

大数据应用与技术丛书

Beyond Spreadsheets with R

R

数据加工与 分析呈现宝典



[美] 乔纳森·卡罗尔(Jonathan Carroll) 著
蒲 成 译

 MANNING

清华大学出版社

大数据应用与技术丛书

R 数据处理与分析 呈现宝典

[美] 乔纳森·卡罗尔(Jonathan Carroll) 著
蒲 成 译

清华大学出版社

北 京

Jonathan Carroll

Beyond Spreadsheets with R

EISBN: 978-1-61729-459-4

Original English language edition published by Manning Publications, USA (c) 2019 by Manning Publications. Simplified Chinese-language edition copyright (c) 2019 by Tsinghua University Press Limited. All rights reserved.

北京市版权局著作权合同登记号 图字: 01-2019-1498

本书封面贴有清华大学出版社防伪标签, 无标签者不得销售。

版权所有, 侵权必究。侵权举报电话: 010-62782989 13701121933

图书在版编目(CIP)数据

R 数据加工与分析呈现宝典 / (美)乔纳森·卡罗尔(Jonathan Carroll) 著; 蒲成 译. —北京: 清华大学出版社, 2019

(大数据应用与技术丛书)

书名原文: Beyond Spreadsheets with R

ISBN 978-7-302-53486-0

I. ①R… II. ①乔… ②蒲… III. ①程序语言—程序设计 IV. ①TP312

中国版本图书馆 CIP 数据核字(2019)第 179542 号

责任编辑: 王 军

封面设计: 孔祥峰

版式设计: 思创景点

责任校对: 牛艳敏

责任印制: 丛怀宇

出版发行: 清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址: 北京清华大学学研大厦 A 座 邮 编: 100084

社 总 机: 010-62770175 邮 购: 010-62786544

投稿与读者服务: 010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈: 010-62772015, zhiliang@tup.tsinghua.edu.cn

印 装 者: 三河市少明印务有限公司

经 销: 全国新华书店

开 本: 170mm×240mm 印 张: 21.25 字 数: 428 千字

版 次: 2019 年 9 月第 1 版 印 次: 2019 年 9 月第 1 次印刷

定 价: 68.00 元

产品编号: 080262-01

译者序

自 R 问世以来，一直是数据科学处理领域中被广泛使用的利器。虽然 R 诞生于 20 世纪 80 年代，但由于其开源特性以及丰富的生态，相关社区也多种多样，因此 R 也一直在与时俱进，并且成为一套完善的数据处理、计算和制图领域专业语言。作为一种统计分析语言，R 集统计分析与图形显示于一体，它可以运行于 Linux、Windows 和 macOS 操作系统上，而且嵌入了一个非常方便实用的帮助系统。

R 语言的使用很大程度上是借助各种 R 插件包的辅助，从某种程度上讲，R 插件包是针对 R 的插件，不同的插件满足不同的需求。因此，在 R 语言的学习过程中，对于各种插件包的学习是必不可少的，甚至从某种程度上来说，要熟练地将 R 应用于我们的工作中，前提就是必须掌握 R 各个插件包的使用。反之，也正是因为插件包的存在以及广大开发者持续地开发出新的插件包，才使得 R 能够得心应手地应对任何数据处理任务。

本书是一本 R 的入门级书籍，主要是讲解 R 的基础知识点以及简单的编程知识，并不涵盖统计学领域的知识。本书内容涵盖了 R 的安装和环境部署，并且大部分篇幅都是在介绍通过 R 的各种插件包对数据进行处理和呈现的方法。相信在通读并深刻理解了本书的知识内容之后，读者就能够熟练运用 R 进行基本的数据分析了。本书提供了许多应用示例，并且每一章结尾处都有归纳总结和练习实践，这样一来，读者就能通过每一章的知识内容并且结合实践练习来巩固在本书中所学习到的知识。

任何一种编程语言及其开发环境都仅仅是我们达成目的的工具而已。因此，作为使用者，我们所要做的是学习并掌握其使用方法。就这方面而言，本书可以称得上是一本非常完备的工具类书籍。对于没有任何 R 使用经验的读者，跟随本书的章节内容就可以完全掌握 R 的基本使用方法，并能熟练地对数据进行基础的分析处理，而这也是作者编写本书的目的。希望会有越来越多的读者投身到 R 的生态系统之中！

在此要特别感谢清华大学出版社的编辑们，在本书翻译过程中他们提供了颇有助益的帮助，没有其热情付出，本书将难以付梓。

由于译者水平有限，难免会出现一些错误或翻译不准确的地方，如果有读者能够指出并勘正，译者将不胜感激。

译者

2019 年 4 月

作者简介



Jonathan Carroll 拥有澳大利亚阿德莱德大学的理论天体物理学博士学位, 目前作为独立承包商提供数据科学领域的 R 编程服务。他为 R 贡献了许多插件包, 并且经常在 Stack Overflow 上解答问题, 他还热衷于科学传播。

前言

数据无处不在，并且它们被以这样或那样的方式用于几乎每一个行业。其中一种最常见的与数字或文本数据进行交互的方式是电子表格软件。这种方式提供了几种有用的特性：以表格视图呈现数据，允许使用那些值执行计算，以及生成数据汇总。而电子表格往往无法提供的是以可重复、可重现或者编程方式执行这些处理的方法(不需要单击或复制粘贴)。电子表格对于展示数据(包括有限的数据汇总)而言是很好；不过当我们希望对数据进行一些真正高级的处理时，需要借助一种编程语言来实现。

数据再加工——处理原始数据——是数据科学的基石。再加工技术包括清洗、排序、分析、过滤，以及让数据变得真正有用所需要做的其他一切处理。人们常说，90%的数据科学工作都是在准备数据，而其余的10%才是对数据进行实际处理。不要低估仔细准备数据的重要性；分析结果取决于这一步是否正确。

使用编程语言来执行数据再加工意味着对数据进行的处理会被记录下来，可以从原始数据源中再现，并且后续可以被检查分析——甚至在需要的时候可以对其进行修改。尝试在电子表格中这样做意味着要么记录下何时要按下哪个按钮，要么输出和输入之间会联系不上。

我喜欢使用R，它在许多方面都是很有用的。我从未想过一门语言可以如此灵活，以至于它在某个时间可以计算t检验，然后接下来又去请求Uber接口。本书的每一个单词都已经被R代码处理过，每一行结果都是由实际的R代码生成的，并且使用第三方R包(knitr)将这些处理放在了一起。我将R用于我的绝大部分工作中，数据再加工和分析工作都用到了它，而这些工作在这么多年里已经从评估渔业资源变成癌症药物试验中的遗传因素评估。如果我受限于在电子表格程序中开展工作，则无法完成这些任务中的任何一个。

阅读完本书之后，读者将了解到足够多的R编程语言的来龙去脉，从而能够将感兴趣的数据放入R中并且得到一项超越电子表格所能完成的分析。

注意：在这里提醒一下手头有本书盗版版本的那些读者。侵犯版权的通常理由都是由那些从中获利的人打着“没人会有任何损失”的旗号而提出来的。“没人会有任

何损失”这句话没有错，不过仅有侵权者获得了利益。本书的编写和出版耗费了大量的精力，如果没有从正规途径购买本书，那么这些读者从阅读本书中所得到的获益将不为人所注意和感激。如果读者手头有本书的非官方副本并且发现这本书很有用，那么请考虑购买一本正版书，这样一来，无论是对于读者自身还是对于可能从正版销售行为中受益的人而言都是有好处的。

致 谢

我要感谢 Manning 出版社给了我撰写本书的机会，尤其要感谢幕后致力于本书整个制作出版过程的大型团队，其中包括编辑 Jenny Stout，由 Kevin Sullivan、Janet Vail 和 Tiffany Taylor 组成的制作团队以及技术审校 Hilde Van Gysel。还要感谢在本书编著期间提供了宝贵反馈的那些倾力付出的专业审稿人，其中包括：Anil Venugopal、Carlos Aya Moreno、Chris Heneghan、Daniel Zingaro、Danil Mironov、Dave King、Fabien Tison、Irina Fedko、Jenice Tom、Jobinesh Purushothaman、John D. Lewis、John MacKintosh、Michael Haller、Mohammed Zuhair Al-Taie、Nii Atttoh-Okine、Stuart Woodward、Tony M. Dubitsky 以及 Tulio Albuquerque。

我还要感谢 Stack Overflow 和 Twitter 上极其有帮助的社区，并且要特别感谢 AsciiDoctor 小组，他们制作了一套了不起的发布工具链。

非常感激为本书提供支持的 R 社区的所有成员，其中大部分都自愿贡献了用于改进和扩展该语言的方案。我收到了来自审稿人、Twitter 关注者以及我的同事们关于本书内容的各种反馈、建议、评论和讨论。有了他们的帮助，本书的出版才变成现实，因此我要感谢他们每一个人。

本书中所提及的 R 插件包的维护者值得特别推荐。tidyverse 的包已经改变了我使用 R 的方式，并且让数据处理变得更加简单。如果没有 knitr 包，就不可能生成本书的代码输出，因此我最想感谢的就是 knitr 包。

我要感谢我的妻子和孩子，她们在我编写本书的约两年时间里给予我很大的支持，如果没有她们的支持，我必定是无法坚持下来的。

最后但同样重要的是，我非常感谢 R 语言本身背后的团队。这是一款开源软件，可供用户免费使用。作为其用户，我们非常感激该团队为了持续维护和改进这一大型项目所付出的不懈努力。可以在 R 中通过 `citation()` 函数找到其引证，该函数会生成以下信息：

```
R Core Team (2017). R: A language and environment for statistical computing.  
R Foundation for Statistical Computing, Vienna, Austria. URL https://  
www.R-project.org/.
```


关于本书

本书读者对象

翻阅本书的人肯定就是本书的读者。鉴于此，我猜测你们手头已经有一些数据(可能是存储为电子表格的形式)，并且并不十分确定要对其做何种处理。没关系，这样甚至更好。可能你希望从数据中了解一些信息；也可能是希望找到一种新的方式与数据进行交互；还可能是希望基于这些数据绘制一张图表。这些目标都很好，不过我还是认为你可能更希望学习如何进行一些编程处理。

本书内容将所有读者都视为无编程经验并且也不熟悉编程领域的专业术语。可能有些读者已经阅读过一些编程书籍并且感到过惶恐不安，因为那些书中的介绍材料是跳跃式讲解的，以试图让读者快速理解某种语言运转方式的每一处细微差异。不要担心，本书内容并非如此，本书将逐步介绍所有的内容，并且引用大量的示例，以便让读者在阅读完本书时能够熟练地对数据进行所期望的处理。

本书不打算涉及统计学知识，那是其他书籍应该讲解的主题。如果读者不具备统计学背景，请不要担心，阅读本书并不需要这些。本书的重点是 R 语言编程，而非统计学(虽然 R 语言在这方面很擅长)。

当读者完成本书阅读时，应该会大体理解编程方面的知识以及如何利用 R 语言进行编程；如何调研、审查以及使用数据来获得见解；如何做好准备以便创建一个稳健、可重现的工作流并且使用数据来强化我们的结论。

本书将介绍如何通过远比任何电子表格软件更具灵活性的方式来使用一个小的数据集并且将其转换成有意义、具备发表质量的图形。只要使用十多个命令，我们就能将图 1 中所示的数据(正如 RStudio 数据查看器中所示，R 中已经提供了 `mtcars` 数据集)转变为图 2 中的图形。

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

图1 R中提供的mtcars数据集(正如RStudio数据查看器中所示)。这些数据提取自1974年的*Motor Trend US*杂志,其中包括32种车型(1973—1974年的品牌和型号)油耗、汽车设计和性能等10个方面

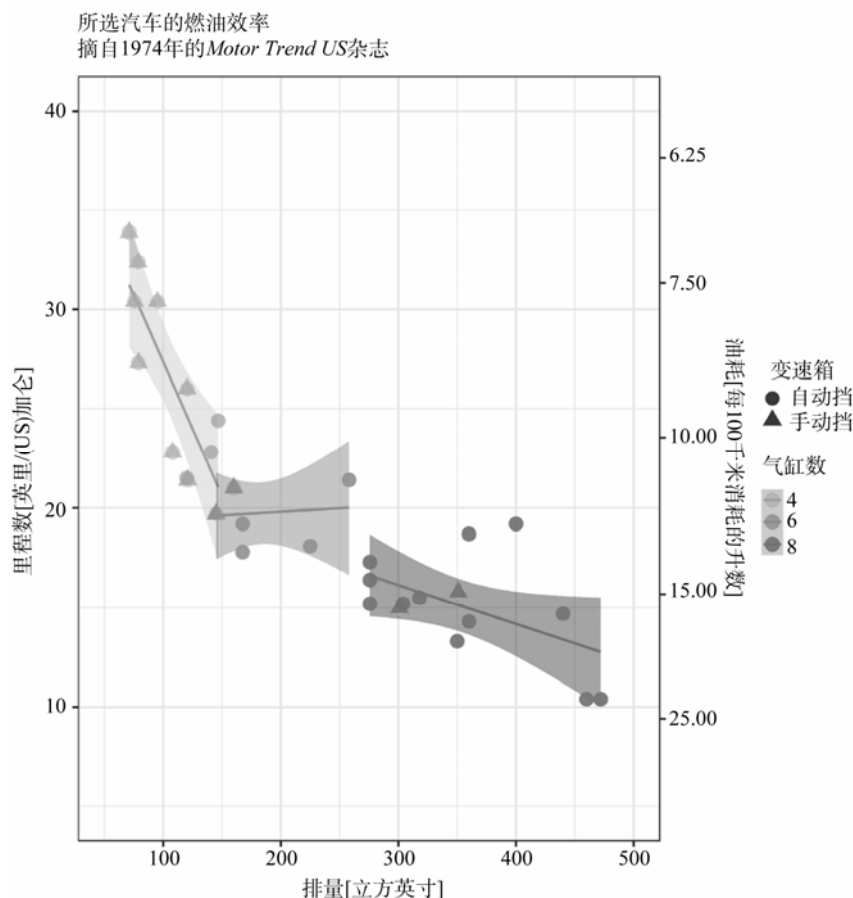


图2 这一 `mtcars` 数据集的可视化图表绘制了针对 32 款车引擎排量(`disp`)的每加仑英里数(`mpg`, 也就是变换了单位的油耗), 这些数据是按气缸数量(`cyl`)来分组的, 并且是根据其变速箱类型(`am`)来区分的, 图表中还有每组气缸数数据的线性拟合。这全是在十几行 R 代码中实现和格式化的

如何阅读本书

本书的每一章内容都以言简意赅的方式呈现给读者, 其中讲解了哪些是重要的方面以及(如果不注意的话)哪些方面可能会变成问题。本书无法涵盖解决问题的每一种方式, 并且本书解决问题的方式可能并非与其他文章所采用的方式相同。不过在本书中, 我首先会尝试向读者展示我所认为的最佳方法, 然后辅以介绍一些读者可能会在其他学习资料中遇到的备选方法。这样做的目的在于让读者成为合格且工作效率较高的 R 用户, 而这可能意味着向读者展示如何以缓慢方式来执行处理(当然同时也介绍了快速方式)。

格式

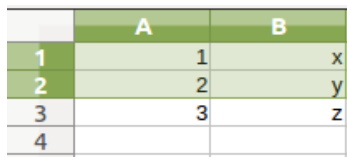
当代码示例生成输出时，这些输出会显示在输入下方，并且具有#>前缀，如果读者自行运行这些代码，那么通常应该预期会看到相同的输出。在撰写本书期间，绝大部分示例的输出都已经通过 R 本身来生成了。在尝试运行以#>开头的代码行时请不要担心，它们会被 R 所忽略。有时示例会显示为带注释的代码块。

特定类型的信息将通过“注意”“警告”和“提示”部分突出显示。

在某些例子中，并未和代码块一起提供其输出，因为实际上并没有运行这些代码。这些代码块仅用作阐释目的。在显示输出的地方，当读者运行代码时应该预期得到类似的结果。

R 生成的错误都会以 Error 这个单词作为开头。在本书中，读者将看到代码里存在大量这样的错误。错误的准确单词可能在不同的版本之间会有所不同。在输入包含这些错误之一的代码块时要注意，因为 R 无法解析该输出。

本书还会讲解电子表格对等功能点可能会是什么样的。本书使用的是 LibreOffice，它看起来就像图 3 这样，不过其概念通常可以扩展到 Excel、Google Sheets 或者读者经常使用的其他任何电子表格软件。



	A	B
1	1	x
2	2	y
3	3	z
4		

图 3 LibreOffice(Linux)中选中的单元格的示例

结构

在循序渐进地阅读本书时，读者会发现其中存在大量的示例，希望读者能够研习一下这些示例。不要只是阅读它们——要亲自在计算机上运行它们，并且查看是否能得到相同的答案。然后尝试基于示例进行一下调整，并且看看是否会得到预期的结果。如果得到不同的结果，那会很好！这意味着你已经找出了可以从中吸取的经验，而下一个任务就是要理解为何会出现这样的结果。

本书将尝试逐步地构建读者关于相关编程和 R 特定术语的知识体系，因此当某些内容看起来不熟悉时，可以主动地回顾温习。

入门准备

以下各项是读者所需要的：

- 本书

- 计算机
- 期望学习一些知识

R 是一种免费语言，并且我们将使用更多的免费软件与其交互。读者可能需要通过互联网连接才能下载这些软件，不过之后大部分示例都可以离线运行。

遵照所出现的示例进行练习，尝试不同的值以查看能否得到预期的结果，修改一些处理并且尝试理解所执行的效果。重启 R 可以解决任何麻烦，因此可以任意进行尝试。

本书不一定会指引读者如何解决所面临的具体问题，不过在阅读完本书之后，读者应该足以理解 R 语言及其生态体系，从而开始探究可能需要用到的其他工具。如果读者是在基因学领域工作，那么很可能需要 Bioconductor 套件包(www.bioconductor.org)所提供的一些更加高级的工具，其中所使用的许多概念和结构都是基于本书所讲解的那些知识点来扩展的(不过本书不会介绍这些扩展概念和结构)。

何处可以找到更多的帮助信息

Stack Overflow(<https://stackoverflow.com>)的 r 标签是一个极其有用的信息源，不过它经常被一些低级或重复性问题以及没人领情的反馈所滥用。在寻求其他人来解决问题之前，请花些时间查找一下该问题是否已经有答案(这种情况经常会出现，因为已经有许多问题被提出过)。

如果这些尝试都失败了，那么在搜索引擎(如 Google)中输入我们所知道的术语和 r 或 rstats 往往会产生一些有用的结果。

R Weekly 站点(<https://rweekly.org>)提供了网络上最有意义的 R 帖子的每周汇总，R-bloggers(<https://r-bloggers.com>)提供了许多受欢迎的 R 相关博客的信息集合，并且会每天更新这些信息。可以关注其中与我们的关注点保持一致的一些博客，这样一定会获取一些有用的技巧信息。

最后，可参与到本地社区之中，要么亲身参与(试试 <https://meetup.com>)，要么在线参与(Twitter)。

关于本书的更多信息

本书是使用 emacs 和 RStudio 以 AsciiDoc 纯文本标记语言来编写的。其中的 R 代码是使用通过 switchr 包所定义的一个自定义包库来执行的，并且是使用 knitr 包将其交织在源中的。

描述定义此自定义库的环境的会话信息如下所示：

```

#> setting      value
#> version      R version 3.4.3 (2017-11-30)
#> system       x86_64, linux-gnu
#> ui           X11
#> language     en_AU:en
#> collate      en_AU.UTF-8
#> tz           Australia/Adelaide
#> date         2018-01-23
#>
#> package      *   version date      source
#> assertthat   0.2.0   2017-04-11  CRAN (R 3.4.3)
#> backports    1.1.2   2017-12-13  CRAN (R 3.4.3)
#> base         *   3.4.3   2017-12-01  local
#> bindr        0.1     2016-11-13  CRAN (R 3.4.3)
#> bindrcpp     0.2     2017-06-17  CRAN (R 3.4.3)
#> broom        0.4.3   2017-11-20  CRAN (R 3.4.3)
#> cellranger   1.1.0   2016-07-27  CRAN (R 3.4.3)
#> cli          1.0.0   2017-11-05  CRAN (R 3.4.3)
#> colorspace   1.3-2   2016-12-14  CRAN (R 3.4.3)
#> commonmark   1.4     2017-09-01  CRAN (R 3.4.3)
#> compiler     3.4.3   2017-12-01  local
#> crayon       1.3.4   2017-09-16  CRAN (R 3.4.3)
#> crosstalk    1.0.0   2016-12-21  CRAN (R 3.4.3)
#> curl         3.1     2017-12-12  CRAN (R 3.4.3)
#> data.table   1.10.4-3  2017-10-27  CRAN (R 3.4.3)
#> datasauRus   *   0.1.2   2017-05-08  CRAN (R 3.4.3)
#> datasets     *   3.4.3   2017-12-01  local
#> devtools     *   1.13.4  2017-11-09  CRAN (R 3.4.3)
#> digest       0.6.14  2018-01-14  CRAN (R 3.4.3)
#> dplyr        *   0.7.4   2017-09-28  CRAN (R 3.4.3)
#> evaluate     0.10.1  2017-06-24  CRAN (R 3.4.3)
#> forcats     *   0.2.0   2017-01-23  CRAN (R 3.4.3)
#> foreign      0.8-67  2016-09-13  CRAN (R 3.3.1)
#> ggplot2     *   2.2.1   2016-12-30  CRAN (R 3.4.3)
#> glue         1.2.0   2017-10-29  CRAN (R 3.4.3)
#> graphics    *   3.4.3   2017-12-01  local
#> grDevices   *   3.4.3   2017-12-01  local
#> grid         3.4.3   2017-12-01  local
#> gtable       0.2.0   2016-02-26  CRAN (R 3.4.3)
#> haven       1.1.1   2018-01-18  CRAN (R 3.4.3)
#> here        *   0.1     2017-05-28  CRAN (R 3.4.3)
#> hms         0.4.0   2017-11-23  CRAN (R 3.4.3)
#> htmltools    0.3.6   2017-04-28  CRAN (R 3.4.3)
#> htmlwidgets *   1.0     2018-01-20  CRAN (R 3.4.3)
#> httpuv       1.3.5   2017-07-04  CRAN (R 3.4.3)
#> httr        *   1.3.1   2017-08-20  CRAN (R 3.4.3)
#> jsonlite     1.5     2017-06-01  CRAN (R 3.4.3)
#> knitr        *   1.18    2017-12-27  CRAN (R 3.4.3)
#> lattice      0.20-35  2017-03-25  CRAN (R 3.3.3)
#> lazyeval     0.2.1   2017-10-29  CRAN (R 3.4.3)
#> leaflet     *   1.1.0   2017-02-21  CRAN (R 3.4.3)
#> lubridate    1.7.1   2017-11-03  CRAN (R 3.4.3)
#> magrittr     1.5     2014-11-22  CRAN (R 3.4.3)
#> mapproj     *   1.2-5   2017-06-08  CRAN (R 3.4.3)
#> maps        *   3.2.0   2017-06-08  CRAN (R 3.4.3)
#> memoise     1.1.0   2017-04-21  CRAN (R 3.4.3)

```

```

#> methods      * 3.4.3      2017-12-01  local
#> mime          0.5        2016-07-07  CRAN (R 3.4.3)
#> misc3d        0.8-4      2013-01-25  CRAN (R 3.4.3)
#> mnormt        1.5-5      2016-10-15  CRAN (R 3.4.3)
#> modelr        0.1.1      2017-07-24  CRAN (R 3.4.3)
#> munsell       0.4.3      2016-02-13  CRAN (R 3.4.3)
#> nlme          3.1-131    2017-02-06  CRAN (R 3.4.0)
#> openxlsx      4.0.17     2017-03-23  CRAN (R 3.4.3)
#> parallel      3.4.3      2017-12-01  local
#> pillar        1.1.0      2018-01-14  CRAN (R 3.4.3)
#> pkgconfig     2.0.1      2017-03-21  CRAN (R 3.4.3)
#> plot3D        * 1.1.1      2017-08-28  CRAN (R 3.4.3)
#> plyr          1.8.4      2016-06-08  CRAN (R 3.4.3)
#> psych         1.7.8      2017-09-09  CRAN (R 3.4.3)
#> purr          * 0.2.4      2017-10-18  CRAN (R 3.4.3)
#> R6            2.2.2      2017-06-17  CRAN (R 3.4.3)
#> Rcpp          0.12.15    2018-01-20  CRAN (R 3.4.3)
#> readr        * 1.1.1      2017-05-16  CRAN (R 3.4.3)
#> readxl        1.0.0      2017-04-18  CRAN (R 3.4.3)
#> reshape2     * 1.4.3      2017-12-11  CRAN (R 3.4.3)
#> rex           * 1.1.2      2017-10-19  CRAN (R 3.4.3)
#> rio           * 0.5.5      2017-06-18  CRAN (R 3.4.3)
#> rlang         * 0.1.6      2017-12-21  CRAN (R 3.4.3)
#> rmarkdown    * 1.8        2017-11-17  CRAN (R 3.4.3)
#> roxygen2     * 6.0.1      2017-02-06  CRAN (R 3.4.3)
#> rprojroot     1.3-2      2018-01-03  CRAN (R 3.4.3)
#> rstudioapi    0.7        2017-09-07  CRAN (R 3.4.3)
#> rvest         0.3.2      2016-06-17  CRAN (R 3.4.3)
#> scales       0.5.0      2017-08-24  CRAN (R 3.4.3)
#> shiny        1.0.5      2017-08-23  CRAN (R 3.4.3)
#> stats        * 3.4.3      2017-12-01  local
#> stringi      1.1.6      2017-11-17  CRAN (R 3.4.3)
#> stringr      * 1.2.0      2017-02-18  CRAN (R 3.4.3)
#> switchr      * 0.12.6     2017-11-07  CRAN (R 3.4.1)
#> testthat     * 2.0.0      2017-12-13  CRAN (R 3.4.3)
#> tibble       * 1.4.1      2017-12-25  CRAN (R 3.4.3)
#> tidyr        * 0.7.2      2017-10-16  CRAN (R 3.4.3)
#> tidyverse    * 1.2.1      2017-11-14  CRAN (R 3.4.3)
#> tools        3.4.3      2017-12-01  local
#> utils        * 3.4.3      2017-12-01  local
#> withr        2.1.1      2017-12-19  CRAN (R 3.4.3)
#> xml2         1.1.1      2017-01-24  CRAN (R 3.4.3)
#> xtable       1.8-2      2016-02-05  CRAN (R 3.4.3)

```

附录 C 中提供了安装这些插件包的特定版本的详细介绍。本书中的示例代码可以在 <https://github.com/BeyondSpreadsheetsWithR/Book> 找到，也可扫封底二维码获取。还有一个问题跟踪系统，读者可将遇到问题的 R 代码直接链接到其中：<https://github.com/BeyondSpreadsheetsWithR/Book/issues>。

本书论坛

购买了本书的读者可以免费访问 Manning 出版社所运营的私有网站论坛，读者可以在其中发表关于本书的评论，提出技术问题，并且接收来自本书作者和其他用户的帮助。要访问该论坛，可以打开 <https://forums.manning.com/forums/beyond-spreadsheets-with-r>，也可以在 <https://forums.manning.com/forums/about> 了解与该论坛有关的更多信息以及该论坛的行为准则。

Manning 可以确保为本书读者提供一个在线场所，以便读者与读者之间以及读者与作者之间可进行有意义的对话。但这并不能保证作者会花很多精力在这个论坛中，因为作者对于该论坛的贡献是自愿的(并且是无偿的)。我们建议读者尝试向作者提出有挑战性的问题，以免作者兴趣索然！只要本书还在发行，那么就可以从出版商网站上访问该论坛以及其中所讨论内容的存档。

封面插图声明

本书封面上的插图标题是“Habit of a Turkish Dancer in 1700 Signior”，插图取自 Thomas Jefferys 的 *A Collection of the Dresses of Different Nations, Ancient and Modern (4 volumes)*，这些书在 1757—1772 年于伦敦出版。标题页表明了，这些是手工上色的铜版画，使用阿拉伯树胶增加厚度。

Thomas Jefferys(1719—1771)被称为“国王乔治三世的地理学家”，他是一位英国制图师，是当时顶尖的地图供应商，他为政府和其他官方机构制版和印刷地图，制作了大量的商业地图和地图集，尤其是北美地图集。作为一名地图绘制师，他的工作激起了人们对他所调查地区的当地服饰习俗的兴趣，这些服饰在这套四卷书集中得到了很好的展示。在 18 世纪末，兴起了一股风潮，人们开始向往远方并享受旅行的乐趣。像 Jefferys 画作这样的收藏品是很流行的，它为旅行者和向往旅行但是没能出发的人们介绍异域居民是什么样子。

Jefferys 画作藏品的多样性生动描绘了两百多年前各个国家的独特性。从那以后，着装上就发生了变化，而当时各国家、各地区丰富的多样性也渐渐趋同，现在很难区分来自不同大陆的人们。如果从乐观的角度看，我们是把文化和视觉上的多样性作为代价，换来了更丰富的私人生活，或者变化更大、更有趣的知识和技术生活。

在如今这个计算机图书封面大同小异的时代，Manning 出版社以两个世纪前丰富多样的地域生活为基础设计图书封面，这令 Jefferys 的画作重新焕发生机，并颂扬了计算机行业的革新性和首创精神。

目 录

第 1 章 数据与 R 语言介绍	1
1.1 什么是数据、数据在哪里 以及如何处理数据	2
1.1.1 什么是数据	2
1.1.2 将周围的一切都视为 数据源	2
1.1.3 数据再加工	4
1.1.4 使用处理得当的数据可以 做些什么	4
1.1.5 数据就是资产	8
1.1.6 可重复的研究和版本 控制	9
1.2 R 语言介绍	11
1.2.1 R 的起源	12
1.2.2 R 能够以及不能完成 哪些工作	13
1.3 R 的运行机制	14
1.4 RStudio 介绍	17
1.4.1 在 RStudio 中使用 R	17
1.4.2 内置插件包(数据和 函数)	22
1.4.3 内置文档	23
1.4.4 简介	24
1.5 亲自尝试	24
1.6 专业术语	25
1.7 本章小结	25

第 2 章 了解 R 数据类型	27
2.1 数据类型	28
2.1.1 数字	28
2.1.2 文本(字符串)	31
2.1.3 类别(因子)	33
2.1.4 日期和时间	35
2.1.5 逻辑值	37
2.1.6 缺失值	38
2.2 存储值(赋值)	38
2.2.1 命名数据(变量)	39
2.2.2 固定不变的数据	44
2.2.3 赋值运算符(<-与=的 对比)	45
2.3 指定数据类型	47
2.4 告知 R 忽略某些内容	51
2.5 亲自尝试	52
2.6 专业术语	53
2.7 本章小结	53
第 3 章 生成新数据值	55
3.1 基础数学算法	55
3.2 运算符优先顺序	58
3.3 字符串串联(连接)	59
3.4 比较	61
3.5 自动转换(强制)	65
3.6 亲自尝试	67
3.7 专业术语	68
3.8 本章小结	68

第4章 理解将要使用的工具：

函数	69
4.1 函数	69
4.1.1 表象之下	72
4.1.2 函数模板	74
4.1.3 参数	77
4.1.4 多个参数	79
4.1.5 默认参数	82
4.1.6 参数名称匹配	83
4.1.7 部分匹配	86
4.1.8 作用域	87
4.2 插件包	92
4.2.1 安装插件包	93
4.2.2 R 如何获知这个函数	96
4.2.3 名称空间	97
4.3 消息、警告和错误	98
4.3.1 创建消息、警告和 错误	99
4.3.2 诊断消息、警告和 错误	101
4.4 测试	103
4.5 项目：泛化一个函数	104
4.6 亲自尝试	105
4.7 专业术语	106
4.8 本章小结	106

第5章 组合数据值 107

5.1 简单集合	107
5.1.1 强制转换	109
5.1.2 缺失值	110
5.1.3 属性	111
5.1.4 名称	112
5.2 序列	113
5.2.1 向量函数	118
5.2.2 向量数学运算	119
5.3 矩阵	121
5.4 列表	124

5.5 data.frame	127
5.6 class 属性	131
5.6.1 tibble 类	133
5.6.2 将结构用作函数参数	137
5.7 亲自尝试	138
5.8 专业术语	139
5.9 本章小结	139

第6章 选取数据值 141

6.1 文本处理	142
6.1.1 文本匹配	142
6.1.2 子字符串	144
6.1.3 文本替换	145
6.1.4 正则表达式	145
6.2 从结构中选取组成部分	148
6.2.1 向量	148
6.2.2 列表	151
6.2.3 矩阵	155
6.3 值的替换	157
6.4 data.frame 和 dplyr	161
6.4.1 dplyr 动词	162
6.4.2 非标准计算	164
6.4.3 管道	166
6.4.4 以困难方式对 data.frame 取子集	169
6.5 替换 NA	172
6.6 条件式选取	173
6.7 汇总值	176
6.8 一个行之有效的示例： Excel 与 R 的对比	179
6.9 亲自尝试	181
6.10 专业术语	183
6.11 本章小结	183

第7章 对大量数据进行处理 185

7.1 整洁数据原则	185
7.1.1 工作目录	187

7.1.2 存储数据格式	189	9.2.3 样式美学	246
7.1.3 将数据读入 R 中	190	9.2.4 添加线条	250
7.1.4 抓取数据	193	9.2.5 添加柱状图	253
7.1.5 检查数据	197	9.2.6 其他图表类型	258
7.1.6 处理数据中奇怪的值 (警示值)	200	9.2.7 刻度	261
7.1.7 转换成整洁数据	201	9.2.8 切面	268
7.2 合并数据	205	9.2.9 额外的选项	272
7.3 写出 R 中的数据	210	9.3 作为对象的图表	275
7.4 亲自尝试	214	9.4 保存图表	278
7.5 专业术语	214	9.5 亲自尝试	278
7.6 本章小结	214	9.6 专业术语	279
9.7 本章小结	279		
第 8 章 根据条件进行处理：控制 结构	217	第 10 章 借助扩展插件包对数据进行 更多的处理	281
8.1 循环	217	10.1 编写自己的插件包	282
8.1.1 向量化	218	10.1.1 创建一个最小化的 插件包	282
8.1.2 整洁的重复：使用 purrr 进行循环	219	10.1.2 文档	283
8.1.3 for 循环	224	10.2 对插件包进行分析	287
8.2 更大或更小的循环作 用域	227	10.2.1 单元测试	287
8.3 条件式执行	229	10.2.2 剖析	290
8.3.1 if 条件	229	10.3 接下来做什么	291
8.3.2 ifelse 条件	233	10.3.1 回归分析	291
8.4 亲自尝试	236	10.3.2 聚类分析	294
8.5 专业术语	237	10.3.3 使用地图	297
8.6 本章小结	238	10.3.4 与 API 进行交互	300
		10.3.5 插件包共享	302
第 9 章 数据可视化：绘图	239	10.4 更多资源	303
9.1 数据准备	239	10.5 专业术语	303
9.1.1 再次介绍整洁数据	240	10.6 本章小结	304
9.1.2 数据类型的重要性	240		
9.2 ggplot2	240	附录 A 安装 R	305
9.2.1 通用构造	241	附录 B 安装 RStudio	307
9.2.2 添加数据点	244	附录 C base R 中的图形	309

第 1 章

数据与 R 语言介绍

本章涵盖：

- 为什么数据分析很重要
- 如何让数据分析变得稳健
- R 如何以及为何适用于数据处理
- Rstudio——R 的人机交互界面

在数据到位后，你希望开始对数据进行一些有意义的处理，对吗？若是如此，那么阅读本书就对了！我们将尽快讲解其方式方法。不过先别急。如果现在就一头扎进去，则会像是囫圇吞枣。比较好的做法是，先理解我们将要使用的原材料和工具。

我们将讲解数据对于数据所有者以及可能会使用该数据的人而言通常意味着什么——因为如果不全面理解所拥有数据的含义，那么基于这些数据所做的分析处理将不会有什么意义(并且最坏的情况下可能会导致全盘错误)。糟糕的数据处理准备只会推迟对于数据的正确处理操作，并且加大所欠的技术债(这可能会让目前的情况变得简单，但之后在应对糟糕的数据时还是必须要偿还这些技术债)。

我们将探讨如何让自己做好应对严谨分析(可重复进行的分析)的准备，然后开始使用目前可用的其中一款最好的数据分析工具：**R** 编程语言。现在，我们来看看“拥有一些数据”这句话的含义。

1.1 什么是数据、数据在哪里以及如何处理数据

之前提到过，你有一些希望对其进行分析处理的数据，这句话其实并不是太准确。这样做是故意的。我确信，即使你没有意识到，也还是拥有着一些数据的。你可能认为数据无非就是存储在 Excel 文件中的内容，但其实数据本身远不止于此。我们所有人都有数据，因为它无处不在。在分析我们自己的数据之前，重要的是要理解其结构(只要我们理解了，R 也就理解了)，这样我们就能切实理解拥有一些数据到底意味着什么。

1.1.1 什么是数据

数据的存在形式多种多样，并非仅是电子表格中的数字和字母。数据也可能存储为另一种文件类型，如逗号分隔的值(CSV)、书中的文字或者网页表格中的值。

注意：我们通常会将逗号分隔的值存储在.csv 文件中。这一格式极其有用，因为它是纯文本——由逗号分隔的值。1.1.6 节中将回过头来介绍其有用的原因。

数据也可能完全不存储——流式数据会像信息流一样流动，如电视所接收和处理的信号、Twitter 提要或者测量设备的输出。如果需要，我们可以存储这些数据，不过通常都希望在数据流入时理解这些数据流。

数据并非总是很干净(实际上，数据大多数时候都是不规范的、单调无趣的)，并且并非总是我们所想要的格式。手头有一些帮助管理数据的工具会是一个很大的优势，并且对于实现可靠目标而言至关重要，不过只有在对数据进行进一步处理前获悉数据所代表的含义时，这一点才会变得有用。“输入的是垃圾，势必也会得到垃圾”这句话的警示是，我们不能指望对糟糕的数据执行分析处理，却又期望得到有意义的结果。你很可能试过，在 Excel 中执行一次计算，却只得到#VALUE!这一结果，这是因为试图用数字除以文本，虽然这些“文本”看起来很像数字。值的类型(文本、数字、图片等)本身可能就是有意义的数据片段，我们将讲解如何最好地使用它们。

那么，什么才是“好的数据”？我们所拥有的值又代表什么？

1.1.2 将周围的一切都视为数据源

我们通过感官(触觉、视觉、听觉、味觉、嗅觉)来体验世界，并且广泛地探究我们周围的环境。这些输入渠道中的每一个都会应对可获取的数据，我们的大脑会处理它们，并且以我们时常认为理所当然的高明的复杂方式将这些信号融合在一起，形成我们对于这个世界的感观和理解。

每次使用其中一种感官时，我们都是在对这个世界进行测量。今天阳光如何？是

否有车在驶近？有什么东西烧起来了？壶里剩余的咖啡是否还够再倒一杯？我们发明了测量工具，以便让我们的生活更为便利并且能够持续处理一些数据——测量温度的温度计、测量重量的秤、测量长度的尺子。

我们还进一步发明了更多的工具用于汇总数据——汽车仪表盘用于简化发动机的内部指标测量；气象站用于汇总温度、风速和气压。在如今这个数字化时代，有海量的数据源可供我们使用。互联网提供了我们可能感兴趣的几乎方方面面的数据，我们也发明了更多的工具来管理这些数据——天气、财经、社交媒体等，所有这些数据都可供我们使用。这个世界确实是由数据构成的。

这并不是说数据是完全有限的。我们会持续添加可用的数据源，并且面对新的问题，我们也可以识别出希望获取的新数据。数据本身也会生成更多的数据。元数据是描述其他一些数据的附加数据——试验对象的数量、测量单位、采样时间、从中收集数据的网站。所有这些也都是数据，并且也都需要被存储、维护以及在其变更时更新。

我们无时无刻不在以各种方式和数据打交道。万维网的其中一个最伟大的成果就是，以最简便的数字化形式为我们收集、整理和汇总数据。想象一下，20 年前，在还未出现智能手机和应用生态系统时，我们是如何叫出租车的。我们要查找出租车公司的电话，然后给他们打电话，告知调度员我们当时在哪里、想去哪里，以及想要在几点乘坐出租车。调度员随后会将我们的请求发送给所有的司机，其中一个司机会接受我们的打车请求。到达目的地之后，我们会付现金或者刷卡，然后得到一张发票。

现在，我们有了设备间的数字化连接、不间断的互联网访问，以及 GPS 定位跟踪，所以其处理过程就简化成打开一款打车应用、输入目的地并且接收预估费用，因为手机已经知道我们所处的位置。该打车程序会接收这些数据并且选取一个合适的位于附近/空闲的司机，交换双方的联系方式以备用，并且将这个司机引导到我们所处的位置。到达目的地后，会在我们的账户中扣除相应的费用，并以电子邮件的形式给我们发送发票。

无论是传统方式还是现代方式，相同的数据都会在不同的参与方之间流动。在现代方式中，需要参与其中的人员较少，因为计算机系统已经获取了相关的数据。我们的手机会与打车应用服务器进行通信，也会与 GPS 系统进行通信以便定位，并且该打车应用服务器会与支付服务器进行通信以便授权该笔支付，同时该打车应用服务器还会与电子邮件服务器通信以便发送发票。

在此过程中的每个节点，都会收集各种数据(在必要时以匿名方式收集)并且将保存这些数据以供后续分析。这个月有多少人叫车去机场？平均的用车距离是多长？平均等车时长是多少？打车费用是苹果设备用户高还是安卓设备用户高？其中一些数据在传统方式下也是可以获取的，但其聚合和对比处理是非常困难的。

许多企业都通过应用程序编程接口(API)对第三方开发人员开放访问权限，这样就

可用更为系统性的方式来获取这些数据。例如，Uber 提供了一个 API，允许第三方软件获取预估车费或者用车历史(会对许可账户进行身份验证)。这就是我们的手机应用能够与 Uber 服务器通信的方式。当然，已经有人编写了一个 R 插件包来调用这个 API，这意味着我们可将 Uber 的数据纳入分析中，或者直接从 R 中发起打车请求(从理论上来说，这是可行的)。

注意：好的软件都具有如何与之交互的文档，这样用户和软件就能够清晰有效地进行通信。这类文档可能会描述可以发送到服务器的请求(以及预期的响应)，或者也可能仅描述如何使用一个函数(以及预期的返回值)。

1.1.3 数据再加工

数据再加工指的是数据的清洗和准备。我们所收集到的大部分数据都无法直接用于分析或展示。通常情况下，所输入的内容需要验证，所汇总的数据需要计算，值需要组合或移除，或者还需要执行重构。这是将数据用于科学处理时常常被忽视的方面，但这一点却至关重要。不恰当的数据处理会导致数据难以使用，并且更糟的是，可能会从中得出错误的结论。

数据再加工、数据整理、数据科学、数据分析、数据处理等术语都差不多是相同意思的不同名称，只不过根据数据的来源和去向的不同，它们具有不同的侧重点和不同的处理路径。大部分分析(无论是精心设计的复杂回归分析，还是简单的可视化呈现)都是从某种形式的数据再加工开始。通常这仅涉及将数据读入软件中，在这种情况下，其中一些处理是基于主观假设前提来执行的(将这些值作为日期处理、将那些值作为文字处理等)。当那些假设的前提失效或者希望以特定方式处理数据时，有能力控制所执行的处理方式就尤为必要了。

无论何时，当数据中具有分组记录时，不管是按年份、患者、动物类别、颜色、汽车品牌还是其他分类进行分组，我们都需要区别对待它们(以某种方式用颜色进行标记，仅包含具有同类典型特征事物的记录，计算出不同分组之间数量的变化程度)，要执行数据再加工，因为需要以某种方式将记录分配到一个特定分组。任何其他的转换、数据清洗或者处理也都可以算作数据再加工。我们很快就会清楚，如果想要结论值得信赖的话，那么对于任何分析来说，其中一大部分处理都会(或者都应该)涉及大量的数据再加工。

1.1.4 使用处理得当的数据可以做什么

到这里，我希望你弄明白的一点是，数据可能是极其重要的。它通常并不仅是表格中的数字。医疗数据通常代表着现实中人们的生命体征以及特定干预所带来的效果，要么是挽救了生命的正面效果，要么是造成了悲剧的负面效果。对于从某种特定

角度来审视这些效果的人而言，它们并非总是一目了然的，因此(专业的或者偶然为之的)数据分析师肩负着从数据中总结出规律以便进行决策的职责。

对于归纳非显而易见的规律而言，数据分析通常是有用的。例如，尽管我们可能识别出了下列这个序列的规律：

```
#> 2 4 6 8 10 12 14 16 18 20
```

即两个数字之间间隔为 2，但以下数据中的规律是什么可能就没那么明显。

```
#> 0.000 0.841 0.909 0.141 -0.757 -0.959 -0.279 0.657 0.989 0.412
```

只有在可视化这些数据(这是使用 `sin()` 函数生成的)之后，我们才能看出其规律，如图 1.1 所示。手头有合适的工具来分析数据意味着可以识别出隐藏的规律、预测新的信息和从数据中学习新的知识。

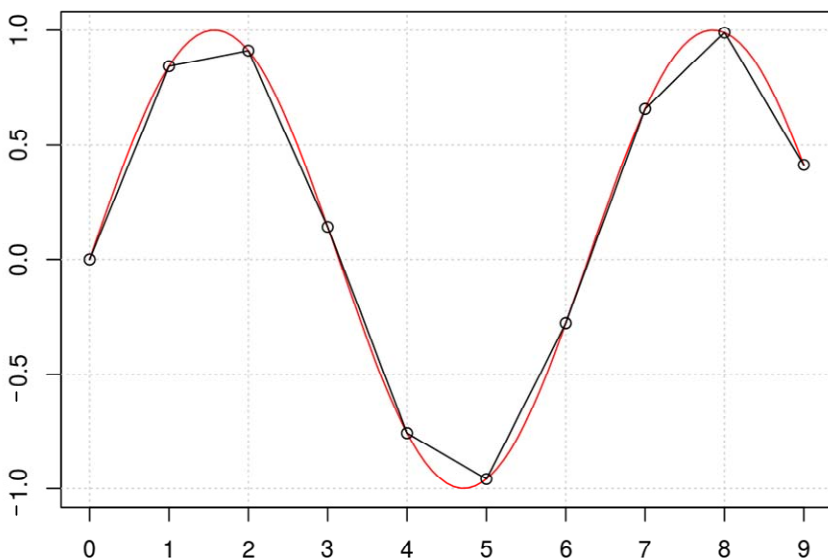


图 1.1 规律出现了。这些点是由 `sin()` 函数在值 0、1、……、9 处生成的。这里也绘制了该平滑的 `sin()` 函数

数据分析的一个经典示例是 John Snow 对于 1854 年在伦敦布罗德大街(Broad Street)所爆发霍乱疫情的分析。在那个年代，排污系统基础设施几乎不存在，并且人们对于传染病的认知相当有限，所以当时在这个特定区域内数以百计的人由于染上霍乱而死去。通过仔细检查霍乱病人的居住位置，John Snow 能够推断出，这些病人之间的共同联系似乎在于其最近的水源是布罗德大街的某个抽水泵。在禁用该抽水泵之后，霍乱病人的数量就显著减少了。在这个示例中，数据是显而易见的——霍乱病人的居住位置——但其规律和联系并不那么明显(参见图 1.2)。

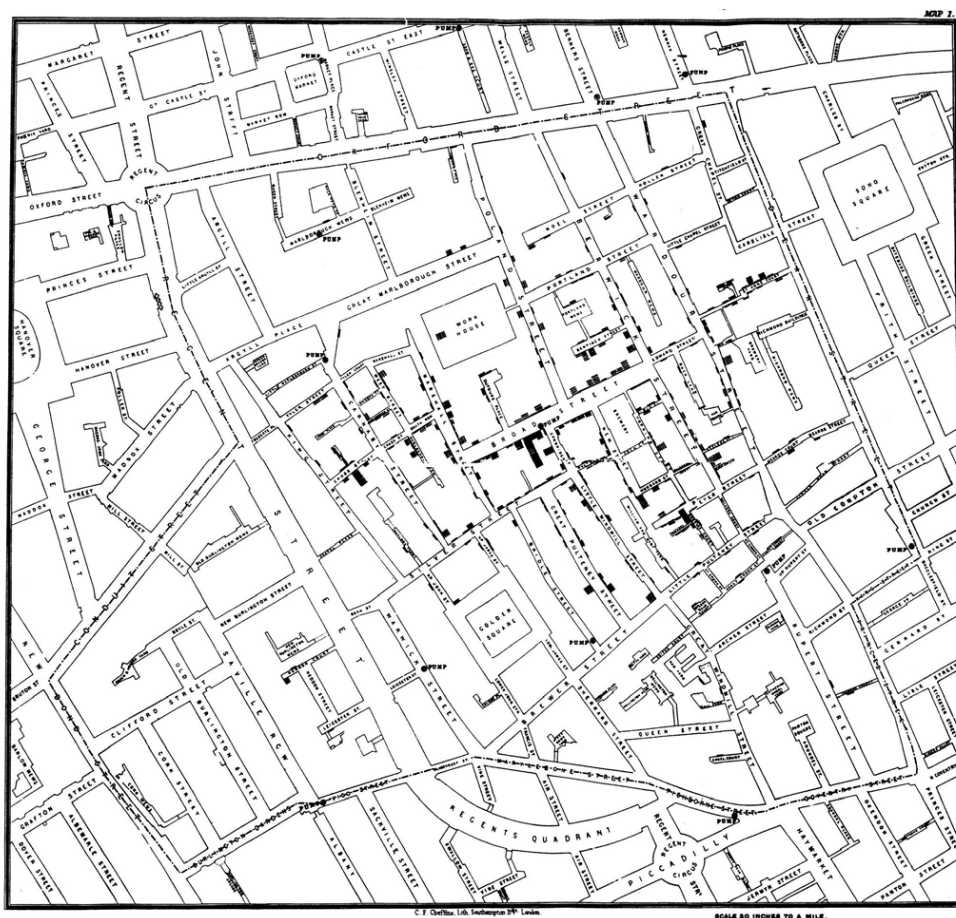


图 1.2 布罗德大街霍乱传染图，由 John Snow 绘制(公共资源)，来自维基共享资源网(Wikimedia Commons)。小点表示抽水泵的位置，并且用堆积条沿街标出了霍乱病例

可能并不令人意外的是，有一些 R 插件包可以与这一数据进行交互。可以在 HistData 包中找到该原始数据，也可以在 cholera 包中找到进一步的图表分析，如图 1.3 所示。

有时，像 Excel 或 Libre Office 这样的电子表格程序对于这一目的而言已经是完全够用的工具。我们可能会将一些表格数字放在一起查看、对其进行分类，也可能会将其绘制成柱状图，这些任务都可以在许多软件程序中轻易实现。不过，当我们希望以更加结构化、正规化、可重复且严谨的方式来与这些数据进行交互时，就要借助编程语言了。R 就是最好的选择。



图 1.3 使用 `HistData`(上图)和 `cholera`(下图)R 包生成的对于布罗德大街霍乱数据的进一步分析

1.1.5 数据就是资产

数据具有很大的影响力，因为它是我们要从中获得见解的信息。很少会出现无论如何都无法获取任何数据的情况(不只是指数字化)，但不同的数据承担着不同的职责。

许多人都依赖天气数据来规划他们每天的行程，例如渔夫可能要借此弄明白他们可以在相对海岸安全范围多远的距离打鱼，或者种植酿酒葡萄的葡萄园主要借此评估由于夜间结霜而损坏其作物的可能性。天气预报并不是最原始的数据源，而是通过整理更为原始的测量所产生的汇编摘要。

同样，财务分析师会提供对股票市场的评估以及对重大投资可能的每日变动情况的见解。这些也都是从模型中生成的，这些模型会摄取有关市场当前状态的高频测量值，并且提供更易于理解和操作的高级汇总摘要。

在这两个例子中，都存在我们要依赖的数据管理人：也就是那些以可预测且稳健方式让原始测量数据变得可用的人。如果这些数据源被损坏(无论是原始源还是处理过的源)，那么那些处于数据使用链中的人就无法对这些数据提供可靠的处理，并且后续可能还存在潜在危害。如果原始记录是手工输入电子表格中的，并且天气预报的创建人要记住需要按的按钮顺序、需要被复制到另一个表格中的单元格以及需要选择哪些行以纳入计算，那么从个人角度讲，我会对这份天气报告没什么信心。

希望你在阅读本书时能够理解我重点讲解这一事实的背后动机——我们都是数据链中的一部分，如果数据在我们手中，而我们却不关注它的话，那么后续的所有步骤都会遭受失败，并且对于那些查询我们数据的人来说，这些失败必然是不明显的。因此我们要寻求对能够获取的所有数据进行稳健的、可重复的且清晰的处理，然后再将处理后的数据发布出去。

尽管对于可重复研究的全面描述需要大量的资源才能阐释清楚，但目前下列指导意见已经足够：

- 记录下如何、何时以及从何处获取数据。
- 在数据处理期间要对所做出的任何决策提供注解。
- 原始数据源保持不变——在处理过程中所创建的任何内容都应该被记录下来，并且都应该是可重复的，理想情况下不需要人工介入。

提示：可重复的研究是信赖所得结果的关键，即使这些结果看起来不是很重要。也可能一年中只有我们自己查看这些结果，但知晓如何生成新数据就如同我们能从数据中获取的信息一样重要。

能够通过一个数据集所经历的变更往回追溯对于证明一次分析的正确性而言极有价值。我们可能最终会得到一份欧洲国家人均收入中位数的图表，但我们是否能够弄明白其刻度比例是如何计算出来的？是否过滤掉了海外的收入数据？这些数据是

样本还是人口普查数据？如果不清楚该分析中有哪些处理步骤，那么最终结果就会引发无法回答的问题。

至关重要的是，所执行的分析都是从正确的数据开始处理的，数据都是以恰当的方式来收集的，并且该分析能够解答我们的问题。当然，该问题也需要是正确的；否则我们就无法获得所期望了解的信息。

正确问题的大致答案通常是模糊不清的，但这也远远强于错误问题的准确答案，虽然总是能得出错误问题的精准答案。

——JOHN TUKEY，普林斯顿大学统计系的创办人

手上有了正确的数据和正确的问题，要如何持续跟踪所有一切呢？为此，我们不仅需要能够正确处理数据和代码，还需要正确应对其随时间而变化的情况。

1.1.6 可重复的研究和版本控制

你是否曾收到文件名类似于 `mydata_final_Thurs20May_phil_fixed_final_v2.xlsx` 这样的文件？这并非最简洁的名称，但它却揭示了一些非常糟糕的事情——该文件的多个副本在四处扩散，每个副本都具有不同版本的数据，其中大部分都过期了，因为已经进行了一些修正或更新，并且不同的版本之间有着未知的变化。如果某个人展示了从其中一个文件所生成的图表，我们能否确定它来自哪个版本？或者说，如果展示的是最新的图表以及在此之前的一张图表，我们能否看出其区别？

答案是，不要依赖文件名来存储版本化信息(文件名不适合用于此目的)。应该转而使用版本控制系统(VCS)，它可以持续跟踪变更，这样我们(以及协作者)就可做到：

- 总是能获取所有文件的最新版本。
- 能够查看版本之间的变化。
- 能够回滚到之前的任意版本。

其中部分能力需要极大地借助纯文本文件(如.txt、.R 以及.csv)的使用，因为版本控制系统可以逐字比较两个版本的行并且显示出有变化的内容。使用二进制文件(如.docx 和.pdf)则会让这一对比变得更困难，但也并非是无用的。

- 纯文本文件——这是将其内容存储为数字、字母以及标点符号的文件，因而在文本编辑器中打开该文件。纯文本文件中的信息可以被读入任何系统中，并且因为它没有格式化，所以对于每个符号所代表的内容或者如何读取该文件而言是没有歧义的。这并不妨碍格式化的存储，不过该格式化也需要是纯文本的，例如使用像**bold text**这样的标签来包围值的标记语言或者使用像**bold**这样的内联修饰符的 Markdown 语言。
- 二进制文件——这是将其内容以二进制形式(0 和 1)存储的文件，该文件只能被合适的软件所解析。无法在文本编辑器中读取它，但其优势在于，它能对

数据格式化进行编码,包括各种不同的格式,如音频、图片或视频。

这里不会讲解具体的 VCS 选择,但我们应该找到适合于自己的一款 VCS。一些流行的 VCS 选择包括下面这些:

- Git(使用 GitHub/GitLab/Bitbucket)
- Subversion(也称为 SVN)
- Mercurial

其中每一个都具有其自己的学习曲线,但在我们首次需要撤销一大批变更或删除操作时会发现其价值。

版本控制的另一个巨大好处就是,我们可以公开共享(如果希望的话)描述我们对数据进行了何种处理的代码,这样一来,感兴趣的人(可能是另一个数据分析师,也可能是 6 个月后的自己)就可以重复我们之前的处理,因为他们有了输入及分析步骤。

你是否曾经在完成了对于一些数据的处理后开始担心可能没有保存文件,因而可能必须再次对所有步骤进行处理(如果还记得那些步骤)?如果在完成这些处理步骤之后忘记了所执行的步骤,那么我们如何才能确信正确执行了它们呢?其他人又怎么能相信我们的处理步骤呢?通过使用命令脚本(可以将其视为我们要让计算机所执行的大量准确处理),我们就能保留分析步骤的记录,其他人也就能在使用相同数据的情况下像我们那样得到相同的结论。

数据更新很常见,并且在发生更新时,很容易就能看出遵循可重复研究方法和未遵循可重复研究方法的人之间的区别。在经历数周的数据处理和数值计算之后,有人可能会注意到第三个数据集的第 12 列中有一个拼写错误,并且发布一份更新后的文件: `data3_fixedTypo.csv`。

可重复研究的好处有很多。未遵循可重复研究的人会执行以下处理:

- 删除所有的输出(或者将其保存在其他地方)。
- 打开新的数据文件。
- 尽可能按照其能记住的所有分析步骤来执行。
- 忘记了第 4 列需要特殊处理。
- 不清楚最终结果与预期相比有很大差别。

遵循可重复研究的人会执行以下处理:

- 在其脚本中修改输入的数据文件名。
- 重新运行包含所有必需步骤及其文档的分析脚本。
- 明白此次唯一发生变化的就是输入数据更新了。

有许多 R 插件包有助于我们在可重复的研究框架中进行处理,稍后将介绍其中一些较为常用的 R 插件包。

数据准备好了,需要解答的问题也准备好了,并且我们打算通过版本控制来专注

于可重复的研究，那么此时唯一需要的东西就是将所有这些结合起来以便生成一些结果的方法：R 编程语言。

1.2 R 语言介绍

R 是一种统计学编程语言，它适用于执行统计计算，在经历了社区发展之后，其意义已远不止于此。作为一种通用语言，R 足够灵活，它几乎可以处理我们所要交互的任何数据：存储数据或流式数据、图片、文本或者数值。

就像大多数编程语言一样，它具有特定的语法(编码方式)，一开始可能会让人觉得困惑或奇怪，不过要相信，很快你就能适应。无论你是否相信，R 的确是更易读的语言之一。

无论是专业用途还是休闲用途，R 的用户量都在快速增长¹。只要有数据的地方，就很可能会有人在使用 R 处理这些数据。R 流行程度的一个很好的指标就是 RStudio(我们用来与 R 交互的软件)的专业用户列表，图 1.4 中显示了其中一些公司的标志。



图 1.4 R 的专业用户(来自 rstudio.com)

¹ 截至 2017 年，R 在 IEEE Spectrum 的前十位编程语言中排名第六(<http://mng.bz/z5sN>)，并且在 2017 年受欢迎编程语言的 TIOBE 索引中排名第八(www.tiobe.com/tiobe-index/)。

其他许多公司都将 R 用作其数据处理能力的一部分。其中一些知名的专业用户及其特定用途包括以下这些¹:

- Genentech——将 R 用于数据再加工和可视化,并且与 R 核心开发人员保持联系。
- Facebook——将 R 用于探究型数据分析和实验分析。
- Twitter——将 R 用于数据可视化和语义聚类。
- 芝加哥市——使用 R 来构建食物中毒监控系统。
- 纽约时报——将 R 用于交互式特性(如 Dialect Quiz 和 Election Forecast)和数据可视化。
- Microsoft——将 R 用于 Xbox 配对系统。
- John Deere——将 R 用于统计学分析(预测农作物产量以及农业设备的长期需求)。
- ANZ Bank——将 R 用于信用风险分析。

R 被广泛用于基因遗传学、渔业学、心理学、统计学以及语言学等学术研究。业余使用者已经发现了大量值得去做的有意义的事情,例如解决数独谜题(<https://dirk.shinyapps.io/sudoku-solver>)和迷宫问题(<https://github.com/Vessy/Rmaze>)、下国际象棋(<http://jkunst.com/rchess>)以及连接到 Uber 这样的在线服务(<https://github.com/DataWookie/ubeR>)。

本节将介绍 R 的运转机制以及与其交互的方式。就像使用其他任何新工具一样,首先要正确理解 R 提供了哪些特性,这样才能为我们的学习之路节省大量时间。为了完全弄清 R 的一些古怪技巧,我们需要回到起点。

1.2.1 R 的起源

R 的前身是 S 编程语言(用于统计学),它是由贝尔实验室的 John Chambers 及其同事所开发的。S 语言是在 1993 年通过 S-PLUS 这一独占性许可而商用化的,该许可被广泛用于各种学科中。当 R 被视为 S 语言的一种开源实现时,其社区活跃度有了显著增长,这意味着用户每天都能看到其底层结构并基于此结构进行构建。不过,这门新的语言还是向下兼容 S 的,并且 R 所保留的大部分古怪之处都可以归因于此。

2000 年 2 月,新西兰奥克兰大学的 Ross Ihaka 和 Robert Gentleman 发布了 R 的第一个稳定版本。自那以后,R 的基础研发一直是由一组志愿者(R 核心开发人员)负责,并且是基于公众所提交的方案建议来进行。外部开发的扩展插件包也在一直不断出现,它们都正式托管在 Comprehensive R Archive Network(CRAN, <https://cran.r-project.org>)上,到 2017 年底大约有 12 000 个这样的扩展插件包;还有许多扩展插件包非正式地托管在像 GitHub 这样的代码共享站点上。

¹ 参见 Data Science Central 的 Deepanshu Bhalla 所发表的“List of Companies Using R”一文, <http://mng.bz/qJ66>。

1.2.2 R 能够以及不能完成哪些工作

编程语言之间的分类有很多并且很大程度上都会令人费解。这些分类也一直存在争议，因为其定义很复杂并且需要拥有计算机科学的学位才能充分理解其内涵。尽管 R(也就是 S)最初是为统计学而构建的，但它仍然可以被视为一种通用语言(General Purpose Language, GPL)，因为它并非被限定于仅完成单一任务。

有些语言的存在是为了完成某个领域(一个特定的相关领域，如财务、技术绘图或者机器控制)中的任务，这些语言被称为特定领域语言(Domain-Specific Language, DSL)。R 比这些语言更灵活，因为我们可以编写代码以让其完成我们所需的任何目标。

R 不是一种 DSL。它是编写 DSL 的语言，是一种更为强大的语言。

某个人可能想着要实现一个财务数据目标，另一个人可能对于自然语言处理感兴趣，而第三个人的目标可能是预测顾客接下来会做何种决策。所有这些的共通之处就是数据，但 R 是如此的灵活，它提供了一种完善的机制来应对所有这些领域。

——JOE CHENG, RStudio 的 CTO

我并非打算向你推销 R；我认为 R 是一种很棒的语言，它让许多任务都变得更为简单，并且其处理任务的方式很优雅。不过你也要明白，它并非解决我们特定问题的唯一方案，它甚至可能并非最佳方案。但是通过学习一门新语言，我们就不会试图为解决问题而硬着头皮提供一个解决方案；相反，我们对语言运行机制了解得越多，就越有助于我们更好地识别出问题的可能解决方案(即使这意味着另一种语言才是更适用的)。编程语言的对比就像是询问苹果或桔子哪种更好——根据惯例，这要视情况而定，或者可能根本就不是这样的。我们需要结合使用。

1. R 能够完成哪些工作

在最基本的层面上，R 是与数据交互的一款有用工具。它将值(数据)和函数(与数据交互的代码)存储为变量(事物名称)和复杂对象(结构)。通过技术术语来表达的话，我们可以说 R 是一种开源的、解释型的、通用的函数式语言。

- 开源——可以免费获取和(如果需要)修改底层源代码。
- 解释型——R 不需要将代码编译为独立程序。有些语言需要将代码内嵌到可执行程序中以便运行。
- 通用——它并非受限于仅处理特定领域中的一件事情。
- 函数式——它使用函数来操作固定不变的数据，而不是依赖系统的当前状态就地修改数据。

R 可被视为工具袋。如果清楚挂在哪里，则可以将更多工具添加到 R 中。我们可以重新整理这些工具以便让其对用户更为友好，并且我们可以根据需要使用少数几个或者多个工具。这样的工具就是插件包，也就是文档化函数(对数据执行操作的代码)

的逻辑分组，可以调用它们来生成一些输出——一个图形、更多的数据、处理信号、网站请求等。

如果没有这些插件包(这里指的是基本包和默认安装的插件包)，那么 R 就只是能力有限的一种框架而已。其真正的强大之处就是基于这一框架所构建的附加插件包，用了它们，才能创建强大的统计函数和发布质量的图形，同时还可以按需对这些附加插件包进行扩展和修改。

2. R 不能完成哪些工作

有了好用的工具袋并不能确保我们获知如何挥动一把榔头，也不能确保我们知晓十字螺丝和梅花螺丝之间的区别。此外，安装了 R 也并不意味着所有的数据分析过程突然之间就变得清晰。

R 几乎可以让我们对数据进行任何处理，它可能是明智的选择，也可能是完全不合理的选择。在某些情况下，它会警告我们正在进行一些可能不希望进行的处理。有时，它会默默地生成垃圾信息并且继续进行下一步处理，就像一切正常一样。这不完全是 R 的过错——许多人都相信，一种好的编程语言应该“按照我们说的做，而非按照我们的意思做”，并且应该让用户来决定什么是正确的以及什么是错误的。

由于 R 的运行方式(稍后将进行介绍)，它并非总是处理数据的最快方法，不过它也不会慢。根据用途的不同，处理速度的快慢可能完全不是一个问题。有时使用 R 的处理时长会比使用其他一些语言多几分钟，但使用 R 的好处在于，R 代码可能更可用。许多 R 插件包都利用了 R 与其他语言交互的能力，并且在使用 R 语言进行处理以及使用另一种更高效语言(如 C)进行处理之间取得一种平衡。

1.3 R 的运行机制

有些编程语言会将代码编译(构建)为一种可执行程序。那样做有优点和缺点，不过这并非 R 的工作方式。相反，R 是一种解释型语言，对于这类语言，计算机一次只处理一个指令(这些指令的序列就是脚本)，而每个指令的处理结果会呈现(返回)给用户。

为了以这种方式进行操作，R 实现了一种读取、估算、打印、循环(REPL)机制，这种机制所做的就是其字面上的处理。图 1.5 中显示了这一流程图。R 会耐心等待我们的输入，在输入完成后，所输入的内容会被读入系统之中并且被估算(执行计算)，而结果会被打印到控制台(如果有)，之后整个处理过程会循环返回等待更多的输入。

对于刚才所说的在执行任何处理之前需要等待输入的语言而言，这可能像是需要具备很多能力一样。之所以如此，是因为 R 程序(启动 R 的 `R.exe` 或者 R 可执行程序)主要是用 C 编写的，而 C 是一种编译语言(而且是一种非常节约内存的语言)。按下

Enter 键会触发 C 代码执行 REPL 操作。

作为一种开源语言，底层运行的源代码可让任何人进行检验。<https://svn.r-project.org/R/branches> 处提供了自 R 0.60.1 开始的所有版本的官方源代码，这意味着如果需要，我们可以查看过去数年中 R 的各个组件发生了哪些变化。这对于专有(闭源)程序而言是不可能的，其内部运转机制仅对那些基于它进行处理的人才可见。对于开源软件，我们甚至可以下载其整个源代码，对其进行修改，并且编译我们自己的私人版本¹。

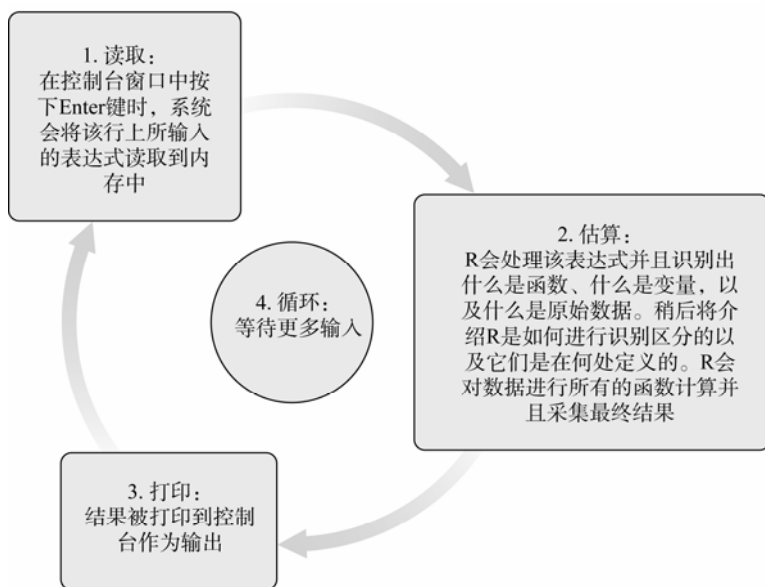


图 1.5 读取、估算、打印、循环

<https://github.com/wch/r-source> 处还有一个更容易获取的只读镜像，它是由 Winston Chang 托管的。它每小时都会与官方的源同步一次。

如果还没有安装 R 的话，请在计算机上安装它。可以参考附录 A 中的说明。

一开始你可能是让 R 每次运行一个命令，但渐渐地就会希望能够告知 R 按顺序执行多个命令。这就需要用到脚本了，这很容易让人联想到演员要说的剧本中的多行台词。R 脚本(通常是以 .R 或 .r 结尾的文件)不过是一系列命令而已，通常每行一个命令，但一个命令也可以被划分成多行，这些命令会被 R 系统按顺序读取并且处理。这不同于电子表格文件的处理方式，电子表格中只会保留当前状态的数据，而不会保留数据的处理过程。脚本中开头的几行会指示 R 如何做好对后续分析的处理准备，之后是如

¹ R 代码的开源许可是 GPLv2，这意味着我们可以对其进行任何处理，只要保留对其进行过处理的人的归属信息，并且不对其进行销售以盈利或者限制其获取权限。

何获取/读取原始数据，然后是如何处理这些数据，最后是如何保存结果以及在何处保存。基于这一工作流，分析就可以重复进行，因为备有原始数据和处理步骤，所以可以重复得到结果。

单用 R 就足以处理一个分析脚本了，可以使用命令行将该脚本传递到 R 处理器。在 Windows 系统上，根据确切版本和安装路径的不同¹，在脚本文件的同一目录中启动命令行，可以使用以下命令：

```
C:\Program Files\R\R-3.4.3\bin\R CMD BATCH yourScriptFile.R
```

在 Linux 或者 Mac 系统上²，可以在命令行中输入以下命令：

```
R -f yourScriptFile.R
```

R 将启动并且处理该脚本文件的内容。

如果是交互式使用 R(也就是手动启动 R，然后 R 会生成一个即时的等待输入指示)，可以使用 `source()` 函数来达到相同的效果：

```
source(file = "~/yourScriptFile.R")
```

目录名中的波浪号(~)是用户主目录的通用占位符。脚本文件可以放在任意文件夹中；我们只需要告知 R 在何处找到脚本文件即可。

尽管你可能很熟悉 Excel 中的按钮菜单，但 R 是一种基于命令的语言。这意味着我们要使用表达式(也就是对数据执行操作的代码片段)来告知 R 要做什么处理并且存储结果。我们花些时间来看看这样的方式是什么样的以及我们将遇到的一些名称，参见图 1.6。不同类型的对象可能会着色突出(语法高亮显示)，这有助于区分代码的不同部分。

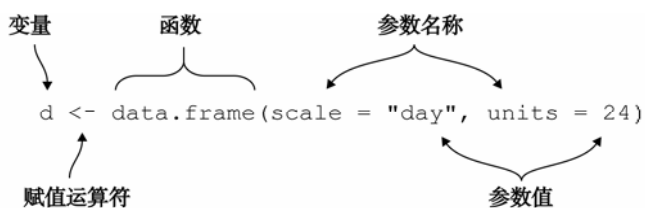


图 1.6 标识出一些术语的 R 代码：变量、赋值运算符(<-)、函数以及参数

图 1.6 中使用的一些术语对于你来说可能是新的概念：

- 变量——指代一段数据的名称。
- 赋值运算符——将值存在变量中的函数。
- 函数——一些与数据交互的代码，使用正括号“(”和反括号“)”来调用它，可能还要使用参数。

1 除非设置 \$PATH 环境变量以搜索这个目录。

2 假设安装目录位于 \$PATH 中。

- 参数——传递给函数的选项，由逗号分隔，可能是通过等号(=)连接的参数名和参数值对(例如 `save = TRUE`)。

以这样的方式使用 R(从所保存的文件中读取命令)自然是可行的，不过若真的要在使用 R 的过程中能够随时得到帮助的话，就需要借助一款对 R 系统进行了封装的附加软件，并且该软件还提供了额外的功能，这款软件就是 RStudio。

1.4 RStudio 介绍

数据是存储在计算机上的(或者存储在计算机可以连接的某个设备或驱动器上)，但是与数据交互需要一些软件来读取数据，解释我们想要对数据进行哪些处理，并且将数据写入某种输出或存储(要么存储为值、图片、声音，要么存储为其他完全不同的形式)。

这些软件可能是多种多样的。

- 使用像 Notepad 或 emacs 这样的文本编辑器来查看原始的、本地存储的数据。
- 在诸如 Excel、Access 或 Google Sheets 的电子表格或者数据库程序中显示格式化数据。
- 使用诸如 Google Chrome 或 Internet Explorer 的浏览器查看通过互联网传输的未编码或转译过的 JSON 数据。
- 使用编程软件来检索和操作数据。

所有这些软件在显示数据和与之交互方面都具有不同的能力。在使用 R 与数据交互时，有一款高度复杂且功能强大的交互式开发环境(IDE)集成了所有这些能力，也就是 RStudio。通过使用 RStudio，我们能够以多种形式查看数据、与之交互、对其进行操作，然后存储数据或者分发数据。这个 IDE 的特色是，提供了一个能感知 R 的文本编辑器来读取/编写脚本，还提供了一个控制台来输入 R 命令。其最出色的一点是，提供了一种方法来检测工作区以及所有已定义变量的当前状态。

如果还没有安装 RStudio 的话，请在计算机上安装它。可以参考附录 A 中的说明。

1.4.1 在 RStudio 中使用 R

RStudio 将其窗口划分成独立的窗口或区域(参见图 1.7¹)。其中的边框可以拖拽以便放大或收缩单个窗口，并且可以根据喜好来重新调整，只要在菜单中依次单击 Tools | Global Options | Pane Layout。有些窗口还可以脱离主窗口变成全屏模式。

下面是默认显示的四个窗口：

- 编辑器——这是编写脚本的位置。脚本就是要按顺序执行的一系列命令。在

1 修改自通过维基共享资源网检索到的 PAC2(www.gnu.org/licenses/agpl.html)的原始截图。

首次打开 RStudio 时, 编辑器会是空白的。依次单击 File | New File | R Script 来开启一个新的文件/脚本。

- 控制台——类似于在终端中出现的 R 提示符。这是逐行输入命令然后按下 Enter 键的地方。R 返回的结果也会呈现在这里。
- 工作区——R 获知的值(我们所定义的数据或者变量)会出现在 Enviroment 选项卡中, 而所执行的命令历史将出现在 History 选项卡中。
- 帮助与绘图——根据所选选项卡的不同, 帮助或绘图区域将显示函数或数据集的文档, 或者最近生成的绘图。该区域也包含 Packages 和 Files 选项卡, 以分别列出所安装的插件包和计算机上的文件。

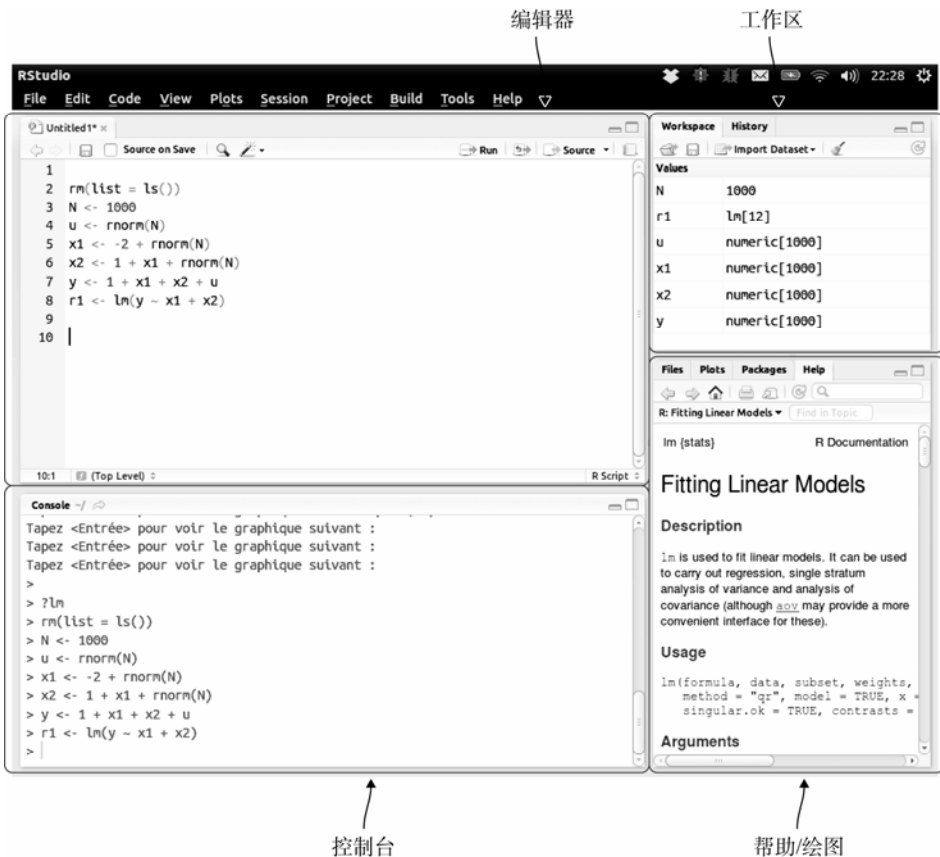


图 1.7 出现在 Ubuntu/Linux 中的 RStudio 窗口

有简单的方法可以在这些窗口之间进行切换。Ctrl+1 会将鼠标指针移动到编辑器窗口以便编写脚本。Ctrl+2 会将鼠标指针移动到控制台窗口以便输入交互式命令。当鼠标指针处于一个函数上时, 按下 F1 键会打开该函数的帮助菜单。还有其他许多键盘快捷组合键可用。试试用 Alt/Option+Shift+K 来打开一个详尽的备忘单。

RStudio 的替代选项

当然, RStudio 并非使用 R 的唯一途径, 不过我的确发现它是最便利的。如果你更适应使用命令行界面(并且可以放弃 RStudio 所提供的附加好处), 那么也可以在终端中很好地使用 R。还可以使用 Emacs Speaks Statistics(ESS) emacs 包将 R 挂载到 emacs 中。在 Windows 上首次安装 R 时, 我们还会发现安装了 RGui, 这是一个简单的图形化界面。

有几个可选的 R 的图形化界面也是被广泛使用的, 如 R Commander 和 Deducer。为了保持一致性(而且因为我坚信 RStudio 更胜一筹), 本书的后续内容会假设你都在使用 RStudio。

RStudio 对于使用 R 来说提供了很多有帮助的便利特性, 不过我们也可以在一个独立终端中完成需要的所有处理。在终端中与 R 进行文字交互, 其结果会与 RStudio 控制台窗口中出现的结果相同, 现在我们重点介绍 RStudio 的方式。

RStudio 直接支持与 Git 和 SVN 的协同使用。我建议你直接在 <http://mng.bz/1s4F> 处阅读 RStudio 关于这方面的内容。

每次用 RStudio 或者独立方式启动 R 会话时(运行 R 程序并且使用 R 语言), 工作区一开始都是空的¹。如果设置的选项被启用(默认是启用的), 那么上次使用 RStudio 打开的文件仍然会出现在编辑器窗口中。工作区窗口不会列示任何对象, 并且控制台将显示以下欢迎消息:

```

版本会列示在此处(以及版本的发布日期)。尽可能尝试使版本
更新到最新, 因为会修复问题和定期优化。注意, 这可能要求
我们同时更新包库
    R version 3.4.3 (2017-11-30) -- "Kite-Eating Tree"
    Copyright (C) 2017 The R Foundation for Statistical Computing
    Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

    Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
  
```

这与所运行的系统是 Windows、Linux 或是 Mac 有关

一分钱一分货。尽管很多人已经为确保 R 如预期般运行付出了很大的精力, 但并没有这样的官方保障承诺

通常会被忽视, 但这些都是很好的提示

命令提示符>表明 R 准备好了并且在等待输入

¹ 除非有意将选项设置为从上次离开处开始, 这是通过加载工作区镜像来实现的。

注意：自 2011 年底开始，R 的版本就不仅是通过一个版本号来指定了，还增加了一个古怪的名称，第一个这样的名称是 Great Pumpkin。这些名称完全是一位核心开发人员突发奇想的念头，尽管没有严格的结构，但它们通常都是以季节性因素作为主题的，并且全都与 Charles M. Schulz 的 *Peanuts* 有关。例如，R 3.4.3(本书使用的版本)的昵称是 Kite-Eating Tree。

当出现提示符(>)时，就表明 R 在等待下一个命令。可以直接在控制台中输入命令(更适用于使用过一次的短命令)并且按下 Enter 键执行它。也可以在编辑器中将命令作为脚本来构建(更适用于较长的分析并且更利于保存步骤)；当鼠标指针处于相关行上时，可以按下 Ctrl+Enter(在 Mac 上则是 Cmd+Enter)组合键来一次执行一个命令，或者使用一个高亮区域来一次执行多个命令。

提示：如果需要在单行中输入多个命令，则可以使用分号(;)来分隔这些命令，如 `a <- 2; b <- 3`。如果开始编写一个长命令并且只输入了一部分——如只输入了正括号“(”却还没输入反括号“)”，或者只输入了一半计算公式(如 `2 +`)——然后按下 Enter 键，那么 R 将认为我们打算在命令中插入一个行分隔符并且将使用 `+` 来替换提示符，以表示它在等待命令的其余部分(对于剩余的部分，我们可以正常输入)。在执行这些命令行之前，R 会将所有的输入行连接在一起。在编辑器窗口中，可以用相同方式来分隔命令(分成多行)，并且将其作为整体或部分来执行。如果在这个模式中卡住了，请不要担心。再次按下 Enter 键只是会插入更多的行分隔符而已。要退出这个模式(或者在任何时候要取消命令输入)，可以按下 Esc 键来清除当前输入并且返回到命令提示符(>)。

如果正在执行复杂计算，那么 R 会通过不显示提示符来表明它处于忙碌状态，直到准备好执行其他命令为止。当 R 再次可用时，它将处理控制台中输入的任何命令，不过我们不应仅依赖当前所运行计算的成功执行。在由于长期运行计算而造成 R 繁忙时，RStudio 还会在控制台上方显示一个小的停止标志，类似于图 1.8 所示的那个。

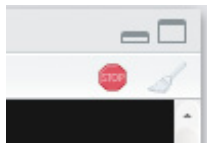


图 1.8 在 R 运行时，可以在控制台上方找到这个符号

每次启动 RStudio(或者从命令行启动 R)时，会话就开始了。在会话中，加载到内存(工作区)的数据(和函数)可被查询和修改。如果你习惯于基于电子表格的环境，其中数据是已经准备好的，只要打开备份的程序就能看见，那在这种情况下要切换到 RStudio，会经历很不一样的思维模式的转换。RStudio 要保存原始数据和步骤，以便生成输出，这意味着我们的工作是可以重复的。只有保存下来的信息才是持久存在的。

注意：已经加载的插件包和数据仅在某个会话中可用，所以如果手动将一个插件包加载到一个会话中(如使用 `library()` 函数，第 4 章将全面介绍该函数)、读入一些数据或者创建任何东西，那么需要在想要使用这个插件包的其他任何会话中重复那些步骤。通过保存脚本(所输入的命令记录)，我们就能轻易地重启会话并且回到曾经的处理位置。脚本仅是保存到硬盘的纯文本命令而已，就像其他任何文件那样独立于 R 会话之外存在。

我们想要 R 在启动会话时运行的命令应该位于主目录¹中一个名为 `.Rprofile` 的文件中。这些命令可能是调用我们总是打算使用的一些插件包上的 `library()`、打印一条消息或者执行一些常见任务。

如果尝试退出 R(使用 `q()` 函数或者关闭 RStudio 窗口)，那么很可能会提示要保存工作区镜像。这使得我们可以保留会话中工作区内定义的(指定的，下一章将会介绍)所有内容，以便后续重新加载以及恢复到上次离开处。这可能是也可能不是我们希望进行的处理，具体取决于我们正在做什么，不过如果我们的确要在不进行保存的情况下重启 R 的话，我们应该尝试以这样一种能够生成相同答案的方式来编写代码。那有助于确保我们的工作流是可重复的。

如果觉得这一设置烦人，则可以通过单击 **Tools | Global Options | General | Save Workspace to .RData on Exit** 来修改这个设置。我建议你将此选项设置为默认不保存，这样我们就能总是开启全新的会话。

当我们希望或者需要开启不带任何已定义对象以及不加载任何额外插件包的全新会话时，可以关闭 R 或 RStudio 并且重新打开它。但是如果是使用 RStudio，那么一个更快捷的选项就是按下 **Ctrl+Shift+F10** 组合键，它会指示 RStudio 重启会话。在某些版本中，这一快捷键还会移除环境中所有已定义的对象。如果不是这样，则可以单击扫帚图标来清除工作区，如图 1.9 所示。



图 1.9 人工清理

比较好的做法是，定期清理对象以便确保脚本是可重复的——当对象已经存在时，我们就能轻而易举地打乱几行命令并且确保仍旧得到预期的结果，但是这会破坏脚本的顺序性，并且如果在定义一个对象前就使用该对象的话，就会出现問題。

提示：在知晓如何使用 R 之后，我们很可能会处理几件不同的事情——有时要同

¹ 这一位置取决于我们的操作系统，不过可以从 R 本身处使用 `Sys.getenv('HOME')` 来定位。

时处理多件事情。RStudio 使得对项目的持续跟踪变得简单。在创建一个 RStudio 项目时，RStudio 会持续跟踪已经打开了哪些文件、它们位于何处，甚至会跟踪专用于该项目的不同窗口的定位。每个项目都有其自己的 R 会话，因此可以独立地为不同上下文分别运行几个会话，而无须担心我们的财务分析会干扰到地图绘制的可视化。

我强烈建议你新的项目中开启每一项具有不同上下文的工作。可以在 RStudio 右上角的菜单中选择项目，如图 1.10 所示。

可以用 File | Project 创建一个新项目，或者用 File | Open Project 打开一个现有项目。

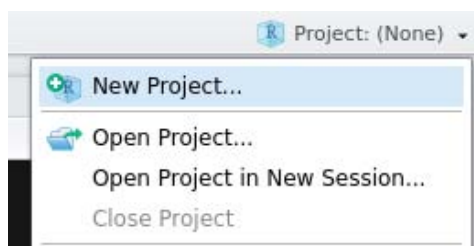


图 1.10 从菜单中选择项目

当意外创建了不合理的代码时，RStudio 会尝试进行提醒，可能是因为这些代码的顺序错误，但是如果变量是在会话的当前状态中定义的，则可能不会有这样的提醒。只有当我们可以会话中执行处理时，会话才会有用，因此我们继续讲解可以告知 R 对数据进行哪些处理。

1.4.2 内置插件包(数据和函数)

R 自带了各种预先安装好的插件包，这些插件包被视为 R 的 base 版本。它还捆绑了一系列有用的插件包，这些插件包有助于执行我们需要进行的大部分处理：

```
#> [1] "base"      "compiler" "datasets"  "graphics"  "grDevices"
#> [6] "grid"      "methods"  "parallel"  "splines"   "stats"
#> [11] "stats4"    "tcltk"    "tools"     "utils"
```

后面我们将讲解与这些插件包有关的更多内容以及它们所包含的函数。对于这些基本包，可以通过将以下命令输入控制台中并且按下 Enter 键来查看函数列表：

```
help(package = "base")
```

该命令会按名称的字母序列出这些函数(R 3.4.3 中约有 450 个)。stats 包提供了约 300 个函数。我们当然不会用到所有这些函数，但应该清楚，即使我们不安装额外的插件包，R 的标准安装中也有大量可以利用的功能。我们不需要做任何特殊操作就能使用这些基本包所提供的函数；在启动 R 的时候，它们就准备好被使用了。

除了有用的函数之外，R 还提供了预先安装好的示例数据集，它们被用于阐释这些函数的使用方式。下面介绍其中一些最常用的：

- **mtcars**——Motor Trend 汽车道路测试。这些数据摘自 1974 年的 *Motor Trend* 杂志，其中包括 32 种车型(1973—1974 年的品牌和型号)油耗、汽车设计和性能等 10 个方面。这是我们很多时候都会在我们自己的示例中用到的数据集。这一数据集是有些过时，不过只有在经常看到它们的时候，我们才认为它们是陈词滥调，对于我们目前的情形而言则并非如此。如果需要快速查阅它而手头又没有 R 会话，那么可以参考本书前言处所提供的整个数据集的截图。
- **iris**——一个著名的数据集，它提供了以厘米为长度单位的测量指标，这些指标包含了各种不同的花萼长度和宽度以及花瓣长度和宽度，分别对应于 3 个鸢尾花品种的 50 朵花。这 3 个品种是山鸢尾花、变色鸢尾花和维吉尼亚鸢尾花。
- **USArrests**——美国各州的暴力犯罪率。这个数据集包含的统计数据是 1973 年美国 50 个州每 100 000 个居民中由于侵犯他人、谋杀等犯罪而被拘捕的人数。其中还提供了居住在市区的人口的百分比。

如果你已经阅读过使用 R 的其他指南，那么可能已经见过这些数据集了。我们也会在示例中使用这些数据集，不过不要害怕使用这些数据集和其他数据集来验证你的新技能。**datasets** 包的帮助菜单中列出了更多的示例数据集(3.4.3 版本中列出了 87 个)的名称并且对其进行了简要描述，只要在控制台中输入以下命令就能查看它们：

```
help(package = "datasets")
```

1.4.3 内置文档

在正确构建的时候，R 插件包、函数以及数据都会自带一些帮助性文档。在 R 会话中使用以下语法可查看这些文档：在问号后面加上要查询的插件包、函数或数据集的名称。例如，要了解与 **mean()** 函数有关的更多信息，可在控制台中输入以下命令：

```
?mean
```

如果?后面的名称是 R 能够感知的插件包、函数或数据集，那么其文档将出现在帮助窗口中——否则，控制台中将会出现一条像如下这样的错误消息：

```
?nonExistentFunction

#> No documentation for 'nonExistentFunction' in specified packages and
#> libraries: you could try '??nonExistentFunction'
```

要搜索文档中的某些文本(要在不知道文档名称的情况下查找这些文本)，可使用双问号。

```
??mean
```

但这样的搜索有一些局限性。一些更好的解决方案还处于开发过程中，如 DataCamp 的解决方案(www.rdocumentation.org)，不过目前大部分问题都可通过阅读手册(?文档)与/或网络搜索来解决。

RStudio 还有一点特性也有助于我们编写 R 代码，它提供了弹出窗口式的语法技巧提示——将鼠标放在函数名称上可查看模板和默认的参数/选项。RStudio 还提供了已知函数和参数的自动补全功能：在输入函数或数据值名称时暂停一下，可查看可能的自动补全，使用向上和向下箭头按钮来从中进行选择，并且按下 Tab 键看看可能的自动补全。当我们可能拼错函数或数据值的名称时，RStudio 也会进行提醒。这些选项都是通过 Tools | Options | Code 来配置的。

1.4.4 简介

R 的一大未被充分使用(但若正确使用却极有价值)的特性是对任何插件包创建详尽指南的能力，这被称为简介(vignette)。简介可以覆盖任何主题，但通常是为了突出插件包所提供的一些重要函数的使用方法，这类似于研究报告或教程。简介有可能长达几页，所以它们所能提供的有效信息远远大于帮助页面所提供的有限信息，因为帮助信息只是一份简短的使用指南而已。

这些简介是在安装插件包的时候构建的，如下所示。

```
# The plot3D package produces 3D plots.  
# Install it using  
install.packages(pkgs = "plot3D")
```

可在控制台中通过 `vignette()` 命令加上简介的主题(名称)来使用简介(可能还需要使用参数 `package` 来传递该简介的归属插件包)。

```
# Vignette: Fifty ways to draw a volcano using package plot3D  
vignette(topic = "volcano")
```

可使用下列命令查看当前所有可用的简介。

```
browseVignettes(all = TRUE)
```

这个命令会加载一个本地托管的网页，上面会显示指向每个可用简介的链接，并且是按照其所对应的插件包来分组显示的。它们会被生成为 PDF 文件、HTML(网页)文件、这些简介的源文件或原始的 R 源数据块。如果希望了解与所安装插件包有关的更多内容，那么简介就是值得查看的地方。

1.5 亲自尝试

如果还没有亲自尝试过的话，请打开 RStudio 并且熟悉一下这个程序。在要保

存我们工作内容的主目录中创建一个新的文件夹，打开一个新的 R 脚本，如图 1.11 所示。

将一些命令输入脚本中(尝试添加一些数值)，并且在对其进行计算时可以看到它们出现在控制台中。按下 **Ctrl+Enter**(或 **Cmd+Enter**)组合键或者单击脚本窗口正上方的 **Run** 按钮，如图 1.12 所示。

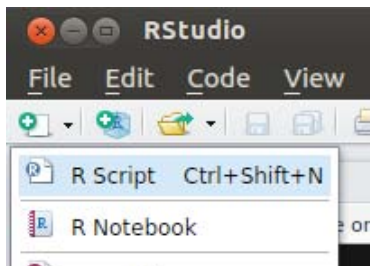


图 1.11 打开一个新脚本



图 1.12 Run 按钮

注意观察命令、结果、错误和警告是如何出现的，代码行是如何计算的，以及 RStudio 是如何显示不同的代码片段的。不要忘记定期保存脚本！

1.6 专业术语

- 脚本——保存在一个文件(通常以.R 或.r 结尾)中的一系列命令。
- 值——一段数据。
- 变量——指向一段数据的名称。
- 函数——与数据交互的一些代码。
- 对象——定义一个变量或函数的复杂(或简单)结构。
- 插件包——可被安装和使用的函数(也可能是数据)集合，它扩展了 R 的功能。
- 简介——使用插件包及其函数的详尽指南，它是与所安装的插件包存储在一起的。

1.7 本章小结

- 数据无处不在。
- 数据处理往往伴随着责任。
- 正确地结构化数据会让数据更具访问性。
- 不同的数据类型需要不同的处理方式。
- 数据可视化可以揭示隐藏的规律。

- 可重复研究意味着结果值得信赖。
- R 被用于许多领域并且历史悠久。
- R 会解释我们的命令，而非将其编译成可执行程序。
- RStudio 会让 R 代码的使用和处理变得更加顺畅。
- 许多插件包都对 R 进行了扩展。
- 可重复研究意味着创建一套可重复的工作体系，而这要通过保存获得结果所需的原始数据和处理步骤来实现，而不是仅存储该结果。
- 每一个运行中的 R 实例都代表着一个会话。已指定变量和已加载插件包/函数仅在一个会话中可用，并且必须在启动新会话时重建它们。
- 控制台中所显示的提示符(>)意味着 R 准备好接收命令了。
- 可在脚本中编写命令或者在控制台中每次运行一个命令。
- 要退出会话，可通过关闭按钮来关闭 RStudio 或者输入 `q()` 命令。
- 要查找与一个函数有关的更多信息，可以在问号后面加上该函数的名称，例如 `?mean`。

第 2 章

了解 R 数据类型

本章涵盖：

- 我们可能会遇到的数据类型
- R 如何使用类型来存储数据
- 如何给变量赋值

在所有的编程语言中，数据都是由计算机使用二进制值存储在内存中的。二进制值只有两种可能的状态，我们可以将这两种状态视为 TRUE 和 FALSE，或者更通俗地讲，就是 1 和 0。在计算机内存中，这两种状态代表电荷的存在或缺失，也就是说它们代表着 ON 和 OFF。

我们可能会在屏幕上看到 42，不过这要归功于计算机将它所存储的内容转换成我们期望看到的内容。这是极其有用的(总好过看到 00000000 00000000 00000000 00101010)，但也有些风险，后文将介绍这一点。

将一段数据存储到计算机内存中的方式有很多。通过指示计算机将数据作为特定类型来处理，我们就能变更对数据进行操作的方式。