大数据、云计算、人工智能、信息安全人才培养丛书
“互联网+”新形态一体化精品教材

R语言程序设计

R YUYAN CHENGXU SHEJI

主编◎冯青文　张　闯　姜俊颖

扫一扫
学习资源库

◆微课视频
◆教学课件
◆电子教案

上海交通大学出版社
SHANGHAI JIAO TONG UNIVERSITY PRESS

大数据、云计算、人工智能、信息安全人才培养丛书

“互联网+”新形态一体化精品教材

R语言程序设计

R YUYAN CHENGXU SHEJI

主　编◎冯青文　张　闯　姜俊颖

扫一扫
学习资源库

◆微课视频
◆教学课件
◆电子教案

上海交通大学出版社
SHANGHAI JIAO TONG UNIVERSITY PRESS

内容提要

本书通过大量的实例讲述 R 语言程序设计需要掌握的各方面知识。全书共 14 章，内容包括初识 R 语言，R 语言基础，单模式数据结构，多模式数据结构，基本数据管理，高级数据管理，日期、时间和因子，程序逻辑结构，R 语言函数，R 语言读取数据，类与对象，图形初阶，ggplot2 包，综合案例。本书语言通俗易懂，案例丰富，实用性强，特别适合 R 语言数据可视化的入门与进阶读者阅读，也可作为高等院校统计学、商务分析、大数据等相关课程的教材。

图书在版编目（CIP）数据

R 语言程序设计 / 冯青文，张闯，姜俊颖主编 . —
上海：上海交通大学出版社，2021.9
ISBN 978-7-313-25143-5

Ⅰ. ① R… Ⅱ. ①冯… ②张… ③姜… Ⅲ. ①程序语言—程序设计—高等学校—教材 Ⅳ. ① TP312

中国版本图书馆 CIP 数据核字（2021）第 136255 号

R 语言程序设计
R YUYAN CHENGXU SHEJI

主　　编：冯青文　张闯　姜俊颖
出版发行：上海交通大学出版社
邮政编码：200030
印　　制：北京华创印务有限公司
开　　本：889 mm × 1194 mm　1/16
字　　数：369 千字
版　　次：2021 年 9 月第 1 版
书　　号：ISBN 978-7-313-25143-5
定　　价：58.00 元

地　　址：上海市番禺路 951 号
电　　话：6407 1208

经　　销：全国新华书店
印　　张：16.5

印　　次：2021 年 9 月第 1 次印刷

大数据、云计算、人工智能、信息安全人才培养丛书

大数据系列专家团队		
姓名	职称	研究方向
刘开南	教授	大数据边缘计算
刘明正	教授	数据分析采集
谢泽奇	教授	数据可视化
李志伟	教授	数据仓库
高丽杰	教授	大数据安全
钱振江	高级工程师	数据清洗
李　魏	教授	数据治理
李晓英	教授	数据可视化
王素华	教授	数据标准
张　晨	教授	大数据分析
靳　帅	教授	大数据技术
赵　鹏	高级工程师	数据可视化分析
张　帅	教授	大数据采集与分析
叶远浓	教授	数据可视化
张　洁	教授	数据仓库
梁军学	教授	大数据安全
张　浩	教授	数据清洗
张仁鹏	教授	数据治理
代　飞	教授	数据可视化
杨杉杉	教授	数据仓库技术
赵　噗	高级工程师	大数据安全

序

随着计算机与信息技术的迅猛发展和普及应用，行业应用系统的规模迅速扩大，行业应用所产生的数据呈爆炸性增长，动辄达到数百 TB 甚至数十至数百 PB 规模的行业、企业大数据已远远超出了传统计算技术和信息系统的处理能力。因此，寻求有效的大数据处理技术、方法和手段已成为现实世界的迫切需求。人们将越来越多地意识到数据对企业的重要性。大数据时代对人类的数据驾驭能力提出了新的挑战，也为人们获得更为深刻、全面的洞察能力提供了前所未有的空间与潜力。“大数据开启了一次重大的时代转型”，大数据将带来巨大的变革，改变我们的生活、工作和思维方式，改变我们的商业模式，影响我们的经济、政治、科技和社会等各个层面。

大数据行业应用需求日益增长，未来越来越多的研究和应用领域将需要使用大数据并行计算技术，大数据技术将渗透到每个涉及大规模数据和复杂计算的应用领域。不仅如此，以大数据处理为中心的计算技术将对传统计算技术产生革命性的影响，广泛影响计算机体系结构、操作系统、数据库、编译技术、程序设计技术和方法、软件工程技术、多媒体信息处理技术、人工智能，以及其他计算机应用技术。

大数据的高速发展也代表着新的生产力、新的发展方向，新一轮产业升级和生产力飞跃过程中，人才作为第一位的资源将发挥关键作用。大数据技术的发展将给研究计算机技术的专业人员带来新的挑战和机遇。国内外 IT 企业对大数据技术人才的需求正快速增长，未来 5 ~ 10 年内业界将需要大量的掌握大数据处理技术的人才。这一人才需求与能够提供的人才数量存在一个巨大的差距。目前，由于国内外高校开展大数据技术人才培养的时间不长，技术市场上掌握大数据处理和应用开发技术的人才还十分短缺。

本套大数据、云计算、人工智能、信息安全人才培养丛书中大数据方向的教材明确了适用专业、培养目标、培养规格、课程体系、师资队伍、教学条件、质量保障等各方面要求，是联合多所高校及多家知名企业共同编撰而成的校企合作教材，充分体现了产教深度融合、校企协同育人的理念，是在校企合作机制和人才培养模式的协同创新下打造的精品教材，既适合高等院校相关专业学生使用，也适用于企业相关岗位专业人才的培养。

计算机科学与技术教授 / 工学博士

大数据（高级）分析师

前　言

R 语言是一种统计绘图用的计算机编程语言。R 语言天生为统计而生，在数据分析、统计建模、数据可视化方面也具有绝对的优势。近年来，R 语言在生态学与环境、进化、遗传学、经济学等领域得到了广泛的应用。截至 2020 年 6 月，R 程序包集成网络（CRAN）上已经有 12 000 多个程序包，并且仍在增加中。此外，在 Bioconductor、R-Forge、Github 等多个网站上还有为数众多的 R 程序包。了解和学习 R 编程无疑能帮助科研人员用好这些宝贵的资源。

本书通过大量的实例讲述 R 语言程序设计需要掌握的各方面知识。全书共 14 章，首先用 13 章介绍了 R 语言程序设计的基础知识，包括初识 R 语言，R 语言基础，单模式数据结构，多模式数据结构，基本数据管理，高级数据管理，日期、时间和因子，程序逻辑结构，R 语言函数，R 语言读取数据，类与对象，图形初阶，ggplot2 包；然后通过一个综合案例讲解了 R 语言程序在实际项目中的应用。

本书在编写上具有以下特色：

（1）通过章前的“学习目标”“知识导图”“本章导读”明确该章要学习的内容，使学生做好学习的准备；通过文中的“说明”“提示”“注意”等模块，丰富知识内容，提高学生的学习兴趣。

（2）在精练语言的基础上，充分利用图、表等形象地描述知识点，帮助学生更好地理解所学内容。本书内容由浅入深，循序渐进，符合学生的认知规律。

（3）计算机技术发展很快，本书着重讲解当前的最新知识和主流技术，使学生学到的知识和技术都与行业密切联系，做到学以致用。

此外，本书作者还为广大一线教师提供了服务于本书的教学资源库，有需要者可致电 13810412048 或发邮件至 2393867076@qq.com。

本书可作为高等院校统计学、商务分析、大数据等相关课程的教材，也可作为其他数据科学从业者的参考用书。由于编写时间仓促，加之网络技术发展迅猛，书中存在的不足和疏漏之处，敬请广大读者批评指正，在此表示衷心的感谢！

目　录

第 3 章　单模式数据结构 / 29

第 4 章　多模式数据结构 / 39

第 5 章　基本数据管理 / 49

第 6 章　高级数据管理 / 65

第 10 章　R 语言读取数据 / 149

第 11 章　类与对象 / 167

第 12 章　图形初阶 / 191

第 1 章 初识 R 语言

学习目标

1. 了解 R 语言简史。
2. 掌握 R 语言在大数据中的应用。
3. 了解 R 语言在线资源。
4. 了解 R 语言的开发。
5. 掌握集成开发环境的安装。

知识导图

笔记

本章导读

R 是一套完整的数据处理、计算和制图软件系统，R 语言是目前世界上使用最广泛的统计编程语言，它是数据科学家的第一选择。由于 R 是一种专业性很强的统计语言，所以学习者必须有统计学的知识基础，不然很多东西会掌握得比较慢。

本章将对 S 语言的诞生、R 语言在大数据中的应用，以及 R 语言手册、在线资料、R 语言联盟和集成开发环境等进行介绍，使读者对 R 语言基础有初步的了解，为后面的学习打下基础。

1.1 R 语言概述

1.1.1 S 语言的诞生

S 语言是一种用来进行数据探索、统计分析、制图的解释型语言，它于 1975—1976 年由贝尔实验室的 RickBecker、JohnChambers 和 AllanWilks 开发。

第一个可运行的 S 语言版本在 1976 年发表，在 GCOS 操作系统上运行，当时它还没有正式名称，曾经被人称为互动式 SCS（interactive SCS，ISCS）、统计运算系统（statistical computing system）、统计分析系统（statistical analysis system，SAS）。直到 1979 年，它才被正式定名为 S 语言。

S 语言丰富的数据类型（如向量、数组、列表、对象等）特别有利于实现新的统计算法，其交互式运行方式、强大的图形交互功能使得我们可以方便地探索数据。

S 语言的实现版本主要是 S-PLUS。它基于 S 语言，并由 MathSoft 公司的统计科学部进一步完善。作为统计学家及一般研究人员的通用方法工具箱，S-PLUS 强调演示图形、探索性数据分析、统计方法、开发新统计工具的计算方法以及可扩展性。

S-PLUS 可以直接进行标准的统计分析，并得到所需结果，它的主要特点是可以交互地从各个方面发现数据中的信息，并可以很容易地实现一个新的统计方法。

1.1.2 R 语言简史

在 S 语言的基础上，新西兰奥克兰大学的 Robert Gentleman 和 Ross Ihaka 及其他志愿人员在 S 语言的基础上开始构思一种新的用于统计学分析的开源语言，因为他们名字的第一个字母都是 R，所以这门语言就叫作 R 语言。R 是 S 语言的一个分支，一般人认为 R 就是 S 语言的一种实现。S 语言的实现版本主要是 S-PLUS，这是一个由 MathSoft 公司开发的一种基于 S 语言的统计学软件，而 R 也是 S-PLUS 的基础，所以它们在程序语法上几乎一样，可能只是在函数方面有细微差别，所以两门语言只要稍加修改，就可以非常容易地移植到对方程序中。

1.1.3 关于大数据

一般认为，大数据（big data）是指无法在一定时间范围内用常规软件工具进行捕捉、管理和处理的数据集合。这里也有许多不同的定义，比如研究机构 Gartner 认为，大数据是需要新处理模式才能具有更强的决策力、洞察发现力和流程优化能力的海量、高增长

率和多样化的信息资产。而麦肯锡全球研究所给出的定义是“一种规模大到在获取、存储、管理、分析方面大大超出传统数据库软件工具能力范围的数据集合，具有海量的数据规模、快速的数据流转、多样的数据类型和价值密度低四大特征。”总之，大数据的核心不在于掌握庞大的数据信息，而在于对这些含有意义的数据进行专业化处理。换言之，如果把大数据比作一种产业，那么这种产业实现盈利的关键在于提高对数据的“加工能力”，通过“加工”实现数据的“增值”。

如果把数据比喻为蕴藏能量的煤矿，煤炭按照不同的性质可分为焦煤、无烟煤、肥煤、贫煤等类，而露天煤矿、深山煤矿的挖掘成本又不一样。与此类似，大数据并不在“大”，而在于“有用”，在于让数据产生更高的价值。所以，数据价值含量、挖掘成本比数据的量更重要。对于很多行业而言，如何利用这些大规模数据是赢得竞争的关键。

大数据的价值主要体现在以下几个方面：

（1）为企业提供基础的数据统计报表分析服务。分析师能够轻易获取数据，生成分析报告，指导产品和运营。产品经理能够通过统计数据完善产品功能，改善用户体验。运营人员可以通过数据发现运营问题，并确定运营的策略和方向。管理层可以通过数据掌握公司业务运营状况，从而进行一些战略决策。

（2）人工智能离不开数据。数据作为人工智能发展的重要基础，在未来的智能化时代也将扮演着重要的角色，所以数据的价值也必然会随着人工智能技术的发展而得到提升。由此可以预测，一家公司如果拥有海量的用户数据，那无异于拥有一座取之不尽的矿山。

（3）未来的大数据能更好地解决社会、商业、科学各类问题。大部分数据都与人类有关，要通过大数据解决人的问题。比如，建立个人的数据中心，每个人的日常生活习惯，身体状态，社交网络，知识能力，爱好性情，情绪波动记录等数据可以充分利用，医疗机构将实时地监测用户的身体健康状况；教育机构针对用户制订培训计划；服务行业提供符合用户习惯的服务；社交网络为志同道合的人群相识相知而提供便利；政府能在用户心理健康出现问题时及时干预；金融机构能为用户的资金提供更有效的使用建议和规划；道路交通平台可以为用户提供合适的出行线路。

1.1.4 R 语言在大数据中的应用

R 语言（简称 R）是专门为数据科学而生的软件，能帮用户完成该领域内统计分析和开发与维护等工作。R 语言拥有 15 000 多个成熟稳定的 R 包，几乎所有类型的数据分析任务都可以在 R 中完成，更强大的是，它还提供了各种数学计算、统计计算的函数，从而使使用者能灵活机动地进行数据分析，甚至创造出符合需要的新的统计计算方法。R 语言是一套完整的数据处理、计算和制图软件系统，其功能叙述如下。

（1）完整连贯的统计分析工具：R 内建多种统计学及数字分析功能，是统计分析、绘图功能的自由开源软件，拥有完整体系的数据分析和挖掘工具，能够有效地存储和处理数据。

（2）向量、矩阵运算等强大的数组运算工具：虽然 R 主要用于统计分析或者开发统计相关的软体，但也有人将其用作矩阵计算。其分析速度可媲美 GNU Octave，甚至商业软件 MATLAB。

（3）优秀的统计制图功能：R 具有丰富的数据挖掘工具包（packages），拥有完整体系的数据统计和分析工具，为数据分析和显示提供了强大的图形功能。

（4）简便而强大的编程语言：可操纵数据的输入和输出，可实现分支、循环，用户可

笔记

自定义功能。

最后，在 R 语言里的软件包生态系统，特别在机器学习方面，R 语言体现出强大的联运效应，即在大数据领域的任何新型研究成果可能都会马上以 R 软件包的形式体现出来。因此，从这个角度看，R 语言始终站在技术发展的尖端位置。

1.2 R 语言社区

1.2.1 R 语言手册

关于 R 语言手册，初学者或者英文水平不高的读者可以借助各个搜索引擎，搜索中文版的语言手册，这些资料很容易获得。但如果想更深入地学习，还是建议看官方的文档，因为翻译过来的内容可能有偏差。R 语言的官方文档地址是 https://www.r-project.org/，官方文档中提供了 R 相关的学习手册，单击“ Manuals”按钮即可浏览，建议所有读者都看一看。另外，如果有不会的函数，在寻找搜索引擎搜索之后，也可以看 R 函数在线帮助（https://www.rdocumentation.org/），进而获得对某个函数或者包的帮助信息。

1.2.2 在线资源

R 语言的在线资源有很多，相信读者在学习的过程中也能搜集到很多资源。常见的如菜鸟教程、W3Cschool 以及慕课等学习网站上都有很多免费的学习资源。在此为大家推荐如下资源网站，供学习使用。

丰富的 R 语言博客资源：http://www.r-bloggers.com

R 语言资源汇总：https://github.com/qinwf/awesome-R

R 语言搜索引擎：http://www.rseek.org

R 语言函数在线帮助：http://www.rdocumentation.org

关于 R 语言的问答：http://stackoverflow.com/questions/tagged/r

一个入门级的 R 语言在线教程：http://tryr.codeschool.com

一个交互式的 R 语言在线教程：https://www.datacamp.com

统计之都：http://cos.name

如果大家在学习或工作中遇到 R 语言的一些问题，除了国内的搜索引擎外，这里推荐 Stack Overflow 网站（https://stackoverflow.com）。Stack Overflow 是一个与程序相关的 IT 技术问答网站，用户可以在该网站免费提交问题，浏览问题，索引相关内容。

1.2.3 R 语言联盟

2015 年 7 月，Linux 基金会宣布成立了 R 语言联盟（R Consortium），用以加强技术和用户交流的合作计划，其宗旨是为 R 社区、R 基金会、团体以及个人使用、维护和分发 R 软件提供帮助。R 语言联盟将与 R 基金会和其他组织共同工作，并向其提供开发、维护和分销 R 软件服务，以及为 R 用户社区提供统一框架。R 语言联盟的创始公司和组织包括 R 基金会、微软、RStudio、谷歌和惠普等。

R 语言被统计学家、分析师以及数据科学家用来发掘数据价值。这是一种免费开源的

编程语言，用于统计计算和为数据分析、建模以及可视化提供互动环境。R 语言联盟将会补充 R 基金会工作的不足，帮助这家位于奥地利的非营利性机构维护这种语言。R 语言联盟将会着力用户外延以及其他项目设计，帮助 R 语言用户。

笔记

R 语言联盟将打造一个新的代码托管平台 R-Hub，用于 R 语言的开发和分发软件包，这项计划得到 Linux 基金会的资助。R-Hub 平台将向 R 软件包提供开发、构建、测试和验证服务，目标是将其打造成“满足 R 社区所有需求的构建平台”。R-Hub 将基于并兼容现有的 CRAN 和 R-Forge 网站。

1.2.4 用户活动

R 语言作为统计和数据挖掘界广泛应用的工具，每年 R 的官方机构都会举办多场 R 语言的学术会议，各个国家及地区也定期有 R 用户的交流活动。在中国的北京、上海、杭州、广州等地，每年都会有多场 R 语言会议，会议内容覆盖数据科学在各行各业的应用，包括天文、地理、医疗、生物、金融、能源、互联网等领域，在高校和业界均有深远影响，促进了 R 语言乃至数据科学在中国的推广和发展。如今 R 语言会议成为 R 语言社区在国内影响力较大的交流盛会，聚学术专家、业界精英、技术大咖于一堂，让更多的数据人参与其中，促进社区内部的交流和进步。

比如，在活动家网站（https://www.huodongjia.com）上，可以搜索到近期将要举办的线上或线下的 R 语言学术活动，有兴趣的读者可以报名参加。当然，大多数学术会议都是收费的，也有部分是免费的。

1.3 集成开发环境

1.3.1 R GUI

本节以 Windows x64 系统为例安装 R GUI，安装包 R 的官方网站是 https://cran.r-project.org/，界面如图 1-1 所示。从这里可以看到当前最新的 R 版本号，如果想了解最新版本的详细信息，可以单击“What's new？”链接查看。

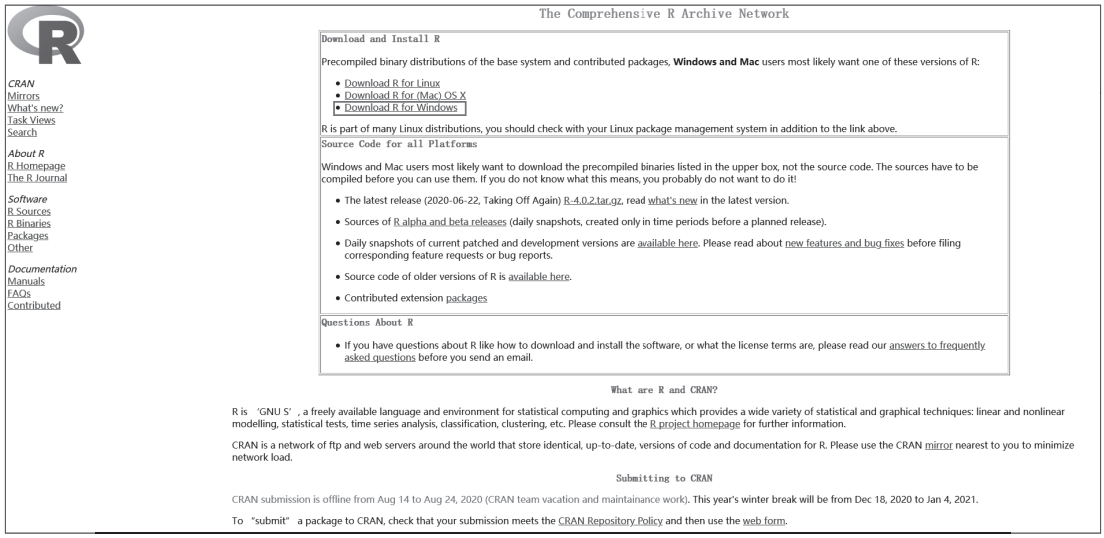


图 1-1　R 的官方网站

笔记

此时选择自己对应的计算机合适的版本。本节以 Windows 系统为例，所以单击“Download R for Windows”链接。之后进入下一界面，如图 1-2 所示。

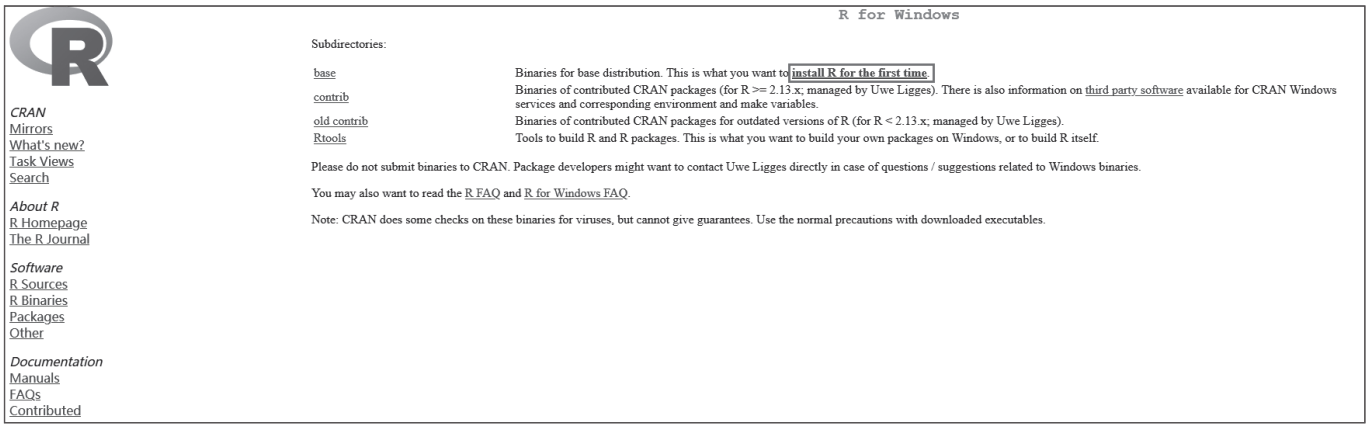


图 1-2　Download R for Windows

单击“install R for the first time”链接，进入新的界面，如图 1-3 所示。

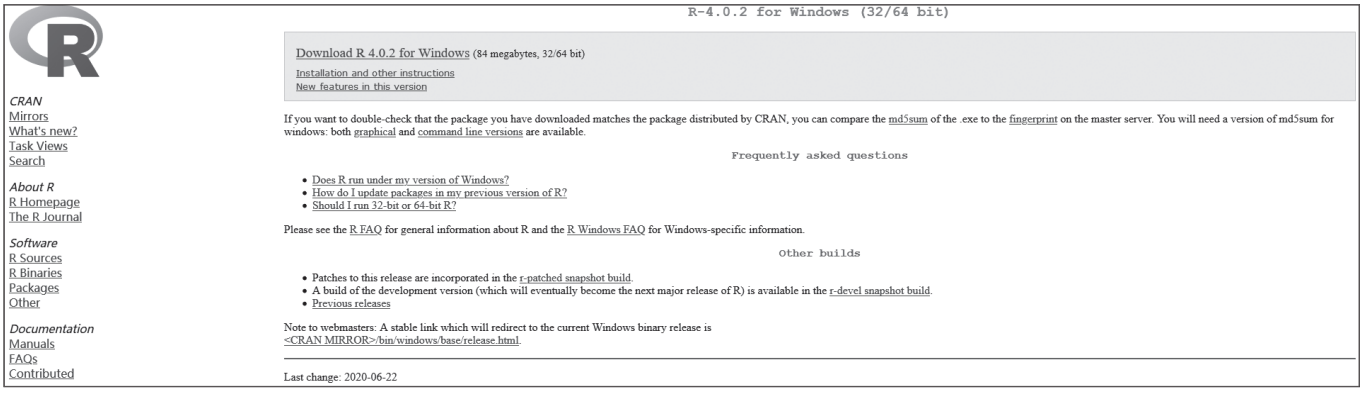


图 1-3　install R the first time

单击“Download R 4.0.2 for Windows（84 megabytes，32/64 bit）”链接，在弹出的对话框中单击“保存文件”按钮，然后等几分钟，下载完成。

打开下载的 R-4.0.2-win.exe。若打开 R-4.0.2-win.exe 的时候出现问题，可以尝试单击鼠标右键，以管理员身份运行。在“选择语言”对话框中，建议使用“中文（简体）”，如图 1-4 所示。

图 1-4　选择语言

选择语言后，单击“确定”按钮，在后边的安装向导对话框中单击“下一步”按钮，当需要选择安装的路径时，建议放在 D 盘，此处将其装至“ D:\Program Files（x86）\R-4.0.2”，接着单击右下角的“下一步”按钮，进入“选择组件”对话框，如图 1-5 所示。

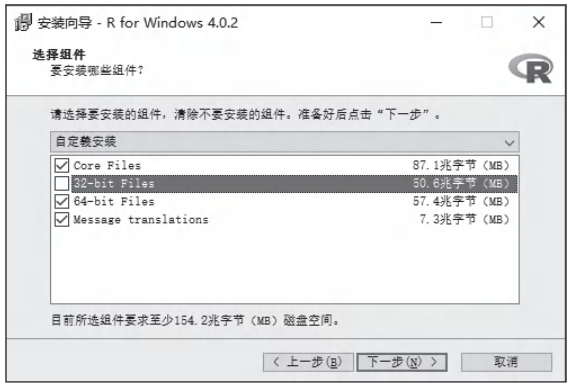


图 1-5　选择组件

在图 1-5 中选择安装的组件，即以 Windows x64 为例，需要取消 32-bit Files 前的“√”，接着一直单击右下角的“下一步”按钮，直到进入“选择附加任务”对话框，如图 1-6 所示。

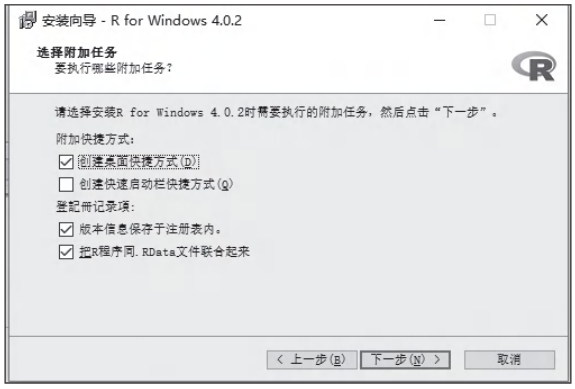


图 1-6　选择附加任务

在选择附加任务中勾选“创建桌面快捷方式”，安装完成后，直接单击 R 桌面的快捷方式就可以打开，这样会更方便一点。当然，如果此处没有选择创建快捷方式，以后可通过程序启动文件，重新创建快捷方式。单击右下角的“下一步”按钮，进入安装等待，直到成功安装完成。

成功安装后，打开桌面的快捷方式，就可以看到如图 1-7 所示的界面。

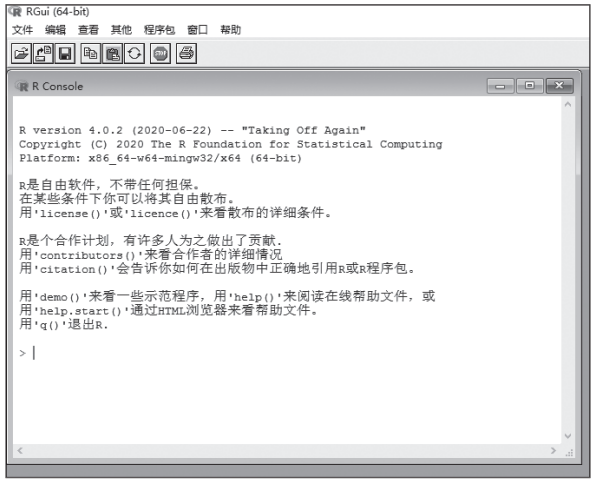


图 1-7　RGui 界面

笔记

1.3.2 RStudio IDE

在 1.3.1 节中，我们成功地安装了 R。使用 R 自带的环境可以进行操作，但 R 自带的环境操作起来可能不是很方便，而 RStudio 很好地解决了这个问题，而且它还具有调试、可视化等功能，支持纯 R 脚本、Rmarkdown（脚本文档混排）、Bookdown（脚本文档混排成书）、Shiny（交互式网络应用）等。

RStudio 是 R 语言的集成开发环境，是一款 R 语言的 IDE，分为面向桌面用户 IDE 和 Linux R 服务器版编辑器两种编辑器，采用 AGPL v3 与 RStudio License Agreement 双协议授权。R 是 RStudio 的基础，必须先安装 R，再安装 RStudio。即使只使用 RStudio，也需要事先为计算机安装好 R。RStudio 只是辅助你使用 R 进行编辑的工具，因为它自身并不附带 R 程序。

下面讲一下如何安装 RStudio。首先，打开 RStudio 官方网站 https://www.rstudio.com/products/rstudio/download/，如图 1-8 所示。

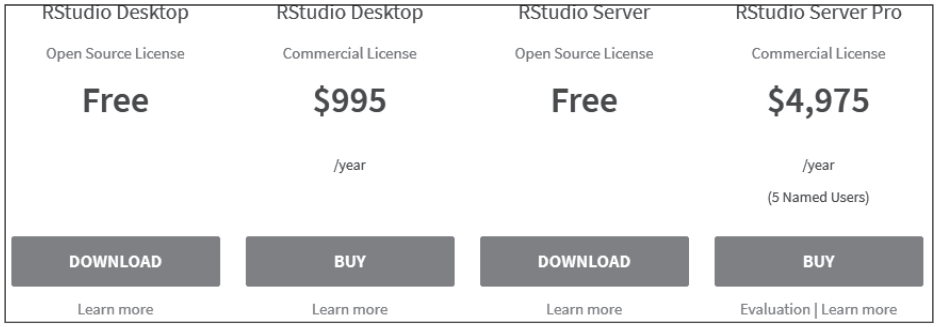


图 1-8　RStudio 官方网站

选择 Free（免费）软件，然后单击“DOWNLOAD”下载，进入如图 1-9 所示的界面。本书成稿时，RStudio 最新版本是 RStudio Desktop 1.3.1073，支持的最低 R 版本是 3.0.1，在 1.3.1 节我们已安装过。此处直接选择 All Installers 中的 RStudio-1.3.1073.exe-Windows 10/8/7，单击开始下载。

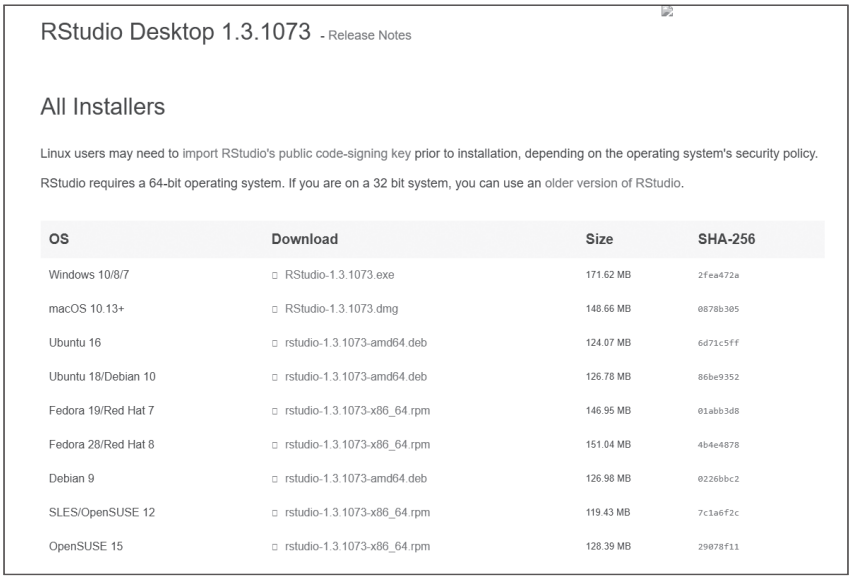


图 1-9　RStudio 下载

笔记

打开刚下载的安装包，如图 1-10 所示。在安装向导中，根据提示单击“下一步”按钮，再选择安装位置，这里仍然建议安装在 D 盘！再次单击“下一步”按钮，然后直接单击“安装”就可以了。

图 1-10　RStudio 安装

安装成功后，在自己安装的路径下选择 bin 文件→ rstudio.exe，右击鼠标，从弹出的快捷菜单中执行“发送到”→“桌面快捷方式”命令，如图 1-11 所示。

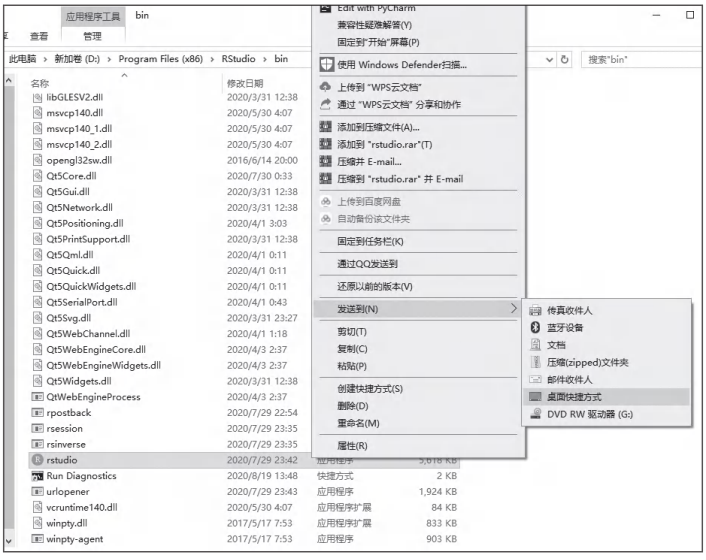


图 1-11　RStudio 快捷方式

然后进入桌面，双击 RStudio 快捷键即可打开如图 1-12 所示的界面。

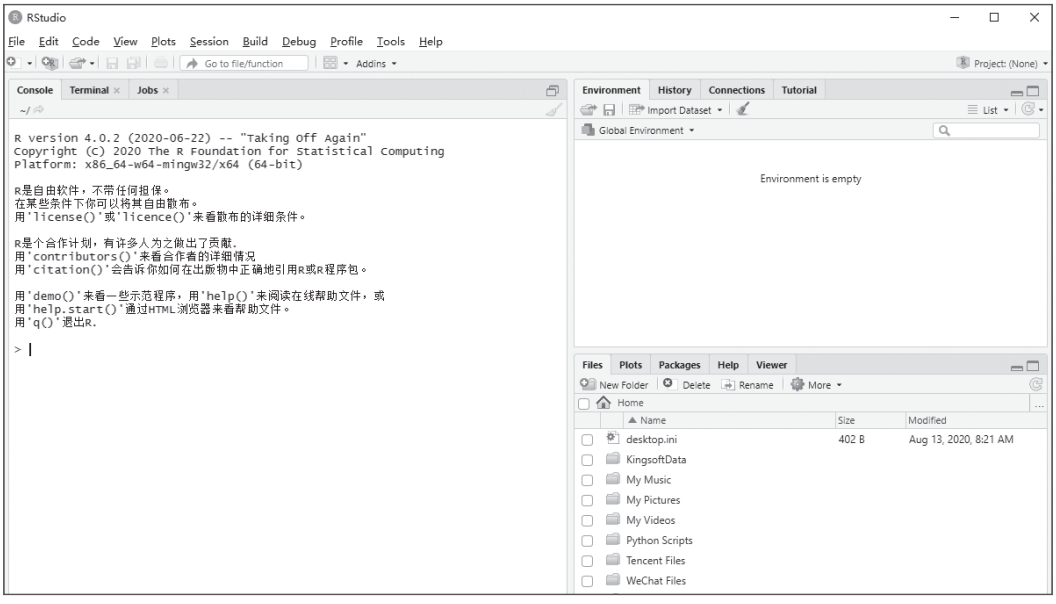


图 1-12　RStudio 界面

至此，RStudio 安装成功。

笔记

1.3.3 Jupyter Notebook

学过 Python 的读者，应该接触或使用过 Jupyter Notebook。Notebook 是一个交互式笔记本，支持运行 40 多种编程语言。按照 Jupyter 创始人的说法，起初他是想做一个综合 Ju（Julia）、Py（Python）和 R 三种科学运行语言的计算工具平台，所以将其命名为 Jupyter。

它可以内置很多编程语言内核（kernel），默认的内核是 Python，所以，当安装 Notebook 后，就可以直接运行 Python 代码。

Jupyter 发展到现在，已经成为一个几乎支持所有语言，能够把软件代码、计算输出、解释文档、多媒体资源整合在一起的多功能科学运行平台。Jupyter Notebook 的工作界面如图 1-13 所示。可以看出，当前还不支持创建 R 语言项目。

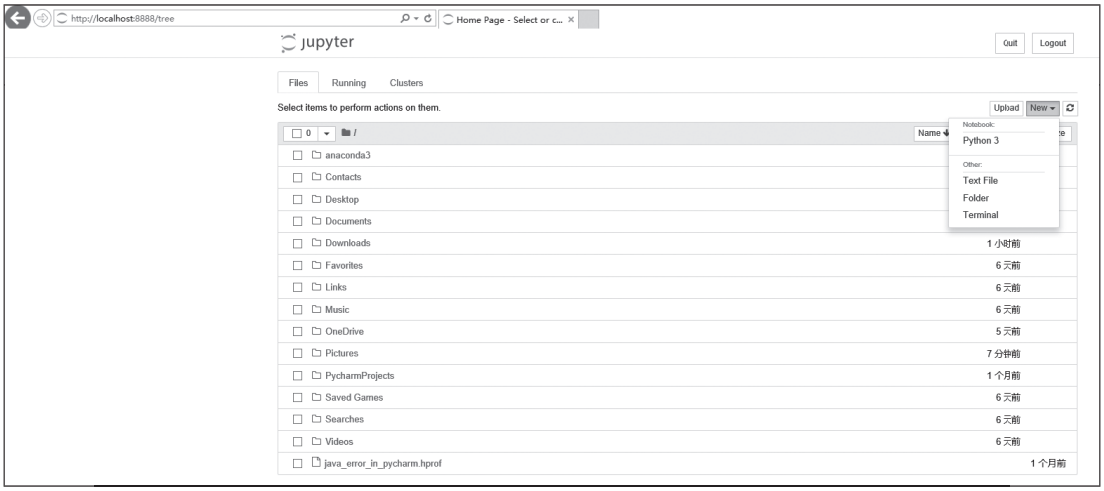


图 1-13　Jupyter Notebook 的工作界面

下面总结了几个 Jupyter Notebook 的优势。

1. 整合了所有资源

在软件开发过程中频繁地进行上下文切换，会影响生产效率。举一个例子，假设需要切换窗口看一些文档，再切换窗口用另一个工具画图，不断地切换窗口就会成为影响效率的因素。而 Jupyter Notebook 则不同，它会将所有用软件编写的资源全部放在一个地方，无须切换窗口就可以轻松找到。

2. 交互性编程体验

在机器学习和数据统计领域，Python 编程的实验性特别强，比如为了测试 100 种不同的方法，有时就需要将一小块代码重写 100 遍。在这种情况下，如果使用传统的 Python 开发流程，每一次测试都要将所有代码重新运行一遍，这样会花费开发者很多时间。

Jupyter Notebook 引进了 Cell 的概念。每次测试可以只运行一小部分代码，并且在代码下方立刻就能看到运行结果。如此强的交互性，满足了 Python 程序员可以专注问题本身，不会被频繁的工具链拖累，也不用在命令行之间来回切换，所有工作都能在 Jupyter Notebook 上完成。

3. 轻松运行他人编写的代码

同样是在机器学习和数学统计领域，我们可能会借鉴他人分享的代码，但当复制过来想运行时，却需要使用 pip 安装一大堆依赖的库，这足以让人抓狂。而 Jupyter Notebook

就可以解决这个问题。

笔记

例如，Jupyter 官方的 Binder 平台以及 Google 提供的 Google Colab 环境，它们可以让 Jupyter Notebook 变得和 Google Doc 在线文档一样。比如用 Binder 打开一份 GitHub 上的 Jupyter Notebook 时，就不需要安装任何 Python 库，直接打开代码就能运行。

通过以上介绍，我们对 Jupyter Notebook 有了初步的了解，下面介绍 Jupyter Notebook 的安装。安装 Jupyter Notebook，推荐使用 Anaconda。Anaconda 是一个基于 Python 的数据处理和科学计算平台，它已经内置了许多非常有用的第三方库，装上 Anaconda，就相当于把 Python 和一些如 Numpy、Pandas、Scrip、Matplotlib 等常用的库自动安装好了，使得安装比常规 Python 安装要容易。当然，Python 不是本书讲述的内容，但因为它是内置于 Jupyter Notebook 的，所以安装 Jupyter Notebook 后默认就安装了 Python。

首先，进入 Anaconda 的官网 https://www.anaconda.com/download/#windows，如图 1-14 所示，下载对应的版本，这里选择的是 64-Bit Graphical Installer（466 MB）。

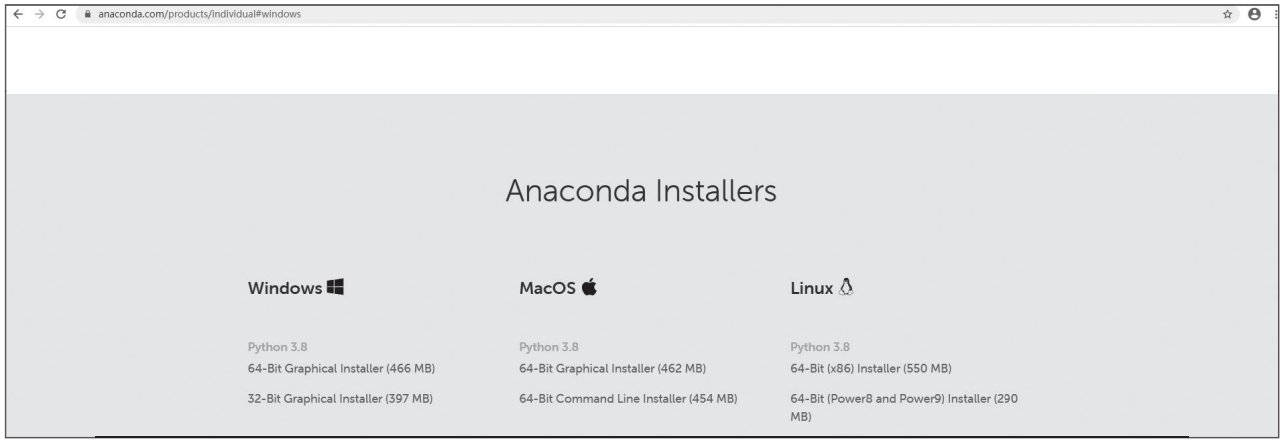


图 1-14　Anaconda 官网

下载完成后，双击下载好的 .exe 文件进行安装，单击“Next”按钮，单击“I agree”，选择“Just Me”，之后单击“Next”按钮。注意，在选择安装目录对话框中，仍然建议安装在 D 盘，单击 Next 按钮进入如图 1-15 所示的界面，建议勾选红色框中的“Add Anaconda to my PATH environment variable”，自动配置环境变量，单击“Install”按钮等待完成。

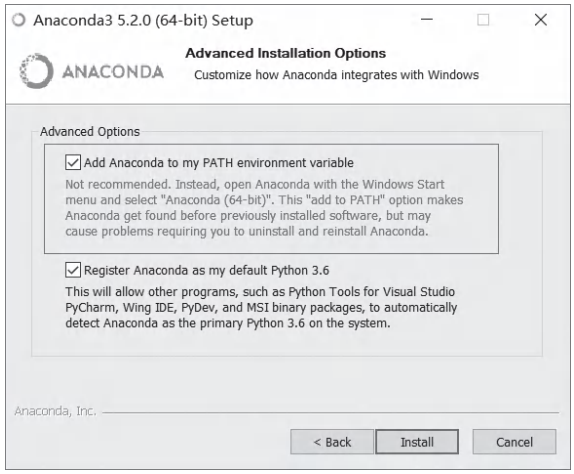


图 1-15　Anaconda 环境变量

笔记

在接下来的对话框中单击“skip”，进入图 1-16，取消两个勾选，最后单击“Finish”按钮。

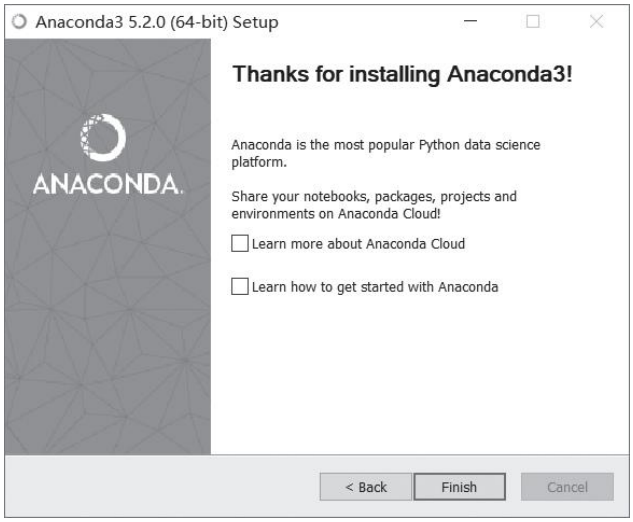


图 1-16　安装完成

安装成功后，在桌面左下方的搜索框中输入“jupyter notebook”，可看到图 1-17，双击打开 Jupyter Notebook 即可看到如图 1-13 所示的界面。

图 1-17　打开 Jupyter Notebook

下面介绍如何在 Jupyter Notebook 中安装 R 语言内核。要想在 Jupyter Notebook 中运行 R 语言其实非常简单，打开 RStudio，依次输入以下代码，安装对应的扩展包即可。

```
install.package('repr','IRdisplay','evaluate','crayon','pbdZMQ','devtools','uuid','digest')
library(devtools)
install_github("IRkernel/IRkernel")
IRkernel::installspec()
```

执行完毕，打开 Jupyter Notebook，就可以新建 R 项目了，如图 1-18 所示。

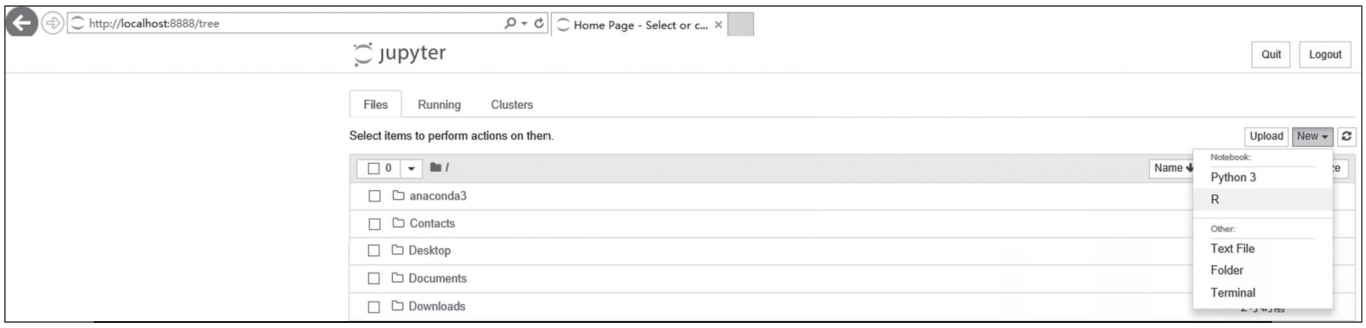


图 1-18　Notebook 中显示 R

1.4　R 语言的开发

在本书的编排上，首先介绍给读者的是 R 语言的初级内容，主要包括了解 R 语言的对象，掌握程序的基本编写逻辑，怎样进行数据导入、整理、转换，怎样进行假设检验，然后由浅入深进一步介绍如何分析，以及如何使用各种包。

R 语言的学习同样遵循二八定律，即百分之八十的精力需要花费在软件之外的统计学理论背景、业务知识学习上，而需要使用 R 软件实现的部分，不要单纯地学 R 语言的基础语法，尽管这些很重要，需要明确的是，统计学的理论背景更加重要，如果理论理解透了，很多事情就水到渠成，迎刃而解了。这一点特别体现在对统计与数据分析的学习上。

对于数据可视化，读者需要在牢固掌握基础语法运用、数据清洗技能的基础上，能够熟练运用一套图形语法。掌握这些之后，就不要将过多精力放在工具和代码本身，而是要更多地关注可视化素养，提升设计审美水平。数据可视化除了依赖实现的工具和平台语法之外，更多的是对于数据源的理解、对于可视化的理解、对于设计理念的融会贯通（怎么配色、怎么排版、怎么搭配字体等）。当然，对于设计、审美、创意这些柔性的东西，很难通过一两本书或者一两套课程完全解决，这些是内化于生活，积累于日常的点点滴滴。当然，如果有意识地通过一些课程、书籍慢慢培养，日积月累也会见效的。

R 语言是统计学家开发的，问世之初就决定了它的使命是统计计算和数据可视化，这算是 R 语言核心功能的两个大方向。

R 包是 R 的精髓，说是 R 的全部，一点也不为过。2006 年 3 月 15 日，第一个 R 包（coxrobust）正式上线 CRAN。近年来，R 包出现了井喷的趋势，目前已经有超过 20000 个 R 包，如果再算上 GitHub 上托管的个人开发的小众包，可能有好几万了，而且还在不断增长，每天都在增长，每一个 R 包都有各自的功能，R 的强大，正是因为这些程序包。对于这两个方向而言，统计计算的学习，基础都在课堂理论与专业背景上，说实话，R 语言只是提供了一个实现的平台而已，它并不改变或者创造新的理论、模型。

虽然这些包里的函数很多，使初学者眼花缭乱，但实际上，随着学习的深入，这些统计计算使用的公式、用到的模型算法，大部分都被封装到一个个扩展包里，导入包之后，仅调用对应函数、设置对应参数即可。这些函数与 Excel 里面的函数没有区别，不必恐惧。

至于参数的调优、模型的检验与优化，这些操作依赖的知识背景也基本来源于课堂学

笔记

习和专业背景，与 R 软件的关系并不大，对于需要自己写算法的情况，也仅是在函数的基础上按照成熟的理论算法进行调优和计算，这与软件无关，而与软件之外的专业背景和行业经验有关。

说到底，对于统计学习这一块，重要的是理论背景、业务经验，真正需要 R 实现的仅是内置的扩展包函数、基础语法。

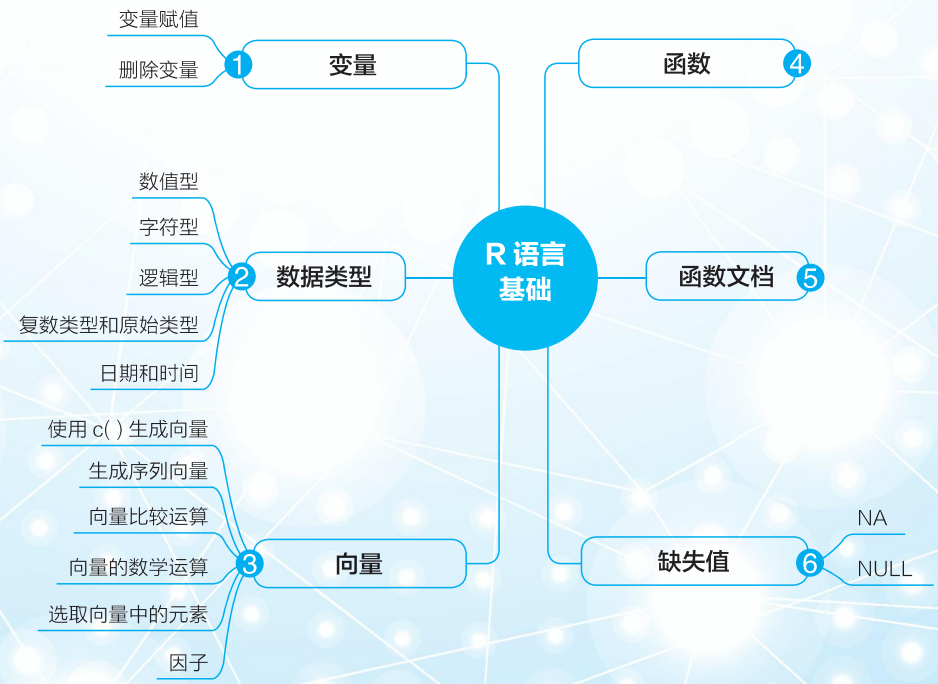
类比一下 SPSS 的学习，一个不懂统计学的人很难学好 SPSS，尽管他知道各种功能模块和菜单。同样，一个不懂统计学和数学的人也很难学好 R 语言，尽管他很熟悉 R 语言的基础语法和很多扩展包所能实现的功能。

第 2 章 R 语言基础

学习目标

1. 掌握变量的使用。
2. 掌握数据类型的使用。
3. 掌握向量的使用。
4. 掌握函数调用的方法。
5. 掌握函数文档的操作。
6. 掌握缺失值的使用。

知识导图



笔记

本章导读

本章将介绍 R 语言最基本的知识和语法。R 语言最基本的数据是向量，本章将首先介绍向量与 R 语言的其他数据类型之间的关系。R 语言中，单个数值没有单独的数据类型，它只不过是向量的一种特例。虽然本章所举的案例都是最常见的问题，但通过这些问题我们能掌握一些解决问题的基本技巧。

2.1 变量

R 软件采用“计算器”模式方便、快捷，但有时需要定义变量并保存变量值，以便重复使用。在 R 软件中，不需要声明变量或者显示地创建变量，只需要将值赋予一个名称，R 软件就会自动生成该名称的变量。

2.1.1 变量赋值

赋值操作由一个小于号（<）和减号（–）构成，两个符号之间没有空格。变量被定义之后，将存储到当前的工作空间中，此时工作空间仅存储在计算机内存中，当退出 R 软件时可保存至本地硬盘。R 语言是动态输入语言，可随意改变变量的数据类型。例如，先给 x[①] 赋一个数值型数值，然后再对其赋一个字符串，在这一过程中 R 软件能够理解用户的意图。

示例代码如下：

```
> x <– 4
> x
[1] 4
> x <–"hello"
> x
[1] "hello"
```

2.1.2 删除变量

删除变量的命令无法撤销，即删除的变量无法找回。可以通过如下命令同时删除多个变量：

```
> rm(a,B,C)
```

另外，rm( ) 函数中有一个 list 参数，它包含所有需要的变量名称。通过 rm( ) 函数和 ls( ) 函数，可删除工作空间中的所有变量。其中，ls( ) 函数可以显示当前工作空间中的所有对象的名称。ls( ) 函数输出的结果是一个字符串向量，其中向量的每个元素代表一个变量名。当工作空间中没有已定义的变量时，函数 ls( ) 会返回一个空向量 character( )。示例代码如下：

```
> ls()
[1] "A"          "chengji"
[3] "d"          "juzhen"
[5] "juzhen2"    "juzhencheng"
[7] "mat"        "v"
[9] "y"
> rm(list = ls())
> ls()
character(0)
```

① 为了与代码中的格式一致，正文中变量 x 等也全部采用正体表示。

2.2 数据类型

笔记

在 R 中，可以将一组数据用 c 函数数组组合在一起，形成一个原子型向量。示例代码如下：

```
number <- c(1,2,4,6)
> number
[1] 1 2 4 6
```

R 中绝大多数的数据结构都是原子型向量。每一个原子型向量都将值存储在一维向量中，并且只能是一种数据类型。R 可以识别六种基本类型的向量，分别是数值型、字符型（character）、逻辑型（logical）、复数类型（complex）、原始类型（raw）、日期和时间型。

2.2.1 数值型

数值型数据又称为双整型（double）。数值可正可负，可包含小数，也可以不包含。在 R 中输入的任何一个数值都默认以 double 型存储，并且可以使用 typeof( ) 函数查看向量是什么类型，示例代码如下：

```
> chengji <- c(69.5,75,99.5,88)
> chengji
[1] 69.5 75.0 99.5 88.0
> typeof(chengji)
[1] "double"
```

另外我们还要明白整型向量是用来存储整型（integer）的数据，也就是数据不需要小数部分。在 R 中，明确设定整型的方法是在该数值之后加上大写的 L，示例代码如下：

```
> int <- c(1L,2L,5L)
> int
[1] 1 2 5
> typeof(int)
[1] "integer"
```

在 R 中，如果不加 L，R 并不会将一个数值设定为整数。为了避免浮点误差，可以只限定使用整型，而不使用带有小数点的双整型。但是，对于很多数据科学的应用场景来说不太现实，并且由于浮点运算所导致的误差并不是非常显著，大多数情况下都使用双整型，而不是整型。

2.2.2 字符型

在 R 中，字符要加双引号，再组合起来构成一个字符型向量。字符型向量中的单个元素称为字符串（string）。字符串不仅可以包含英文字母，也可以由数字或者符号组成。在 R 中，任何加双引号的对象都会被当作字符串，无论引号内是什么元素。示例代码如下：

笔记

```
> text <- c("Hello","World")
> text
[1] "Hello" "World"
> typeof(text)
[1] "character"
```

2.2.3 逻辑型

逻辑型向量用来存储 TRUE 和 FALSE，这是 R 中布尔数据的存储类型。在比较数据时，其结果就是逻辑型数据。示例代码如下：

```
> 3 > 4
[1] FALSE
> 3 < 4
[1] TRUE
```

只要在 R 中输入 TRUE 或者 FALSE，就会被当作逻辑型数据。R 也会默认把 T 和 F 分别当作 TRUE 和 FALSE 的简写。示例代码如下：

```
> logic <- c(TRUE,FALSE,T,F)
> logic
[1]  TRUE FALSE  TRUE FALSE
> typeof(logic)
[1] "logical"
```

2.2.4 复数类型和原始类型

R 还可以识别另外两种类型的原子型向量：复数类型和原始类型。复数类型向量用来存储复数。要生成一个复数类型向量，只将某个数字与带 i 的虚数项相加即可。示例代码如下：

```
> comp <- c(1+1i,1+2i,1+3i)
> comp
[1] 1+1i 1+2i 1+3i
> typeof(comp)
[1] "complex"
```

原始类型向量用来存储数据的原始字节。原始类型向量的生成较为复杂，但是如果要生成一个长度为 n 的空原始类型向量，可以用 raw(n) 表示。示例代码如下：

```
> raw(4)
[1] 00 00 00 00
> typeof(raw(4))
[1] "raw"
```

2.2.5 日期和时间

笔记

除了以上的数据类型外，R 用一个特殊的类表示日期和时间数据。

运行 Sys.data( ) 可以得到当前的日期。Date 类存储了从 1970 年 1 月 1 日以来计算的天数，更早的日期表示为负数，也就是说，Date 类型是一个整数。

运行 Sys.time( ) 可以得到计算机的当前时间，虽然显示出来的信息是一串字符，但它的类型是 double 型。它的类是 POSIXct 和 POSIXt。POSIXct 类记录了以标准时间时区（UTC）为准的，从 1970 年 1 月 1 日开始时的秒数，即 POSIXct 类型是整数。POSIXt 把日期和时间存储为一个列表，其中包括秒、分、时和月份等，POSIXt 是使用列表表示日期和时间的。示例代码如下：

```
> Data1 <- Sys.Date()
> Data1
[1]"2020-09-02"
> typeof(Data1)
[1] "double"
> time <- Sys.time()
> time
[1] "2020-09-02 18:52:45 CST"
> typeof(time)
[1] "double"
> class(time)
[1] "POSIXct" "POSIXt"
```

Sys.Date( ) 可以返回当天的日期，date( ) 可以返回当天的日期和时间，Sys.time( ) 可以返回当天的时间。

2.3 向量

2.3.1 使用 c( ) 生成向量

向量的生成，除了上面介绍的方法之外，用 c( ) 函数也可以构建一个向量。向量不仅是 R 的一种数据结构，还是 R 软件的重要组成部分。向量中可以包含数值、字符串、逻辑值，但不能由多种格式混合组成。通过 c( ) 函数添加元素构成向量的代码如下：

```
> c(1,2,3,5,8,9)
[1] 1 2 3 5 8 9
> c("Hello","world","!")
[1] "Hello" "world" "!"
> c(TRUE,FALSE,TRUE)
[1]  TRUE FALSE  TRUE
```

c( ) 函数中的参数自身是向量，c( ) 会将多个向量合为一个向量。示例代码如下：

笔记

```
> v1 <- c(1,3,5)
> v2 <- c(2,4,6)
> c(v1,v2)
[1] 1 3 5 2 4 6
```

如果多个向量的数据类型不同，那么 R 软件对于混合型向量会进行如下的数据类型转换：

```
> v1 <- c(1,3,5)
> v2 <- c(2,4,6)
> c(v1,v2)
[1] 1 3 5 2 4 6
> v3 <- c(TRUE,FALSE,TRUE)
> v4 <- c("Hello","world","!")
> c(v1,v2)
[1] 1 3 5 2 4 6
> c(v1,v3)
[1] 1 3 5 1 0 1
> c(v1,v4)
[1] "1"          "3"          "5"          "Hello"
[5] "world" "!"
> c(v3,v4)
[1] "TRUE"   "FALSE"  "TRUE"   "Hello"    "world"    "!"
```

2.3.2 生成序列向量

（1）使用表达式 n:m 生成 n，n+1，n+2，…，m 的一个向量。示例代码如下：

```
> f <- 1:9
> f
[1] 1 2 3 4 5 6 7 8 9
> f2 <- 9:1
> f2
[1] 9 8 7 6 5 4 3 2 1
> f3 <- c(2:8)
> f3
[1] 2 3 4 5 6 7 8
```

通过以上代码可以看出，冒号运算符生成的是增量为 1 的数列向量，并且 R 软件能自动识别 9 大于 0，并以递减的形式生成数列。

（2）通过 seq(from=,to=,by=) 函数生成向量。seq( ) 函数通过第三个参数规定数列元素的增量。示例代码如下：

```
> f4 <- seq(from=0,to=10)
> f4
 [1]  0  1  2  3  4  5  6  7  8  9  10
> f5 <- seq(from=1,to=9,by=3)
> f5
[1] 1  4  7
```

还可以在 seq( ) 函数中规定输出数列的长度，R 软件会自动识别并根据要求生成等增量的数列。示例代码如下：

笔记

```
> f6 <- seq(from=0,to=100,length.out = 5)
> f6
[1]   0   25   50   75   100
```

（3）通过 rep( ) 函数生成重复某个值的数列。示例代码如下：

```
> rep(2,3)
[1] 2 2 2
```

2.3.3 向量比较运算

通过比较运算符（==、!=、<、>、<=、>=）对两个向量中的每个元素进行比较，返回结果是每个元素间比较结果的逻辑值向量。

比较运算符可以比较两个值，并根据结果返回 TRUE 或 FALSE。示例代码如下：

```
> a <- 3
> b <- 4
> a < b
[1] TRUE
> a == b
[1] FALSE
> a <= b
[1] TRUE
> a != b
[1] TRUE
> a > b
[1] FALSE
> a >= b
[1] FALSE
```

还可以对两个向量进行比较运算，即将两个向量中每两个对应的元素进行比较，并以逻辑值向量方式返回比较结果。示例代码如下：

```
> a <- c(3, 6 ,10)
> b <- c(7,4,8)
> a == b
[1] FALSE FALSE FALSE
> a > b
[1] FALSE  TRUE  TRUE
```

2.3.4 向量的数学运算

基本的数学运算符可以对向量中的元素进行逐个计算，并以向量的形式输出结果。所

笔记

有的基本数学运算符都能应用于向量对中。这些数学运算符对两个向量中相应的每个元素对进行计算。示例代码如下：

```
> v <- c(11,12,13,14,15)
> w <- c(1,2,3,4,5)
> v+w
[1] 12 14 16 18 20
> v-w
[1] 10 10 10 10 10
> v*w
[1] 11 24 39 56 75
> v/w
[1] 11.000000  6.000000  4.333333  3.500000  3.000000
> w^v
[1] 1     4096  1594323  268435456  30517578125
```

2.3.5 选取向量中的元素

从向量中选取一个或多个元素的基本方法是根据元素在向量中的位置使用方括号选出元素，如 a[2] 代表了 a 向量中的第二个元素。需要注意的是，向量第一个元素的位置索引值为 1，不是其他编程语言的 0。示例代码如下：

```
> ff <- c(1,2,3,5,8)
> ff[1]
[1] 1
> ff[4]
[1] 5
```

另外，也可以同时选择一个向量中的多个元素，向量的索引本身也可以是一个向量，并根据下标向量中所指定的位置选择向量中的元素。示例代码如下：

```
> ff[1:3]
[1] 1 2 3
> ff[c(1,3,5)]
[1] 1 3 8
```

负索引是去掉向量中相应索引的元素。例如，下标为 –1，那就选择除了第一个元素外的所有其他元素。示例代码如下：

```
> ff[-2]
[1] 1 3 5 8
```

还可以使用逻辑向量从数据中选择元素。与索引逻辑向量取值为 TRUE 的元素对应的原始数据向量的元素将被选择。示例代码如下：

```
> ff<4
[1]  TRUE  TRUE  TRUE FALSE FALSE
> ff[ff<4]
[1] 1 2 3
> ff %%2 == 0
[1] FALSE  TRUE FALSE FALSE  TRUE
> ff[ff %%2 == 0]
[1] 2 8
```

笔记

2.3.6 因子

因子（factor）在 R 中用来存储分类信息。例如，从性别上，可以把人分为男人和女人；从年龄上，又可以把人分为未成年人（<18 周岁）、成年人（≥ 18 周岁）。可以将因子看作与性别类似的概念，它只可以取某些特定的值，如男性或者女性，这些值之间可能有一些特殊的顺序规定。因子的这种特殊性非常适合记录某项研究中研究对象满足的不同处理水平或者其他类型的分类变量。

R 把表示分类的数据称为因子，因子的行为有时像字符串，有时像整数。因子是一个向量，通常，每个元素都是字符类型，也有其他数据类型的元素。因子具有因子水平（levels），用于限制因子的元素的取值范围。R 强制：因子水平是字符类型，因子的元素只能从因子水平中取值，这意味着，因子的每个元素要么是因子水平中的字符（或转换为其他数据类型），要么是缺失值，这是因子的约束，是语法上的规则。

向 factor( ) 函数传递一个原子型向量即可生成一个因子。R 会将向量中的值重新编码为一串整数值，再将编码的结果存储在一个整数向量中。此外，R 还会将一个 levels 属性和一个 class 属性添加到该整型向量，其中 levels 属性包含显示因子值的一组标签，而 class 属性包含类 factor。

1. 创建因子

可以通过 factor( ) 函数创建因子。factor( ) 函数的第一个参数必须是字符向量，通过 levels 参数显式设置因子水平，格式如下：

```
factor(x=character( ),levels,labels=levels,exclude=NA,ordered=is.ordered(x),nmax=NA)
```

参数注释如下。

x：向量，通常是有少量唯一值的字符向量。

levels ：水平，字符类型，用于设置 x 可能包含的唯一值，默认值是 x 的所有唯一值。如果 x 不是字符向量，那么使用 as.character(x) 把 x 转换为字符向量，然后获取 x 向量的水平。x 向量的取值与 levels 有关。

labels：水平的标签，字符类型，用于对水平添加标签，相当于对因子水平重命名。

exclude：排除的字符。

ordered：逻辑值，用于指定水平是否有序。

nmax：水平的上限数量。

笔记

例如，因子 sex 的值是向量 c('f','m','f','f','m')，因子水平是 c('f','m')，代码如下：

```
> sex <- factor(c('f','m','f','f','f'),levels = c('f','m'))
> sex
[1] f m f f f
Levels: f m
```

2. 查看因子水平

因子水平可以通过函数 levels(factor) 查看。示例代码如下：

```
> levels(sex)
[1] "f" "m"
```

水平的级数相当于 level 的长度，可以由 nlevels( ) 函数查询。示例代码如下：

```
> nlevels(sex)
[1] 2
```

3. 因子水平的标签

使用 factor( ) 函数创建因子，可以使用 labels 参数为每个因子水平添加标签，labels 参数的字符顺序要和 levels 参数的字符顺序保持一致。

4. 有序因子

通常，因子一般是无序的，这可以通过 is.ordered( ) 函数验证。示例代码如下：

```
> is.ordered(sex)
[1] FALSE
```

因子的顺序实际上是指因子水平的顺序，有序因子的因子水平是有序的。在特殊情况下，有些因子的水平在语义上大于或小于其他水平，R 支持按顺序排列的因子，使用 ordered( ) 函数，或通过给 factor( ) 函数传入 order=TRUE 参数，把无序因子转换为有序的因子。

（1）通过 ordered( ) 函数把现有因子转换为有序因子。ordered( ) 函数不能指定特定因子水平的顺序，通常情况下，因子中先出现的水平小于后出现的水平。

例如，通过 ordered( ) 函数把 sex 因子转换为有序的因子的代码如下：

```
> ordered(sex)
[1] f m f f m
Levels: f < m
```

（2）通过 factor( ) 函数创建有序因子，通过 levels 指定因子的顺序。示例代码如下：

```
> sex <- factor(c('f','m','f','f','m'),levels=c('f','m'),ordered=TRUE)
> sex
[1] f m f f m
Levels: f < m
```

2.4 函数

笔记

与大多数编程语言一样，R 语言编程的核心是一组指令的集合，用来读取输入、执行计算、返回结果。使用函数时，在函数名后的括号中输入相应的数据即可。传递到函数中的数据称为该函数的参数，可以有多个参数，参数之前用逗号隔开。参数可以是原始数据，R 对象甚至是另一个 R 函数的返回结果。如果存在函数嵌套，R 将从最内层的运算开始解析，直到最外层的运算为止。下面是一些 R 常用的函数。

sum( )——求和。示例代码如下：

```
> a <- c(1,2,3,4,5)
> sum(a)
[1] 15
```

mean( )——求均值。示例代码如下：

```
> mean(a)
[1] 3
```

sort( )——正排序。示例代码如下：

```
> sort(a)
[1] 1 2 3 4 5
```

sort(a,decreasing=TRUE)——逆序。示例代码如下：

```
> sort(a,decreasing = TRUE)
[1] 5 4 3 2 1
```

rev( ) ——相反顺序。示例代码如下：

```
> rev(a)
[1] 5 4 3 2 1
```

round( )——四舍五入。示例代码如下：

```
> a <- c(2.2,3.5,3.6,5.1)
> round(a)
[1] 2 4 4 5
```

另外，使用函数时，需要知道调用哪个参数，假如使用了一个该函数不能识别的参数名，就会得到一条错误的输出信息。对于一个函数，如果不能确定应该如何设置其中的参数，可以使用 args( ) 函数查看这个函数的所有参数名。具体来说，只要将函数的名称放在 args( ) 函数的括号中即可。示例代码如下：

```
> args(round)
function (x, digits = 0)
NULL
```

笔记

从上面代码可以看出，round 的 digits 参数已经设置为 0。因为它本身有一个默认值，所以是可选参数。可根据需要将值赋予这个可选参数，如果不明确赋值，该可选参数就会使用其默认值。例如，round( ) 函数会默认将数值四舍五入到小数点后的 0 位。要替代该默认值，可以将 digits 设置为其他值。示例代码如下：

```
> x <- c(1.356,5.278,3.498)
> round(x,digits = 2)
[1] 1.36 5.28 3.50
```

应当注意的是，在调用一个包含多个参数的函数时，应该写出每个参数的名称，这样有助于理解代码。再者，如果没有写出参数名称，那么 R 会按顺序将用户输入的值与函数中的参数匹配，但是用户使用的参数顺序很可能与 R 的顺序不一致，这可能导致值被传递给错误的参数。如果详细地写了参数名称，就可以防止这样的情况发生。R 会自动将用户输入的值与其参数名相匹配，而无论参数的顺序怎样。

2.5 函数文档

R 函数很多，且每个函数的功能也很多，如果要详细了解函数的全部功能，可以使用 help 命令查询函数的帮助文档；用 args( ) 函数快速获取函数的参数；用 example( ) 函数查看函数的使用示例。

例如，想了解 mean( ) 函数的全部功能，可使用 help 命令查询该函数，或者以简写的形式在函数名前面添加“?”即可。示例代码如下：

```
> help(mean)
> ?mean
```

该命令可打开函数的帮助文档，或者将函数的解释直接显示在控制台中。R 软件的函数帮助文档在最后一般会给出具体的使用例子。R 的一个很诱人的特性是，可以使用 example 命令查看函数例子的具体执行，它可以演示该函数的功能。示例代码如下：

```
> example(mean)
mean> x <- c(0:10, 50)
mean> xm <- mean(x)
mean> c(xm, mean(x, trim = 0.10))
[1] 8.75 5.50
```

2.6 缺失值

在统计数据集中，缺失值在 R 中表示为 NA，而 NULL 代表不存在的值，而不是存在但未知的值。

2.6.1 NA

在 R 的很多统计函数中，我们要求函数跳过缺失值 NA。示例代码如下：

```
> x <- c(35,NA,33,187,15)
> x
[1]  35  NA  33 187  15
> mean(x)
[1] NA
> mean(x,na.rm = T)
[1] 67.5
> x <- c(35,NULL,33,187,15)
> mean(x)
[1] 67.5
```

笔记

在第一个调用中，因为 x 中有一个缺失值 NA，导致 mean( ) 无法计算平均值。但通过把可选参数 na.rm 设置为真，可以计算其余元素的值。相比之下，R 会自动跳过空置 NULL。

下面几个 NA 值的模式都不一样：

```
> x <- c(3,NA,24)
> mode(x[1])
[1] "numeric"
> mode(x[2])
[1] "numeric"
> y <- c("ac",NA,"bd")
> mode(y[1])
[1] "character"
> mode(y[2])
[1] "character"
```

2.6.2 NULL

NULL 的一个用法是在循环中创建向量，其中每次迭代都在这个向量上增加一个元素。以下代码是建立一个偶数向量：

```
> x <- NULL
> for (i in 1:10) if (i%%2==0) x<-c(x,i)
> x
[1]  2  4  6  8 10
```

如果在上面的代码中使用 NA，而不是 NULL，则会得到多余的 NA。示例代码如下：

```
> x <-NA
> for (i in 1:10) if(i%%2==0) x<-c(x,i)
> x
[1] NA  2  4  6  8 10
```

笔记

NULL 值被作为不存在而计数，并且没有模式。示例代码如下：

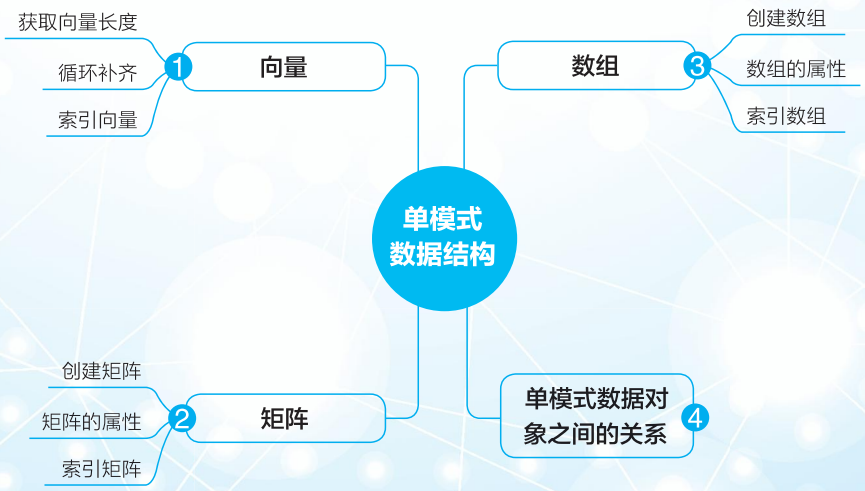
```
> x <- NULL
> length(x)
[1] 0
> y <- NA
> length(y)
[1] 1
```

第 3 章 单模式数据结构

学习目标

1. 掌握向量的属性和索引向量的概念。
2. 掌握数组的创建。
3. 掌握数组的属性。
4. 掌握索引数组的使用。
5. 掌握矩阵的属性。
6. 掌握矩阵的创建。
7. 掌握索引矩阵。

知识导图



笔记

本章导读

第 2 章已经简单介绍了向量，本章将介绍 R 语言中单模式的数据结构：向量、矩阵和数组。在 R 语言中，数组可以看成一个由递增下标表示的数据项的集合。矩阵是向量的一种特例，矩阵将数值存储在一个二维数组中，这个概念与线性代数的矩阵是一样的。

3.1 向量

第 2 章已经介绍了向量的创建方法和向量的部分属性，本章重点介绍索引向量。

3.1.1 获取向量长度

使用函数 length( ) 可获取向量的长度。尤其是在写一般函数的代码时，经常需要知道向量参数的长度。示例代码如下：

```
> x <- c(1,5,7)
> length(x)
[1] 3
```

3.1.2 循环补齐

在对两个向量使用运算符时，如果要求这两个向量具有相同的长度，R 会自动循环补齐，即重复较短的向量，直到它与另一个向量长度相匹配。例子中较短的向量被循环补齐，因此运算其实是 c(1,3,5,1,3)+c(6,0,9,23,44) 这样执行的。示例代码如下：

```
> c(1,3,5)+c(6,0,9,23,44)
[1]  7  3 14 24 47
Warning message:
In c(1, 3, 5) + c(6, 0, 9, 23, 44) : 长的对象长度不是短的对象长度的整倍数
```

3.1.3 索引向量

R 中最重要、最常用的一个运算符是索引，通常使用它来选择给定向量中特定索引的元素，构成子向量。索引向量的格式是向量 1[向量 2]，它返回的结果是，向量 1 中索引为向量 2 的那些元素。负数的下标代表可以把相应元素剔除。索引向量有四种类型。

1. 逻辑的向量

在这种情况下，索引向量必须与从中选取元素的向量具有相同的长度。在索引向量中，返回值是 TRUE 的元素所对应的元素将被选出，返回值为 FALSE 的元素所对应的元素将被忽略。示例代码如下：

```
> x <-c(2,NA,3)
> y <-x[!is.na(x)]
> y
[1]2 3
```

以上代码创建了一个名为 y 的对象，对象中包含 x 中的非缺失值，同时保持顺序。如果 x 中包含缺失值，y 的长度将小于 x。

笔记

2. 正整数的向量

这种情况下索引向量中的值必须在集合 {1,2,…,length(x)} 中，在返回的向量中包含索引向量中指定的元素，并且在结果中按照索引向量中的顺序排列。索引向量的长度可以任意，返回的向量与索引向量有相同的长度。例如，x[3] 是 x 的第 3 个元素。示例代码如下：

```
> x <-c(2,4,6,8)
> x[c(1,3)]
[1] 2 6
> x[2:3]
[1] 4 6
> v <–3:4
> x[v]
[1]6 8
> x[–1]
[1] 4 6 8
```

3. 负整数的向量

这种索引向量的作用是把某些值去掉。例如，>y<-x[-(1:5)] 表示向量 y 取得前 5 个元素以外的值。

4. 字符串的向量

这种可能性只存在于拥有 names 属性并由它区分向量中元素的向量。这种情况下，一个由名称组成的子向量收到了与正整数的索引向量相同的效果。示例代码如下：

```
> fruit <- c(5,10,1,20)
> names(fruit) <- c("orange","banana","apple","peach")
> lunch <- fruit[c("apple","orange")]
```

由字母和数字组成的名称比单纯的数值索引更好记是它最大的优点。这个功能在使用数据框的时候非常好用。

3.2 矩阵

在计算机处理时，把矩阵当作一种特殊的向量，包含两个属性：行数和列数。所以，矩阵也和向量一样，有模式的概念，例如数值型和字符型。

3.2.1 创建矩阵

矩阵的行和列的下标都是从 1 开始的，矩阵在 R 中是按列存储的，也就是说，先存储第一列，再存储第二列，以此类推。

创建矩阵的方法之一就是使用 matrix( ) 函数。示例代码如下：

笔记

```
> x <- matrix(c(1,2,3,4,5,6),nrow = 2,ncol = 3)
> x
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

这里把第一列（即 1 和 2）与第二列（3 和 4）以及第三列（5 和 6）连接在一起。因此数据是（1，2，3，4，5，6）。由于 R 是按列存储的，这就决定了这六个数在矩阵中的位置。如果指定了矩阵中的全部元素，那么列数 ncol 和行数 nrow 这两个参数只需给出其中一个就够了。

另外一种创建矩阵的方法是为矩阵的每一个元素赋值：先声明一个矩阵，并且给出行数和列数，然后再为矩阵的每一个元素赋值。示例代码如下：

```
> y <- matrix(nrow = 2,ncol = 3)
> y[1,1] <-1
> y[2,1] <-2
> y[1,2] <-3
> y[2,2] <-4
> y[1,3] <-5
> y[2,3] <-6
> y
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

3.2.2 矩阵的属性

1. 矩阵运算

矩阵 A 的转置：t（A）。

矩阵 A 的求逆：solve（A）。

矩阵 A 与矩阵 B 相乘：A%*%B。A*B 是逐个元素的乘法运算，然而 A%*%B 是矩阵乘法。所有这些函数都返回一个矩阵。

一个 n 阶对角（单位）矩阵：diag（n）。

2. 给矩阵的行和列赋名字

在 R 中可以把名称赋值给一个矩阵的行和列，这可以增强矩阵的可读性。每个矩阵都有一个 rownames 属性和一个 colnames 属性。把字符串向量赋值给矩阵的相应属性的语法如下：

```
> rownames(mat)<- c("rowname","rowname",...,"rowname")
> colnames(mat)<- c("colname","colname",...,"colname")
```

给矩阵 x 的行和列赋名字。示例代码如下：

```
> colnames(x) <- c("num","name","score")
> rownames(x) <- c("zhangsan","lisi")
> x
         num name score
zhangsan   1    3     5
lisi       2    4     6
```

3.2.3 索引矩阵

笔记

索引向量的运算方法也同样适用于矩阵。示例代码如下：

```
> y
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> y[,2:3]
     [,1] [,2]
[1,]    3    5
[2,]    4    6
```

这里提取矩阵 y 中第 2、3 列的所有元素组成了一个子矩阵。如下所示：

```
> y[1,]
[1] 1 3 5
```

上面的例子提取了矩阵 y 的一行。

向量的负值索引用来排除某些元素，这种操作也同样适用于矩阵。示例代码如下：

```
> y
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> y[,-2]
     [,1] [,2]
[1,]    1    5
[2,]    2    6
```

3.3 数组

在 R 语言中的数组和其他语言中的数组含义不太一样。向量的概念更像其他编程语言中的数组，在 R 语言中的数组更像多维数组。

3.3.1 创建数组

1. 设置 dim 属性生成数组

将向量转换成一个 n 维数组的方法是：用 dim( ) 函数将相应的维度属性赋给该向量。具体来说，就是把这个向量的 dim 属性设置为 n。R 会将该向量中的元素重新排列到 n 维。例如，可以将向量 m 重新组织成一个数组。示例代码如下：

```
> m <- c(1:20)
> m
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
[19] 19 20
> dim(m) <- c(2,2,5)
```

笔记

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

这样的数据结构是 5 维的，R 始终将第一个值赋给行数，将第二个值赋给列数。总的来说，R 对行列进行维度操作时，行的优先权要高于列。

2. array( ) 函数生成数组

array( ) 函数用来生成一个 n 维数组。比如，可以利用 array( ) 函数将数值排列到一个三维立方体空间中，或者一个四维、五维或 n 维的超立方体空间中。在 R 语言中，数组使用 array( ) 函数创建。array( ) 函数的语法格式如下所示：

```
array(data=NA,dim=length(data),dimnames=NULL)
```

其中，data 为创建数组的元素；dim 为数组的维数，是数值型向量；dimnames 是各维度中的名称标签列表。

示例代码如下：

```
arr1 <- array(1:10) # 相当于一维向量：1 2 3 4 5 6 7 8 9 10
arr2 <- array(1:10, dim=c(2,5)) # 相当于 2 行 5 列的矩阵
arr3 <- array(1:24, dim=c(3,4,2)) # 创建了一个 3 × 4 × 2 维的数组
```

3.3.2 数组的属性

在创建数组时可以给数组的每一维的每个水平取名字。示例代码如下：

```
> dim1<-c("A1","A2","A3")
> dim2<-c("B1","B2","B3")
> dim3<-c("C1","C2")
> arr4<-array(1:18, dim=c(3,3,2), dimnames = list(dim1, dim2, dim3))
```

3.3.3 索引数组

笔记

数组的元素可以通过中括号指定下标存取，各个下标用逗点隔开。示例代码如下：

```
arr4[2,3,1] # 获取单个元素的值：8
arr4[2,1,] # 获取第 1 个维度的第 2 个水平和第 2 个维度的第 1 个水平的所有元素值
arr4[2,,] # 获取第 1 个维度的第 2 个水平的所有组合元素值
arr4[,2,] # 获取第 2 个维度的第 2 个水平的所有组合元素值
arr4["A2","B3","C2"] # 通过水平名称的组合获取元素值
```

修改数组中元素的值。示例代码如下：

```
arr4[2,3,1] <- 18
arr4[2,3,] <- c(110,111)
```

除此之外，也可以利用索引向量取得指定的子数组。示例代码如下：

```
1 arr <- array(1:25, c(5, 5))
2 subarr <- arr[1:3, 2:4]
```

数组 arr 为

```
[,1] [,2] [,3] [,4] [,5]
[1,]   1   6   11  16  21
[2,]   2   7   12  17  22
[3,]   3   8   13  18  23
[4,]   4   9   14  19  24
[5,]   5   10  15  20  25
```

而子数组 subarr 为

```
[,1] [,2] [,3]
[1,]   6  11  16
[2,]   7  12  17
[3,]   8  13  18
```

若不指定下标，则 R 会选取对应下标的所有元素，例如：

```
1 arr <- array(1:25, c(5, 5))
2 subarr2 <- arr[1:3, ]
```

子数组 subarr2 为

```
   [,1] [,2] [,3] [,4] [,5]
[1,]   1   6  11  16  21
[2,]   2   7  12  17  22
[3,]   3   8  13  18  23
```

若所有下标都不指定 arr[,,]，则表示整个数组，这与忽略下标直接使用 arr 效果是一样的。

如果一个多维数组只给了一个下标或索引向量，在这种情况下，R 会把数组当作向量使用，而忽略维度属性。示例代码如下：

笔记

```
1 arr <- array(1:8, c(2, 4))
2 arr[3] <- 10
```

此时 arr 为

```
   [,1] [,2] [,3] [,4]
[1,]   1  10   5   7
[2,]   2   4   6   8
```

数组也可以使用索引数组的方式存取任意不规则元素集合，这种方式直接使用范例说明比较清楚，首先假设有一个 4 × 5 的二维数组：

```
x <- array(1:20, dim = c(4, 5))
```

x 为

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

可以使用索引数组的方式将 x[1，3]、x[2，2] 与 x[3，1] 取出来构成一组新的向量，首先产生一个索引数组 i：

```
i <- array(c(1:3, 3:1), dim = c(3, 2))
```

数组 i 为

```
   [,1] [,2]
[1,]   1   3
[2,]   2   2
[3,]   3   1
```

索引数组的行（row）数必须等于被存取数组的维度，其列数则没有限制，每一列（column）指定一个元素，一列中的各个数字就是对应元素的下标。以索引数组取出指定的元素：

y <- x[i]

新的向量 y 为

[1] 9 6 3

除了取出元素，索引数组也可以改变数组中指定元素的值：

```
x[i] <- c(30, 31, 32)
```

此时 x 变为

```
   [,1] [,2] [,3] [,4] [,5]
[1,]   1   5  30  13  17
[2,]   2  31  10  14  18
[3,]  32   7  11  15  19
[4,]   4   8  12  16  20
```

3.4 单模式数据对象之间的关系

笔记

矩阵包含两个附加的属性：行数和列数。数组是 R 里更一般的对象，矩阵是数组的一个特殊情形。数组可以是多维的。例如，一个三维数组可以包含行、列和层，而矩阵只有行和列两个维度。

向量、矩阵和数组都只能存储单一类型的数据，当数据中存在大量数值时，计算任务就会更加简单。这是因为 R 知道这些数据对象中只有一种数据类型，因此在数据操作时更加直接、高效。