R 语言与精算数据科学

张连增　　李浩男

2024-09-02

目录

Chapter 1

前言

R 语言是一种专门用于统计分析和数据可视化的编程语言和软件环境。它由统计学家 Ross Ihaka 和 Robert Gentleman 于 1993 年开发，并且是基于 S 语言的一种实现。R 语言在学术界、统计学界、数据科学以及相关领域中被广泛使用。

1.1 为什么选择 R?

1. **强大的统计功能**：R 语言内置了丰富的统计分析功能，支持回归分析、假设检验、时间序列分析、聚类分析等多种统计方法，且不断有新的包和函数被开发和维护。
2. **数据可视化**：R 语言提供了强大的数据可视化功能，尤其是通过 `ggplot2` 包可以创建高度定制化和复杂的图表。R 语言支持的图表种类非常多样，包括散点图、柱状图、箱线图、热图等。
3. **开放性与扩展性**：R 是一个开源项目，拥有庞大的用户和开发者社区。CRAN（Comprehensive R Archive Network）上有数千个扩展包，覆盖了几乎所有的数据分析需求，从生物信息学到金融分析。
4. **与其他语言和工具的兼容性**：R 可以与其他编程语言（如 Python、C++）以及数据管理工具（如 SQL、Hadoop）集成，这使得它能够处理大型数据集，并与其他数据处理工作流无缝衔接。
5. **交互性和易用性**：R 语言有丰富的交互式环境（如 RStudio），用户可以方便地编写、调试代码，并实时查看分析结果。此外，R 语言的语法相对直观，适合统计学背景的用户。
6. **广泛的应用领域**：R 在金融、医药、生物信息学、社会科学等多个领域都有广泛应用。它被用于学术研究、数据科学项目以及商业分析中。

1.2 精算与 R 语言

精算学是一个集数学、统计学、经济学、金融学及计算机科学于一体的综合性学科，它专注于对金融风险和不确定性进行量化评估，为保险、养老金、投资等行业提供决策支持。而 R 语言作为一种强大的统计分析和图形展示工具，在精算领域的应用日益广泛，以下是几个主要原因：

- 1. 数据分析和建模：精算师需要处理大量数据，包括保险赔付记录、客户资料、市场趋势等。R 语言提供了丰富的统计和数据分析包（如 `ggplot2` 用于数据可视化，`dplyr` 和 `tidyr` 用于

数据清洗和整理，`lm`、`glm` 等函数用于线性模型和非线性模型拟合），能够高效地帮助精算师进行数据清洗、探索性数据分析、模型构建和验证等步骤。

- 2. 风险管理和预测：精算的核心工作之一是评估未来风险并制定相应的保险费率、准备金等。R 语言能够支持各种复杂的预测模型，如时间序列分析（`forecast` 包）、生存分析（`survival` 包）以及机器学习算法（通过 `caret`、`randomForest` 等包），这些工具对于预测保险损失、评估长寿风险等至关重要。

- 3. 报告和可视化：精算师需要向管理层、监管机构或客户报告分析结果。R 语言不仅能处理复杂的数据分析，还能生成高质量的图形和报告，通过 R Markdown、Shiny 等工具，可以轻松地创建交互式报告和仪表板，使数据分析和结果呈现更加直观和易于理解。

- 4. 行业趋势和前沿技术：随着大数据和人工智能的发展，精算行业也在不断探索新技术的应用。R 语言作为开源社区活跃的统计软件，不断吸收和整合新的算法和技术，如深度学习、自然语言处理等，为精算师提供了接触和应用前沿技术的平台。

- 5. 就业竞争力：掌握 R 语言将极大地提升精算专业学生的就业竞争力。在保险、金融等行业，具备数据分析能力的人才越来越受到重视。能够熟练运用 R 语言进行数据分析、模型构建和报告呈现的学生，在求职过程中将更具优势。

在 CRAN 网站中包含了专门针对精算学的 R 语言项目集合，参考https://cran.r-project.org/web/views/ActuarialScience.html。其中提供了一系列 R 包，覆盖了寿险精算中的生存分析、死亡率模型；非寿险计算中的损失模型、产品定价、准备金计算；再保险与极值理论；精算相关的数据集等内容。

Chapter 2

你好, R!

2.1 R 软件: 安装 R 和 RStudio

R 可以从以下网站下载: http://cran.r-project.org/。R 可用于所有平台: Unix/Linux、Windows 和 Mac。在本课程中，我们还将使用 RStudio，一个 R 的界面。RStudio 可以从 www.rstudio.org 获取。RStudio 可作为桌面程序用于所有平台: Unix/Linux、Windows 和 Mac。在本教程中，我们将重点关注 R 的 Windows 实现。

2.2 R GUI 软件交互界面

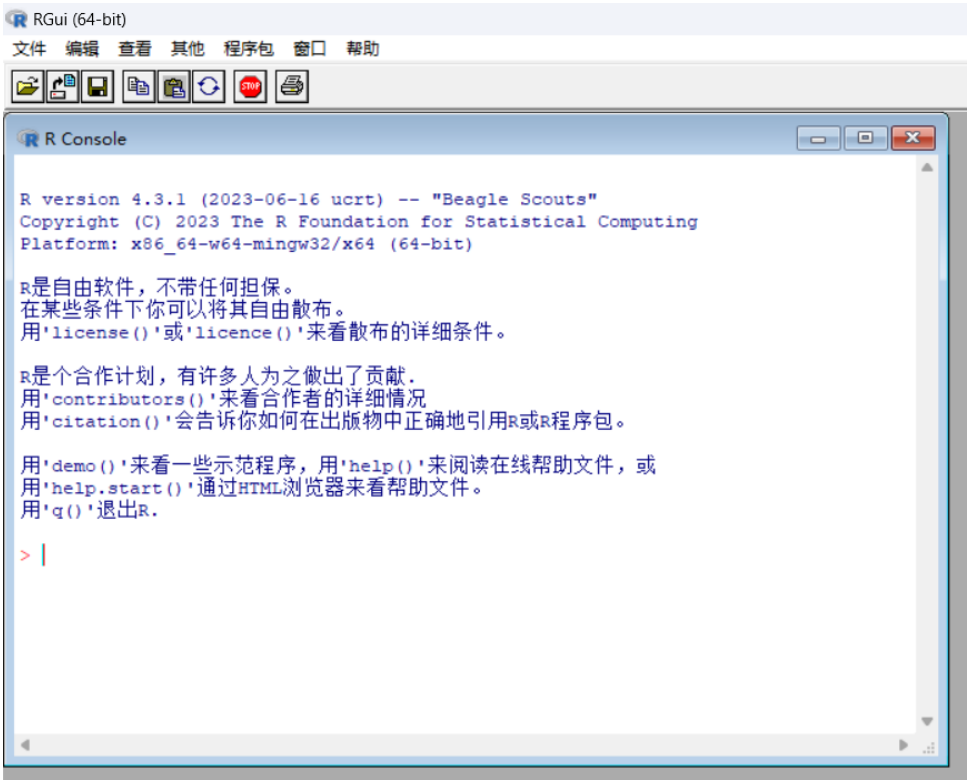


图 2.1: R 软件启动界面

当我们启动 R Studio 软件后，可以看到提示符 >，这个提示符现在正在等待你的命令，如图2.1。下面我们可以用一行命令实现一系列高级操作，生成了若干张图片。

```
demo(graphics)
```

这行代码帮助我们完成了 `demo` 函数的调用，在 R 中，调用函数时需要加上括号，括号内是函数的参数，我们会在后续的学习深入了解函数的概念，并实现自编 R 函数。如果我们要了解更多关于 `demo` 函数的参数和选项的信息，我们可以查看 `demo` 的帮助。要获取帮助，只需使用 `help` 命令或在命令前输入问号 `?`。感兴趣的话可以试试`?demo` 或 `help(demo)`，你会得到更多类似的命令。

2.2.1 中断和退出 R

如果你希望中断一个正在执行的代码，在 Windows 或 MacOS 上按 `ESC` 键，或者点击工具栏的“停止标志”（它是红色的）。如果你希望退出 R，你可以点击窗口框架右上角的红色 X（Windows），按 CMD+q（MacOS）。在所有平台上，你可以输入：

```
q()
```

在这个界面中，我们可以直接输入代码并得到结果，但是使用起来并不方便，推荐使用 RStudio。

2.3 RStudio 软件

RStudio 是为 R 语言设计的一个跨平台的集成开发环境（IDE），它将许多功能强大的编程工具集成到一个直观、易于学习的界面中。用户可以在其中编辑、运行 R 的程序文件，可以跟踪运行，还可以构造文字、图表融合在一起的研究报告等。一个运行中的 RStudio 界面见图2.2。

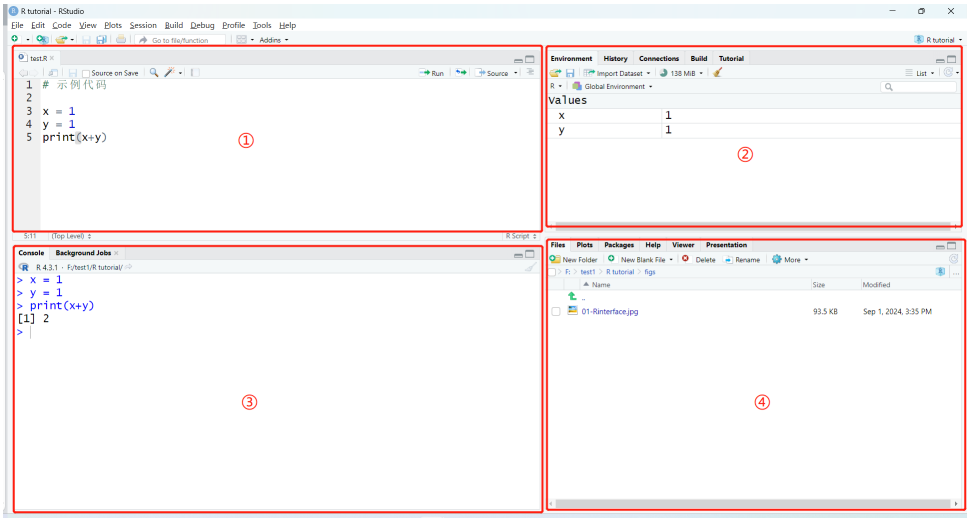


图 2.2: Rstudio 软件界面

4 个窗口分别是：

- 编辑器窗口：用于查看或编辑脚本，如 R 脚本等（如果未出现该窗口，用快捷键 Ctrl+Shift+N 新建一个脚本）。
- 控制台窗口：用于输入命令，以及显示 R 脚本的执行情况。

- 工作区窗口：一共有 4 个标签页，Environment 为环境窗口，可以暂时简单理解为查看变量的窗口；History 为历史窗口，用于以前运行过的命令；其他标签页暂不介绍。
- 文件等窗口：一共有 6 个标签页，Files 用于查看与管理文件；Plot 用于查看输出的绘图；Packages 用于管理 R 扩展包；Help 用于查看帮助文档（R 软件的文档与 RStudio 的文档都在这里）。

2.3.1 控制台窗口

开始输入一个命令，例如 `co`，按下 Tab 键，它会建议以 `co` 开头的函数。然后再输入 `r` 并选择 `cor.test` 以使用该函数。如图2.3。

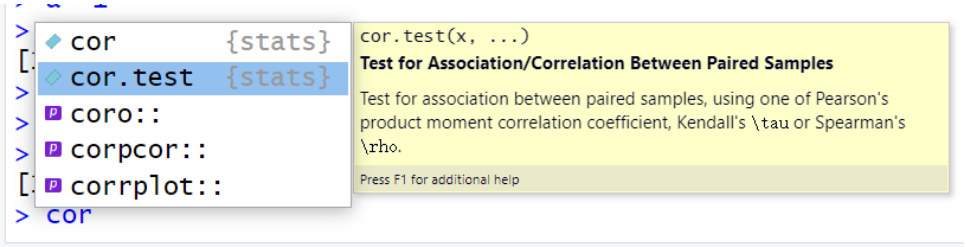


图 2.3: Tab 键使用案例

Tab 不仅会自动补全函数和变量，还会提供函数输入参数的提示。使用 Tab 键选定函数后，可以按下 **F1** 来获取该函数的帮助文件，按下 **F2** 可以显示该函数的源代码。

RStudio 中有许多实用的键盘快捷键，比如显示快捷键菜单的快捷键是 Alt + Shift + K，更多相关的内容可以参考http://rstudio.org/docs/using/keyboard_shortcuts。我们也可以在 Tools -> Modify Keyboard Shortcuts 中修改快捷键。

2.3.2 编辑器窗口

左上角的面板是一个编辑器，可用于编辑 R 脚本 (.R)、纯文本文件 (.txt)、HTML 网页文件、Tex 文件 (.tex) 或 RMarkdown 文件 (.rmd)，后面两种文件类型可以转换为 PDF 文件。

通常我们的 R 代码在编辑器中完成，运行结果在控制台展示，例如可以在编辑器中输入以下代码：

```
x <- 1
y <- 2
# 打印两个变量的和
print(x+y)
```

上述代码中，`#` 为注释符号，后面的内容不会被当作代码执行。`<-`为 R 语言的赋值运算符，用于将右边的值分配（赋值）给左边的变量。在 R 中，也可以使用 `=` 来赋值，效果与 `<-`相同，在大多数情况下，`=` 和 `<-` 是可以互换使用的。在 R 中，`<-` 更为传统且广泛使用，特别是在脚本和函数定义中，它被认为更加符合 R 的习惯用法。

运行上述代码有两种方式，

- 选中所有代码，在编辑器左上角点击 `Run`，或使用快捷键 Ctrl + Enter；
- 无需选中代码，在编辑器左上角点击 `Source`。

这两种方式的主要区别是，第一种方式更加自由随意，可以运行选中的任意代码，第二种方式会直接运行整个文件中的所有代码。运行以上代码可以得到以下结果：

```
## [1] 3
```

2.3.3 工作区窗口

在右上角，有一个包含 Environment 和 History 选项卡的菜单。Environment 中列出了当前 R 会话（session）中的对象。你可以加载、保存或清除工作区中的对象。请注意，在工作区面板下，有一个 **Import Dataset** 的选项。History 中列出了在控制台中输入的所有命令。你可以选择加载、保存、搜索或删除历史记录。可以通过高亮显示一行或多行命令，并将它们发送到 **控制台**或者**编辑器**，来轻松执行历史命令。当然，在控制台也可以使用上箭头来获得历史命令。

如果我们执行了刚才编辑器中的代码，那么 Environment 会包含两个对象：x 和 y，History 包含了我们执行过的 4 行代码。

通常，我们在执行一个 R 程序之前会清空当前环境的中的所有变量，以避免变量污染。在 R 中，`rm(list = ls())` 是一个用于清除当前工作空间中所有对象的命令，其中，`ls()` 函数会列出当前工作空间中所有的对象名称（变量、函数等）。

```
ls()
```

```
## [1] "x" "y"
```

**`rm()` 函数用于删除指定的对象。由于 `ls()` 返回当前工作空间中所有对象的名称，因此 `rm(list = ls())` 的作用就是删除空间中的所有对象。

```
rm(list = ls())
ls()
```

```
## character(0)
```

`character(0)` 表示这是一个没有任何元素的字符类型向量。

注意：此操作不可逆，删除的对象将无法恢复，除非你之前已经保存了它们。

2.3.4 文件等窗口

右下角的窗口包括 Files，Plots，Packages 和 Help 等内容。

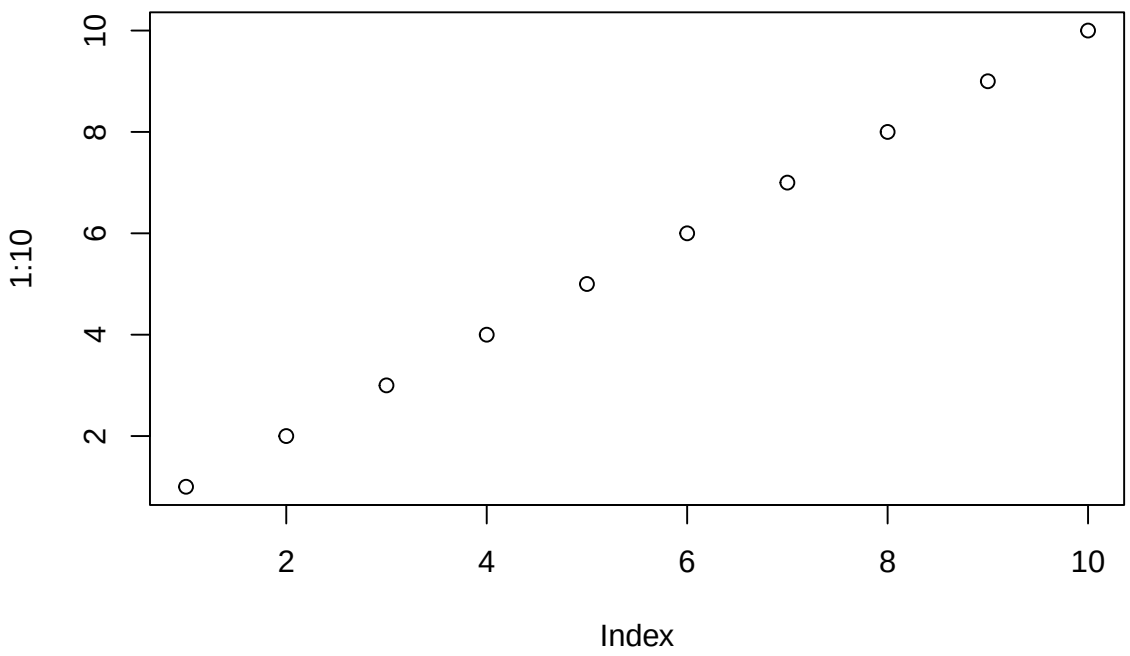
2.3.4.1 Files

Files 是一个文件浏览器，允许你创建新文件夹、重命名文件夹或删除文件夹。点击菜单最右侧三个点的图标（...）可以浏览文件夹。在 More 菜单中，你可以设置当前的工作目录（更多内容将在下文介绍）。如果你双击一个文本文件、R 文件、或 html 文件，它将自动在编辑器中打开。

2.3.4.2 Plots

Plots 窗口显示在 R 中生成的图片，并且可以缩放、导出或删除图片。例如，控制台窗口中输入以下命令来得到一张图片：

```
plot(1:10)
```



2.3.4.3 Packages

Packages 列出了你计算机中安装的所有软件包（package）。打勾的那些软件包是当前 R 会话中加载的。R 包是 R 语言功能扩展的基本单位，它们可以包含函数、数据集、编译代码和帮助文档等。

要在 R 中使用一个包，首先需要安装它，然后加载它：

- **安装包：**

```
install.packages(" 包名")
```

例如：

```
install.packages("ggplot2")
```

- **加载包：**

```
library(包名)
```

例如：

```
library(ggplot2)
```

2.3.4.4 Help

当我们使用`?demo` 时，RStudio 会自动跳转到帮助界面。

在 R 语言中，帮助文件（help files）是用于提供关于函数、数据集或其他包内容的详细说明文档。帮助文件的主要构成包括：

- 1. **标题（Title）**：简要描述函数或数据集的内容。
- 2. **描述（Description）**：对函数或数据集的简要介绍。
- 3. **用法（Usage）**：展示函数的调用方式，包括参数列表。
- 4. **参数（Arguments）**：详细说明函数参数的作用和含义。
- 5. **返回值（Value）**：描述函数的输出结果。
- 6. **示例（Examples）**：提供代码示例，演示函数的实际使用。

当你开始第一个 R 项目时，你需要了解如何组织 R 文件以及保存 R 脚本和输出结果。

2.4 定义工作目录

在开始 R 编程时，首先要做的是确保能够找到你的数据，并且你的输出结果会被保存到计算机硬盘上的有用位置。因此，需要设置一个**工作目录**。

在 RStudio 中，创建一个 RStudio 项目（Project）文件夹是开始新项目的一个很好的方法，这有助于管理 R 代码、数据和结果。当然，通常的任务可以不使用项目选项，但我们仍然需要能够设置一个当前工作目录（所有输出文件都将保存在该目录中）。

有多种方式可以在 RStudio 中定义工作目录。要更改目录：

- 1. 在 RStudio 中：`Session -> Set Working Directory -> Choose Directory`；
- 2. 在 RStudio 中，点击 Files 标签（在右下角窗口）。使用文件浏览器窗口查看目录内容并导航到你想设置为主目录的目录。首先，使用文件浏览器窗口，并点击右上角的三个点图标，导航到正确的目录；然后，当你进入正确的目录并看到你的数据文件后，点击 `More`（蓝色齿轮图标），然后选择 `Set As Working Directory`。

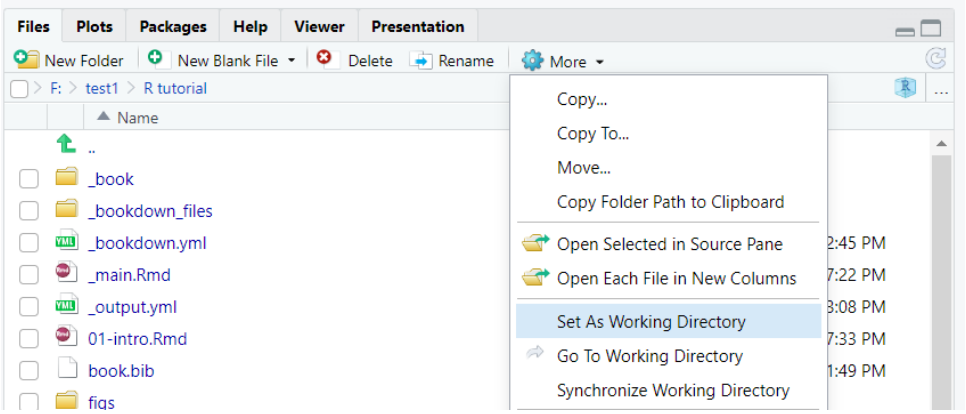


图 2.4: 查看文件并设置工作目录

- 3. 使用打开 R 文件的方式启动 RStudio，此时默认的工作目录就是 R 文件的存储路径。
- 4. 使用代码来设置工作目录。

```
# 获取现在的工作目录
getwd()
```

```
## [1] "F:/R_tutorial/Book/R tutorial"
```

```
# 设置工作目录
setwd("C:/Users")
getwd()
```

```
## [1] "C:/Users"
```

特别提示：R 中无法使用 Windows 系统中用于分隔文件路径中的文件夹的反斜杠（\）。因为在 R 中，反斜杠是转义字符，通常用来表示特殊字符（例如 \n 表示换行）。在下面的代码中，路径 C:\Users\Rwork 中的 \U 被 R 解释为一个无效的转义序列，所以该代码会返回一个错误。

```
setwd("C:\Users\Rwork")
```

```
## Error: '\U' used without hex digits in character string (<text>:1:11)
```

我们可以通过以下两种方法之一来解决这个问题：

- 使用双反斜杠（\\）：将单个反斜杠替换为双反斜杠，以告诉 R 这是一个普通的反斜杠，而不是转义字符。

```
setwd("C:\\Users\\Rwork")
```

- 使用正斜杠（/）：在 R 中，路径中的反斜杠可以用正斜杠 (/) 替代。即使在 Windows 系统上，这也是有效的。

```
setwd("C:/Users/Rwork")
```

当然感兴趣的读者也可以使用函数 file.path 来自动合成路径。

```
file.path("C:","Users","Rwork")
```

```
## [1] "C:/Users/Rwork"
```

2.4.1 使用相对路径

在计算机文件系统中，路径（Path）是用来定位文件或文件夹的字符串。路径可以是相对的（Relative Path）或绝对的（Absolute Path）。

绝对路径是从根目录（或根文件夹）开始的完整路径，用于直接指向文件系统中的任何文件或文件夹，而不依赖于当前工作目录。它包含了从根目录到目标文件或文件夹的所有目录名，并以正确的顺序排列，中间用路径分隔符（在 Windows 中是，在 Unix、Linux、macOS 中是/）分隔。

例如，在 Windows 系统中，一个绝对路径可能是这样的：

```
C:/Users/YourName/Documents/file.txt
```

相对路径是相对于当前工作目录的路径。它不需要从根目录开始，而是从当前目录开始，指向目标文件或文件夹。这意味着如果你知道当前的工作目录，就可以很容易地理解相对路径。

例如，如果当前工作目录是 C:/Users/YourName/Documents，并且你想要访问同一目录下的 file.txt 文件，在运行代码时则可以直接使用以下相对路径来引用该文件：

```
file.txt
```

如果想要访问子目录 Reports 中的 report.pdf 文件，路径将是：

```
Reports/report.pdf
```

在 R 语言中使用相对路径是一种便捷的方式，可以避免在不同计算机或项目中由于绝对路径的差异而导致代码无法正常运行。相对路径是相对于当前工作目录（working directory）设置的路径。

以下是如何在 R 中使用相对路径的几个关键步骤和技巧：

- 1. 设置工作目录，工作目录是相对路径的起点。
- 2. 使用 . 和 .. 进行相对路径引用在 R 中，可以使用 . 表示当前目录，.. 表示上一级目录。例如：

```r
# 引用当前目录下的文件
data <- read.csv("./data/myfile.csv")

# 引用上一级目录下的文件
data <- read.csv("../data/myfile.csv")

# 引用上两级目录下的文件
data <- read.csv("../../data/myfile.csv")
```

在使用 R 完成一个项目时，通常会将项目组织成多个文件夹，如 data/, scripts/, output/ 等。此时，相对路径在不同文件夹间访问文件就显得非常重要。

此外，使用 here 包也可以实现管理相对路径，它会自动定位到项目的根目录，并以此为基础构建相对路径，避免了手动设置工作目录的麻烦。here() 函数会自动在项目的根目录下查找路径，并且无论你从哪个脚本执行代码，都可以保持路径的一致性。

```r
# 加载 here 包
library(here)

# here 自动定位到工作目录，实现相对路径的效果
here("data", "myfile.csv")
```

2.5 保存 R 对象和工作空间

在 RStudio 中保存变量主要涉及到将数据、模型结果或其他任何 R 对象保存到你的工作环境中，以便在后续的 R 会话（session）中能够重新访问或使用它们。然而，严格来说，“保存变量” 这个术语通常指的是将变量存储在你的 R 会话中，以便在当前或重新加载的 RStudio 环境中使用。如果我们想要将变量持久化存储到文件中，以便在不同的 R 会话或不同的计算机上重新加载，那么需要将变量导出到文件中。

在 RStudio 中，当你运行一个赋值语句（如 `x <- 10`），变量 `x` 就会被保存在当前 R 会话的内存中。只要不关闭 RStudio 或清除工作空间（Workspace），这个变量就会一直存在，并且可以在后续的代码中使用。

如果想要保存当前 R 会话中的变量，我们可以通过 RStudio 软件操作或 R 命令来实现。

在 RStudio 软件操作：找到 `Session` 菜单，然后选择 `Save Workspace as...` 来保存当前的工作空间到一个.RData 文件。之后，我们可以通过 `Load Workspace` 来加载这个文件。

通过 R 命令：使用 `save.image()` 函数或 `save()` 函数。`save.image()` 会保存当前工作空间中的所有变量到一个默认的文件（通常是.RData），而 `save()` 函数允许你指定要保存的变量和文件名。例如，`save(x, file="my_data.RData")` 会保存变量 `x` 到 `my_data.RData` 文件中。

```
rm(list = ls())                  # 删除当前 R 环境中的所有对象
x <- 10                          # 创建一个新的数值型变量 x，并将其赋值为 10
save(x, file = "my_data.RData") # 保存变量 x 到 my_data.RData 的文件中
rm(list = ls())                  # 再次删除当前 R 环境中的所有对象
x                                # 这里尝试访问变量 x，在实际运行时会报错
```

```
## Error in eval(expr, envir, enclos): object 'x' not found
```

使用 `load()` 函数可以将.RData 文件加载工作空间，如 `load("my_data.RData")`。

```
load("my_data.RData") # 加载之前保存的 RData 文件，该文件中包含了变量 x
x                     # 尝试访问变量 x，访问成功
```

```
## [1] 10
```

2.6 R 包

在 R 中，包是一个包含函数、数据集和其他代码的集合，用于扩展 R 的功能。R 附带了一小部分基本包，如下表：

包名	描述
base	基本 R 函数
datasets	基本 R 数据集
grDevices	基础和网格图形的图形设备
graphics	R 基础图形函数
methods	R 对象的正式定义方法和类，及其他编程工具
stats	R 统计函数
utils	R 实用函数

下面，我们将给出关于 R 包一些基本操作的介绍：

1. **安装包：**

- 通过 CRAN 安装：CRAN（The Comprehensive R Archive Network）是 R 语言的主要包仓库，包含了大量的 R 包。在 R 的工作窗口或 RStudio 中，可以使用 `install.packages()` 函数来安

装 CRAN 上的 R 包。例如，安装 `ggplot2` 包，可以使用命令 `install.packages("ggplot2")`。

- 从本地安装：如果 R 包已经以压缩文件（如.tar.gz 或.zip）的形式下载到本地，也可以通过 RStudio 的 "Packages" 窗口或 `install.packages()` 函数的 `repos=NULL` 参数来从本地安装。
- 从 GitHub 安装：GitHub 是一个面向开源及私有软件项目的托管平台，许多开发者会在 GitHub 上发布自己的 R 包。安装 GitHub 上的 R 包需要使用 `devtools` 包中的 `install_github()` 函数。例如，安装 `rfCountData` 包，对应的网址为https://github.com/fpechon/rfCountData，可以使用命令 `devtools::install_github("fpechon/rfCountData")`。

2. **加载包**：安装完 R 包后，需要在使用前将其加载到 R 的工作环境中。这可以通过 `library()` 函数或 `require()` 函数来完成。例如，加载 ggplot2 包，可以使用命令 `library(ggplot2)` 或 `require(ggplot2)`。

3. **R 包的查看** 在 R 中，可以使用多种方式来查看已安装的 R 包及其相关信息：

- 使用 `installed.packages()` 函数可以列出所有已安装的 R 包及其版本信息;
- 使用 `search()` 函数可以列出当前工作环境中已加载的 R 包。
- 使用`.libPaths()` 函数可以查看 R 包的安装路径。如果需要更改包的安装路径，可以使用`.libPaths(new="` 新路径`")` 进行设置。
- 使用 help(package="包名") 函数可以打开该包的帮助文档，了解包的功能和使用方法。

4. **R 包的更新和删除**

- 使用 `update.packages()` 函数来更新所有已安装的 R 包，或者使用 `install.packages("` 包名`", dependencies=TRUE)` 函数来单独更新某个包及其依赖项。
- 使用 `remove.packages("` 包名`")` 函数可以删除已安装的 R 包。请注意，在删除包之前，请确保不再需要该包中的任何函数或数据集。