第 1 章

R 语言简介

R 语言作为一种功能强大的开源编程语言和环境，已经成为数据分析、统计建模和可视化等领域的重要工具。它的灵活性、可扩展性和丰富的功能使得越来越多的数据科学家、统计学家和研究人员选择使用 R 语言来处理和分析数据。本章旨在帮助读者快速掌握 R 语言的基本概念及环境搭建。

1.1 R 语言概述

R 是一种用于统计分析和绘图的语言和操作环境。作为一个用于统计计算和制图的优秀工具，R 语言具有免费且源代码开放等特点，在数据科学领域拥有广泛用户。本节首先介绍 R 语言的特点及其在医学数据分析中的应用特点。

1.1.1 R 语言的诞生

R 语言诞生于 1980 年左右的 S 语言的一个分支，可以认为 R 是 S 语言的一种实现，被广泛应用于统计领域。S 语言是由 AT&T 贝尔实验室开发的一种用来进行数据探索、统计分析和作图的解释型语言。最初 S 语言的实现版本主要是 S-PLUS。

S-PLUS 是一个商业软件，它基于 S 语言，并由 MathSoft 公司的统计科学部进一步完善。后来，新西兰奥克兰大学的 Robert Gentleman 和 Ross Ihaka 及其他志愿人员开发了 R 系统。

R 可以看作贝尔实验室（AT&T BellLaboratories）的 Rick Becker、John Chambers 和 Allan Wilks 开发的 S 语言的一种实现。因此，两者在程序语法上可以说是几乎一样的，可能只是在函数方面有细微差别。程序可以十分容易地移植到 S 程序中，而很多 S 程序只要稍加修改也能运用于 R。

1.1.2 R 语言的特点

R 语言在统计分析、数据科学和数据可视化领域具有广泛的应用和高度的认可。该语言具有以下特点：

（1）开源和免费：R 语言是开源软件，任何人都可以免费使用和修改。这促进了社区的协作和共享，使得 R 语言拥有丰富的包和资源。

（2）强大的统计分析功能：R 语言内置了丰富的统计分析功能，从基础统计到复杂的模型，都有相应的函数和包支持。

（3）灵活的图形功能：R 语言提供了强大的数据可视化功能，通过基础绘图函数和扩展包（如 ggplot2），用户可以创建各种高质量的图形和图表。

（4）丰富的扩展包：CRAN（R 语言的官方软件包存储库）上有数以千计的扩展包，这些包涵盖数据处理、机器学习、生物信息学、金融分析等各个领域。

（5）良好的社区支持：R 语言拥有一个活跃的用户和开发者社区，提供了大量的在线资源、教程和支持。

（6）与其他语言的集成：R 语言可以与其他编程语言（如 C、C++、Java、Python 等）进行集成，灵活性高。

（7）数据处理和清洗：R 语言提供了强大的数据处理和清洗功能，特别是通过 tidyverse 系列包（如 dplyr、tidyr），可以高效地进行数据操作。

1.1.3 R 语言绘图系统

在用 R 语言绘图时，首先会使用由 grDevices 包提供的一系列基本绘图函数，如颜色、字体和图形输出格式等。在 grDevices 包的基础上有多种绘图选择。

一般来说，R 语言有传统绘图系统和网格绘图系统两种主要的绘图系统。这两种绘图系统相互独立，以不同的方式进行绘图。这两种绘图系统对应 R 语言核心包的 graphics 包和 grid 包。

（1）graphics 包是 R 语言的内置绘图包，且每次启动 R 语言都会自动加载。它可以生成多种类型的图表，并且提供了许多美化图形细节的函数。

（2）grid 包则提供了一系列不同的绘图函数。grid 包并没有提供一套完整的绘图函数，通常不能直接用于绘图。因此，在 grid 包的基础上又发展出了 lattice 和 ggplot2 两个应用广泛的程序包。

其中，lattice 包由 D.Sarkar 根据 Cleveland 的格子图发展而来，ggplot2 包由 H.Wickham 根据 L.Wilkson 的图形语法发展而来。

上述两个绘图系统还衍生出了许多其他的绘图工具。例如，搭载于传统绘图系统之上的 maps、diagram、plotrix、gplots 和 poxmap 等扩展包，以及搭载于网格绘图系统之上的 vcd 和 grImport 扩展包等。另外，还有一些扩展包提供了 R 语言与第三方绘图系统的接口。

1.1.4 R 语言与医学数据分析

医学领域的数据日益呈现出大规模、高维度、多样化的特点。如何高效地分析和可视化这些数据，是当前医学研究中的重要挑战。R 语言作为一种功能强大、灵活性高的编程语言，已广泛应用于医学数据分析和可视化中。

1. 临床研究与统计分析

R 语言在临床试验和流行病学研究中的广泛应用得益于其强大的统计分析功能。它可以完成线性回归、逻辑回归、生存分析和贝叶斯模型等常见统计任务。例如：

（1）使用 survival 包进行生存曲线分析和风险模型构建。

（2）利用 meta 包进行荟萃分析，评估医学干预效果的总体趋势。

2. 流行病学建模

在流行病学研究中，R 语言被广泛用于疾病传播建模和风险评估。例如：

（1）使用 EpiModel 包模拟传染病的传播动态。

（2）借助 sf 和 tmap 包分析疾病地理分布并绘制地图。

3．基因组学与转录组学

基因组学与转录组学数据分析是 R 语言的优势领域之一。通过 Bioconductor 平台，研究者可以轻松处理和分析高维组学数据。例如：

（1）使用 DESeq2 和 edgeR 进行差异表达分析。

（2）借助 ComplexHeatmap 绘制复杂的基因热图。

（3）使用 clusterProfiler 进行基因功能富集分析。

4．机器学习与预测建模

R 语言通过 caret、randomForest 和 xgboost 等包，为医学数据提供机器学习的支持。研究者可以利用 R 构建诊断模型、预测疾病风险或优化个体化治疗方案。例如，开发基于患者基因特征和病史的预测模型，用于临床决策支持。

5．影像数据分析

R 语言还开始涉足医学影像分析，借助 ANTsR 和 oro.dicom 包，研究者可以处理 DICOM 格式的医学影像数据，提取感兴趣的影像特征，并结合其他患者数据进行综合分析。

1.1.5 R 语言与医学数据可视化

R 语言因其灵活性和扩展性，已经成为医学数据可视化领域的重要工具。无论是数据分布与趋势的基本展示，还是更高级的生存分析、基因组学可视化以及地理分布图的绘制，R 语言都能提供丰富的解决方案。通过这些可视化方法，研究者可以更深入地挖掘数据中的模式和规律，为医学研究提供支持，也为临床实践提供直观且易于理解的参考依据。

1．数据分布与趋势可视化

R 语言的强大数据可视化能力，使研究者能够以图形的方式直观展示和探索数据的分布与趋势。在医学研究中，这种能力尤为重要，因为医疗数据通常具有复杂的多维度特性，借助可视化工具，研究者可以快速发现数据的内在规律。常用工具如下：

（1）ggplot2：R 中最广泛使用的可视化工具之一，基于“图层语法”的设计，支持高度自定义的静态图表制作。研究者可以绘制直方图、箱线图、散点图等，观察患者人口学特征，如年龄分布、性别比例等。

（2）plotly：提供动态交互式图表功能，尤其适合对数据趋势的动态探索。例如，研究疾病发展过程中，可以绘制交互式的时间序列图以观察随访患者某项生物标志物的变化趋势。

例如，通过 ggplot2 绘制患者的年龄分布直方图，或用 plotly 生成动态折线图展示不同治疗方案对血糖水平的影响。

医学研究中的典型应用场景如下。

（1）患者人口学分析：通过条形图或箱线图直观展示患者的性别、年龄或 BMI 等基本特征的分布。

（2）时间趋势分析：在流行病学研究中，使用折线图或面积图跟踪疾病的年度或月度发病率变化趋势。

（3）多维数据探索：散点图结合颜色或尺寸的编码，直观呈现患者特征之间的关系，例如年龄、BMI 和疾病发生风险的关联。

2．生存分析图形化展示

生存分析是医学研究中的一个重要领域，用于研究患者从某个起始事件（如确诊）到某个终止事件（如死亡）的时间分布。R 语言中的 survminer 包为生存分析的图形化提供了强有力的支持。

Kaplan-Meier 生存曲线可以直观反映特定患者群体在某时间段内的生存概率变化，曲线之间的显著差异提示潜在影响因素。例如，研究化疗方案对乳腺癌患者 5 年生存率的影响。

Kaplan-Meier 曲线是一种直观展示生存概率随时间变化的常用图形。通过 survminer 包，研究者可以实现：

（1）绘制生存曲线，显示不同组别（如不同治疗方案或年龄组）的生存概率差异。

（2）添加对数秩检验（Log-rank Test）的 P 值，展示组间统计学显著性。

（3）优化曲线标注，如在关键时间点添加生存概率。

典型应用场景如下。

（1）治疗效果比较：展示不同治疗方案（如药物 A 与药物 B）对患者生存期的影响。

（2）危险因素分析：将患者按某项指标（如是否患有糖尿病）分组，绘制对应的生存曲线以评估该因素的影响。

3．基因表达热图与维恩图

在基因组学和转录组学研究中，可视化工具的使用是数据解释的重要环节。R 语言中提供了多种工具包，如 ComplexHeatmap 和 VennDiagram，用于展示基因表达数据和数据集之间的关系。

（1）ComplexHeatmap 是一个功能强大的热图工具包，支持高维数据的精细可视化和复杂注释。通过绘制热图展示差异表达基因（DEGs）在不同样本中的表达水平。热图的颜色渐变可以清晰地反映基因表达量的高低，为研究者提供直观的视角。例如，在癌症研究中，绘制肿瘤组织与正常组织中基因表达的热图，观察基因表达的模式差异。

（2）VennDiagram 包可以绘制多个数据集之间的交集与并集关系，即韦恩图。在基因组数据分析中，维恩图可以展示多个实验条件下筛选出的差异基因的交集。例如，研究在不同药物处理条件下共有的和特异性表达的基因。

总之，R 语言作为一种强大的数据分析和可视化工具，其应用并不仅限于传统的生存分析、基因组学可视化或疾病地图等领域。事实上，随着医学研究的多样化和工具包生态的不断扩展，研究者可以根据自己的需求找到适合特定领域的可视化工具和方法。

1.2 搭建 R 语言环境

R 语言可以在 CRAN（Comprehensive R Archive Network）网站免费下载。CRAN 是一个包含发布版本、资源包、文档和源代码的网络集合，由几十个镜像网站组成，提供下载安装程序及相应版本的资源包。镜像更新频率一般为 1~2 天。

1.2.1 安装程序下载

单击左侧 CRAN 下的 Mirrors 链接进入镜像列表。CRAN 针对 Windows、Mac OS 和 Linux 等系统平台有编译好的相应二进制安装包，根据自己的系统平台选择下载安装即可。下面以 Windows 平台为例，向读者介绍 R 语言的下载与安装步骤。

步骤 01 在 IE 浏览器中输入网址（https://www.r-project.org/），按 Enter 键后进入 R 语言官网，如图 1-1 所示。

步骤 02 在主页中单击左侧 Download 下的 CRAN，或者单击右侧的 download R 超链接，进入 CRAN Mirrors 页面。镜像是按照国家或地区进行分组的，在左侧找到 China，选择其中的一个镜像（推荐选用清华大学镜像）单击，如图 1-2 所示。

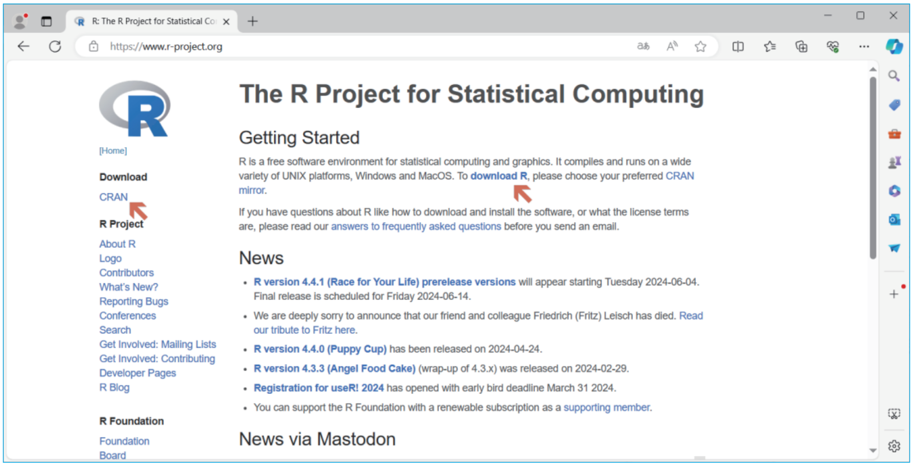
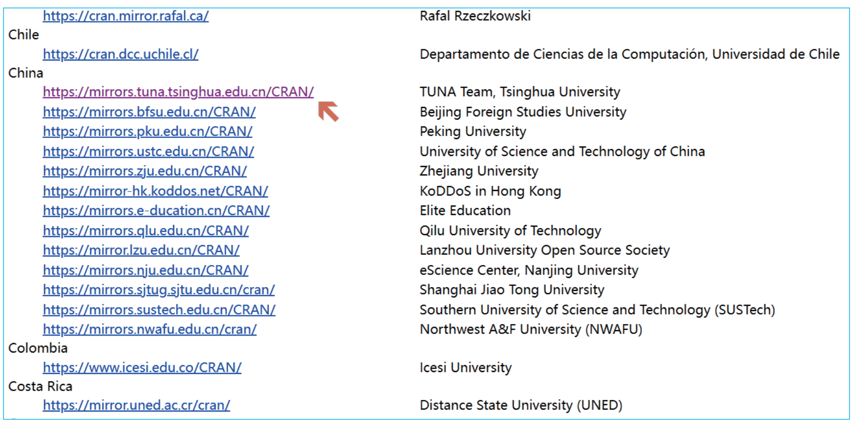


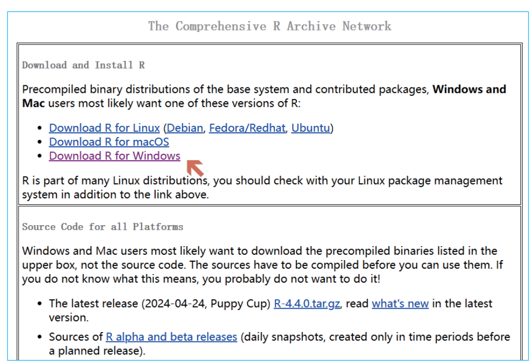
图 1-1　R 语言官网



https://cran.mirror.rafal.ca/	Rafal Rzeczkowski
Chile	
https://cran.dcc.uchile.cl/	Departamento de Ciencias de la Computación, Universidad de Chile
China	
https://mirrors.tuna.tsinghua.edu.cn/CRAN/	TUNA Team, Tsinghua University
https://mirrors.bfsu.edu.cn/CRAN/	Beijing Foreign Studies University
https://mirrors.pku.edu.cn/CRAN/	Peking University
https://mirrors.ustc.edu.cn/CRAN/	University of Science and Technology of China
https://mirrors.zju.edu.cn/CRAN/	Zhejiang University
https://mirror-hk.koddos.net/CRAN/	KoDDoS in Hong Kong
https://mirrors.e-ducation.cn/CRAN/	Elite Education
https://mirrors.qlu.edu.cn/CRAN/	Qilu University of Technology
https://mirror.lzu.edu.cn/CRAN/	Lanzhou University Open Source Society
https://mirrors.nju.edu.cn/CRAN/	eScience Center, Nanjing University
https://mirrors.sjtug.sjtu.edu.cn/cran/	Shanghai Jiao Tong University
https://mirrors.sustech.edu.cn/CRAN/	Southern University of Science and Technology (SUSTech)
https://mirrors.nwafu.edu.cn/cran/	Northwest A&F University (NWAFU)
Colombia	
https://www.icesi.edu.co/CRAN/	Icesi University
Costa Rica	
https://mirror.uned.ac.cr/cran/	Distance State University (UNED)

图 1-2　选择镜像站点

步骤 03 在出现的界面中根据自己的操作系统选择适应的版本，本书为 Windows 平台，因此单击 Download R for Windows 链接，如图 1-3 所示。



The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2024-04-24, Puppy Cup) R-4.4.0.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).

图 1-3　选择适应的平台版本

步骤04 在弹出的页面中单击 base 或 install R for the first time 链接，如图 1-4 所示。继续在弹出的下一页面中单击 Download R-4.4.0 for Windows 链接，如图 1-5 所示，即可将安装文件下载到本地计算机。

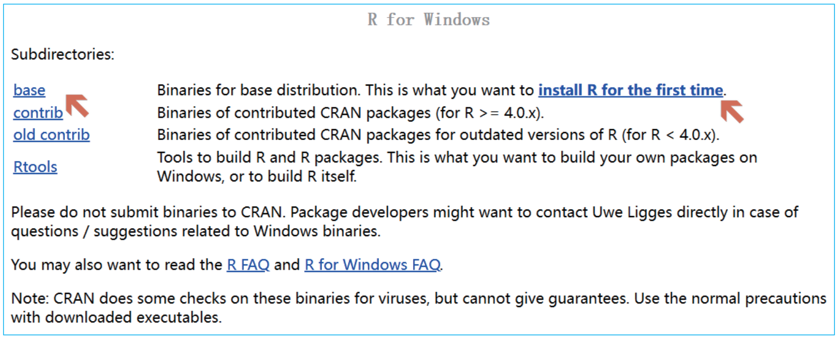


图 1-4 选择下载版本

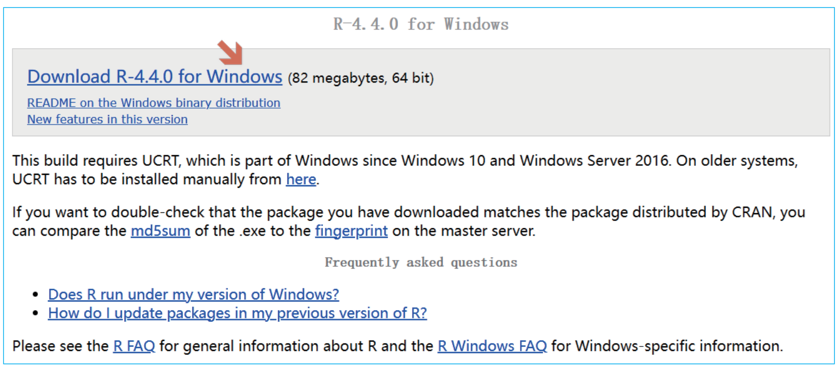


图 1-5 下载链接

1.2.2 R 语言安装与启动

步骤01 在刚下载完成的安装包 R-4.4.0-win 上双击，或者右击，在弹出的快捷菜单中执行“以管理员身份运行”命令。

步骤02 在弹出“选择语言”对话框中默认选择“中文（简体）”，单击“确定”按钮进入安装设置过程，依次单击“下一步”按钮即可，无须额外设置。

步骤03 安装完成后，会在桌面上出现快捷启动方式按钮，双击该按钮即可启动 R 语言，首次启动后的 RGui 界面如图 1-6 所示。能够正常启动说明安装成功。

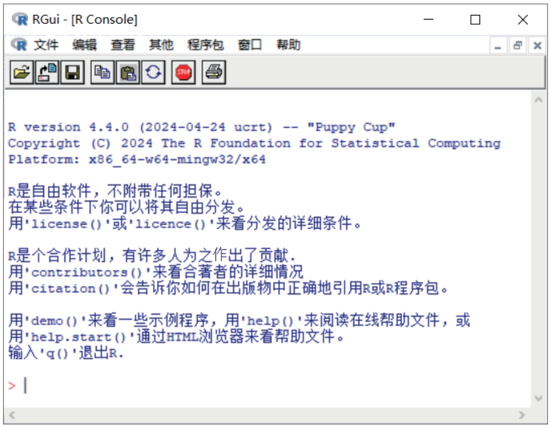


图 1-6 R 语言主界面（RGui 界面）

说明 在 Windows 平台中安装 R 语言时，除了安装必要的核心文件外，还会安装一个名为 Rgui.exe 的可执行文件，该程序文件位于 C:\Program Files\R\R-4.4.0\bin\x64（默认安装）下。双击该文件，即可进入 R 语言自带的 GUI 界面，即 R 语言主界面。

RGui 界面的上方为主菜单栏和快捷工具按钮。下方为 R 语言运行的控制台（R Console），R 语言运行的输入和输出均在此操作。

R 的所有分析和绘图均由 R 命令实现，使用时在提示符“>”后输入命令代码，每次可以输入一条命令，也可以连续输入多条命令，命令之间用分号“;”隔开，命令输入完成后，按 Enter 键，R 就会运行该命令并输出相应的结果。

【例 1-1】在控制台进行命令的输入示例。

在控制台中依次输入以下代码，并显示输出结果。

```
> 3+9                  # 在提示符后输入命令，按 Enter 键
[1] 12                 # 显示的输出结果，[1] 表示输出结果的第一行
```

如果要输入的数据超过一行时，可以在适当的地方按 Enter 键，并在下一行继续输入，R 会在断行的地方用“+”表示连接。例如，计算 3+9+33+96+34+45+56+45-33-21 的值，分 3 行输入，控制台上的显示为：

```
> 3+8+34+98+           # 此处最后的 "+" 表示为输入完成，后续还有输入，类似于续行符
+ 34+45+56+45-         # 此处最前面的 "+" 表示接上一行输入
+ 33-21
[1] 267
>
```

1.2.3 辅助工具 RStudio

R 语言自带的 RGui 操作界面相对简单，伴随着 R 语言的广泛应用，众多的 R 语言辅助工具应运而生。其中最具代表性的为 RStudio 公司的 RStudio 套件及微软的 Visual Studio R 套件。

1. RStudio 下载与安装

步骤01 在 IE 浏览器中输入网址（https://posit.co），按 Enter 键后进入 RStudio 官网。在页面右上方找到并单击 DOWNLOAD RSTUDIO 按钮，下载软件。

说明 作者使用的当前版本为 RStudio-2024.04.1-748。

步骤02 在刚下载完成的安装包 RStudio-2024.04.1-748 上双击，或者右击，在弹出的快捷菜单中执行“以管理员身份运行”命令。

步骤03 在弹出的“Rstudio 安装”对话框中单击“下一步”按钮，进入安装设置过程，随后依次单击“下一步”按钮即可，无须额外设置。

步骤04 安装完成后，会在 Windows 系统“开始”菜单栏中出现 RStudio 快捷启动方式按钮 rstudio ，单击该按钮即可启动 RStudio，首次启动后的 RGui 界面如图 1-7 所示。能够正常启动说明安装成功。

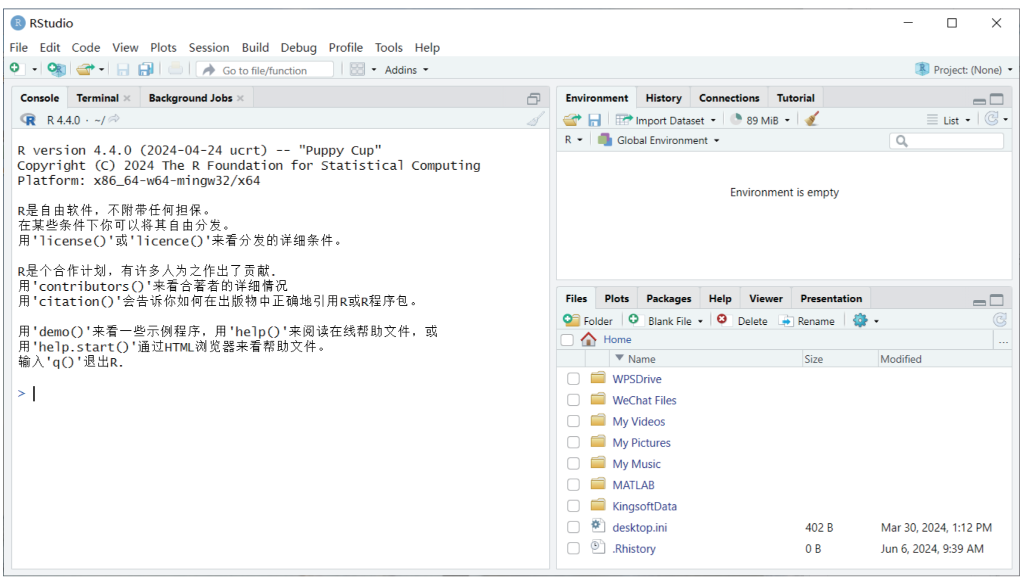


图 1-7 RStudio 主界面

2. RStudio 主界面介绍

执行菜单栏中的 File → New File → RScript 命令，或单击左上角的（新建）按钮，在弹出的菜单中执行 R Script 命令，在窗口的左上方即可出现脚本编辑区，如图 1-8 所示。

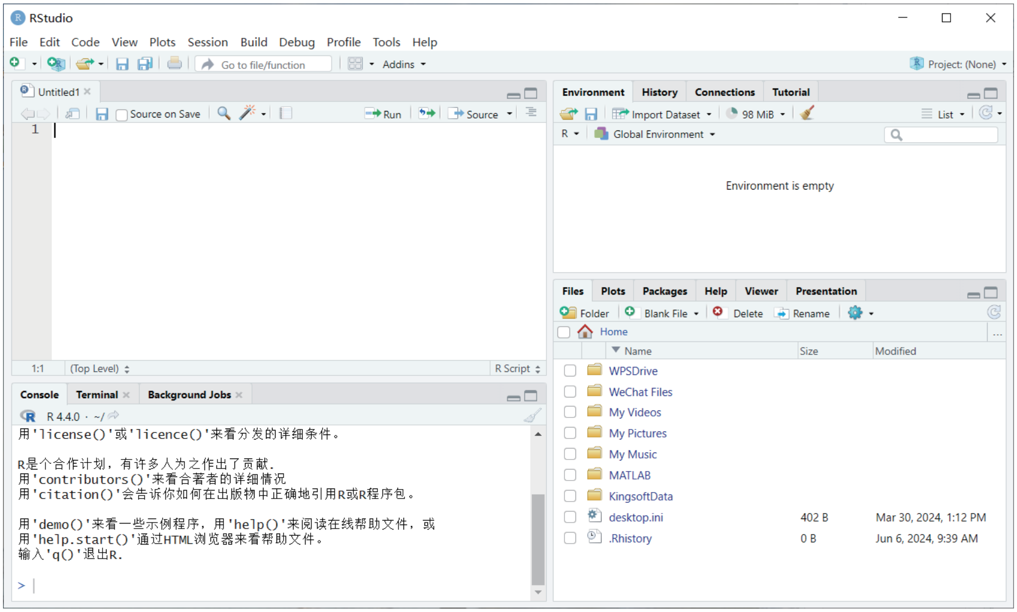


图 1-8　打开脚本窗口的主界面

默认打开的操作界面最上方区域为RStudio的菜单栏和快捷工具栏，该区域主要有File（文件）、Edit（编辑）、Tools（工具）、帮助（Help）等菜单，该区域在保存文件、发布程序及结果、安装包时使用。下方工作区被划分为 4 个子区域。

（1）左上方可以称为代码编写区（代码编辑器），可以编写 R 脚本、Rmd 文档、R Notebook 等不同类型的文件，并且可以进行程序运行和调试等操作。该区上方还有文件保存、查找、运行等快捷方式。例如，编辑程序脚本完成后，单击 Run 按钮即可运行该脚本。

（2）左下方为运行结果输出区域（控制台），该区域既可以输入并执行命令，查看命令行的运行结果，也可以输出程序脚本运行结果。这里包含所有运行过的命令，方便对历史记录的检查。

（3）右上方为当前工作空间相关信息，可显示当前工作环境（Environment）加载的 R 语言程序包、R 语言对象（列表、因子、数据框、矩阵、向量等），也可查看 R 语言运行的历史（History）等信息。

（4）右下方为当前用户工作目录和 R 语言程序包的相关信息，包括环境、文件、绘图、包、帮助、查看等选项卡窗口。可以查看当前工作目录下的文件、已安装的 R 语言程序包，单击 Packages 选项卡下的 Install 和 Update 按钮可分别安装和更新 R 语言包，在该区域还可以查看当前绘图和输出、查找 R 函数帮助等。

3. 主界面设置

RStudio 支持自定义界面布局，执行菜单栏中的 Tools → Global Options 命令，在弹出的 Options 对话框中选择 Pane Layout 选项，即可根据自己的喜好进行界面窗口的设置，如图 1-9 所示。

另外，在 Appearance 选项组下可以进行界面字体等的设计，在 Packages 下可以进行镜像地址设置，在国内可以设置为China (Beijing 1)，以提高下载速度。

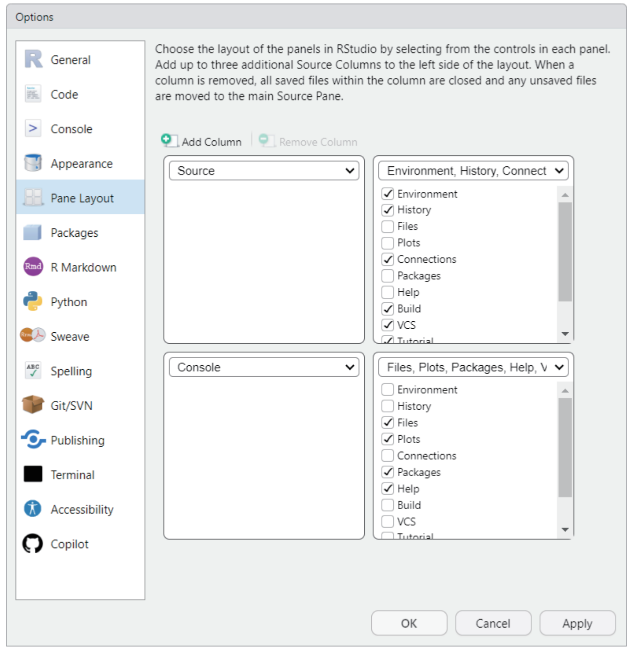


图 1-9 Options 对话框

1.2.4 包的安装与加载

R 语言中的包（Package）是指包含 R 数据集、函数等信息的集合。大部分统计分析和绘图都可以使用已有的 R 包来实现。R 语言还拥有功能强大的第三方包，如 ggplot2 等，第三方包需要先下载安装后方可使用。

一个 R 包中可能包含多个函数来进行多种分析和绘图，而对于同一问题的分析和绘图，也可以使用不同的包来实现，通常是根据个人的需要和偏好来选择所需要的包。

1. 查看已安装的包

安装 R 时，默认自带了一系列包（如 base、datasets、graphics、stats、utils、grDevices、methods 等），这些包提供了种类繁多的默认函数和数据集，分析时无须加载即可直接使用包中的函数。其他包则需要事先安装并加载后才能使用。

查看 R 中已经安装过的包时，可以使用 library() 或 .packages(all.available=TRUE) 函数。

【例 1-2】查看已安装的包。

在控制台中依次输入以下代码，并显示输出结果。

```
> library()                                     # 在新窗口列出包的名称
> .packages(all.available=TRUE)                 # 在命令窗口列出包的名称
[1]  "base"        "boot"         "class"        "cluster"       "codetools"
[6]  "compiler"    "datasets"     "foreign"      "graphics"      "grDevices"
[11] "grid"        "KernSmooth"   "lattice"      "MASS"          "Matrix"
[16] "methods"     "mgcv"         "nlme"         "nnet"          "parallel"
[21] "rpart"       "spatial"      "splines"      "stats"         "stats4"
[26] "survival"    "tcltk"        "tools"        "translations"  "utils"
```

使用 help() 函数可以在 R 官网上查阅包的功能简介，其语法格式为：

```
help(package=p_name)                            # p_name 为包的名称
```

2. 使用函数安装包

通常 R 包都来自 CRAN，在使用 R 时，可根据需要随时在线安装所需的包，选择相应的镜像站点即可完成包的下载和安装。读者可以一次性下载安装多个包，下载时将多个带引号的包名称用逗号隔开即可。下载安装包的语法格式为：

```
install.packages("p_name")                   # 包的名称 p_name 必须使用双引号引起来
install.packages("p_name1","p_name2",...,"p_namen")          # 一次安装多个包
```

【例 1-3】安装 ggplot2 和 ggraph 两个包。

在控制台中依次输入以下代码：

```
> install.packages("ggplot2")                   # 安装 ggplot2 包，安装一次即可
> install.packages(c("ggplot2","ggraph"))       # 同时安装 ggplot2 和 ggraph 两个包
```

3. 使用 RStudio 安装包

执行菜单栏中的 Tools → Install Packages 命令，在弹出的 Install Packages 对话框中输入想要安装的包，然后单击 Install 按钮，系统将会自动安装指定的包和相关依赖包，如图 1-10 所示。

当需要一次性下载安装多个包时，需要在下载第三方包的界面框内输入多个包名称，并以逗号或空格隔开。

> **说明** 读者也可以在主窗口右下方选择 Packages 选项卡，然后单击 Install 按钮安装所需要的包。

如果包不能自动安装，可以手动从网上搜索并下载 .zip 或 .tar.gz 文件到本地，再手动安装（不建议）。执行菜单栏中的 Tools → Install Packages 命令，在弹出的 Install Packages 对话框中修改 Install from 为 Package Archive File（.zip; .tar.gz），然后浏览安装，如图 1-11 所示。

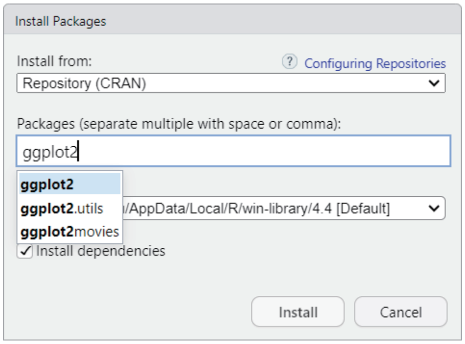


图 1-10 Install Packages 对话框

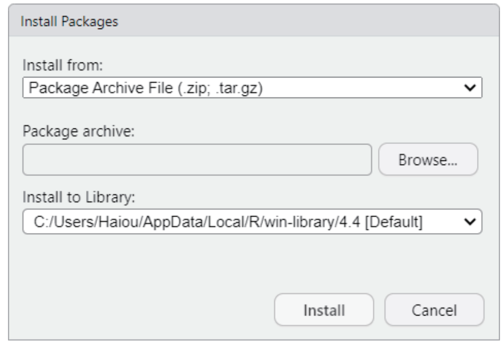


图 1-11 手动安装参数设置

4．通过 GitHub 安装包

有些 R 包只存放在 GitHub，也有很多 CRANR 包的最新开发版位于 GitHub，读者可以先安装 devtools 或 remotes 包，再通过 install_github() 函数安装 GitHub 来源的包。

```
devtools::install_github("tidyverse/dplyr")        # 或
remotes::install_github("tidyverse/dplyr")
```

说明 函数中“::”的前面是包名，这是不单独加载包而使用包中函数的用法。tidyuerse 是 GitHub 用户名，dplyr 为该用户中名为 dplyr 的 repository（仓库），即包名。另外，通过“包名 :::”前缀可以访问包中的内部函数。

注意 不是所有的仓库都是 R 包，识别 R 包的标志是否含有 DESCRIPTION 文件。

读者也可以将整个包文件夹从网页下载并解压缩到当前路径（或完整路径），再从本地安装，语法格式为：

```
install.packages("dplyr-master",repos=NULL,type="source")
```

在生物信息领域有专门的包，可以从 Bioconductor 网站获取，安装时首先安装 BiocManager 包，再通过 install() 函数安装 Bioconductor 来源的包，语法格式为：

```
BiocManager::install("p_name")
```

5．更新包

当需要从 CRAN 或其他指定的仓库下载和安装包的最新版时，可以使用 update.packages() 函数。其语法格式为：

```
update.packages("p_name")                    # 更新名为 p_name 的包
update.packages()                            # 更新所有包
```

6．加载第三方包

在完成安装后，要使用该包时，需要使用 library() 函数或 requir() 函数将其加载到 R 中。其语法格式为：

```
library(package_name)       # 加载指定的包，如果包不存在，会报错并停止执行代码
require(package_name)       # 加载指定的包，当包不存在时不报错，而是返回一个逻辑值
```

注意 在加载第三方包时，每次只能加载一个包，如需加载多个包，必须多次调用 library() 或 require() 函数。

【例 1-4】将 ggplot2 和 ggraph 两个包加载到 R 中。

在控制台中依次输入以下代码：

```
> library(ggplot2)                   # 加载 ggplot2 包
> library(ggraph)                    # 加载 ggraph 包
```

7．卸除包与卸载包

这里，卸载包表示卸载已安装到 R 中的包；卸除包表示卸除已经加载到内存的包，卸除不是卸载，只是存储释放。

当希望卸载已安装的包时，可以采用 remove.packages() 函数，其语法格式为：

```
remove.packages("package_n ame",lib=file.path("package path"))
```

例如，卸载 ggplot2 包的语句为：

```
remove.packages("ggplot2")
```

当希望卸除加载的包时，可以采用 detach() 函数，其语法格式为：

```
detach("package_name")
```

例如，卸除 ggplot2 包的语句为：

```
detach("package:ggplot2")
```

1.3 对象与变量

如果要对输入的数据进行多种分析，如计算标准差、绘制柱状图等，每次分析都要输入

数据就非常麻烦。这时，可以将多个数据组合成一个数据集，然后将数据集赋值给一个名字，这就是所谓的 R 对象（object）。对对象进行赋值，此时的对象可以称为变量。

1.3.1 对象

R 语言中的所有事物都可以称为对象，如向量、列表、函数、环境等。R 语言的所有代码都是基于对象（object）操作的，对计算机内存的访问同样是通过对象实现的。

【例 1-5】R 语言对象应用示例。

输入代码如下：

```
> c("海鸥","麻雀","鸽子","海燕")                    # 包含 4 个元素的字符型向量
[1] "海鸥" "麻雀" "鸽子" "海燕"
> c(5)                          # 只有 1 个元素的数值型向量，或者直接输入数字 5
[1] 5

> list(c("海鸥","麻雀","鸽子"),c(5),"I'm Chinese.")   # 包含 3 个元素的列表
[[1]]
[1] "海鸥" "麻雀" "鸽子"

[[2]]
[1] 5

[[3]]
[1] "I'm Chinese."

> function(x,y)                                          # 函数
+ {
+   (x^2+y)
+ }
function(x,y)
{
    (x^2+y)
}
> new.env()                                              # 环境
<environment: 0x00000172e87ccc70>
```

上述代码中，c() 是一个 R 函数，表示将其中的数据合并成一个向量。

约定：本书中，提示符“＞”表示其后需要输入，提示符“+”表示其后输入为上一行的延续，符号“# ”表示注释，无须输入。

1.3.2 变量

R 对象可以是一个数据集、模型、图形等，在分析前需要给对象赋值。R 的标准赋值符号为“<-”（也可以使用“=”进行赋值），推荐使用“<-”。

R 对象实际上就是给对象取的一个名字（如 x），然后对其赋值（可以是数值、向量、矩阵或数据框等）。赋值后的对象可以称为变量，它是调用对象的重要手段。

【例 1-6】 赋值方法示例。

在控制台中依次输入以下代码，并显示输出结果。

```
> x1 <- 6                                          # 将数值赋给 x
> x1
[1] 6
> x2 <- c("海鸥","麻雀","鸽子","海燕")             # 将字符型向量赋给 x
> x2
[1] "海鸥" "麻雀" "鸽子" "海燕"
> y1 <- y2 <- y3 <- 6                              # 同时将一个值赋给多个变量
> y1
[1] 6
> y2
[1] 6
> y3
[1] 6
```

说明 变量名与变量值前后可以互换，同时赋值符号“<-”需要变为“->”，不推荐使用。继续在控制台中依次输入以下代码，并显示输出结果。

```
> 6 -> x4                                          # 将数值赋给 x
> x4
[1] 6
> z <- c(68,61,82,66,72,44,66,57)                  # 将数值向量赋给 z
> mean(z)                                          # 计算平均数
[1] 64.5
> sum(z)                                           # 求和
[1] 516
```

变量名称是以字母或点号(.)开头，并以数字、字母及下画线(_)的任意组合组成的名称。在 R 语言中，变量的命名有以下规则：

(1)变量名的首字符只能使用字母或点号，变量名的次字符及之后的字符只能包含数字、字母或下画线。

（2）变量名区分大小写，如 name 和 Name 代表两个不同的变量对象。

（3）变量的命名建议与其含义相近，如用 Gender 表示性别变量，而不用 ga、x 等。

（4）系统的保留字（如 if、for 等）不能作为变量名。

1.4 获取帮助信息

R 语言拥有功能强大、种类繁多的第三方包，方便不同的用户选择合适的包解决工作中的实际问题。这也造成了需要记忆太多的包或函数，甚至函数的具体用法和参数含义也要掌握。当不记得这些函数或包时，就需要通过 R 语言强大的资源支持系统获取对应的帮助信息。

1.4.1 使用内置帮助函数

在 R 语言的学习中，熟练掌握帮助的使用方法是至关重要的。R 语言自身包含大量的内置帮助函数，这些函数均可以在离线环境下使用，常用的内置帮助函数如表 1-1 所示。

表1-1 内置帮助函数功能及用法

函　　数	功　　能	示　　例
help.start	显示R语言的网页帮助	help.start()
?	查找某个函数帮助	?t.test或?"t.test?"t.test ""
help	查找某个函数帮助	help(t.test)或help("t.test")
??	查找与某个函数有关的关键字	??t.test或??"t.test"
help.search	查找与某个函数有关的关键字	help.search("t.test")
apropos	查找与输入字段相匹配的函数与变量	apropos("t.test")
find	查找与输入字段相匹配对象（变量或函数）所属的环境或包	find("t.test")
example	运行函数示例（所有函数）	example(t.test)或example(t.test)
demo	运行函数演示（部分函数）	demo(nlm)或demo("nlm")
RSiteSearch	在http://search.r-project.org上检索输入的关键字	RSiteSearch("Hosmer-Lemeshow")
data	列出当前已加载包中所含的所有可用示例数据集	data()
vignette	列出当前已安装包中所有可用的vignette文档	vignette()

R 自带了很多数据集，并附有数据集的分析和绘图示例，可作为学习 R 时的练习使用。利用 data() 函数及 help() 函数可以了解数据集的信息。

```
data()                               # 列出当前已加载包中所含的所有可用示例数据集
data(date_name)                      # 查看名为 date_name 的数据集数据
help(date_name)                      # 查看数据集 date_name 的详细信息
```

【例 1-7】获取帮助信息示例。

在控制台中依次输入以下代码，并显示输出结果。

```
> help(lda,package='MASS')           # 直接查询某个函数的帮助文档
> help.search('geom_bar')            # 从所有的已下载包中搜寻 geom_bar 函数
> RSiteSearch('Neural Network')      # 在线搜索包含关键词的帮助文档
> data(Titanic)                      # 查看泰坦尼克号的数据详细信息
> example(t.test)                    # 运行函数 t.test 的示例
```

1.4.2 获取自带数据集信息

R 语言自带了许多数据集，这些数据集可以通过 datasets 包访问。本书中很多的示例是通过自带数据集演示的。利用 data() 函数可以查看已加载包中的数据集。

```
data()                               # 不带参数时，列出所有已加载包中的数据集
data(package="datasets")             # 查看 datasets 包中的数据集
data(datasetsName)                   # 加载名称为 datasetsName 的数据集
```

另外，利用 str() 函数可以查看数据集的结构，利用 summary() 函数可以查看数据集的统计摘要。

【例 1-8】查看数据集操作。

在控制台中依次输入以下代码，并显示输出结果（略）。

```
data()                               # 查看所有已加载包中的数据集
data(package="datasets")             # 查看 datasets 包中的数据集
data(iris)                           # 加载 iris 数据集
head(iris)                           # 查看 iris 数据集的前几行
str(iris)                            # 查看 iris 数据集的结构
summary(iris)                        # 查看 iris 数据集的摘要
```

1.4.3 R 语言相关软件和资料

除内置函数帮助外，R 语言还拥有丰富的外部学习资源。为方便读者学习，下面提供一些 R 语言相关软件和资料的常用网址：

（1）R 语言官方及 RStudio 官方提供了丰富的学习资料。

（2）R 语言邮件列表收集了多年来积累的关于 R 语言的各种问题及解决方法，读者可以订阅这些邮件列表，以获取帮助。

（3）Rseek 站点是一个 R 语言的网页搜索引擎，可以查找各种函数，以及 R 语言邮件列表中的讨论和博客文章。

（4）R-bloggers 是 R 语言主要的博客社区，也是关注 R 语言的社区资讯和小技巧的最佳方式。另外，Stack Overflow 与 R-bloggers 类似，也是一个活跃的 R 语言社区。

（5）R 语言入门中文论坛是专门为国内 R 语言用户提供的在线沟通和交流的平台，当遇到问题时可以与大家交流分享。

1.5 本章小结

本章详细介绍了 R 语言的基础知识，包括 R 语言概述、获取与安装、对象与变量以及获取帮助信息等内容。本章为 R 语言的基础入门知识，通过本章的学习，可以为后续学习和应用 R 语言实现医学数据分析与可视化奠定基础。

第 2 章 创建数据集

在数据分析和建模的过程中，数据集是整个分析流程的核心。无论是进行基础的统计分析、构建机器学习模型，还是进行复杂的可视化呈现，都离不开数据的支持。在 R 语言中，数据集的创建和处理尤为重要，R 不仅提供了多种方式来导入、生成和操作数据，还拥有丰富的函数和包来处理不同类型的数据。通过灵活地使用这些工具，研究者可以高效地管理、清洗和变换数据，为后续的分析打下坚实的基础。

本章将重点介绍如何在 R 语言中创建数据集，并探讨常见的数据结构，如向量、矩阵、数据框和列表的特点与用途。掌握这些基础知识，能够帮助读者在实际工作中灵活处理数据，并为后续的分析步骤做好准备。

视频

2.1 数据集的概念

在数据分析中，数据集是信息的载体，它承载着我们需要研究和探索的所有数据信息。所谓数据集，指的是按一定格式组织起来的数据集合，通常由行和列构成，其中每一行代表一个观察样本，每一列则表示某个特定的特征或变量。在 R 语言中，数据集的概念极为广泛，涵盖从简单的向量到复杂的数据框架等多种形式。

理解数据集的概念，不仅有助于熟练掌握数据的基本操作，还能为后续的分析和建模打下坚实的基础。通过学习如何在 R 中创建和管理数据集，我们能够更好地应对从数据获取、清洗到分析的各个环节，从而更高效地解读数据，得出有价值的结论。

数据集在 R 语言中最直观的体现就是对象（Object）和变量（Variable）。这是编程的基础概念，几乎所有的数据和操作都围绕它们展开。对象是 R 中存储数据的容器，而变量则是用来引用这些对象的标签。通过对对象和变量的深入理解，能帮助我们更好地进行数据处理和分析。

2.1.1 认识 R 语言的对象

所有事物在 R 语言中都可以称为对象，如向量、列表、函数、数据框、环境等。所见都为对象。对于操作者来说，我们使用 R 语言导入的数据就是 R 语言的对象，以及 R 语言运行过程中产生的所有数据都是 R 语言的对象。

【例 2-1】R 语言产生对象。

在控制台窗口中输入以下代码，同时显示输出结果：

```
> c(1,2,3,4,5)                                    # 创建一个数值向量
[1] 1 2 3 4 5

> matrix(1:6,nrow=3,ncol=2)                       # 创建一个 3 行 2 列的矩阵
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6

> list(c("海鸥","鸽子"),c(5),"I'm Chinese.")      # 创建一个包含 3 个部分的列表
[[1]]
[1] "海鸥" "鸽子"
[[2]]
[1] 5
[[3]]
[1] "I'm Chinese."

> function(x,y) {
+     z=x+y
+     return(z)
+ }                                               # 创建一个函数
function(x,y) {
```

```
        z=x+y
        return(z)
    }
```

上述代码中，c() 是一个 R 函数，表示将其中的数据合并成一个向量。

2.1.2 认识 R 语言的变量

2.1.1 节我们认识了 R 语言的对象，对象都是存于 R 语言内存的数据。我们在分析时如何调取这些数据为我们工作呢？那就是通过将 R 语言中的对象对应给变量，就是给每个对象起一个具体的名字，由变量代表对象来操控数据。

在 R 语言中，可以使用赋值运算符“<-”或“=”来创建变量并将值赋给它。变量名可以包含字母、数字和下画线，但不能以数字开头。

【例 2-2】R 语言创建变量。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> A <- c(1,2,3,4,5)                                          # 给变量 A 赋值
> A
[1] 1 2 3 4 5

> B <- matrix(1:6,nrow=3,ncol=2)                             # 将矩阵赋值给 B
> B
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6

> C <- list(c("海鸥","鸽子"),c(5),"I'm Chinese.")            # 将列表赋值给 C
> C
[[1]]
[1] "海鸥" "鸽子"
[[2]]
[1] 5
[[3]]
[1] "I'm Chinese."

> hanshu <- function(x,y) {
+   z=x+y
+   return(z)
+ }                                                   # 将函数赋值给 hanshu
```

```
> hanshu
function(x,y) {
  z=x+y
  return(z)
}

> # 变量创建结束后，直接对变量进行操作，就等同于对变量对应的对象操作
> a <- 2
> b <- 4
> a+b
[1] 6
> a*b
[1] 8
```

变量名称是以字母或点号(.)开头，并以数字、字母及下画线(_)的任意组合组成的名称。变量名中不能有特殊符号，不能以数字开头。

2.2 R 语言的数据结构

R 语言的数据结构是数据分析的核心，它决定了如何组织、存储和处理数据。在 R 中，不同类型的数据结构提供了灵活且强大的工具来应对多样化的数据分析需求。无论是处理单一类型的数据还是混合类型的复杂数据集，选择合适的数据结构至关重要。

R 中的数据结构主要包括向量、矩阵、数组、数据框和列表，如图 2-1 所示。每一种结构都具有独特的属性和应用场景，理解它们的使用方式可以大大提高数据处理的效率和灵活性。本节将深入探讨这些基本的数据结构及其在 R 编程中的应用，帮助你在实际分析中更加自如地操作各种数据。

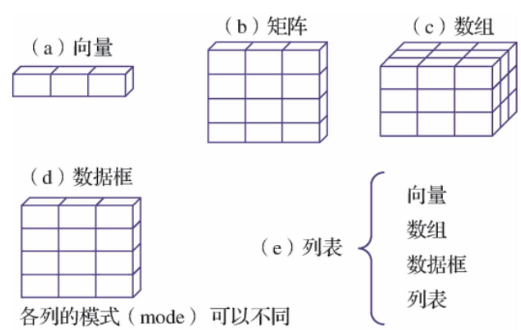


图 2-1 不同数据格式展示图

2.2.1 数据类型

在 R 语言中，数据类型是决定数据如何存储和操作的基本属性。了解不同的数据类型有助于更高效地进行数据处理和分析。

R 语言提供了多种数据类型来表示不同形式的数据，例如数值、字符、布尔值等。熟悉这些数据类型能够帮助你避免代码中的类型错误，并能根据需求选择最合适的存储方式。R 语言中的基本数据类型包括：

- 数值型（Numeric）：用于存储整数和浮点数；数值型又分为整数型（integer）和双精度型（double）两种。
- 字符型（Character）：用于存储文本或字符串。
- 逻辑型（Logical）：用于存储布尔值，即 TRUE 或 FALSE。
- 复数型（Complex）：用于存储带有实部和虚部的复数。
- 日期型（Data）：用于保存时间数据。

【例 2-3】数据类型应用。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> num <- 42                                           # 数值型数据
> print(num)                                          # 输出：42
[1] 42
> char <- "R 语言 "                                   # 字符型数据
> print(char)                                         # 输出："R 语言 "
[1] "R 语言 "
> bool <- TRUE                                        # 逻辑型数据
> print(bool)                                         # 输出：TRUE
[1] TRUE
> comp <- 1+2i                                        # 复数型数据
> print(comp)                                         # 输出：1+2i
[1] 1+2i
> Birthday <- as.Date('1985-6-18')                    # 日期型数据
> Birthday
[1] "1985-06-18"
> Score <- as.integer(x=c(160,85,125))                # 整数型数据
> Score
[1] 160  85 125

> # 查看数据类型
> print(class(num))                                   # 输出："numeric"
[1] "numeric"
> print(class(char))                                  # 输出："character"
[1] "character"
> print(class(bool))                                  # 输出："logical"
[1] "logical"
> print(class(comp))                                  # 输出："complex"
[1] "complex"
```

```
> print(class(Birthday))                    # 输出："Date"
[1] "Date"
> print(class(Score))                       # 输出："integer"
[1] "integer"
```

R 语言的数据类型多样，每种类型都有其独特的应用场景。掌握这些基本的数据类型能够帮助你更好地理解和处理数据。在实际操作中，根据具体的需求选择合适的数据类型，不仅可以提高代码的运行效率，还能增强数据分析的准确性。

2.2.2 向量

向量是 R 语言中最基础、最常用的数据结构之一，它由一组同类型的元素构成，无论是数值、字符还是逻辑数据，向量都可以将这些数据存储在一个有序的集合中。由于向量是数据分析和处理的基础，掌握如何创建和操作向量是学习 R 语言的关键。

R 提供了多种创建向量的方法，每种方法适用于不同的场景。下面将详细介绍几种常见的方式，并配以具体的示例。

1）c() 函数

c() 函数（concatenate）是最常用的创建向量的方法，它可以将多个数值、字符或逻辑值合并成一个向量。

【例 2-4】生成多种不同类型的向量。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> vec_numeric <- c(1,2,3,4,5)                      # 数值向量
> vec_numeric
[1] 1 2 3 4 5

> vec_char <- c("A","B","C","D")                   # 字符向量
> vec_char
[1] "A" "B" "C" "D"

> vec_logical <- c(TRUE,FALSE,TRUE)                # 逻辑向量
> vec_logical
[1]  TRUE FALSE  TRUE
```

2）rep() 函数

rep() 函数用于重复元素或向量，适合生成重复模式的数据，其语法结构如下：

```
rep(x,times=,each=,...)
```

其中，参数含义如表 2-1 所示。

表2-1 选项说明

选　　项	描　　述
x	一个向量（可以是任何类型，包括列表）、因子，或者（仅针对rep函数）POSIXct、POSIXlt或Date对象；或者一个包含此类对象的S4对象
times	一个整数值向量，若长度为length(x)，则表示每个元素重复的（非负）次数；若长度为1，则表示整个向量重复的次数。负值或NA值会导致错误。接受双精度向量，其他输入将被强制转换为整数或双精度向量
each	非负整数。x的每个元素将重复each次。其他输入将被强制转换为整数或双精度向量，并取其第一个元素。如果为NA或无效值，则按1处理

【例 2-5】生成不同序列和不同间距的向量。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> vec_seq3 <- seq(1,10,by=2)              # 从 1 ～ 10，步长为 2 的序列
> vec_seq3
[1] 1 3 5 7 9

> vec_seq4 <- seq(0,1,length.out=5)       # 从 0 ～ 1，生成 5 个等间距的数值
> vec_seq4
[1] 0.00 0.25 0.50 0.75 1.00
```

3）seq() 函数

seq() 函数用于创建规则序列，提供了更灵活的方式控制步长和序列的长度，其语法结构如下：

```
seq(from=1,to=1,by=((to-from)/(length.out-1)),
    length.out=NULL,along.with=NULL,...)
```

其中，参数含义如表 2-2 所示。

表2-2 选项说明

选　　项	描　　述
from,to	序列的起始值和（最大）终止值。除非仅提供from作为未命名参数，否则该参数的长度为1
by	序列的期望长度。一个非负数，对于seq和seq.int，如果是小数，将向上取整
by	从该参数的长度中获取序列的长度

【例 2-6】生成多种重复性向量。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> vec_rep1 <- rep(3,times=5)                    # 重复数值 3 五次
> vec_rep1
[1] 3 3 3 3 3

> vec_rep2 <- rep(c(1,2,3),times=2)             # 重复向量 1、2、3 两次
> vec_rep2
[1] 1 2 3 1 2 3

> vec_rep3 <- rep(c(1,2,3),each=2)              # 每个元素分别重复两次
> vec_rep3
[1] 1 1 2 2 3 3
```

4）使用 : 运算符

运算符可以生成一个连续的整数序列。

【例 2-7】生成连续性变量。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> vec_seq1 <- 1:10                              # 生成从 1 ～ 10 的向量
> vec_seq1
 [1]  1  2  3  4  5  6  7  8  9 10

> vec_seq2 <- 10:1                              # 生成从 10 ～ 1 的递减向量
> vec_seq2
 [1] 10  9  8  7  6  5  4  3  2  1
```

5）提取向量的子集

提取向量的子集是数据操作中常见的需求，R 提供了多种方法来实现这一功能。

【例 2-8】3 种不同的提取方法

在控制台窗口中输入以下代码，同时显示输出结果。

```
> # 1. 基于索引提取。通过指定向量元素的索引，可以提取特定位置的元素，R 的索引从 1 开始
> vec_numeric[2]                       # 提取向量 vec_numeric 的第 2 个元素
[1] 2
> vec_numeric[1:3]                     # 提取第 1 ～ 3 个元素
[1] 1 2 3
> vec_numeric[c(1,3,5)]                # 提取第 1、3 和 5 个元素
[1] 1 3 5

> # 2. 基于逻辑条件提取。使用逻辑表达式可以筛选出符合条件的元素
> vec_numeric[vec_numeric > 2]         # 提取大于 2 的元素
[1] 3 4 5
```

```
> vec_char[vec_char=="A"]                                    # 提取等于 "A" 的元素
[1] "A"
>
> # 3. 基于名称提取。如果向量中的元素带有名称，可以通过名称进行提取
> names(vec_numeric) <- c("one","two","three","four","five") # 为向量元素命名
> vec_numeric["two"]                                         # 通过名称提取
two
  2
```

2.2.3 矩阵

在 R 语言中，矩阵（Matrix）是一种重要的数据结构，用于存储二维的同类型数据。与向量类似，矩阵中的所有元素必须是相同类型的（例如数值、字符或逻辑值）。矩阵广泛应用于统计学和数据分析领域，尤其适合表示线性代数中的矩阵运算或表格格式的数据。

R 语言提供了多种方法来创建矩阵，其中最常用的函数是 matrix()。该函数允许用户通过指定行、列等参数来构建矩阵。其语法结构如下：

```
matrix(data,nrow=1,ncol=1,byrow=FALSE,dimnames=NULL)
```

其中，参数含义如表 2-3 所示。

表2-3　选项说明

选　　项	描　　述
data	构成矩阵的元素，可以是向量
nrow	矩阵的行数
ncol	矩阵的列数
byrow	逻辑值，若为TRUE，按行填充矩阵；若为FALSE（默认值），按列填充矩阵
dimnames	一个可选的列表，用于指定矩阵行列的名称

【例 2-9】通过 matrix() 函数创建矩阵。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> mat1 <- matrix(1:6,nrow=2,ncol=3)                    # 按列填充矩阵
> mat1
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> mat2 <- matrix(1:6,nrow=2,ncol=3,byrow=TRUE)         # 按行填充矩阵
> mat2
     [,1] [,2] [,3]
[1,]    1    2    3
```

```
[2,]    4    5    6
> mat3 <- matrix(1:9,nrow=3,ncol=3,
+               dimnames=list(c("Row1","Row2","Row3"),
+                             c("Col1","Col2","Col3")))  # 带行名和列名的矩阵
> mat3
     Col1 Col2 Col3
Row1    1    4    7
Row2    2    5    8
Row3    3    6    9
```

矩阵的元素可以通过行列索引来访问，语法为 matrix[row,column]。

【例 2-10】通过行列索引来访问矩阵元素。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> mat1[1,2]                                  # 访问第一行第二列的元素
[1] 3
> mat1[1,]                                   # 提取第一行
[1] 1 3 5
> mat1[,2]                                   # 提取第二列
[1] 3 4
```

R 中提供了对矩阵进行数学运算的函数，常见的包括矩阵加法、乘法以及转置。

【例 2-11】矩阵基本属性运算。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> mat_sum <- mat1+mat2                       # 矩阵加法
> mat_sum
     [,1] [,2] [,3]
[1,]    2    5    8
[2,]    6    9   12
> mat_t <- t(mat1)                           # 矩阵转置
> mat_t
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
> mat_mult <- mat1*mat2                      # 矩阵乘法（元素相乘）
> mat_mult
     [,1] [,2] [,3]
[1,]    1    6   15
[2,]    8   20   36
```

使用 cbind() 和 rbind() 可以按列或按行合并矩阵。

【例 2-12】使用 cbind() 和 rbind() 按列或按行合并矩阵。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> mat_cbind <- cbind(mat1,mat2)                              # 按列合并矩阵
> mat_cbind
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    3    5    1    2    3
[2,]    2    4    6    4    5    6
> mat_rbind <- rbind(mat1,mat2)                              # 按行合并矩阵
> mat_rbind
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
[3,]    1    2    3
[4,]    4    5    6
```

矩阵在数据分析、统计学和线性代数中具有广泛应用，例如线性回归中的系数矩阵、图像处理中的像素矩阵、数据处理中的多维表格。

矩阵是 R 语言中处理二维同类型数据的基本结构，理解如何创建、操作和应用矩阵，对于进行有效的数据分析至关重要。掌握矩阵的基本操作，可以让你在处理复杂数据时更加得心应手。

2.2.4 数组

在 R 语言中，数组（Array）是一种能够存储多维数据的结构，支持二维及以上的维度。数组与矩阵类似，但可以扩展到更高的维度，如三维或四维，用于表示复杂的多维数据。数组在科学计算、图像处理和其他需要多维数据的领域具有广泛应用。

创建数组的常用函数是 array()，它能够将一个向量重塑为指定维度的数组，可以简单地认为数组就是好多矩阵拼在一起的一个结构。基本语法如下：

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

其中，参数含义如表 2-4 所示。

表2-4 选项说明

选　　项	描　　述
data	构成数组的元素，通常是一个向量。如果省略，则默认填充NA
dim	一个整数向量，指定数组的维度
dimnames	一个可选的列表，用于为数组的各个维度命名

【例 2-13】通过 array() 函数创建数组。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> arr1 <- array(1:12,dim=c(2,3,2))          # 创建一个 2×3×2 的三维数组
> arr1
,,1
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
,,2
     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12
> # 创建一个带有维度名称的三维数组
> arr2 <- array(1:12,dim=c(2,3,2),
+               dimnames=list(c("Row1","Row2"),
+                             c("Col1","Col2","Col3"),
+                             c("Layer1","Layer2")))
> arr2
,,Layer1
     Col1 Col2 Col3
Row1    1    3    5
Row2    2    4    6
,,Layer2
     Col1 Col2 Col3
Row1    7    9   11
Row2    8   10   12
```

【例 2-14】访问数组中的具体数据。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> arr1[1,2,1]                               # 访问第一行第二列第一层的元素
[1] 3
> arr1[1,,]                                 # 提取第一行的所有元素（所有层）
     [,1] [,2]
[1,]    1    7
[2,]    3    9
[3,]    5   11
> arr1[,2,]                                 # 提取第二列的所有元素（所有层）
     [,1] [,2]
[1,]    3    9
[2,]    4   10
```

【例 2-15】可以通过索引直接修改数组中的元素。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> arr1[1,2,1] <- 99                   # 将第一行第二列第一层的元素修改为 99
> arr1
,,1
     [,1] [,2] [,3]
[1,]    1   99    5
[2,]    2    4    6
,,2
     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12
```

【例 2-16】使用 dim() 函数获取或修改数组的维度。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> dim(arr1)                               # 获取数组的维度
[1] 2 3 2
> dim(arr1) <- c(2,2,3)                   # 修改数组维度，将其变为 2×2×3
> arr1
,,1
     [,1] [,2]
[1,]    1   99
[2,]    2    4

,,2
     [,1] [,2]
[1,]    5    7
[2,]    6    8

,,3
     [,1] [,2]
[1,]    9   11
[2,]   10   12
```

【例 2-17】数组支持类似于向量和矩阵的运算，例如元素的加法、减法等。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> arr3 <- array(1:12,dim=c(2,3,2))          # 创建一个 2×3×4 的数组
> arr_sum <- arr3+arr3>                     # 对两个数组进行元素相加

> arr_sum
,,1
```

```
          [,1] [,2] [,3]
     [1,]    2    6   10
     [2,]    4    8   12
     ,,2
          [,1] [,2] [,3]
     [1,]   14   18   22
     [2,]   16   20   24
```

数组在处理高维数据时非常有用，常见的应用场景包括：图像处理中的多维像素数据存储（如 RGB 图像的三维数组）、时间序列分析中的多维数据表示、科学计算和物理模型中的多维数据模拟。

数组是 R 语言中用于表示多维同类型数据的重要结构，尤其在需要处理三维或更高维数据的场景中非常有用。通过掌握数组的创建与操作，能够高效地组织和处理复杂数据，为数据分析和科学计算提供有力支持。

2.2.5 数据框

数据框（DataFrame）是 R 语言中最常用的二维数据结构，它类似于 Excel 中的表格，能够存储不同类型的列。例如，数据框的某一列可以存储数值型数据，而另一列则可以存储字符型数据。由于其灵活性，数据框被广泛应用于数据分析、数据整理和统计建模等领域。

R 语言中可以通过 data.frame() 函数来创建数据框。该函数允许用户将多个向量组合在一起，每个向量代表数据框中的一列。基本语法如下：

```
data.frame(...,row.names=NULL,check.rows=FALSE,check.names=TRUE,
           stringsAsFactors=default.stringsAsFactors())
```

其中，参数含义如表 2-5 所示。

表2-5 选项说明

选　　项	描　　述
...	构成数据框的列，通常是向量、列表等
row.names	用于指定行名，可以是字符向量或数值向量
check.rows	用于验证每列的行数是否一致
check.names	若为TRUE，则自动检查并调整列名以确保合法性
stringsAsFactors	指定字符串是否自动转换为因子类型

【例 2-18】使用 data.frame() 函数创建数据框。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> df <- data.frame(Name=c("Alice","Bob","Charlie"),
+                  Age=c(25,30,35),
+                  Gender=c("Female","Male","Male"))   # 创建一个简单的数据框
> df
     Name Age Gender
1   Alice  25 Female
2     Bob  30   Male
3 Charlie  35   Male
```

【例 2-19】使用 df[row,column] 访问数据框中的元素，通过列名选择某一列。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> df[1,]                                            # 提取第一行数据
   Name Age Gender
1 Alice  25 Female

> df[,2]                                            # 提取第二列数据
[1] 25 30 35

> df$Name                                           # 通过列名提取
[1] "Alice"   "Bob"     "Charlie"
```

【例 2-20】通过赋值方式将新列添加到数据框中。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> df$Income <- c(50000,60000,70000)                        # 添加新列：收入
> df
     Name Age Gender Income
1   Alice  25 Female  50000
2     Bob  30   Male  60000
3 Charlie  35   Male  70000
```

【例 2-21】R 提供了 cbind() 和 rbind() 函数，分别用于按列和按行合并数据框。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> df2 <- data.frame(Name=c("David","Eva"),
+                   Age=c(40,28),
+                   Gender=c("Male","Female"))             # 创建一个数据框

> df_combined <- rbind(df2,df2)                            # 按行合并两个数据框
> df_combined
   Name Age Gender
1 David  40   Male
2   Eva  28 Female
3 David  40   Male
4   Eva  28 Female
```

【例 2-22】使用 summary() 函数对数据框进行基本的汇总统计。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> summary(df)                                          # 汇总统计
     Name                 Age           Gender               Income
 Length:3           Min.   :25.0   Length:3           Min.   :50000
 Class :character   1st Qu.:27.5   Class :character   1st Qu.:55000
 Mode  :character   Median :30.0   Mode  :character   Median :60000
                    Mean   :30.0                      Mean   :60000
                    3rd Qu.:32.5                      3rd Qu.:65000
                    Max.   :35.0                      Max.   :70000

> mean(df$Age)                                         # 计算某列的平均值
[1] 30
```

数据框的应用领域非常广泛。

- 数据读取和清洗：R 语言中的 read.csv() 函数可以将外部 CSV 文件读取为数据框，方便后续数据处理。
- 数据统计和建模：在数据分析过程中，数据框用于存储输入数据，并作为统计分析和机器学习建模的基础。
- 数据可视化：数据框可以直接传递给 ggplot2 等可视化包，生成各类图表。

数据框是 R 语言中处理多类型数据的核心数据结构。无论是在数据导入、处理还是分析中，数据框都是不可或缺的工具。通过对数据框的操作，用户可以高效地管理和分析结构化数据，为进一步的统计建模和可视化奠定基础。

2.2.6 因子

因子（Factor）是 R 语言中特有的一种数据结构，用于处理分类数据。它是有序或无序的离散数据的存储方式，常用于表示分类变量或分组变量。因子非常适合处理字符型分类数据，因为它不仅存储类别，还可以存储类别的水平（Levels），使得 R 在统计分析中能够高效处理和显示这些数据。

R 语言使用 factor() 函数来创建因子。该函数将字符向量或整数向量转换为因子，并为其赋予不同的水平。基本语法如下：

```
factor(x,levels=,labels=,ordered=FALSE,exclude=NA)
```

其中，参数含义如表 2-6 所示。

表2-6 选项说明

选 项	描 述
x	一个字符型或数值型向量
levels	因子的水平，即不同分类的类别。如果不指定，默认会使用x中的唯一值
labels	用于为因子水平指定标签，常用于为因子赋予更具描述性的名称
ordered	逻辑值，表示因子是否有序。默认为FALSE
exclude	指定需要排除的值，默认是NA

【例 2-23】使用 factor() 函数创建因子。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> fruit <- factor(
+      c("apple","banana","apple","orange","banana"))     # 创建无序因子
> fruit
[1] apple  banana apple  orange banana
Levels: apple banana orange

> levels(fruit)                                         # 查看因子的水平
[1] "apple"  "banana" "orange"

> size <- factor(c("small","medium","large","medium","small"),
+                levels=c("small","medium","large"),
+                ordered=TRUE)                           # 创建有序因子
> size
[1] small  medium large  medium small
Levels: small < medium < large

> levels(size)                                          # 查看因子的水平和顺序
[1] "small"  "medium" "large"
> is.ordered(size)
[1] TRUE
```

【例 2-24】R 语言还支持对因子进行排序、统计以及更改因子水平等操作。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> levels(fruit) <- c("Apple","Banana","Orange")     # 修改水果因子的水平名称
> fruit
[1] Apple  Banana Apple  Orange Banana
Levels: Apple Banana Orange

> size[1] < size[2]                            # 比较因子水平，"small"<"medium"
[1] TRUE

> table(fruit)                                 # 计算水果因子的频数
```

```
> fruit
 Apple Banana Orange
     2      2      1

> df <- data.frame(Name=c("A","B","C","D","E"),
+                  Fruit=fruit,
+                  Size=size)              # 创建包含因子的数据框
> df
  Name  Fruit   Size
1    A  Apple  small
2    B Banana medium
3    C  Apple  large
4    D Orange medium
5    E Banana  small
```

因子相对于普通的字符型向量，有以下几个优势。

- 节省存储空间：因子水平存储为整数，而不是字符，大大减少了存储空间的占用。
- 统计分析的优势：在回归分析和建模中，因子可以用于处理分类变量，能够提供更高效的计算。
- 数据一致性：因子限制了分类数据只能包含指定的水平，避免了无意中引入新的类别，保证了数据一致性。

因子是 R 语言中处理分类数据的重要工具。通过将分类数据转换为因子，不仅能节省内存，还能更好地参与统计分析。掌握因子的创建、修改和操作，能够帮助你更有效地处理各种分类数据，在统计建模中也将带来极大的便利。

2.2.7 列表

在 R 语言中，列表（List）是一种非常灵活且强大的数据结构。与向量和数据框不同，列表能够容纳不同类型的数据元素，例如数值、字符、向量、数据框，甚至是其他列表。因此，列表在处理复杂的数据结构时显得尤为重要。R 的列表为研究者和开发者提供了高度灵活的数据存储方式，适用于处理混合类型的数据或将多个对象组合在一起。

在 R 中，列表可以通过 list() 函数来创建，list() 函数能够接受不同类型的数据并将其组合到一个列表中。基本语法如下：

```
list(...)
```

其中，参数…表示对象的数据格式。

【例 2-25】使用 lsit() 函数创建列表。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> my_list <- list(num=42,
+                 char="R 语言 ",
+                 vec=c(1,2,3),
+                 df=data.frame(x=1:3,y=c("A","B","C"))
+                 )                          # 创建包含不同数据类型的列表
> my_list                                    # 查看列表内容
$num
[1] 42
$char
[1] "R 语言 "
$vec
[1] 1 2 3
$df
  x y
1 1 A
2 2 B
3 3 C
```

在该例中，my_list 包含一个数值、一个字符向量、一个数值向量和一个数据框。这展示了列表的灵活性，它可以容纳不同类型的对象。

通过 list() 函数，我们可以创建各种各样的组合数据，这使得列表成为处理复杂数据的基础。

【例 2-26】R 语言中提取列表中的元素常用的 3 种方式。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> # 使用 $ 提取元素：适用于提取具名列表中的元素
> my_list$num                                # 提取具名元素
[1] 42

> # 使用 [[]] 提取单个元素：适用于具名或无名列表，可以精确提取单个元素
> my_list[[1]]                               # 提取列表中的第一个元素
[1] 42
> my_list[["df"]]                            # 提取数据框
  x y
1 1 A
2 2 B
3 3 C

> # 使用 [] 提取子列表：当使用单一方括号时，结果是一个子列表，包含指定元素
> my_list[1:3]                               # 提取第一个和第三个元素的子列表
```

```
$num
[1] 42
$char
[1] "R 语言 "
$vec
[1] 1 2 3
```

【例 2-27】同样，也可以用上述方式修改列表中的元素。

在控制台窗口中输入以下代码，同时显示输出结果。

```
> my_list$num <- 100                        # 修改列表中的数值
> my_list[[3]] <- c(4,5,6)                  # 通过 [[]] 修改向量
> my_list                                   # 查看修改后的列表
$num
[1] 100
$char
[1] "R 语言 "
$vec
[1] 4 5 6
$df
  x y
1 1 A
2 2 B
3 3 C
```

列表是 R 语言中非常重要的数据结构，它能够存储多种类型的数据，灵活性极高。通过对列表的掌握，我们可以更高效地处理复杂的数据组合。此外，R 中提供的多种提取、修改元素的方式，使得列表在实际数据处理过程中具有极大的优势。在数据分析和建模中，列表常用于存储和组织不同类型的数据对象，这为高效的数据管理奠定了基础。

2.3 获取内置数据集

R 语言自带了丰富的内置数据集，这些数据集涵盖从经济学到生物学等多个领域，适用于学习 R 语言和进行数据分析实践。获取和使用这些内置数据集，不仅有助于我们快速上手 R 语言中的数据操作，还能为实际的分析工作提供参考。

在 R 中，使用 data() 函数可以列出、加载和查看内置数据集。该函数支持列出所有可用的数据集，并将其加载到当前的工作环境中。其语法结构如下：

```
data(dataset,package=NULL,envir=environment(),verbose=FALSE)
```

其中，参数含义如表 2-7 所示。

表2-7　选项说明

选　　项	描　　述
dataset	指定要加载的数据集的名称（字符型），如果留空，将列出所有可用的数据集
npackage	可选参数，指定从哪个包中加载数据集。如果留空，将从所有已加载的包中搜索数据集
envir	默认是当前环境
verbose	布尔值，是否显示加载数据集的详细信息，默认是FALSE

如果不指定 dataset 参数，直接运行 data()，将列出当前 R 会话中所有可用的数据集。可以使用该方法来查看有哪些内置数据集可以使用。

【例 2-28】查看内置数据集。

在控制台窗口中输入以下代码，同时显示输出结果。

```
data()                                          # 列出所有可用的数据集
```

要加载某个具体的数据集，只需将数据集的名称传递给 data() 函数。

【例 2-29】加载 mtcars 数据集。

在控制台窗口中输入以下代码，同时显示输出结果。

```
data(mtcars)                                    # 加载 mtcars 数据集
head(mtcars)                                    # 查看 mtcars 数据集的前几行
```

如果数据集位于某个特定的包中，而该包还没有加载，可以通过 package 参数指定包名。

【例 2-30】加载 MASS 包中名为 Boston 的数据集。

在控制台窗口中输入以下代码，同时显示输出结果。

```
data(Boston,package="MASS")                     # 从 MASS 包中加载 Boston 数据集
head(Boston)                                    # 查看 Boston 数据集的前几行
```

R 语言的内置数据集为学习和实践 R 编程提供了便捷的工具。通过 data() 函数，可以轻松获取和使用这些数据集，快速进行数据分析。掌握内置数据集的使用，可以更好地理解 R 语言中的数据操作，为后续的分析打下坚实的基础。

2.4　基本运算符

R 语言提供了多种运算符，用于执行不同类型的操作，包括算术运算、比较运算、逻辑运算和赋值运算。掌握这些基本运算符对于数据处理和编程至关重要。本节将详细介绍这些运算符及其使用方法。

1）算术运算

算术运算包括加、减、乘、除、取余和幂运算，运算符如表 2-8 所示。

表2-8 算术运算符

运算符	含义	示例	结果	运算符	含义	示例	结果
+	加法	1+2	3	−	减法	3-1	2
*	乘法	2*3	6	/	除法	9/3	3
%%	取余数	9%%2	1	%/%	取整数	9%%2	4
^	幂	2^4	16	**	幂	2**4	16

2）逻辑运算

逻辑运算包括与、或和非运算，运算符如表 2-9 所示。

表2-9 逻辑运算符

运算符	含义	示例	结果
&	与，当所有数据为TRUE时结果为TRUE	TRUE&FALSE	FALSE
|	或，当有一个数据为TRUE时结果为TRUE	TRUE&FALSE	TRUE
!	非，为输入结果的反向	!TRUE	FALSE

3）关系运算

关系运算用于比较两个值的大小，运算符如表 2-10 所示。

表2-10 关系运算符

运算符	含义	示例	结果	运算符	含义	示例	结果
==	等于	1==1	TRUE	!=	不等于	1!=1	FALSE
<	小于	1<2	TRUE	<=	小于或等于	4<=5	TRUE
>	大于	1>2	FALSE	>=	大于或等于	4<=5	FALSE

这些运算符是 R 语言中常用的工具，在数据分析和编程中经常用到。掌握它们可以提高代码编写的效率和清晰度。

2.5 本章小结

通过本章的学习，对 R 语言的对象、变量以及 R 语言最常见、常用的数据结构有了最基础的认识，可以自由对不同类型的数据结构进行数据索引提取以及在不同类型的数据中更改数据集本身数据，为后续的数据分析打下了坚实的基础。