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 语言是一种用于统计分析和绘图的语言和操作环境。作为 GNU 系统的一部分，R 语言是自由、免费、开源的软件，是一个用于统计计算和制图的优秀工具。

1.1.1 R 语言的诞生

R 语言起源于 20 世纪 80 年代，作为 S 语言的一个分支而诞生。R 可以被视为 S 语言的一个实现，并在统计领域得到广泛应用。S 语言最初由 AT&T 贝尔实验室开发，是一种用于数据探索、统计分析和图形制作的解释型语言。S 语言最初的商业实现版本是 S-PLUS。

S-PLUS 是一个基于 S 语言的商业软件，由 MathSoft 公司的统计科学部进行进一步开发。随后，在新西兰奥克兰大学，由 Robert Gentleman 和 Ross Ihaka 以及其他志愿者共同开发了 R 系统。

R 语言可以看作是 AT&T 贝尔实验室的 Rick Becker、John Chambers 和 Allan Wilks 所开发的 S 语言的一个实现版本。因此，R 和 S 在程序语法上几乎完全相同，尽管在函数方面可能存在一些细微的差异。这些差异使得程序能够在两种语言之间轻松移植，而且许多 S 语言的程序只需稍作修改即可在 R 语言中使用。

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，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提供支持。

（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 上有数以千计的扩展包，这些包涵盖了数据处理、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）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。这两种绘图系统分别对应 R 语言核心包中的 graphics 包和 grid 包。

- 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。
- grid 包则提供了一系列不同的绘图函数。因为 grid 包并没有提供一套完整的绘图函数，不能直接用于绘图，所以在 grid 包的基础之上又发展出了 lattice 和 ggplot2 这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 扩展包等，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 图形语法

一张统计图形是从数据（data）到几何对象（geometric object，缩写为 geom）的图形属性（aesthetic attributes，缩写为 aes）的一个映射（mapping）。图形中还可能额外包含数据的统计变换（statistical transformation，缩写为 stats），最终绘制在某个特定的坐标系（coordinate system，缩写为 coord）中，并通过分面（facet）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组成的。

（1）数据：图形最基础的部分就是想要可视化的数据，以及一系列将数据中的变量对应到图形属性的映射。

（2）图层（layer）：由几何元素和统计变换组成。

（3）几何对象：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：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：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格线。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：将绘图窗口划分为若干个子窗口，描述如何将数据分解为不同子集，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，但有时会进行调整以绘制自己想要的图形。

1.2 搭建 R 语言环境

R 语言可以在 CRAN（Comprehensive R Archive Network）网站上免费下载。CRAN 是一个网络集合体，包含发布版本、资源包、文档和源代码。它由遍布全球的几十个镜像站点组成，这点站点提供下载安装程序和相应版本的资源包。镜像站点一般 1~2 天更新一次。

1.2.1 安装程序的下载

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（package），读者根据自己的系统平台选择对应安装包下载安装即可。下面以 Windows 平台为例，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。

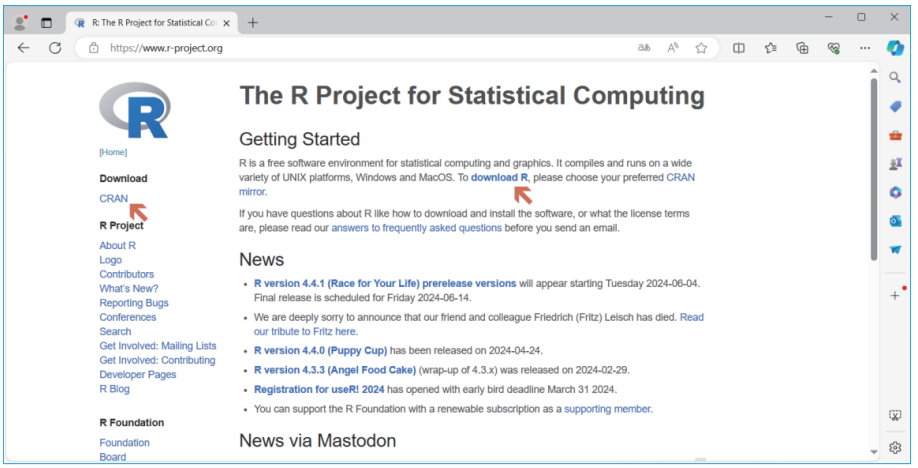

图 1-1 R 语言官网

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，在页面左侧找到 China，单击其中的一个镜像（推荐选用清华大学镜像），如图 1-2 所示。

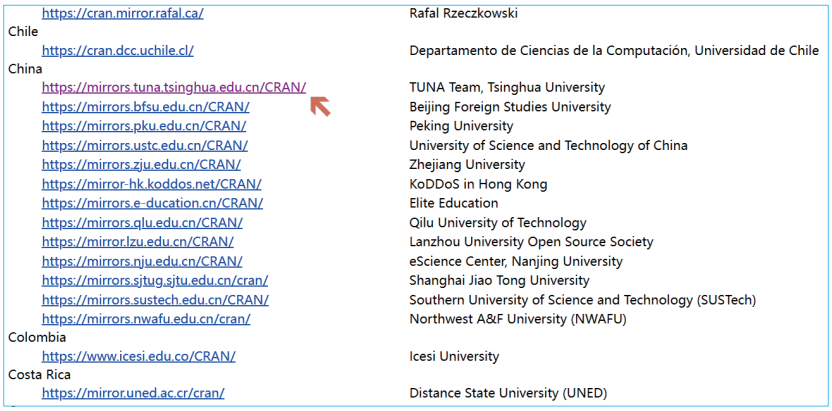

图 1-2　选择镜像站点

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

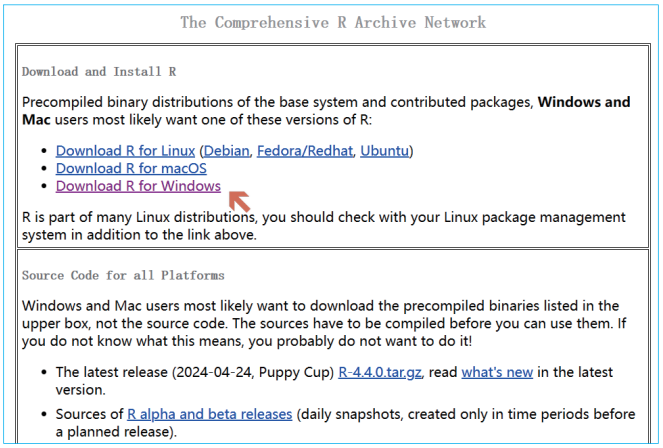

图 1-3　选择对应的平台版本

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。

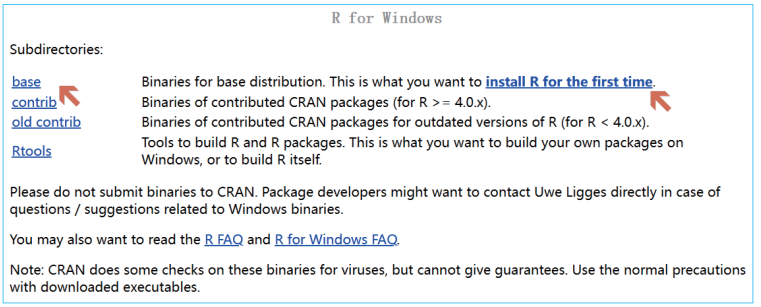

图 1-4 选择下载版本

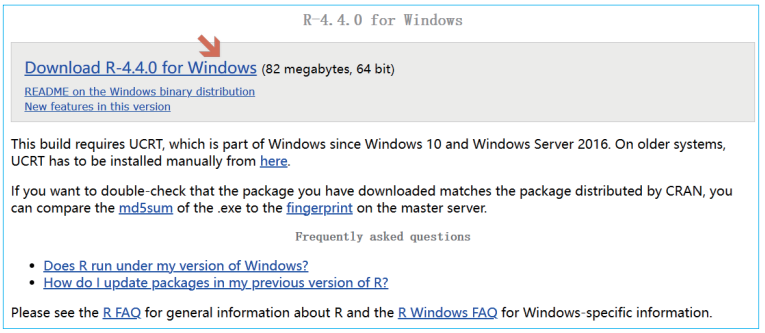

图 1-5 下载链接

1.2.2 R 语言的安装与启动

安装与启动 R 语言的步骤如下：

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

步骤 02 在弹出的“选择语言”对话框中选择“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图标，双击该图标即可启动 R 语言。首次启动后的 RGui 界面如图 1-6 所示。能够正常启动就说明安装成功。

说明 在 Windows 平台中安装 R 语言时，除了安装必要的核心文件之外，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-4.4.0\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。

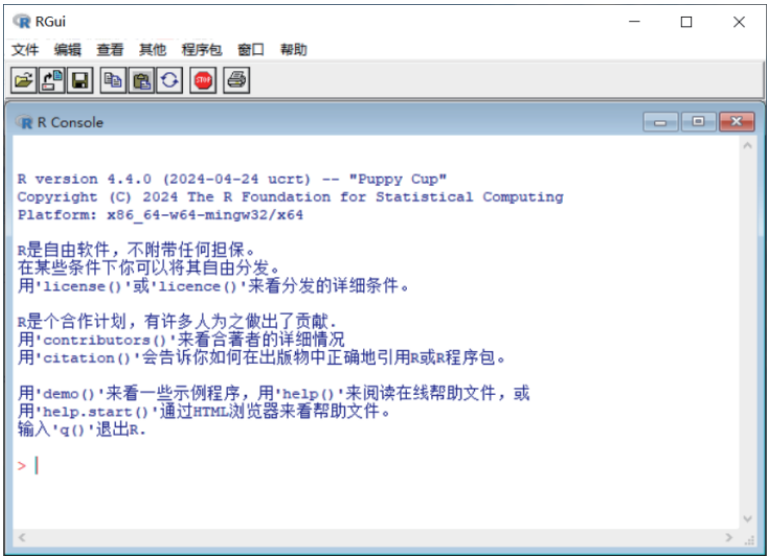

图 1-6　R 语言主界面（RGui 界面）

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 语言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，每次可以输入一条命令，也可以连续输入多条命令，各命令之间用分号（;）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。

在控制台中输入以下代码，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，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+8+34+98+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，类似续行符
+ 34+45+56+45-            # 此处最前的 "+" 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。下面介绍 RStudio 套件的安装。

1. Rstudio 的下载与安装

下载与安装 Rstudio 的步骤如下：

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 编者当前使用的版本为 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；或者右击该安装包，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 系统的“开始”菜单栏中出现 RStudio 快捷启动图标 rstudio，单击该图标即可启动 RStudio。首次启动后的 RStudio 界面如图 1-7 所示。能够正常启动就说明安装成功。

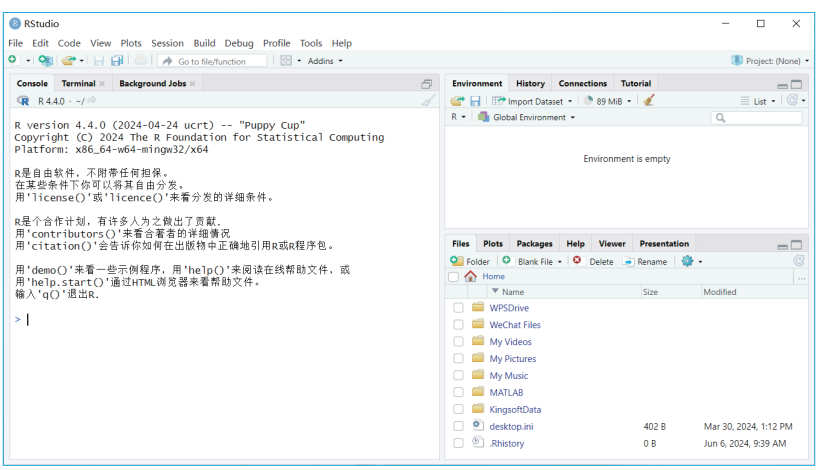

图 1-7 RStudio 主界面

2．Rstudio 主界面介绍

执行菜单栏中的 File → New File → R Script 命令，或单击主界面左上角的（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。

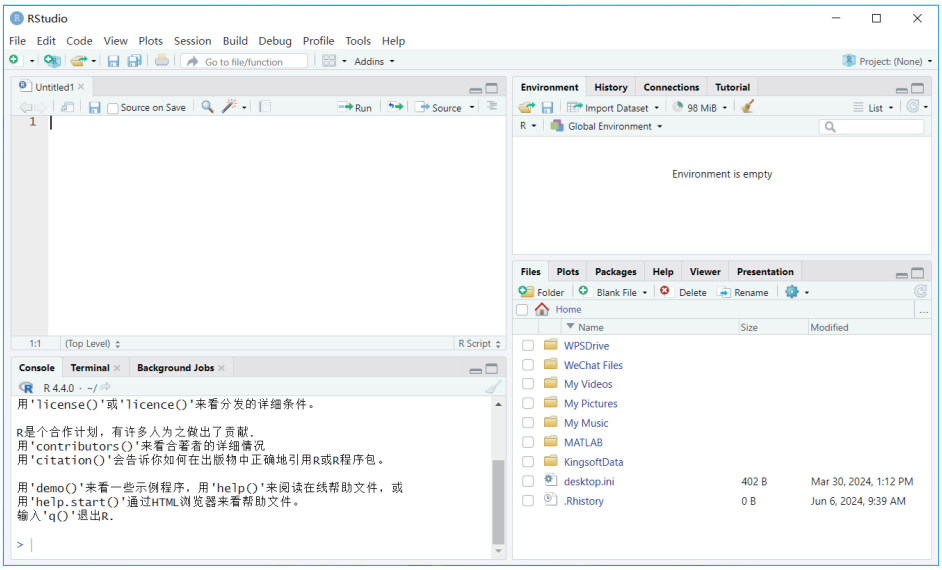

图 1-8　打开脚本窗口的主界面

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，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显示区域，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显示区域，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。例如，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。

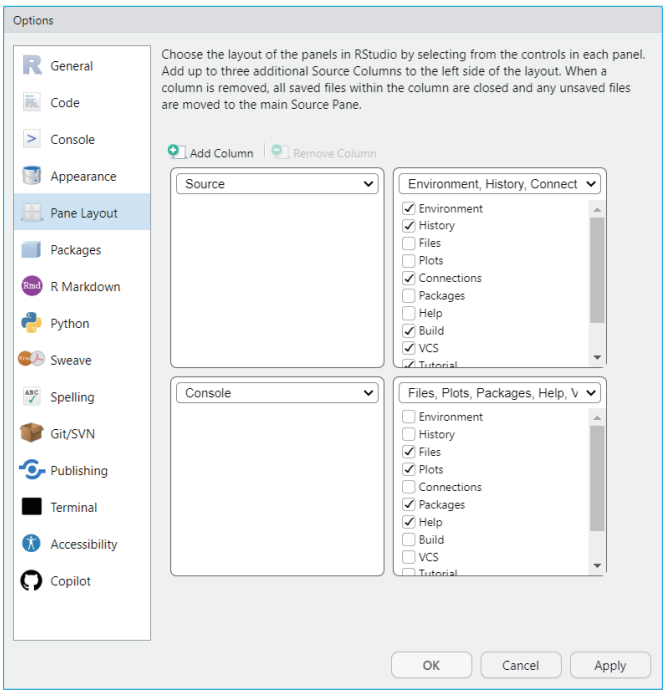

图 1-9 Options 对话框

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

1.2.4 包的安装与加载

R 语言中的包是指包含 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，然后方可使用。

一个 R 包中可能包含多个函数，能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，然后才能使用。

要查看 R 中已经安装的包，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自动完成包的下载和安装。读者可以一次性下载和安装多个包，下载时将多个带引号的包名称用逗号隔开即可。下载和安装包的语法格式如下：

```
install.packages("p_name")              # 包的名称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、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，然后浏览安装，如图 1-11 所示。

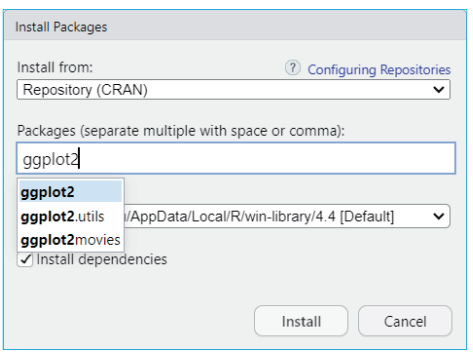

图 1-10 Install Packages 对话框

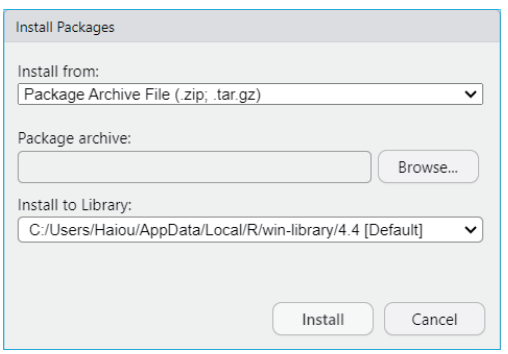

图 1-11 手动安装参数设置

4．通过 GitHub 安装包

有些 R 包只存放在 GitHub，也有很多 CRANR 包的最新开发版也位于 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“::”前面的是包名，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 包的标志是是否含有 DESCRIPTION 文件。

读者也可以将整个包文件夹从网页下载下来，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。

在控制台中输入以下代码：

```
> library(ggplot2)                           # 加载安装 ggplot2 包
> library(ggraph)                            # 加载安装 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_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、绘制柱状图等，每次分析都要输入数据就会非常麻烦。这时，可以将多个数据组合成一个数据集，然后对数据集赋予一个名字，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是可以称作对象，如向量、列表、函数、环境等。R 语言的所有代码都是基于对象的操作，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。

在控制台中输入以下代码：

```
> 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。

在本书中，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），然后对它赋值（可以是数值、向量、矩阵或数据框等）。赋值后的对象称为变量，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 变量名与变量值可以前后互换，同时赋值符号“<-”需要变为“->”，不推荐使用。继续在控制台中输入以下代码，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 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，这些数据集的数据结构如图 1-12 所示。

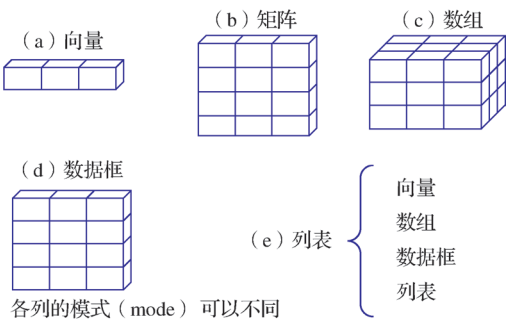

图 1-12 数据集类型

1.4.1 数据类型

在R语言中，常见的数据类型包括数值型(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。

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 语言中使用 class() 函数来识别数据类型；使用 is.*() 函数来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 语言中还包含有几种常见的特殊值，如表 1-1 所示。

表1-1　特殊的数据类型

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

【例 1-7】数据类型应用示例。

在控制台中输入以下代码，并显示输出结果。

```
> 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
> is.Date(x=Birthday)
[1] FALSE

> 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"
```

最常用的日期型数据类型是 Date（仅存储日期）和 POSIXct（同时存储日期与时间）。在处理日期型数据时，经常需要使用 as.Date() 函数将读入的数据从数值型转换成日期型，有时还需要进一步提取日期型数据的年、月、周等数据信息。

使用 as.numeric() 函数或 as.integer() 函数可以将日期型数据转换成数值型。使用 strftime(x,format="") 函数可以定义日期型数据的格式，如 strftime(a,'%Y') 表示只显示年份。

【例 1-8】日期型数据应用示例。

在控制台中输入以下代码，并显示输出结果。

```
> a <- as.Date("2024-06-06")
> class(a)                                         # 输出 a 的数据类型
[1] "function"
> b <- as.POSIXct("2024-06-06 16:26")
> class(b)                                         # 输出 b 的数据类型
[1] "POSIXct" "POSIXt"
> a_Year <- as.integer(strftime(a,'%Y'))
> a_Year                                           # 输出年份
[1] 2024
a_month <- as.integer(strftime(a,'%m'))
> a_month                                          # 输出月份
[1] 6
a_week<- as.integer(strftime(a,'%W'))
> a_week                                           # 输出周数
[1] 23
```

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 1-9】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。

在控制台中输入以下代码，并显示输出结果。

```
> 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)
```

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 表示在未知公差的情况下，通过它来设定等差数列的元素个数。

【例 1-10】序列生成法输入向量示例。

在控制台中输入以下代码，并显示输出结果。

```
> 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
[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
[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 1-11】通过重复生成法录入公司 2020 — 2022 年各季度的销售额。

在控制台中输入以下代码，并显示输出结果。

```
> 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 1-13 所示，读者应观察 Year 和 Quarter 这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 1-13 创建的数据框对象

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。

- 位置索引是指在方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，则需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)]”。
- bool 索引是指方括号（[]）内的值不是整数型下标，而是 TRUE 或 FALSE 值，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，进而可以得到数值型、字符型或日期时间型的向量。

其中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）通过在方括号中指定正整数来返回由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由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由向量中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 1-12】向量子集的提取示例。

在控制台中输入以下代码，并显示输出结果。

```
> 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相应元素的向量子集
 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 语言中，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，指定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 语言中，使用 array() 函数可以创建数组，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 语言中通过 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 是指要强制转换为矩阵的数据框；rownames.force 是一个布尔型参数，如果设置为 TRUE，则强制将数据框的列名称作为矩阵的行名称；如果设置为 FALSE 或 NA（默认值），则矩阵的列名称将与数据框的变量名称保持一致。

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 1-13】创建矩阵与数组示例。

在控制台中输入以下代码，并显示输出结果。

```
> 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，所不同的是向量子集是基于一维数据的提取，而矩阵子集是基于二维数据的提取。

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

【例 1-14】矩阵子集的提取示例。

在控制台中输入以下代码，并显示输出结果。

```
> 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 5 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 2~3 行，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
```

取出矩阵中的单行或单列时，对应的 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，例如 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。而向量和矩阵中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 部分参数的含义

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 为待转换为数据框的对象，可以是列表或矩阵。

【例 1-15】数据框创建示例——手动构造学生信息的向量。

在控制台中输入以下代码，并显示输出结果。

```
> 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 1-14 所示。通过 data.frame() 函数方便地将 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 1-14　创建的数据框对象

注意　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。继续在控制台中输入以下代码，并显示输出结果。

```
> 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。例如，查看数据框 Stu_info 的数据结构，可继续在控制台中输入以下代码，并显示输出结果。

```
> 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，还有其他函数可用于查看数据框的类型、行数、列数等信息。在控制台中输入以下代码，并显示输出结果。

```
> class(Stu_info)                # 使用 class() 函数可以查看数据框的类型
[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。继续在控制台中输入以下代码，并显示输出结果。

```
> 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。为保证合并有意义，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、数据框中的任意一种数据对象。

【例 1-16】创建包含常数、字符型向量、矩阵和数据框 4 个元素的列表。

在控制台中输入以下代码，并显示输出结果。

```
# 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
```

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，因此第 3 个元素的输出是以“[[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。

其中，单方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；双方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 1-17】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。

在控制台中输入以下代码，并显示输出结果。

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

在返回的结果中，第一个元素通过单方括号的索引方式返回列表型对象；第二个元素通

过双方括号的索引方式返回字符型的向量；第三个元素通过双方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），则可以使用双方括号或美元符号；若列表元素没有名称（如 List_object 中的 [[2]]），则只能使用双方括号的索引方式。

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

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，则“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）时，则“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

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，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 1-18】将向量编码为因子。

在控制台中输入以下代码，并显示输出结果。

```
> 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，需要将 factor() 函数中的参数设置为 ordered=TRUE（默认 ordered=FALSE）。

继续在控制台中输入以下代码，并显示输出结果。

```
> 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 基本运算

R 语言中的基本运算包括算术运算、逻辑运算、关系运算和赋值运算等。通过这些基本运算，可以让数据分析和统计计算变得更加直观和高效。

1. 算术运算

算术运算包括加、减、乘、除、取余和幂运算。算术运算符如表 1-3 所示。

表1-3 算术运算符

运 算 符	描 述	示 例	结 果	运 算 符	描 述	示 例	结 果
+	加	3+2	5	%%	取余数	7 %% 2	1
−	减	5−2	3	%/%	整除	7 %/% 2	3
*	乘	3*2	6	^或**	幂	2^3或2**3	8
/	除	6/2	3				

2. 逻辑运算

逻辑运算包括与、或和非运算。逻辑运算符如表 1-4 所示。

表1-4　逻辑运算符

运 算 符	描　述	示　例	结　果
&	元素级的与运算	TRUE & FALSE	FALSE
|	元素级的或运算	TRUE | FALSE	TRUE
!	非运算	!TRUE	FALSE
&&	短路与运算	TRUE && FALSE	FALSE
||	短路或运算	TRUE || FALSE	TRUE

3．关系运算

关系运算用于比较两个值。关系运算符如表 1-5 所示。

表1-5　关系运算符

运 算 符	描　述	示　例	结　果	运 算 符	描　述	示　例	结　果
==	等于	3==3	TRUE	!=	不等于	3 !=2	TRUE
>	大于	3 > 2	TRUE	>=	大于或等于	3 >=2	TRUE
<	小于	2 < 3	TRUE	<=	小于或等于	2 <=3	TRUE

4．赋值运算

赋值运算用于将值赋给变量。赋值运算符如表 1-6 所示。

表1-6　赋值运算符

运 算 符	描　述	示　例	结　果
<-	赋值运算	x <-5	x的值为5
=	赋值运算	y=3	y的值为3
->	赋值运算（右）	4-> z	z的值为4
<<-	全局赋值运算	x <<-6	x在全局环境中赋值为6
->>	全局赋值运算	7 ->> y	y在全局环境中赋值为7

【例 1-19】运算符应用示例。

在控制台中输入以下代码：

```
# 算术运算
a <- 5
b <- 2
print(a + b)                    # 输出：7
print(a - b)                    # 输出：3
print(a * b)                    # 输出：10
```

```
print(a / b)                          # 输出：2.5
print(a %% b)                         # 输出：1
print(a %/% b)                        # 输出：2
print(a ^ b)                          # 输出：25

# 关系运算
print(a==b)                           # 输出：FALSE
print(a !=b)                          # 输出：TRUE
print(a > b)                          # 输出：TRUE
print(a < b)                          # 输出：FALSE
print(a >=b)                          # 输出：TRUE
print(a <=b)                          # 输出：FALSE

# 赋值运算
x <- 10
y=20
30 -> z
print(x)                              # 输出 10
print(y)                              # 输出 20
print(z)                              # 输出 30
x <<- 15
print(x)                              # 输出：15（全局赋值）
40 ->> w
print(w)                              # 输出：40（全局赋值）
```

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。但这也造成了需要记忆太多的包或函数的问题，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至关重要。R 语言自身包含了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 1-7 所示。

表1-7 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()

（续表）

函数名称	功　能	示　例
?	查找某个函数的帮助	?t.test或?"t.test?"t.test ""
help	查找某个函数的帮助	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 的练习之用。利用 help() 函数可以了解数据集的信息。

```
help(date_name)                    # 查看数据集 date_name 的详细信息
```

【例 1-20】获取帮助信息示例。

在控制台中输入以下代码，输出结果略。

```
> 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
```

1.6.2 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 1-21】查看数据集示例。

在控制台中输入以下代码，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
```

1.6.3 R 语言相关软件和资料

除内置帮助函数外，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站：

（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 语言的各种问题及其解决方法，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。

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、基本运算以及获取帮助信息等内容。本章讲解的 R 语言的基础入门知识，通过学习本章内容，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。