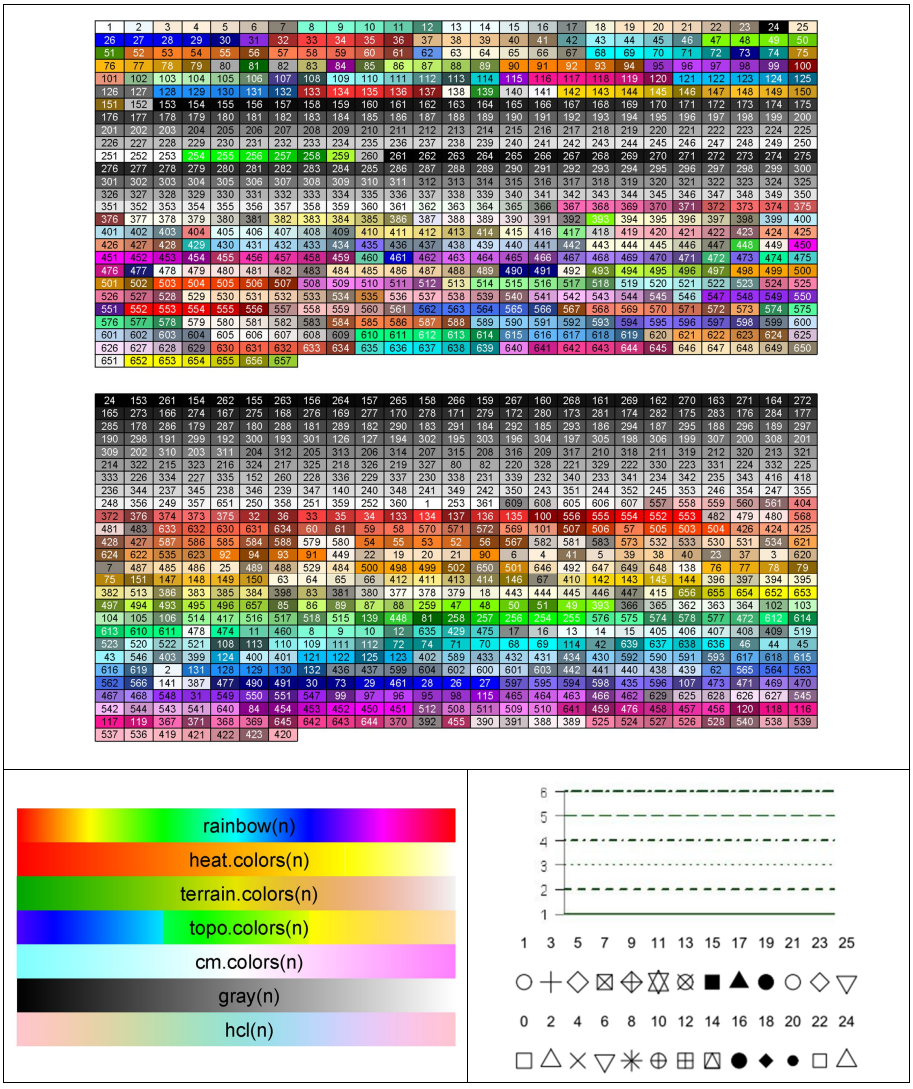


图 3.7: plot() 函数示例：lty 参数

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')
```

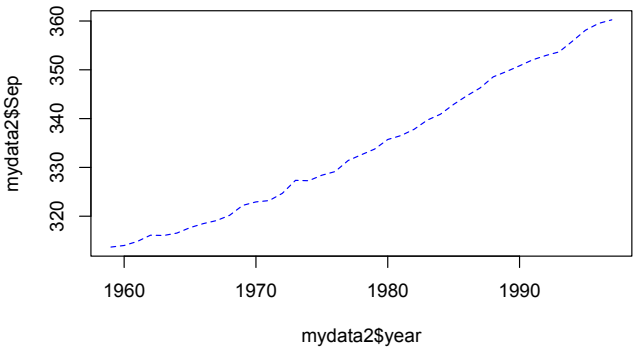


图 3.8: plot() 函数示例：蓝色数据点

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）：

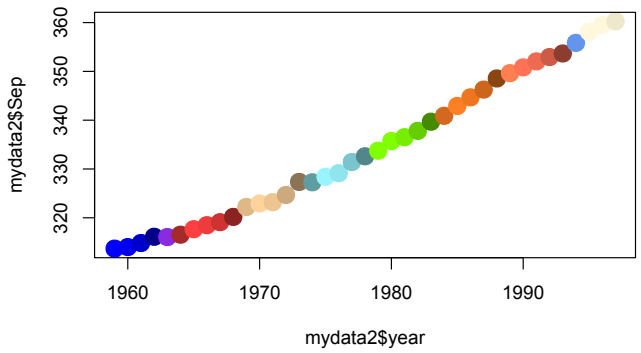


图 3.9: `plot()` 函数示例：多色数据点

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

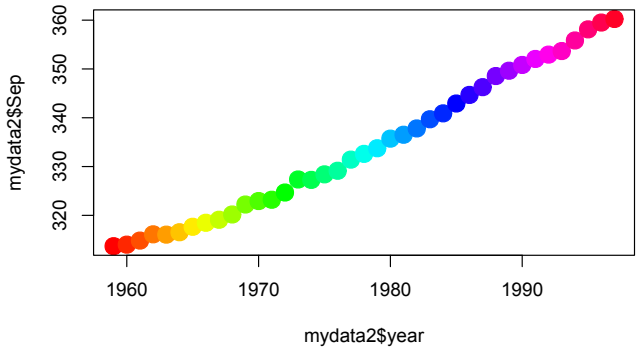


图 3.10: `plot()` 函数示例：`col` 参数彩虹数据点

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。

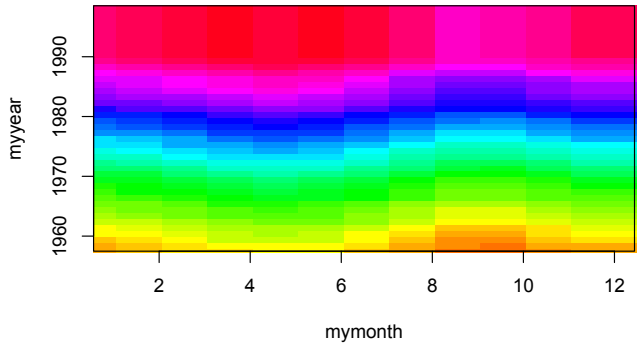


图 3.11: `plot()` 函数示例：三维图

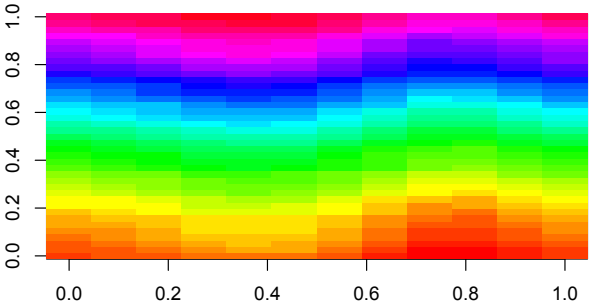


图 3.12: `image()` 函数示例

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。

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：

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"
```

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/

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

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)]
```

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/`