第 1 章
R 语言基础

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 语言的基本概念和技巧。

1.1 R 语言概述

R 是一种用于统计分析、绘图的语言和操作环境。R 是属于 GNU 系统的一个自由、免费、源代码开放的软件，它是一个用于统计计算和统计制图的优秀工具。

1.1.1 R 语言的诞生

R 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，被广泛使用于统计领域。S 语言是由贝尔实验室（AT&T Bell Laboratories）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一个 R 系统。

R 可以看作贝尔实验室的 Rick Becker、John Chambers 和 Allan Wilks 开发的 S 语言的一种实现。因此，两者在程序语法上几乎一样，可能只是在函数方面有细微差别，程序十分容易移植到 R 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 作为一种统计分析软件，是集统计分析与图形显示于一体的。它可以运行于 UNIX、Windows 和 Macintosh 操作系统上，而且嵌入了一个非常方便实用的帮助系统，相比于其他统计分析软件，R 还有以下特点：

（1）R 是一种开放性软件。这意味着它是完全免费、开放源代码的。可以在它的网站及其镜像中下载任何有关的安装程序、源代码、程序包及其文档资料。标准的安装文件自身就带有许多模块和内嵌统计函数，安装之后可以直接实现许多常用的统计功能。

（2）R 是一种可编程的语言。作为一个开放的统计编程环境，R 的语法通俗易懂，是很容易学会和掌握语言的语法。掌握之后，便可以编制自己的函数来扩展现有的语言。这也是它的更新速度比一般统计软件（如 SPSS、SAS 等）快得多的原因。大多数最新的统计方法和技术都可以在 R 中直接得到。

（3）R 的函数和数据集集成在程序包中。只有当一个包被载入时，它的内容才可以被访问。一些常用的、基本的程序包已经被收入了标准安装文件中，随着新的统计分析方法的出现，标准安装文件中所包含的程序包也随着版本的更新而不断变化。在新版安装文件中，已经包含的程序包有：base（R 的基础模块）、mle（极大似然估计模块）、ts（时间序列分析模块）、mva（多元统计分析模块）、survival（生存分析模块）等。

（4）R 语言拥有强大的数据分析功能。R 是专门为统计和数据分析开发的语言。基于免费开源的特点，R 语言已经形成了强大的社区，各行各业的优秀研究者无时无刻地贡献着自己编写的功能强大的包（研究成果），这些包涵盖各行各业前沿的分析方法，使用时如同站在巨人的肩膀上。从统计分析到机器学习再到深度学习，从金融分析到生信分析，从文本挖掘到社交网络分析再到并行计算等，R 语言无所不“包”。

（5）R 语言拥有强大的数据可视化能力。R 语言的绘图能力非常强大，尤其是 ggplot2 及其扩展包包含各种各样方便实用的绘图方法，便于研究者更清楚地理解自己所面对的数据。R 还包括多重可交互的数据可视化包，如 plotly 可直接将 ggplot2 的图像进行可交互地呈现。

（6）R 具有很强的互动性。除图形输出是在其他窗口外，它的输入输出是在同一个窗口中进行的，如果输入语法中出现错误，马上会在窗口中得到提示，对以前输入过的命令有

记忆功能，可以随时再现、编辑修改以满足用户的需要。输出的图形可以直接保存为 JPG、BMP、PNG 等图片格式，还可以直接保存为 PDF 文件。另外，R 与和其他编程语言和数据库之间有很好的接口。

由于 R 语言的第三方包非常多，难免会存在一些质量较差的包，还有一些包的更新跟不上 R 版本的更新。作为一种解释性的高级语言，使用者有时会认为它的计算速度较慢。随着计算机硬件不断提升、R 并行计算包的出现以及 apply 函数族强大的并行计算能力，会逐渐满足使用者在运行速度上的需求。

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 语言包括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，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。

这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。

R 语言绘图系统提供的绘图函数可以分为高级绘图函数和低级绘图函数，前者能够绘制出完整的图形，而后者是在已有图形上添加额外的图形。

（1）graphics 包既提供了高级绘图函数，又提供了低级绘图函数。

（2）grid 包仅提供低级绘图函数，高级绘图函数则留给建立在其基础之上的 lattice、ggplot2 以及其他扩展包。

1.1.4 图形语法

一张统计图形是从数据到几何对象的图形属性的一个映射。图形中还可能额外包含数据的统计变换，最终绘制在某个特定的坐标系中，并通过分面来生成数据不同子集的图形。也就是说，一张统计图形是由以下独立的图形部件所组成的。

（1）数据（data）。图形最基础的部分是想要可视化的数据（data）以及一系列将数据中的变量对应到图形属性（aesthetic attributes，aes）的映射（mapping）。

（2）图层（layler）。由几何元素和统计变换组成。

（3）几何对象（geometric object，geom）。在图形中实际看到的图形元素，如点、线、多边形等。

（4）统计变换（statistical transformation，stats）。对数据进行的某种汇总，如分组计数、线性回归等。统计变换为可选部分，但很有用。

（5）标度（scale）。将数据的取值映射到图形空间，如用颜色、大小或形状来表示不同的取值。展现标度的常用方式为绘制图例和坐标轴，它们实际上是从图形到数据的一个映射，从图形中可以读取原始数据。

（6）坐标系（coordinate system，coord）。描述数据如何映射到图形所在的平面，同时提供读图所需的坐标轴和网络线。通常使用笛卡儿坐标系，也可以变换为极坐标和地图投影等其他类型。

（7）分面（facet）。将绘图窗口划分为若干子窗口，描述如何将数据分解为各个子集，以及如何对子集作图并联合进行展示。分面也称为条件作图或网格作图。

（8）主题（theme）。主题控制着各点的精细显示，如字体、背景、颜色、网格线等。虽然 ggplot2 的默认设置基本满足需求，但有时需要进行调整来绘制自己想要的图形。

本书重点介绍 ggplot2 包，其语法特点如下：

（1）采用图层的设计架构，以 ggplot() 函数开始，图层之间通过“+”进行叠加，后面叠加的图层在前面的图层上方。一般通过 geom_...() 函数或 stat_...() 函数添加图层。

（2）将表征数据和图形细节分开，能快速将图形展示出来。通过 stat_...() 函数可以将常见的统计结果添加到图形中。

（3）拥有丰富的扩展包，创建的图形更加美观。通过调整颜色（color）、字体（font）和主题（theme）等辅助包可以帮助读者快速定制个性化的图表。

1.2 R 语言的获取与安装

R 语言可以在 CRAN（Comprehensive R Archive Network）网站上免费下载，CRAN 是拥有发布版本、资源包、文档和源代码的网络集合，它由几十个镜像网站组成，提供下载安装程序和相应版本的资源包，镜像更新频率一般为 1~2 天。

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。

1.2.1 安装程序下载

（1）在 IE 浏览器中输入网址（https://www.r-project.org/），按回车键后进入 R 语言官网，图 1-1 所示。

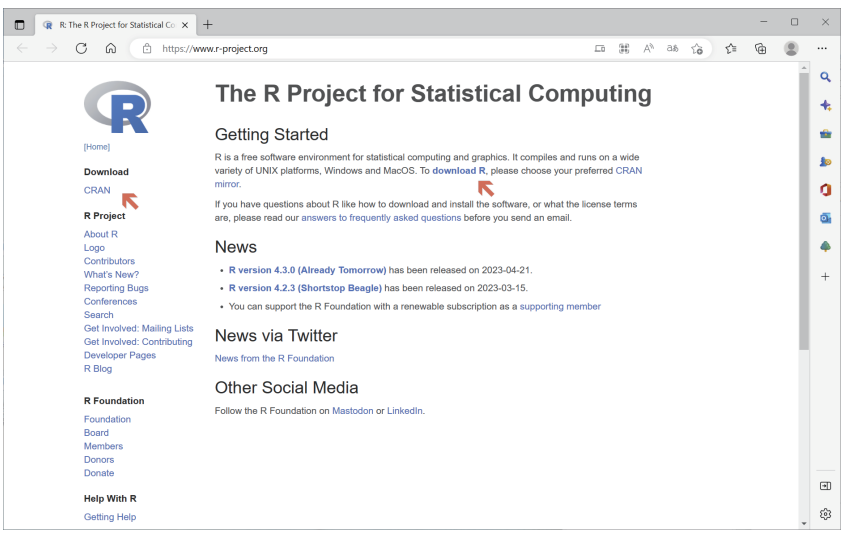

图 1-1　R 语言官网

（2）在主页单击左侧 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。

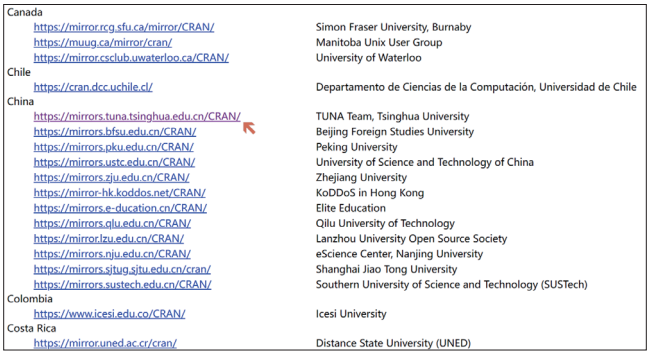

图 1-2 选择镜像站点

（3）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。

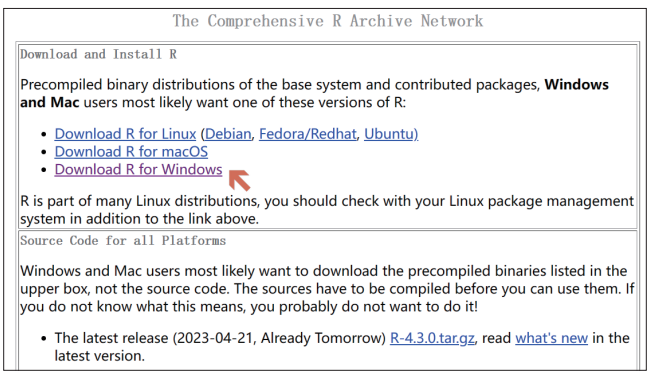

图 1-3 选择适应的平台版本

（4）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。继续在弹出的下一个页面中单击 Download R-4.3.0 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。

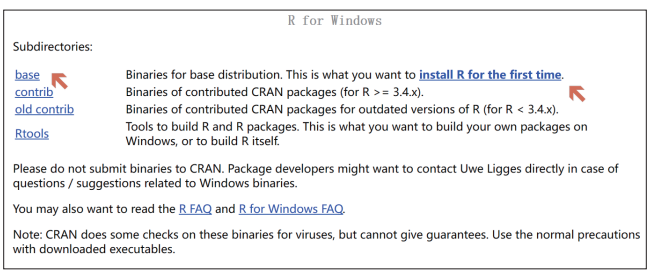

图 1-4 选择下载版本

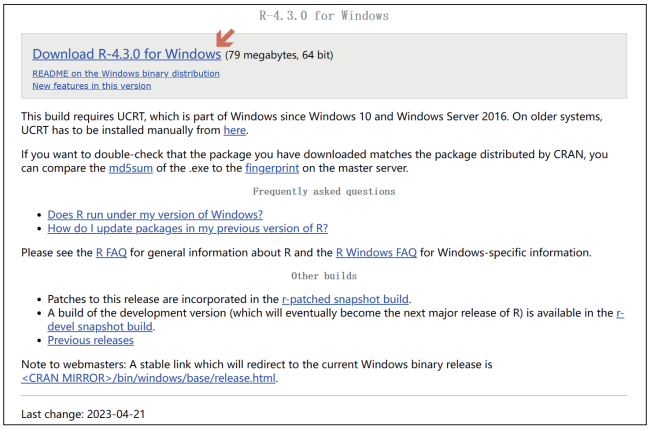

图 1-5　下载链接

1.2.2　R 语言的安装与启动

（1）在刚下载完成的安装包 R-4.3.0-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。

（2）在弹出的“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。

（3）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 RGui 界面，首次启动的 RGui 界面如图 1-6 所示。能够正常启动说明安装成功。

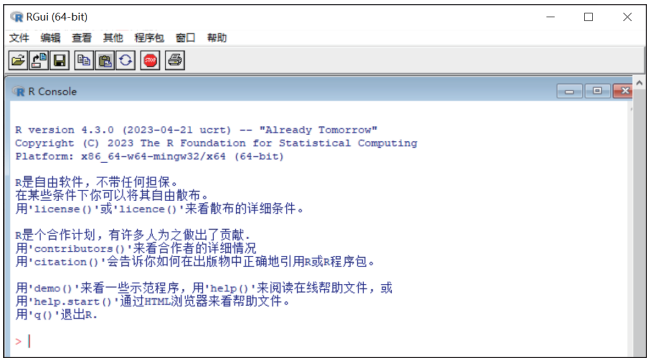

图 1-6　R 语言主界面（RGui 界面）

说明 在 Windows 平台中安装 R 语言时，除安装必要的核心文件外，还会安装一个叫作 Rgui.exe 的可执行文件，该程序文件位于 C:\Program Files\R\R-4.3.0\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】控制台命令输入示例。

在控制台中输入：

```
＞ 3+8                    # 在提示符后输入命令，按 Enter 键
[1] 11                    # 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，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。

```
＞ 3+8+34+98+             # 此处的 "+" 表示后续还有输入
34+45+56+45-
34-42
```

控制台上的显示为：

```
> 3+8+34+98 +
+34+45+56+45-
+34-42
[1] 247
>
```

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。下面介绍 RStudio 套件的下载与安装。

1. RStudio 的下载与安装

（1）在 IE 浏览器中输入网址（https://www.rstudio.com），按回车键后进入 RStudio 官网。在页面中找到并单击 DOWNLOAD RSTUDIO 按钮，下载该软件。

当前版本为 RStudio-2023.03.0-386。

（2）在刚下载完成的安装包 RStudio-2023.03.0-386 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。

（3）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。

（4）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，单击该按钮即可启动 RStudio，首次启动后的 RStudio 界面如图 1-7 所示，能够正常启动说明安装成功。

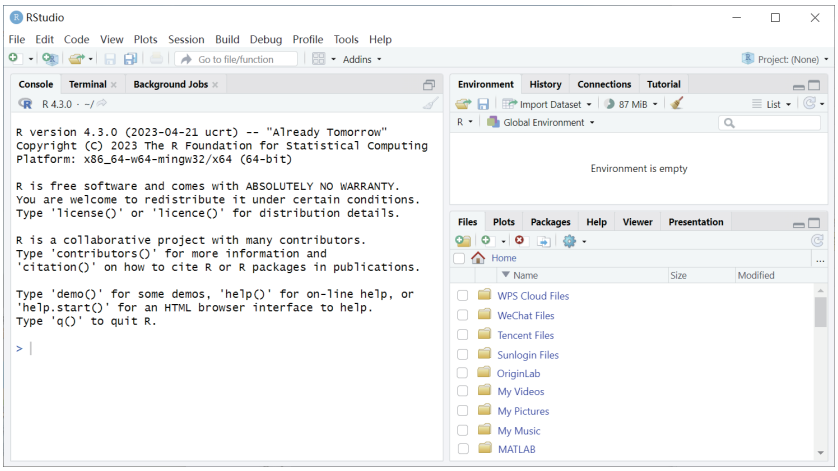

图 1-7　RStudio 主界面

2．RStudio 主界面介绍

执行菜单栏中的 File → New File → R Script 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。

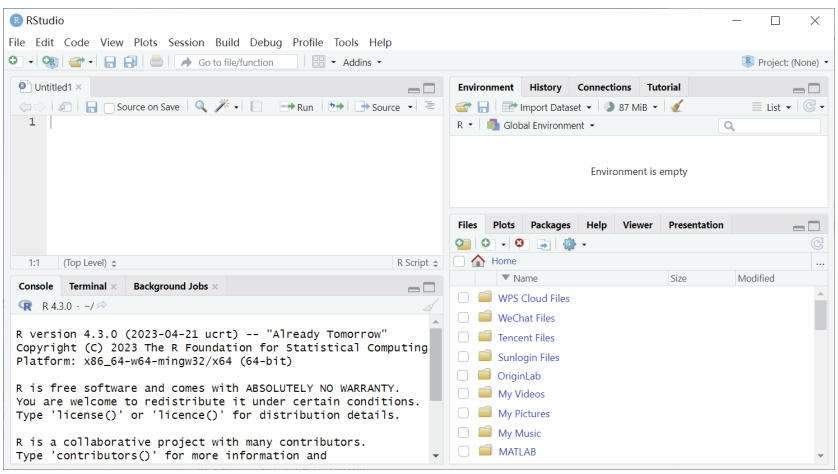

图 1-8　打开脚本窗口的主界面

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，该区域主要有文件、编辑、工具、帮助等菜单，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）左上方可以称为程序编写区，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的运行结果。这里包含所有运行过的命令，方便对历史记录进行检查。

（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。

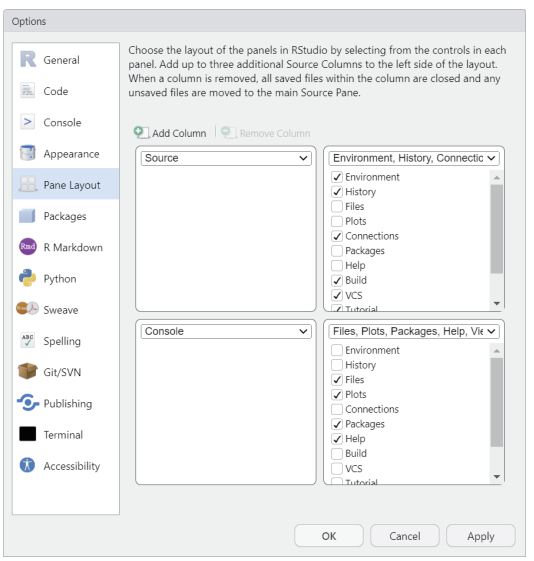

图 1-9 Options 对话框

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，第三方包需要下载并安装后才能使用。

一个 R 包中可能包含多个函数能做多种分析和绘图，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的。

1. 查看已安装的包

安装 R 时，默认自带一系列包（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。

```
> 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

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 gplots 两个包。

输入代码如下：

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

本书中的示例经常会调用不同的包，示例中不再提供安装包的方法，读者在学习过程中自己安装用得到的包即可。

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。

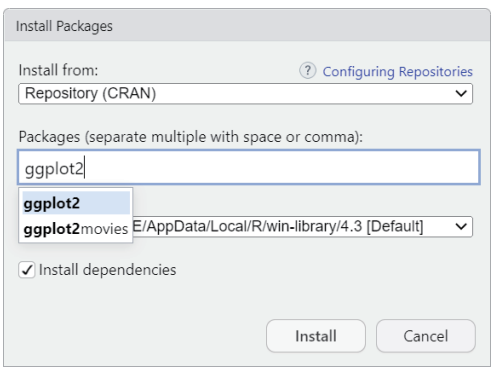

图 1-10 Install Packages 对话框

4．加载第三方包

在安装完成后，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 函数或 require 函数。

【例 1-4】将 ggplot2 和 gplots 两个包加载到 R 中。

输入代码如下：

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

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

5．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_name",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做多种分析，如计算标准差、绘制条形图等，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）。

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: 0x000001bbcfd76fc0>
```

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。

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。

输入代码如下：

```
> x1 <- 6                                          # 将数值赋给 x1
> x1
```

```
[1] 6
> x2 <- c("海鸥","麻雀","鸽子","海燕")     # 将字符型向量赋给 x2
> 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
```

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。例如：

```
> 6 -> x4                                  # 将数值赋给 x4
> 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 数据结构

在 R 中分析数据或创建图形时，首先要提供参与分析或绘图的数据集（Data Set）。R 可以处理的数据集类型包括向量（Vector）、矩阵（Matrix）、数组（Array）、数据框（Data

Frame）、因子（Factor）、列表（List）等，它们的数据结构如图 1-11 所示。

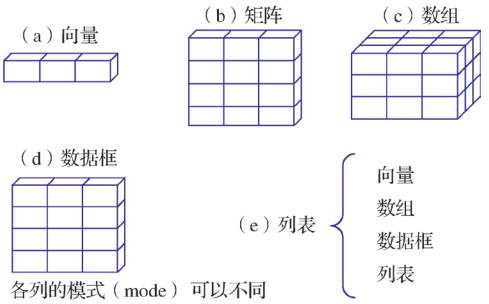

图 1-11 输出结果

1.4.1 数据类型

在 R 中，常见的数据类型有数值型（Numeric）、字符型（Character）、逻辑型（Logical）、复数型（Complex）或日期时间型（Data）等，其中数值型又分为整数型（Integer）和双精度型（Double）两种。数据类型的识别、判断以及类型之间的转换是数据分析中必不可少的内容。

在统计分析或运算过程中，可能需要对向量中的数据类型进行识别和判断。R 中使用 class() 函数识别数据类型，使用 is.*() 函数判断是否为某个指定的数据类型，使用 as.*() 函数进行数据类型的转换。其中 * 为数据类型的英文表示。

另外，R 语言中还包含几种常见的特殊值，如表 1-1 所示。

表1-1 特殊的数据类型

符 号	含 义	判断函数
NA	缺失值	is.na()
NULL	空值	is.null()
NaN	不确定值	is.nan()
Inf	无限值	is.inf()

【例 1-7】数据类型应用示例。

输入代码如下：

```
> Name <- c('Jeff','Tom','Mary')
> class(Name)                                    # 使用 class 函数识别数据类型
[1] "character"
```

```
> Birthday <- c('1985-6-18','1992-4-11','1986-12-8')
> class(Birthday)                              # 数据类型识别
[1] "character"
> Income <- c(16000,8500,12500)
> class(Income)                                # 数据类型识别
[1] "numeric"

> is.character(x=Name)
[1] TRUE
> is.integer(x=c(160,85,125))
[1] FALSE
> is.numeric(x=c(160,85,125))
[1] TRUE

> install.packages("lubridate")      # 安装第三方包，用于日期时间型数据的处理
> library(lubridate)                           # 加载包
> Score <- as.integer(x=c(160,85,125))         # 类型强制转换
> Score                                        # 返回向量中的元素
[1] 160  85 125
> class(Score)                                 # 类型识别
[1] "integer"
> Birthday <- as.Date(Birthday)                # 类型强制转换
> Birthday
[1] "1985-06-18" "1992-04-11" "1986-12-08"
> class(Birthday)                              # 类型识别
[1] "Date"
```

1.4.2 向量

向量是 R 语言中重要的数据结构，可以是数值数据、字符数据或逻辑值，很多情况下都会涉及向量的处理和运算。向量的创建可以通过手动输入、序列生成、重复生成和目标抽取（向量索引）等方法实现。

1. 手动输入法

R 语言允许用户通过手动方式将数据存储到向量中，例如将姓名存储到变量为 Name 的向量中，或将性别存储到变量为 Gender 的向量中。手动构建向量通过 c() 函数实现。

【例 1-8】手动输入向量示例。将 3 个客户的姓名、性别、出生日期和收入保存到各自的变量中。

输入代码如下：

```
> Name <- c('Jeff','Tom','Mary')
> Gender <- c('男','男','女')
> Birthday <- c('1985-6-18','1992-4-11','1986-12-8')
> Income <- c(16000,8500,12500)
```

（1）代码中的“<-”符号表示赋值，即将符号右边的向量赋值给左边的变量，表示赋值的符号还可以使用“=”“->”，读者根据自己的习惯选择即可。

（2）对于字符型的值或日期时间型的值，必须用引号引起来（如前 3 个变量），而数值型的值则不需要。

2．序列生成法

利用符号“:”或函数 seq() 生成具有规律的数值型数据，这就是序列生成法。其中，英文状态下的冒号“:”用于生成步长为 1 或 -1 的连续数据，seq() 函数用于生成指定步长或长度的等差数列。seq() 函数的语法格式为：

```
seq(from,to)                    # 不含 by、length 参数的 seq 函数
seq(from,to,by)                 # 含 by 参数的 seq 函数
seq(from,to,length)             # 含 length 参数的 seq 函数
seq(from,by,length)             # 含 by、length 参数的 seq 函数
```

其中，from 指定等差数列的初始值；to 指定等差数列的结束值；by 指定等差数列的公差；length 表示在公差未知的情况下，可以通过其设定等差数列的元素个数。

【例 1-9】序列生成法输入向量示例。

输入代码如下：

```
> X1 <- 1:8; X2 <- 1:-8
> X1
[1] 1 2 3 4 5 6 7 8
> X2
[1]  1  0 -1 -2 -3 -4 -5 -6 -7 -8
> X3 <- seq(from=1,to=8)              # 创建从 1 到 8，默认步长为 1 的序列
> X3
[1] 1 2 3 4 5 6 7 8
> X4 <- seq(from=1,to=8,by=2)         # 创建从 1 到 8，步长为 2 的序列
```

```
> X4
[1] 1 3 5 7
> X5 <- seq(from=1,to=8,length=2)   # 创建从 1 到 8，长度为 2 的序列
> X5
[1] 1 8
> X6 <- seq(from=1,by=8,length=2)   # 创建起点为 1、步长为 8、长度为 2 的序列
> X5
[1] 1 9
```

3．重复生成法

重复生成法是利用 rep() 函数将某个对象进行指定次数的重复，进而减少人工输入的一种方法。rep() 函数的语法格式为：

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

其中，x 指定需要循环的对象，times 指定 x 的循环次数（x 的整体在循环），each 指定 x 中元素的循环次数（依次将 x 的元素进行循环）。

【例 1-10】通过重复生成法录入公司 2020—2022 年各季度的销售额。

输入代码如下：

```
> Year <- rep(x=2020:2022,each=4)          # 生成 2020—2022 年的年份信息
> Quarter <- rep(x=1:4,times=3)            # 生成第 1~4 季度的季度信息
> Sales <- c(9.6,8.2,11.1,12.9,13.4,16.2,20.6,31.8,30.6,35.4,39.6,29.5)
                                           # 手动输入销售额信息
> DF <- data.frame(Year,Quarter,Sales)     # 将 3 个变量组装为数据框对象
> View(DF)                                 # 预览数据
```

创建的数据框对象如图 1-12 所示，读者应观察 Year、Quarter 两个向量创建的差异。

	Year	Quarter	Sales
1	2020	1	9.6
2	2020	2	8.2
3	2020	3	11.1
4	2020	4	12.9
5	2021	1	13.4
6	2021	2	16.2
7	2021	3	20.6
8	2021	4	31.8
9	2022	1	30.6
10	2022	2	35.4
11	2022	3	39.6
12	2022	4	29.5

图 1-12　创建数据框对象

4．目标抽取法（向量索引）

在介绍目标抽取前，先介绍一下向量索引。向量中的元素是按照顺序排列的，通过索引的方法可以将向量中的元素提取出来。在 R 语言中，索引使用方括号“[]”表示，包括位置索引与 bool 索引两种方法。

（1）位置索引是指在中括号内标明目标元素的下标，如向量第 5 个元素可以写为“[5]”，当取出向量中的多个元素时，需要将整数型的下标值写成向量的形式，如向量的第 2、4、6 个元素可以写为“[c(2,4,6)]”。

（2）bool 索引是指方括号“[]”内的值不是整数型下标，而是 TRUE 或 FALSE，索引时取出 TRUE 所对应的值。bool 索引经常会与比较运算符（>、>=、<、<=、==、!=）配合使用，相比于位置索引，bool 索引使用得更加频繁。

回到目标抽取法，它是指从已知向量中提取子集（由该向量的部分元素组成新的向量），或从矩阵中抽取一行或一列，或从数据框中抽取一列，进而得到数值型、字符型或日期时间型的向量。

矩阵或数据框的操作将在后文介绍。向量子集的提取通过方括号来实现，具体方式如下：

（1）通过在中括号中指定正整数来返回向量指定位置的元素组成的子向量（R 语言中向量的元素起始位置为 1）。

（2）通过在方括号中指定逻辑值向量来返回向量中对应的逻辑值为 TRUE 的元素组成的子向量。

（3）通过在方括号中指定负整数来返回向量除去指定位置的元素组成的子向量。方括号内不能同时包含正整数和负整数。

（4）如果向量已经命名，则可以通过在方括号中指定元素的名称来返回相应的子向量。

【例 1-11】向量子集的提取示例。

输入代码如下：

```
> X <- 1:12
> names(X) <- c('A','B','C','D','E','F','G','H','I','J','K','L')
> X
 A  B  C  D  E  F  G  H  I  J  K  L
 1  2  3  4  5  6  7  8  9 10 11 12
> X[5:8]                          # 通过正整数提取向量子集
E F G H
```

```
5 6 7 8
> X[c(6,6,8)]                          # 通过正整数提取向量子集
F F H
6 6 8
> X[-c(6,8)]                           # 通过负整数提取不包含响应元素的向量子集
 A  B  C  D  E  G  I  J  K  L
 1  2  3  4  5  7  9 10 11 12
> Y <- c(rep(TRUE,3),rep(FALSE,2))     # 创建逻辑向量
> Y
[1]  TRUE  TRUE  TRUE FALSE FALSE
> X[Y]                                 # 通过逻辑向量提取向量子集
 A  B  C  F  G  H  K  L
 1  2  3  6  7  8 11 12
```

1.4.3 矩阵与数组

前面介绍的向量实际上就是一个一维数组。而矩阵是一个二维数组，其中的每个元素都是相同的数据类型。

1．创建矩阵和数组

在 R 中，用 matrix() 函数可以创建矩阵，其语法格式为：

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

其中，data 指定一个用于构造矩阵的一维向量；nrow 指定矩阵的行数（默认为 1）；ncol 指定矩阵的列数（默认为 1）；byrow 为布尔型的参数，指在矩阵构造过程中元素是按列填充（byrow=FALSE）还是按行填充（byrow=TRUE）；dimnames 用于设置矩阵的行和列的名称，需将行、列名称以列表的形式传递给该参数。

数组与矩阵类似，但数组的维数可以大于 2。在 R 中，使用 array() 函数创建数组，其语法格式为：

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

其中，data 是一个包含数组中数据的向量；dim 指定每个维度的最大长度；dimnames 是各维度名称标签的一个列表。

另外，在 R 中通过 as.matrix() 函数可以将数据框强制转换为矩阵，其语法格式为：

```
as.matrix(x,rownames.force=NA)
```

其中，x 指定一个数据框，并将其强制转换为矩阵；rownames.force 为布尔型参数，将 x 强制转换为矩阵后，矩阵是否包含字符型的列名称，默认为 NA，表示矩阵列名称与数据框变量名称一致。

使用 t() 函数可以实现矩阵的转置，其语法格式为：

```
t(mat)                              # 将矩阵 mat 转置
```

【例 1-12】创建矩阵与数组示例。

输入代码如下：

```
> X <- matrix(5:16,nrow=3,ncol=4)
> X
     [,1] [,2] [,3] [,4]
[1,]    5    8   11   14
[2,]    6    9   12   15
[3,]    7   10   13   16
> XX <- t(X)                        // 矩阵转置
> XX
     [,1] [,2] [,3]
[1,]    5    6    7
[2,]    8    9   10
[3,]   11   12   13
[4,]   14   15   16
> Y <- array(letters[1:16],dim=c(2,4,2))
> Y
,,1
     [,1] [,2] [,3] [,4]
[1,] "a"  "c"  "e"  "g"
[2,] "b"  "d"  "f"  "h"
,,2
     [,1] [,2] [,3] [,4]
[1,] "i"  "k"  "m"  "o"
[2,] "j"  "l"  "n"  "p"
```

2．提取子集（矩阵索引）

矩阵子集的抽取（索引）与向量子集的抽取（索引）相似，不同的是向量子集是基于一维数据提取的，而矩阵子集则是基于二维数据提取的。

矩阵子集的提取方法为 [row_index,col_index]，其中 row_index 控制矩阵提取的行，col_index 控制矩阵要取的列。

【例 1-13】矩阵子集的提取示例。

输入代码如下：

```
> Mat <- matrix(1:24,ncol=6)          # 创建 4×6 的矩阵
> Mat
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    5    9   13   17   21
[2,]    2    6   10   14   18   22
[3,]    3    7   11   15   19   23
[4,]    4    8   12   16   20   24
> Mat[3,]                             # 取出第 3 行的数据
[1]  3  7 11 15 19 23
> Mat[,2]                             # 取出第 2 列的数据
[1] 5 6 7 8
> Mat[,5]
[1] 17 18 19 20
> Mat[3,4]                            # 取出第 3 行第 4 列的数据
[1] 15
> Mat[2:3,2:5]                        # 取出第 2~3 行、第 2~5 列的数据
     [,1] [,2] [,3] [,4]
[1,]    6   10   14   18
[2,]    7   11   15   19
> Mat[1:dim(Mat)[1]%%2==1,1:dim(Mat)[2]%%2==0]    # 取出奇数行偶数列的数据
     [,1] [,2] [,3]
[1,]    5   13   21
[2,]    7   15   23
```

提示　取出矩阵中的单行或单列时，应的 col_index 或 row_index 不需要设置。其中 dim 函数以向量形式返回矩阵的行数和列数，dim(Mat)[1] 表示仅返回矩阵 Mat 的行数。

1.4.4　数据框

数据框即数据表，表中不同的字段可以是不同的数据类型，因此构成数据框的各字段可

以是不同数据类型的向量。而向量和矩阵的元素不允许同时出现多种数据类型。

构造数据框可以利用 data.frame() 函数手动创建，或者利用 as.data.frame() 函数将矩阵或列表强制转换为数据框，也可以通过读取外部数据形成数据框。

函数 data.frame() 的语法格式为：

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

部分参数的含义如表 1-2 所示。

表1-2 参数含义

参 数	含 义
...	指定多个用于构造数据框的长度相等的向量
row.names	指定数据框的行名称（默认为1~n的整数）
check.rows	bool型参数，确认是否检查行名称row.names与数据框的行数一致（默认为FALSE）
check.names	bool型参数，确认是否检查数据框列名称的合理性和重复性（默认为TRUE）
fix.empty.names	bool型参数，当数据框没有列名称时，确认是否将其修正为V1、V2……（默认为TRUE）
stringsAsFactors	bool型参数，确认是否将字符串向量强制转换为因子型向量（默认为TRUE）

函数 as.data.frame() 的语法格式为：

```
as.data.frame(x,row.names=NULL)
```

其中，x 指定待转换为数据框的对象，可以是列表或矩阵。

【例 1-14】数据框创建示例——手动构造学生信息的向量。

输入代码如下：

```
> ID <- 1:6
> Name <- c('Jeff','Tom','Mary','Mike','Mike','Kris')
> Gender <- c('Male','Male','Female','Male','Male','Female')
> Birthday <- c('1995-6-18','1995-4-11','1996-2-8',
                '1995-8-11','1996-1-23','1995-12-19')
> Height <- c(177,182,168,179,173,165)
> Weight <- c(65.3,74.2,57.8,70.4,68.9,55.4)
> Stu_info <- data.frame(ID,Name,Birthday,Gender,Height,Weight)
                                        # 将向量组合为数据框
> View(Stu_info)                        # 数据预览
```

创建的数据框对象如图 1-13 所示。通过 data.frame() 函数方便将 6 个向量组合为一张数据表，并且表中的字段包含字符型、数值型和日期型。

	ID	Name	Birthday	Gender	Height	Weight
1	1	Jeff	1995-6-18	Male	177	65.3
2	2	Tom	1995-4-11	Male	182	74.2
3	3	Mary	1996-2-8	Female	168	57.8
4	4	Mike	1995-8-11	Male	179	70.4
5	5	Mike	1996-1-23	Male	173	68.9
6	6	Kris	1995-12-19	Female	165	55.4

图 1-13　创建数据框对象

组合为数据框的向量元素个数必须相等，否则会返回错误信息。

当数据框中的行和列较多时，使用 head() 函数可以只显示数据框的前几行，使用 tail() 函数可以只显示数据框的后几行。

```
> head(Stu_info,2)              # 只显示数据的前 2 行，不指定值时，默认显示前 6 行
  ID Name  Birthday Gender Height Weight
1  1 Jeff 1995-6-18   Male    177   65.3
2  2  Tom 1995-4-11   Male    182   74.2
> tail(Stu_info,2)              # 只显示数据的后 2 行，不指定值时，默认显示后 6 行
  ID Name   Birthday Gender Height Weight
5  5 Mike  1996-1-23   Male    173   68.9
6  6 Kris 1995-12-19 Female    165   55.4
```

当数据量比较大时，使用 str() 函数可以只查看数据的结构。如查看数据框 Stu_info 的数据结构，可使用下面的代码：

```
> str(Stu_info)                 # 查看 Stu_info 的数据结构
'data.frame':   6 obs. of  6 variables:
 $ ID      : int  1 2 3 4 5 6
 $ Name    : chr  "Jeff" "Tom" "Mary" "Mike" ...
 $ Birthday: chr  "1995-6-18" "1995-4-11" "1996-2-8" "1995-8-11" ...
 $ Gender  : chr  "Male" "Male" "Female" "Male" ...
 $ Height  : num  177 182 168 179 173 165
 $ Weight  : num  65.3 74.2 57.8 70.4 68.9 55.4
```

结果显示，Stu_info 是一个数据框，共有 6 个变量，每个变量又有 6 个观测值。

另外，还有其他函数可以查看数据框的类型、行数、列数等其他信息。

```
> class(Stu_info)              # 使用 class() 函数查看数据框的类型
[1] "data.frame"
> nrow(Stu_info)               # 查看数据框的行数
[1] 6
> ncol(Stu_info)               # 查看数据框的列数
[1] 6
> dim(Stu_info)                # 查看数据框的行数和列数
[1] 6 6
```

当需要对数据框中的特定变量进行分析或绘图时，使用“$”符号指定要分析的变量。

```
> Stu_info$Height              # 指定身高（Height，列）
[1] 177 182 168 179 173 165
> Stu_info[,5]                 # 同上
> Stu_info[,5 : 6]             # 通过下标指定身高（Height）及体重（Weight）
  Height Weight
1    177   65.3
2    182   74.2
3    168   57.8
4    179   70.4
5    173   68.9
6    165   55.4
> Stu_info[,c(5,6)]            # 同上
> Stu_info[5,]                 # 指定第 5 行的数据
  ID Name  Birthday Gender Height Weight
5  5 Mike 1996-1-23   Male    173   68.9
> Stu_info[c(2,4),]            # 指定第 2 行、第 4 行的数据
  ID Name  Birthday Gender Height Weight
2  2  Tom 1995-4-11   Male    182   74.2
4  4 Mike 1995-8-11   Male    179   70.4
```

使用 rbind() 函数可以将不同的数据框按行合并，使用 cbind() 函数可以将不同的数据框按列合并。为保证有意义地合并，当按行合并时，数据框中的列变量必须相同；当按列合并时，数据框中的行变量必须相同。

1.4.5 列表

列表用以存储包括常数、向量、矩阵、数据框在内的任意一种数据对象，甚至可以嵌套列表。列表的元素可以是异质的，行数也可以不同。

1．构造列表

列表的构造使用 list() 函数，其语法格式为：

```
list(…)
```

其中，…为常数、向量、矩阵、数据框在内的任意一种数据对象。

【例 1-15】创建包含常数、字符型向量、矩阵和数据框 4 个元素的列表。

输入代码如下：

```
# 创建列表元素的对象
> Constant <- 20
> Vector <- c('本科','本科','硕士','本科','博士')
> Mat <- matrix(data=1:9,ncol=3)
> DF <- data.frame(ID=1:5,Age=c(22,23,26,23,28),
                   Gender=c('女','男','男','女','男'),
                   Income=c(10500,9800,18000,14000,26000))
> List_object <- list(A=Constant,B=Vector,Mat,D=DF)          # 构造列表
> List_object
$A
[1] 20

$B
[1] "本科" "本科" "硕士" "本科" "博士"

[[3]]
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

$D
  ID Age Gender Income
1  1  22     女  10500
2  2  23     男   9800
3  3  26     男  18000
4  4  23     女  14000
5  5  28     男  26000
```

说明 在构造列表时，第 3 个元素并没有将 Mat 传递给一个新的变量名，因此第 3 个元素的输出以“[[3]]”作为名称。

2. 列表索引

列表的索引有单方括号“[]”、双方括号“[[]]”和美元符号“$”3 种形式，区别在于返回的元素是列表型数据结构还是其本身的数据结构。

单括号索引方式返回的一定是列表型对象，而非元素的原始结构；双括号索引或美元符号索引方式返回的一定是元素的原始结构。

【例 1-16】利用上例中创建的列表 List_object 演示列表的索引，检查返回列表中元素的数据结构。

输入代码如下：

```
> Return_A <- List_object[1]          # 中括号索引
> class(Return_A)
[1] "list"
> Return_B <- List_object[[2]]        # 双中括号索引
> class(Return_B)
[1] "character"
> Return_C <- List_object[[3]]        # 双中括号索引
> class(Return_C)
[1] "matrix" "array"
> Return_D <- List_object$D           # 美元符号索引
> class(Return_D)
[1] "data.frame"
```

如结果所示，第一个元素通过单中括号的索引方式返回列表型对象，第二个元素通过双中括号的索引方式返回字符型向量，第三个元素通过双中括号的索引方式返回矩阵型对象，第四个元素通过美元符号索引方式返回数据框对象。

注意 在返回原始的数据结构时，若列表元素有名称（如 List_object 中的 A、B 和 D），则可以使用双括号或美元符号；若列表元素没有名称（如 List_object 中的 [[2]]），则只能使用双括号的索引方式。

1.4.6 因子

在数据分析中，变量或数据（变量的观测结果）基本可以分为类别变量（Categorical Variable）与数值变量（Metric Variable）两大类。

（1）类别变量是取值为对象属性或类别以及区间值（Interval Value）的变量，也称定性变量（Qualitative Variable）。例如性别可以分为“男”“女”两类，“性别”就是类别变量，当把成绩（满分 100 分）等级分为 85~100（优）、75~84（良）、60~74（中）及 60 以下（差），“成绩等级”为数值区间，因而也属于类别变量，类别变量的观测值就是类别数据。

类别变量根据取值是否有序可分为无序类别变量和有序类别变量。无序类别变量的各类别间是不可以排序的，而有序类别变量的各类别间是有序的，如成绩分为“优”“良”“中”“差”就是有序的。取区间值的变量自然是有序类别变量。

（2）数值变量是取值为数字的变量，变量的观测结果称为数值数据（Metric Data）或定量数据（Quantitative Data）。数值变量根据其取值的不同可以分为离散变量和连续变量。离散变量是只能取有限个值的变量，其取值可以列举；连续变量是可以在一个或多个区间中取任何值的变量。

类别变量在 R 中称为因子（Factor），因子的取值称为水平（LEVEL），很多分析或绘图都可以按照因子的水平进行分类处理。使用 factor() 函数可以将向量编码为因子。

【例 1-17】将向量编码为因子示例。

输入代码如下：

```
> va <- c("优","良","中","差")     # 创建向量 va
> va
[1] "优" "良" "中" "差"
> fac1 <- factor(va)               # 将向量 a 编码为因子
> fac1
[1] 优 良 中 差
Levels: 差 良 优 中
> as.numeric(fac1)                 # 将因子 a 转换为数值
数值
[1] 3 2 4 1
```

可以发现，上述因子是无序的。若将有序因子转换为数值，则需要将 factor() 函数中的参数设置为 ordered=TRUE（默认 ordered=FALSE）。

```
> fac2 <- factor(va,ordered=TRUE,levels=va)        # 将向量 va 编码为有序因子
> fac2
[1] 优 良 中 差
Levels: 优 < 良 < 中 < 差
> as.numeric(fac2)                                 # 将因子 a 转换为数值
[1] 1 2 3 4
```

1.5 数据存取与抽样

前面介绍了各类 R 数据的创建方法，而在实际工作中，要分析或绘图的数据是已有数据，因此在分析前，只需要将这类数据读入 R 即可。R 可以读取多种类型的数据，也可以读取数据库中的数据，还可以在网上爬取数据。

1.5.1 数据存取

图表绘制通常使用外部保存的数据文件，R 可以读取不同格式（包括 CSV、TXT，以及 Excel、SQL、HTML 等数据文件）的外部数据。

1. 读入 R 格式的数据

R 语言系统除自带数据集外，本身还提供 *.RData 和 *.rds 两种数据存储格式。通过 load() 函数和 readRDS() 函数可以分别实现 *RData 格式和 *rds 格式数据的读取。

（1）RData 格式文件属于非标准化存储，既可以存储数据，又可以存储当前工作空间中的所有变量。

（2）RDS 格式文件属于标准化存储，仅用于存储单个 R 对象，且存储时可以创建标准化档案。

当数据本身为 R 格式，或已将其他格式数据保存为 R 格式，可以直接使用 load() 函数将指定路径下的数据读入（加载）R 中。

2. 读取 CSV/TXT 格式的数据

CSV 或 TXT 格式的数据是学习或工作中常见的文本型数据。其中，CSV 格式数据是一种通用的数据格式，其他很多类型的数据均可转换为 CSV 格式。使用 read.table() 与 read.csv() 函数可以很容易将 CSV、TXT 格式数据读入 R 中，这里只介绍 read.csv()。

```
read.table(file,header=FALSE,sep="",quote="\"'",dec=".",
           row.names,col.names,as.is=!stringsAsFactors,tryLogical=TRUE,
           na.strings="NA",colClasses=NA,nrows=-1,
           skip=0,check.names=TRUE,fill=!blank.lines.skip,
           strip.white=FALSE,blank.lines.skip=TRUE,
           comment.char="# ",
           allowEscapes=FALSE,flush=FALSE,
           stringsAsFactors=FALSE,
           fileEncoding="",encoding="unknown",text,skipNul=FALSE)

read.csv(file,header=TRUE,sep=",",quote="\"",
         dec=".",fill=TRUE,comment.char="",...)
```

部分参数的含义如表 1-3 所示。

表1-3　参数含义

参　数	含　义
file	指定需要读取的文件路径（需包含路径和文件名，如'D:/Rdata/test.csv'，路径需采用反斜杠“/”，或者双斜杠“\\”）
header	指定是否需要将原始数据集中的第一行（字段名称）作为表头，对于read.csv函数，默认为TRUE
sep	指定原始数据集中字段间的分隔符，对于read.csv函数，默认为','
quote	指定值的引号方式，对于read.csv函数，默认为双引号
dec	指定浮点型数据的小数点格式，默认为英文状态下的点
fill	在原始数据集中，如果行内值的个数不相等，是否用空白填充，对于read.csv函数，默认为TRUE
comment.char	通过指定字符型的注释符，使得读取数据时跳过这些注释符开头的行记录；对于read.csv函数，默认为空字符（""）
stringsAsFactors	确定是否需要将字符型变量强制转换为因子型变量，默认为FALSE

【例 1-18】文件读取示例。

输入代码如下：

```
> TableA <- read.csv("D:/Rdata/d_table.csv") # 读取含有标题的 CSV 格式数据
> TableB <- read.csv("D:/Rdata/d_table.csv",header=FALSE)
                                              # 读取不含标题的 CSV 格式数据
> load("D:/Rdata/d_table.RData")              # 读取加载 R 格式数据
```

3. 读取 Excel 格式数据

使用 xlsx 包中的 read.xlsx() 函数和 read.xlsx2() 函数导入 .xlsx 格式的数据文件。在实际

工作中建议使用 CSV 格式导入数据文件。

```
TableA <- read.xlsx("D:/Rdata/Data.xlsx",sheetIndex=1)
```

也可以使用 write.xlsx() 函将数据文件导出为 XLSX 格式：

```
write.xlsx(TableA," D:/Rdata/Data.xlsx",sheetName="Sheet Name")
```

注意 在使用 R ggplot2 绘图时，通常使用一维数据列表的数据框。当导入的数据表是二维数据列表时，需要使用 reshape2 包的 melt() 函数或者 tidyr 包的 gather() 函数将二维数据列表的数据框转换成一维数据列表。

4．保存数据

当在 R 中录入新数据，或者想要对读入的数据以指定格式保存在指定的路径中时，可以使用 write.table() 函数。当需要以 CSV 格式保存在指定的路径中时，建议使用 write.csv() 函数。

```
write.table(x,file="",append=FALSE,quote=TRUE,sep=" ",
            eol="\n",na="NA",dec=".",row.names=TRUE,
            col.names=TRUE,qmethod=c("escape","double"),
            fileEncoding="")

write.csv(...)
```

部分参数的含义如表 1-4 所示。

表1-4 参数含义

参 数	含 义
x	指定需要保存的数据名称，可以是矩阵格式，也可以是数据框格式
file	指定数据保存后的文件名称（含文件格式，如CSV或TXT等），可带路径
append	bool型参数，是否需要将数据追加到已存在的外部数据集中，默认为FALSE；在write.csv函数中，该参数值不能修改
quote	传递bool型值，或者数值向量，默认为TRUE，即对于字符型变量，变量中的值会添加双引号；如果参数接受的是数值向量，则表示对应下标的字符型变量值将添加双引号
sep	指定输出数据集中各变量之间的分隔符，默认为空格；在write.csv函数中，参数值不能修改
na	指定输出数据集中，缺失值的表示方法，默认为NA
dec	指定输出数据集中，小数点的表示方法，默认为英文状态下的句号点

【例 1-19】文件保存示例。

输入代码如下：

```
> write.csv(dN_table,file="D:/Rdata/d_table.csv")
                                  # 将数据保存为 CSV 格式，并存放在指定路径中
> save(dN_table,file="D:/Rdata/d_table.RData")
                                  # 将数据保存为 R 格式，并存放在指定路径中
```

1.5.2 数据抽样

1. 生成随机数

工作中，有时需要生成各类分布的随机数做模拟分析，R 中产生随机数时，只需在相应分布函数前加字母“r”即可，如生成均值为 0、标准差为 1 的正态分布随机数，代码如下：

```
rnorm(n,mean=0,sd=1)    # 产生 n 个服从正态分布的随机数，其均值为 0，标准差为 1
```

如果需要每次运行都产生相同的一组随机数，可在生成随机数之前使用 set.seed() 函数设定随机数种子。例如：

```
set.seed(9)
```

使用相同的随机数种子，每次运行都会产生一组相同的随机数。

【例 1-20】产生随机数示例。

输入代码如下：

```
> rnorm(6)              # 产生 6 个标准正态分布随机数
[1] -0.6100317 -0.7621257  0.5379864 -0.9037370  0.4651411  1.0644905
> set.seed(9)           # 设定随机数种子
> rnorm(8,25,2)         # 产生 8 个均值为 25、标准差为 2 的正态分布随机数
[1] 26.24514 23.43874 24.46430 24.29931 22.85433 25.25595 28.95925 25.66443
> runif(6,0,2)          # 在 0~20 产生 6 个均匀分布随机数
[1] 1.7958039 1.8487793 0.9142511 0.1712717 0.9751896 0.9961681
```

2. 随机抽样

实验获取的数据会比较庞大，实际应用需要从中抽取一个简单随机样本以作为分析样本。R 的 sample() 函数可以实现随机抽样，其语法格式为：

```
sample(x,size,replace=FALSE,prob=NULL)
```

其中，x 是由一个或多个元素组成的向量，size 是要抽取的元素个数（样本量），replace 确定是否采取放回抽样，设置为 TRUE 表示有放回抽样，默认为 FALSE（不放回抽样），prob 是要抽取元素的概率权重向量。

【例 1-21】随机抽样示例。

输入代码如下：

```
> set.seed(10)                    # 设定随机数种子
> N <- rnorm(100,6,2)             # 产生 100 个均值为 6、标准差为 2 的正态分布随机数
> n1 <- sample(N,size=8)          # 无放回随机抽取 8 个数据
> n1
[1] 5.796478 4.644771 4.255682 7.785852 6.296336 7.511563 5.350912 2.786646
```

1.6 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。

1.6.1 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 1-5 所示。

表1-5 内置帮助函数的功能及用法

函数名称	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的对象（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 1-22】获取帮助信息示例。

输入代码如下：

```
> help(lda,package='MASS')    # 直接查询某个函数的帮助文档
> help.search('geom_bar')     # 从所有已下载的包中搜寻 geom_bar 函数
> RSiteSearch('Neural Network')      # 在线搜索包含关键词的帮助文档
> help(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
```

1.6.2 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站点：

（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。

1.7 本章小结

本章详细介绍了 R 语言的基础知识，包括 R 语言的概述、获取与安装、对象与变量、数据结构、数据存取与抽样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 语言实现数据可视化打下坚实的基础。