第 1 章

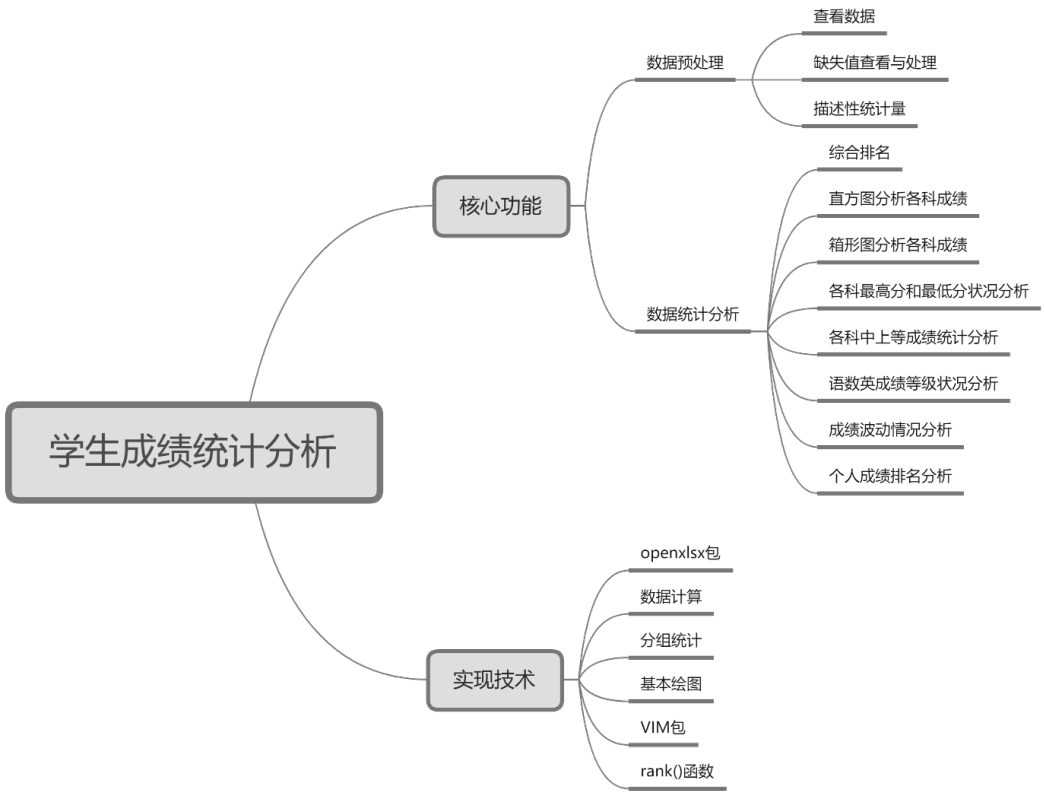
学生成绩统计分析

——openxlsx + 数据计算 + 分组统计 + 基本绘图

数据分析的应用越来越广泛，通过对学生成绩数据进行全面的分析和可视化，能够为教育工作者提供有价值的参考，帮助他们全面深入了解学生的学习情况，并挖掘其潜在的问题和优势。本章将使用 openxlsx 包结合数据计算、分组统计与基本绘图实现学生成绩的统计分析。

本项目的核心功能及实现技术如下：

项目微视频



1.1 开 发 背 景

学生成绩统计分析是教育领域中非常重要的一项工作，通过对学生成绩数据进行深入分析和可视化，可以帮助学校和老师更好地了解学生的学习情况，从而及时发现问题，并采取相应的措施改进教学质量。本项目将主要使用第三方 R 包 openxlsx，同时结合数据计算、分组统计和基本绘图实现学生成绩的统计分析，其中包括综合排名、直方图分析各科成绩、箱形图分析各科成绩、各科最高分和最低分状况分析等。

1.2 系 统 设 计

1.2.1 开发环境

本项目的开发及运行环境如下：

☑ 操作系统：推荐 Windows 10、11 及以上版本。
☑ 编程语言：R 语言。
☑ 开发环境：RStudio。
☑ 第三方 R 包：openxlsx、VIM、dplyr。

1.2.2 分析流程

学生成绩统计分析的首要任务是数据准备，然后进行数据预处理工作，即查看数据、缺失值查看与处理和描述性统计量，以确保数据质量，最后进行数据统计分析。

本项目分析流程如图 1.1 所示。

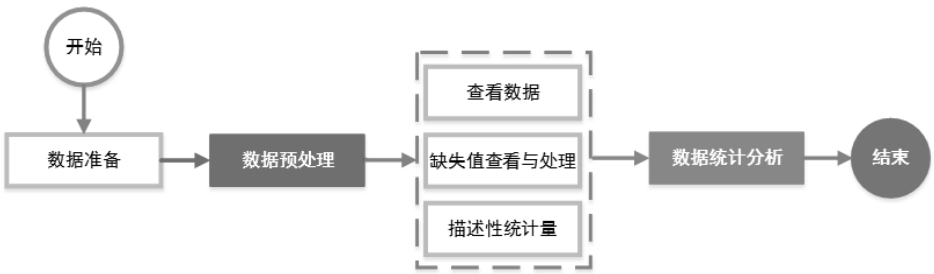


图 1.1 学生成绩统计分析流程

1.2.3 功能结构

本项目的功能结构已经在章首页中给出。本项目实现的具体功能如下：

☑ 数据预处理：首先查看数据概况，包括行数、列数、所有列名以及数据集中每个变量的数据类型；然后查看和处理缺失值；最后查看学生成绩的最小值、最大值、中位数、平均数等。
☑ 数据统计分析：包括综合排名、直方图分析各科成绩、箱形图分析各科成绩、各科最高分和最低分状况分析、各科中上等成绩的学生统计分析、语数英成绩等级状况分析、成绩波动情况分析和个人成绩排名分析。

1.3 技 术 准 备

1.3.1 技术概览

学生成绩统计分析通过读取 Excel 文件中的学生成绩数据，然后进行数据计算、分组统计并绘制相应

的图表来详细分析学生各科成绩的情况，其中主要使用了第三方 R 包 openxlsx、数据计算、分组统计和基本绘图函数，这些知识在《R 语言数据分析从入门到精通》一书中有详细的讲解，对这些知识不太熟悉的读者可以参考该书对应的内容。

除此之外，本项目实现了通过可视化图表对缺失值进行探索，主要使用了第三方 R 包 VIM，在对学生成绩进行综合排名时使用了 rank()函数。下面对这两部分内容进行详细的介绍并举例说明，以确保读者顺利完成本项目的开发，同时拓展相关知识以便更好地利用 R 语言进行数据分析。

1.3.2 VIM 包

当数据量较大时，想要详细了解数据集中数据的缺失值情况，通过可视化图表探索缺失值是绝佳的选择。第三方 R 包 VIM 提供了大量的可视化缺失值函数，如 aggr()函数、barMiss()函数、scattMiss()函数、histMiss()函数、matrixplot()函数、marginplot()函数、marginmatrix()函数，下面介绍几个常用的可视化缺失值函数。

1. aggr()函数

aggr()函数不仅可以展示每个变量里缺失值的个数（或比例），还可以展示多个变量组合下缺失值的个数（或比例），例如下面的代码：

```
library(VIM)
dataset <- sleep[, c("Dream", "NonD", "BodyWgt", "Span")]
aggr(dataset)
```

运行程序，结果如图 1.2 所示。

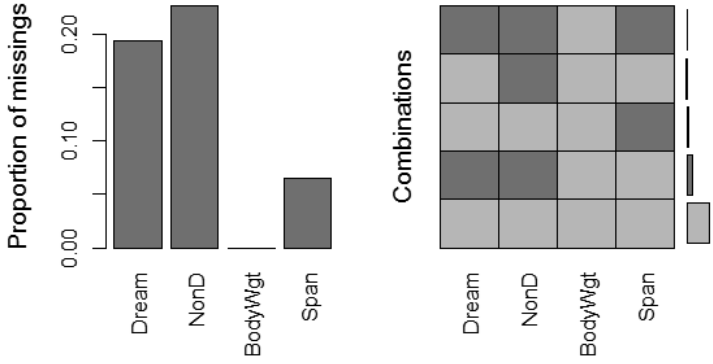


图 1.2 aggr()函数可视化缺失值

说明

aggr()函数在可视化缺失值时使用了三种颜色，每一种颜色代表一个属性，具体如下：

☑ 观测值用蓝色高亮显示。

☑ 缺失值用红色突出显示。

☑ 计算值用橙色突出显示。

2. barMiss()函数

barMiss()函数提供了一个柱形图，主要分析某一变量在另一变量不同取值情况下的缺失值情况。例如下面的代码：

```
x <- sleep[, c("Exp", "NonD", "Sleep")]
barMiss(x, only.miss = FALSE)
```

运行程序，结果如图 1.3 所示。

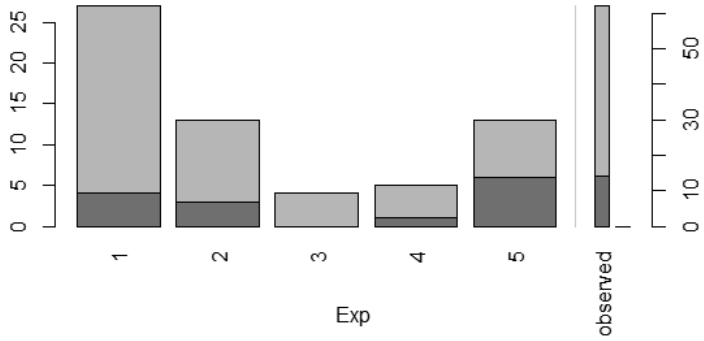


图 1.3　barMiss()函数可视化缺失值

3. marginplot()函数

marginplot()函数提供了一个散点图，在图形边界展示两个变量的缺失值情况。例如下面的代码：

```
marginplot(sleep[c("Gest","Dream")], pch=c(20),
col=c("darkgray", "red", "blue"))
```

运行程序，结果如图 1.4 所示。

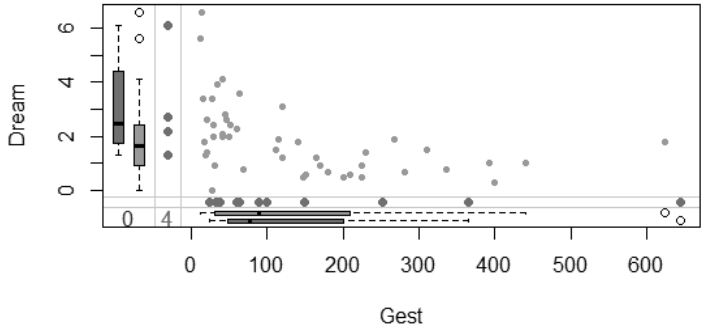


图 1.4　marginplot()函数可视化缺失值

1.3.3　rank()函数详细解析

数据分析过程中，经常需要对数据进行排名。排名主要实现对数据进行排序，然后标记数据在排序中的位置，如学生成绩排名。在 R 语言中，可以通过 rank()函数实现数据排名，默认情况下，rank()函数按照升序排列数据，即排名从低到高。如果需要降序排名，可以使用负号（−）。语法格式如下：

```
rank(x, na.last = TRUE,ties.method = c("average", "first", "last", "random", "max", "min"))
```

参数说明：

- ☑ x：一维数组或向量（数值、复数、字符或逻辑向量）。
- ☑ na.last：如果参数值为TRUE，则将数据中的缺失值放在排名最后；如果参数值为FALSE，则将缺失值放在第一位；如果参数值为 NA，则缺失值会被移除；如果参数值为 keep，则不参与排名，保持为 NA。

☑ ties.method：排名方法，参数值说明如下。

- average：相同元素都取该组中的平均水平，该水平可能是个小数。
- first：最基本的排序，小数在前大数在后，相同元素先者在前后者在后。
- last：与 first 类似，不同的是相同元素后者在前先者在后。
- random：相同元素随机编排次序，避免了“先到先得”，“权重”优于“先后顺序”的机制增大了随机的程度。
- max：小数在前大数在后，相同元素都取该组中最好的水平，即通常所讲的并列排序。
- min：小数在前大数在后，相同元素都取该组中最差的水平，可以增大序列的等级差异。

下面通过具体的示例详细介绍 rank()函数。例如，使用 rank()函数计算学生数学成绩排名，代码如下：

```
scores <- c(99, 88, 104, 75, 118,60)      # 数学成绩
ranks <- rank(-scores)                    # 降序排名，即高分在前低分在后
print(ranks)
```

运行程序，结果为：

```
3 4 2 5 1 6
```

上述代码输出的是每个学生数学成绩的排名，按照成绩从高到低排列。那么，如果成绩相同，则默认情况下排名相同，并且排名以平均数表示，代码如下：

```
scores <- c(99, 88, 104, 75, 118,60,88)   # 数学成绩
ranks <- rank(-scores)                    # 降序排名，即高分在前低分在后
print(ranks)
```

运行程序，结果为：

```
3.0 4.5 2.0 6.0 1.0 7.0 4.5
```

从运行结果得知：数学成绩 88 分的有两名同学，原排名为第 4 和第 5，那么，默认情况下取平均数 4.5，这种排名显然不科学。此时可以设置 ties.method 参数值为 min，这样，当有相同的成绩时，将根据成绩排名的最小值来确定相同成绩的排名，代码如下：

```
ranks <- rank(-scores,ties.method = "min")
print(ranks)
```

运行程序，结果为：

```
3 4 2 6 1 7 4
```

从运行结果得知：数学成绩 88 分的两名同学，排名为并列第 4。

那么，如果成绩中存在缺失值，rank()函数又该如何处理呢？这种情况下，可以通过设置 na.last 参数来决定如何处理缺失值排名的问题，例如下面的代码：

```
# 数学成绩
scores <- c(99, NA, 104, 75, 118,60)
# 排名
ranks <- rank(-scores)
print(ranks)
# 缺失值放在第一位
ranks <- rank(-scores,na.last = FALSE)
print(ranks)
# 删除缺失值
ranks <- rank(-scores,na.last = NA)
print(ranks)
```

```
# 不参与排名，保持为 NA
ranks <- rank(-scores,na.last = "keep")
print(ranks)
```

运行程序，结果如图 1.5 所示。

```
> # 数学成绩
> scores <- c(99, NA, 104, 75, 118,60)
> # 排名
> ranks <- rank(-scores)
> print(ranks)
[1] 3 6 2 4 1 5
> # 缺失值放在第一位
> ranks <- rank(-scores,na.last = FALSE)
> print(ranks)
[1] 4 1 3 5 2 6
> # 删除缺失值
> ranks <- rank(-scores,na.last = NA)
> print(ranks)
[1] 3 2 4 1 5
> # 不参与排名，保持为NA
> ranks <- rank(-scores,na.last = "keep")
> print(ranks)
[1]  3 NA  2  4  1  5
```

图 1.5　通过 na.last 参数处理缺失值排名的问题

1.4　前 期 工 作

1.4.1　安装第三方 R 包

本项目所需的第三方 R 包前面已经进行介绍，这里应逐一进行安装。例如，安装第三方 R 包 openxlsx，代码如下：

```
install.packages("openxlsx")
```

按 Enter 键，将显示一个 CRAN 镜像站点的列表，选择一个适合的镜像站点，如图 1.6 所示，单击“确定”按钮开始安装。

如果需要一次安装多个第三方 R 包，示例代码如下：

```
install.packages(c("包 1","包 2"))
```

图 1.6　CRAN 镜像列表

1.4.2　新建工程

（1）运行 RStudio 新建一个工程以方便存储项目文件。选择 File→New Project 菜单项，然后单击 New Directory→New Project，选择一个位置以创建工程，如图 1.7 所示。

（2）打开新建工程窗口，输入文件夹名称（如“数据分析项目”），选择工程存放的路径（如 D:/R 程序/RProject），如图 1.8 所示，然后单击 Create Project 按钮，创建工程。

（3）创建完成后，会自动打开该工程，如图 1.9 所示。

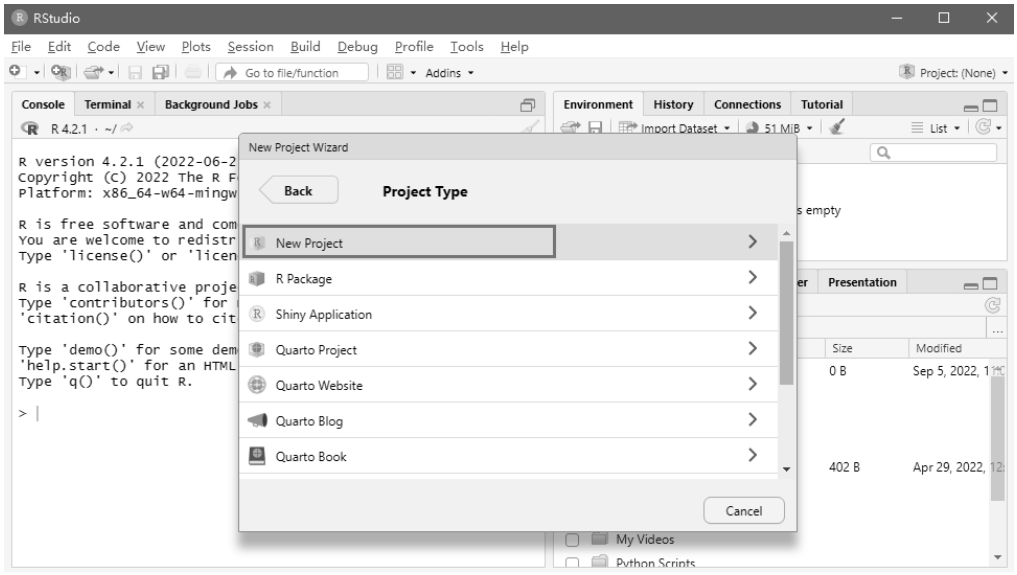


图 1.7　新建工程

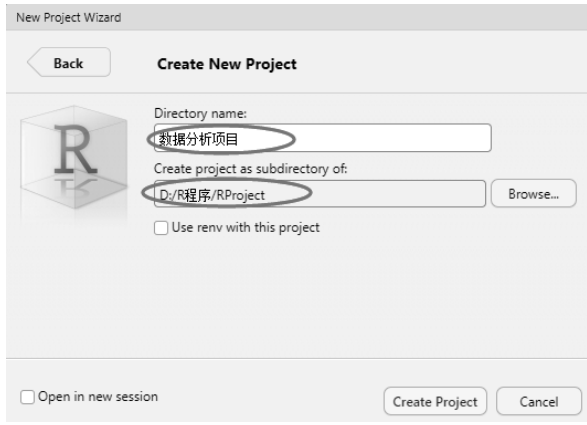


图 1.8　选择工程存放路径

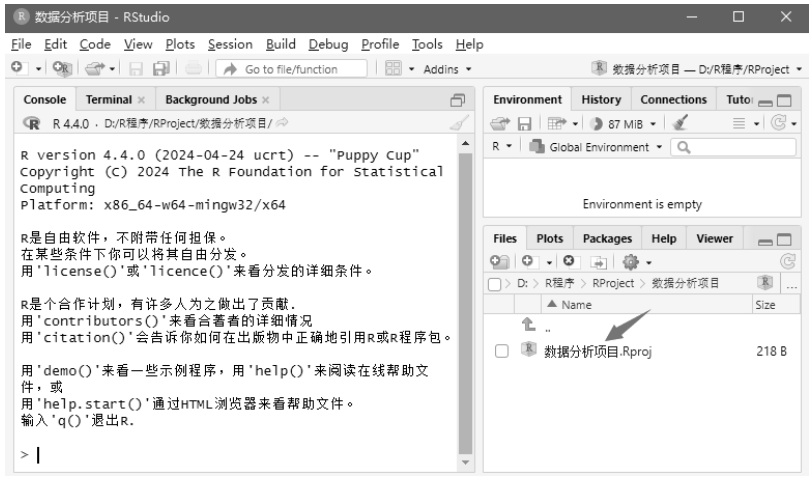


图 1.9　“数据分析项目”工程

1.4.3 新建项目文件夹

开发本项目前应首先在工程（如“数据分析项目.Rproj”）所在文件夹中新建一个项目文件夹（如“学生成绩统计分析”文件夹），以保存项目所需的 R 脚本文件，实现过程如下。

（1）运行 RStudio，选择 File→Open Project 菜单项，选择已经创建好的工程（如“数据分析项目.Rproj”），然后在资源管理窗口中单击 Files 面板中的新建文件夹按钮，如图 1.10 所示。

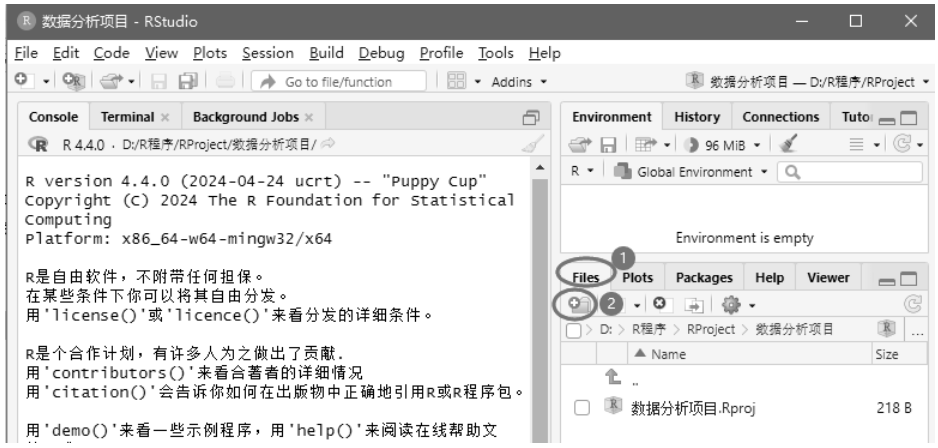


图 1.10　单击 Files 面板中的新建文件夹按钮

（2）打开 New Folder 对话框，输入“学生成绩统计分析”，如图 1.11 所示，然后单击 OK 按钮，项目文件夹就创建完成了。

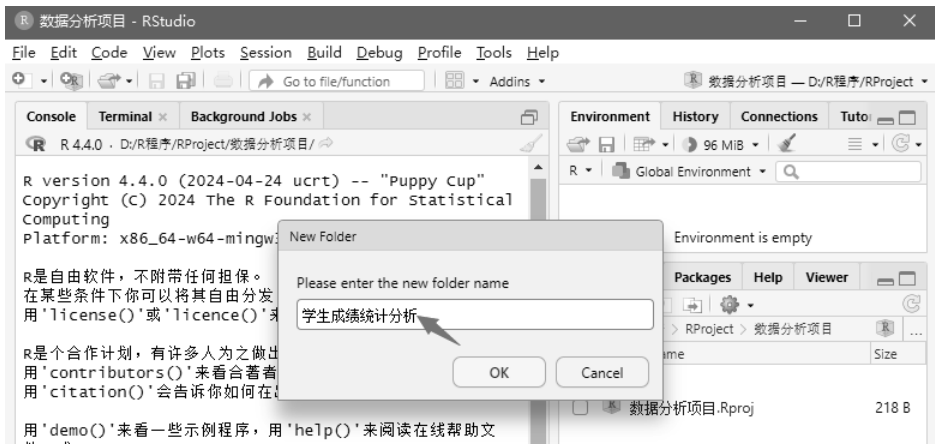


图 1.11　创建“学生成绩统计分析”项目文件夹

1.5 数 据 准 备

1.5.1 数据集介绍

学生成绩统计分析的数据主要来源于某高中学生成绩表，包括“成绩表.xlsx”和“测试成绩.xlsx”，如

图 1.12 所示。

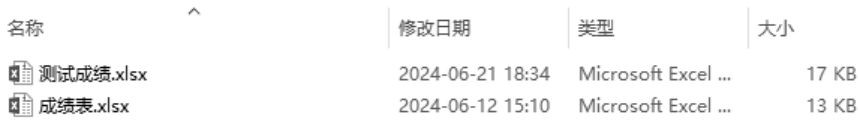
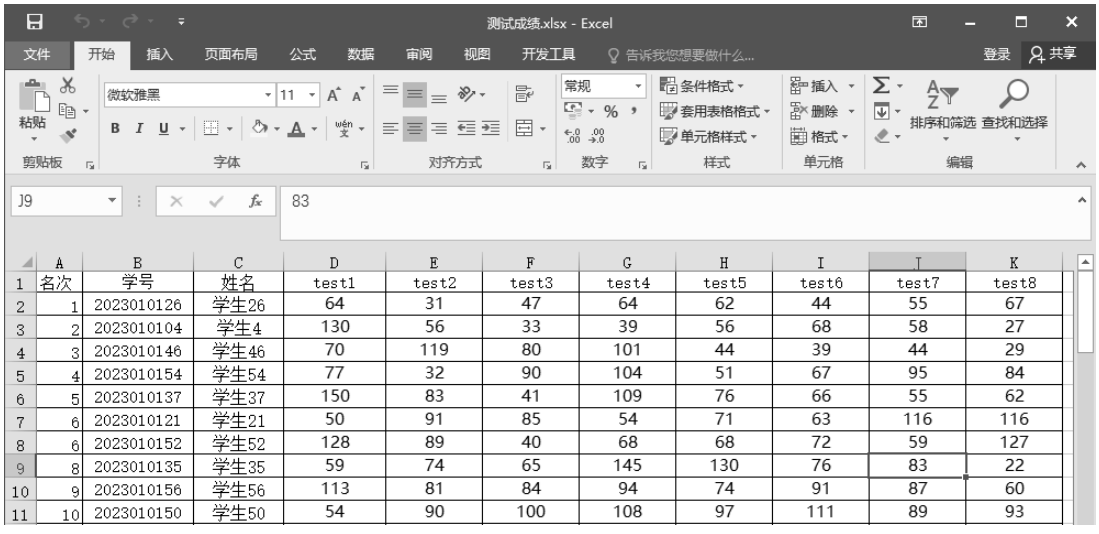


图 1.12　学生成绩统计分析的数据文件

部分数据截图如图 1.13 和图 1.14 所示。

	A	B	C	D	E	F	G	H
1	学号	姓名	语文	数学	英语	物理	化学	生物
2	2023010101	学生1	99.5	63	78.5		60	48.5
3	2023010102	学生2	93.5	138	83	57	74	53
4	2023010103	学生3	96.5	89	101.5	59	70	71.5
5	2023010104	学生4	99.5	148	85	90	57	60
6	2023010105	学生5	101	65	70	53	60	51.5
7	2023010106	学生6	88	79	90.5	51	79	67
8	2023010107	学生7	91.5	70	100.5	57	70	56
9	2023010108	学生8	97.5	99	88.5	61	76	69.5
10	2023010109	学生9	97	110	83.5	57	70	70.5
11	2023010110	学生10	105.5	101	98	61	76	63.5

图 1.13　“成绩表.xlsx”部分数据截图



	A	B	C	D	E	F	G	H	I	J	K
1	名次	学号	姓名	test1	test2	test3	test4	test5	test6	test7	test8
2	1	2023010126	学生26	64	31	47	64	62	44	55	67
3	2	2023010104	学生4	130	56	33	39	56	68	58	27
4	3	2023010146	学生46	70	119	80	101	44	39	44	29
5	4	2023010154	学生54	77	32	90	104	51	67	95	84
6	5	2023010137	学生37	150	83	41	109	76	66	55	62
7	6	2023010121	学生21	50	91	85	54	71	63	116	116
8	6	2023010152	学生52	128	89	40	68	68	72	59	127
9	8	2023010135	学生35	59	74	65	145	130	76	83	22
10	9	2023010156	学生56	113	81	84	94	74	91	87	60
11	10	2023010150	学生50	54	90	100	108	97	111	89	93

图 1.14　“测试成绩.xlsx”部分数据截图

说明

“成绩表.xlsx”和“测试成绩.xlsx”位于资源包项目所在的文件夹，开发本项目前应首先将它们复制到项目文件夹中，如图 1.15 所示。

图 1.15　将数据文件复制到项目文件夹

1.5.2　读取数据

在了解了数据集后，接下来读取数据，主要使用 openxlsx 包的 read.xlsx()函数，实现过程如下（源码位置：资源包\Code\01\view_data.R）。

（1）在项目文件夹（“学生成绩统计分析”文件夹）中新建一个 R 脚本文件，命名为 view_data.R。

（2）使用 openxlsx 包的 read.xlsx()函数读取 Excel 文件，代码如下：

```
# 加载程序包
library(openxlsx)
# 读取 Excel 文件
df <- read.xlsx("学生成绩统计分析/成绩表.xlsx",sheet=1)
```

（3）显示前 6 条数据，代码如下：

```
# 显示前 6 条数据
head(df)
```

运行程序，结果如图 1.16 所示。

```
        学号  姓名  语文 数学  英语 物理 化学 生物
1 2023010101 学生1  99.5   63  78.5   NA   60 48.5
2 2023010102 学生2  93.5  138  83.0   57   74 53.0
3 2023010103 学生3  96.5   89 101.5   59   70 71.5
4 2023010104 学生4  99.5  148  85.0   90   57 60.0
5 2023010105 学生5 101.0   65  70.0   53   60 51.5
6 2023010106 学生6  88.0   79  90.5   51   79 67.0
```

图 1.16　显示前 6 条数据

还可以以表格的形式显示数据，代码如下：

```
View(df)
```

运行程序，结果如图 1.17 所示。

比起图 1.16，图 1.17 中的数据看上去更清晰更直观。不仅如此，通过数据查看器还可以实现数据筛选和排序。例如，筛选“数学”成绩 100～120 的数据，第 1 步单击 Filter，第 2 步单击“数学”文本框，第 3 步在直方图中单击数据区间，如 100-120，如图 1.18 所示，之后将显示筛选结果，如图 1.19 所示。


view_data.R* × df ×

Filter

	学号	姓名	语文	数学	英语	物理	化学	生物
1	2023010101	学生1	99.5	63	78.5	NA	60	48.5
2	2023010102	学生2	93.5	138	83.0	57	74	53.0
3	2023010103	学生3	96.5	89	101.5	59	70	71.5
4	2023010104	学生4	99.5	148	85.0	90	57	60.0
5	2023010105	学生5	101.0	65	70.0	53	60	51.5
6	2023010106	学生6	88.0	79	90.5	51	79	67.0
7	2023010107	学生7	91.5	70	100.5	57	70	56.0
8	2023010108	学生8	97.5	99	88.5	61	76	69.5
9	2023010109	学生9	97.0	110	83.5	57	70	70.5
10	2023010110	学生10	105.5	101	98.0	61	76	63.5

图 1.17　在数据查看器中显示数据

	学号	姓名	语文	数学	英语	物理	化学	生物
9	2023010109	学生9	97.0			57	70	70.5
10	2023010110	学生10	105.5			61	76	63.5
16	2023010116	学生16	95.0			57	72	58.5
22	2023010122	学生22	90.0			66	65	63.0
26	2023010126	学生26	113.5			70	78	72.0
28	2023010128	学生28	97.0			68	69	65.5
30	2023010130	学生30	96.0	101	80.0	63	80	67.5
39	2023010139	学生39	80.0	117	68.0	70	65	12.0
46	2023010146	学生46	97.0	102	99.5	78	84	75.5
57	2023010157	学生57	113.5	110	77.0	66	60	61.0

图 1.18　筛选“数学”成绩 100～120 的数据


view_data.R* × df ×

Filter

	学号	姓名	语文	数学	英语	物理	化学	生物
9	2023010109	学生9	97.0	110	83.5	57	70	70.5
10	2023010110	学生10	105.5	101	98.0	61	76	63.5
16	2023010116	学生16	95.0	102	69.0	57	72	58.5
22	2023010122	学生22	90.0	117	59.5	66	65	63.0
26	2023010126	学生26	113.5	107	106.5	70	78	72.0
28	2023010128	学生28	97.0	108	65.0	68	69	65.5
30	2023010130	学生30	96.0	101	80.0	63	80	67.5
39	2023010139	学生39	80.0	117	68.0	70	65	12.0
46	2023010146	学生46	97.0	102	99.5	78	84	75.5
57	2023010157	学生57	113.5	110	77.0	66	60	61.0

图 1.19　筛选结果

也可以在右侧的 Environment 面板中单击 图标启动数据查看器，如图 1.20 所示。

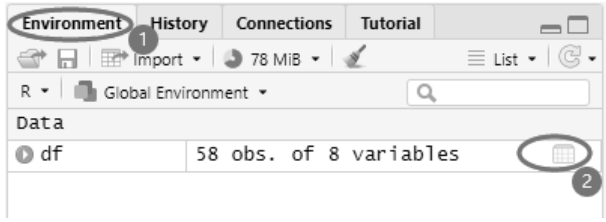


图 1.20　启动数据查看器

1.6　数据预处理

1.6.1　查看数据

查看数据概况，包括行数、列数、所有列名以及数据集中每个变量的数据类型，以便更清晰地了解数据，主要使用 nrow()函数、ncol()函数、names()函数和 sapply()函数，代码如下（源码位置：资源包\Code\01\view_data.R）：

```
# 行数
nrow(df)
# 列数
ncol(df)
# 查看所有列名
names(df)
# 查看数据集中每个变量的数据类型
sapply(df, class)
```

运行程序，结果如图 1.21 所示。

```
> # 行数
> nrow(df)
[1] 58
> # 列数
> ncol(df)
[1] 8
> # 查看所有列名
> names(df)
[1] "学号" "姓名" "语文" "数学" "英语" "物理" "化学" "生物"
> # 查看数据集中每个变量的数据类型
> sapply(df, class)
      学号       姓名       语文       数学       英语
"numeric" "character"  "numeric"  "numeric"  "numeric"
      物理       化学       生物
"numeric"  "numeric"  "numeric"
```

图 1.21　查看数据

从运行结果得知：数据共 58 行 8 列，以及所有的列名和数据类型。

接下来使用 str()函数查看数据整体概况，代码如下：

```
str(df)
```

运行程序，结果如图 1.22 所示。

```
'data.frame':   58 obs. of  8 variables:
 $ 学号: num  2.02e+09 2.02e+09 2.02e+09 2.02e+09 2.02e+09 ...
 $ 姓名: chr  "学生1" "学生2" "学生3" "学生4" ...
 $ 语文: num  99.5 93.5 96.5 99.5 101 ...
 $ 数学: num  63 138 89 148 65 79 70 99 110 101 ...
 $ 英语: num  78.5 83 101.5 85 70 ...
 $ 物理: num  NA 57 59 90 53 51 57 61 57 61 ...
 $ 化学: num  60 74 70 57 60 79 70 76 70 76 ...
 $ 生物: num  48.5 53 71.5 60 51.5 67 56 69.5 70.5 63.5 ...
```

图 1.22　查看数据整体概况

从运行结果得知：数据共 58 行 8 列，以及所有的列名、数据类型和具体数据。

1.6.2　缺失值查看与处理

1. 缺失值识别

下面使用 is.na()函数查看缺失值，代码如下（源码位置：资源包\Code\01\view_data.R）：

```
is.na(df)
```

运行程序，结果如图 1.23 所示。

	学号	姓名	语文	数学	英语	物理	化学	生物
1	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
2	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
3	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
4	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
5	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
6	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
7	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
8	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
9	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
10	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
11	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
12	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
13	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
14	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
15	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
16	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
17	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
18	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
19	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
20	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE

图 1.23　使用 is.na()函数查看缺失值

从运行结果得知：TRUE 表示数据存在缺失，FALSE 表示数据不存在缺失。如果数据很少，缺失值一目了然，例如“物理”和“化学”存在缺失值。但是如果数据量较大，看起来就不是很方便了，此时需要借助 table()函数，代码如下：

```
table(is.na(df))
```

运行程序，结果如下：

```
FALSE  TRUE
  459     5
```

table()函数帮我们统计了数据不缺失和缺失的个数，值分别为 459 和 5，也就是说学生成绩表有 5 个缺失值，具体这 5 个缺失值在哪里尚且不清楚，此时需要进一步分析。

2. 缺失值探索分析

下面对缺失值进行探索分析，主要使用 VIM 包的 aggr()函数对缺失值进行统计与可视化，以直观观察数据缺失情况，代码如下（源码位置：资源包\Code\01\view_data.R）：

```
library(VIM)                                          # 加载 VIM 包
val=aggr(df,prop = FALSE,number=TRUE,cex.axis=.7)     # 缺失值可视化
val                                                   # =缺失值统计
```

运行程序，结果如下：

```
Missings in variables:
 Variable Count
     物理     3
     化学     1
     生物     1
```

可视化效果如图 1.24 所示。

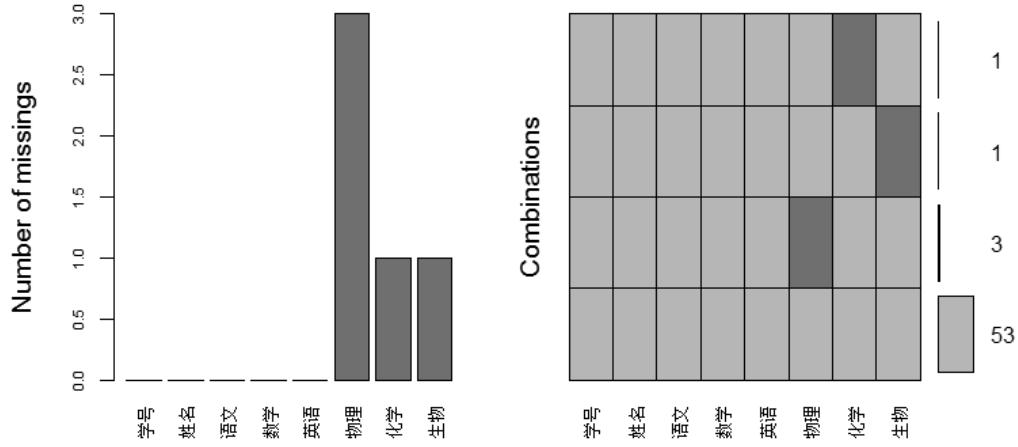


图 1.24　使用 aggr()函数对缺失值进行统计并可视化

从运行结果得知：“物理”有 3 个缺失值，“生物”和“化学”各有一个缺失值。下面对缺失值进行填充处理。

3. 缺失值填充

通过上述检测发现部分学科成绩缺失，可能是因为某种原因学生没有参加考试，此时我们使用特定数值（即 0）进行填充，方法是将缺失值替换为 0，代码如下（源码位置：资源包\Code\01\view_data.R）：

```
df[is.na(df)] <- 0                                    # 将缺失值替换为 0
table(is.na(df))                                      # 再次统计缺失值
write.xlsx(df,"学生成绩统计分析/成绩表 1.xlsx") # 写入 Excel 文件
```

1.6.3　描述性统计量

描述性统计量主要查看学生成绩的最小值、最大值、第一四分位数、中位数、平均数等，主要使用 summary()函数，代码如下（源码位置：资源包\Code\01\view_data.R）：

```
summary(df)
```

运行程序，结果如图 1.25 所示。

```
      学号                姓名                语文            数学            英语
Min.   :2.023e+09   Length:58          Min.   : 80.00   Min.   : 43.00   Min.   : 48.50
1st Qu.:2.023e+09   Class :character   1st Qu.: 93.50   1st Qu.: 75.25   1st Qu.: 78.00
Median :2.023e+09   Mode  :character   Median : 97.00   Median : 85.50   Median : 87.25
Mean   :2.023e+09                      Mean   : 97.14   Mean   : 86.95   Mean   : 85.51
3rd Qu.:2.023e+09                      3rd Qu.:100.88   3rd Qu.: 98.50   3rd Qu.: 97.75
Max.   :2.023e+09                      Max.   :113.50   Max.   :148.00   Max.   :110.50

      物理            化学            生物
Min.   :30.00   Min.   :24.00   Min.   :12.00
1st Qu.:54.00   1st Qu.:60.00   1st Qu.:52.50
Median :61.00   Median :70.00   Median :59.00
Mean   :60.51   Mean   :68.54   Mean   :59.21
3rd Qu.:69.00   3rd Qu.:78.00   3rd Qu.:68.00
Max.   :99.00   Max.   :90.00   Max.   :83.00
NA's   :3       NA's   :1       NA's   :1
```

图 1.25　描述性统计量

从运行结果得知：对于数值型数据，summary()函数给出的统计信息包括最小值（Min）、第一四分位数（1st Qu）、中位数（Median）、平均值（Mean）、第三四分位数（3rd Qu）和最大值（Max）。例如，语文最低分 80，平均分 97.14，最高分 113.5。同时，summary()函数也可以对缺失值进行统计，例如，物理“NA's　:3”表示有 3 个缺失值。

1.7　数据统计分析

1.7.1　综合排名

综合排名包括各科成绩排名以及语文、数学、英语总分排名和总分排名，实现过程如下（源码位置：资源包\Code\01\rank_data.R）。

（1）在项目文件夹下新建一个 R 脚本文件，命名为 rank_data.R。

（2）使用 openxlsx 包的 read.xlsx()函数读取 Excel 文件，代码如下：

```
# 加载程序包
library(openxlsx)
# 读取 Excel 文件
df <- read.xlsx("学生成绩统计分析/成绩表 1.xlsx",sheet=1)
```

（3）分别计算语文、数学、英语总分和排名，主要使用 rank()函数，然后使用 View()函数以表格形式显示数据，代码如下：

```
# 各科成绩排名
df$语文排名 <- round(rank(-df$语文,ties.method='min'),0)
df$数学排名 <- round(rank(-df$数学,ties.method='min'),0)
df$英语排名 <- round(rank(-df$英语,ties.method='min'),0)
df$物理排名 <- round(rank(-df$物理,ties.method='min'),0)
df$化学排名 <- round(rank(-df$化学,ties.method='min'),0)
df$生物排名 <- round(rank(-df$生物,ties.method='min'),0)
# 计算语数英总分和排名
df$语数英总分 <- df$语文+df$数学+df$英语
df$语数英排名 <- round(rank(-df$语数英总分,ties.method='min'),0)
# 计算总分和排名
df$总分 <- df$语文+df$数学+df$英语+df$物理+df$化学+df$生物
df$总分排名 <- round(rank(-df$总分,ties.method='min'),0)
# 以表格形式显示数据
View(df)
```

运行程序，结果如图 1.26 所示。

rank_data.R* × df ×

Filter

语文排名	数学排名	英语排名	物理排名	化学排名	生物排名	语数英总分	语数英排名	总分	总分排名
20	54	40	56	42	47	241.0	48	349.5	58
43	2	36	32	23	41	314.5	3	498.5	13
33	25	8	29	28	10	287.0	20	487.5	18
20	1	33	2	47	28	332.5	1	539.5	2
13	53	48	43	42	44	236.0	50	400.5	48
53	38	24	44	12	17	257.5	38	454.5	30
46	48	9	32	28	39	262.0	35	445.0	34
28	13	26	27	19	14	285.0	23	491.5	16
29	5	34	32	28	12	290.5	15	488.0	17
7	11	15	27	19	23	304.5	4	505.0	12
55	55	19	39	24	23	235.0	52	425.5	40
23	50	42	3	47	49	245.0	47	433.5	36
33	13	21	4	57	55	287.0	20	424.0	42
16	48	54	1	45	51	234.0	53	436.5	35

图 1.26　综合排名（部分数据截图）

从运行结果得知：通过语文、数学、英语总分的排名和总分排名可以看出有些学生存在偏科现象。

（4）将计算结果和排名写入 Excel 文件中，代码如下：

```
write.xlsx(df,"学生成绩统计分析/排名.xlsx")
```

运行程序，计算结果和排名将写入 Excel 文件中，打开 Excel 文件，结果如图 1.27 所示。

排名.xlsx - Excel

文件　开始　插入　页面布局　公式　数据　审阅　视图　开发工具　告诉我您想要做什么...　登录　共享

R2　58

	I	J	K	L	M	N	O	P	Q	R
1	语文排名	数学排名	英语排名	物理排名	化学排名	生物排名	语数英总分	语数英排名	总分	总分排名
2	20	54	40	56	42	47	241	48	349.5	58
3	43	2	36	32	23	41	314.5	3	498.5	13
4	33	25	8	29	28	10	287	20	487.5	18
5	20	1	33	2	47	28	332.5	1	539.5	2
6	13	53	48	43	42	44	236	50	400.5	48
7	53	38	24	44	12	17	257.5	38	454.5	30
8	46	48	9	32	28	39	262	35	445	34
9	28	13	26	27	19	14	285	23	491.5	16
10	29	5	34	32	28	12	290.5	15	488	17
11	7	11	15	27	19	23	304.5	4	505	12
12	55	55	19	39	24	23	235	52	425.5	40
13	23	50	42	3	47	49	245	47	433.5	36
14	33	13	21	4	57	55	287	20	424	42
15	16	48	54	1	45	51	234	53	436.5	35
16	43	57	10	49	52	32	247.5	46	399.5	49

Sheet 1

就绪　100%

图 1.27　排名.xlsx

1.7.2 直方图分析各科成绩

通过绘制各科成绩直方图查看各科成绩的分布情况，主要使用 hist()函数，实现过程如下（源码位置：资源包\Code\01\hist_data.R）。

（1）在项目文件夹下新建一个 R 脚本文件，命名为 hist_data.R。

（2）使用 openxlsx 包的 read.xlsx()函数读取 Excel 文件，代码如下：

```
# 加载程序包
library(openxlsx)
# 读取 Excel 文件
df <- read.xlsx("学生成绩统计分析/成绩表 1.xlsx",sheet=1)
```

（3）使用 par()函数创建 2 行 3 列的绘图区域，然后使用 hist()函数绘制各科成绩直方图，代码如下：

```
# 2 行 3 列的绘图区域
par(mfrow = c(2,3))
# 绘制各科成绩直方图
hist(df$语文,main="",xlab="语文")
hist(df$数学,main="",xlab="数学")
hist(df$英语,main="",xlab="英语")
hist(df$物理,main="",xlab="物理")
hist(df$化学,main="",xlab="化学")
hist(df$生物,main="",xlab="生物")
```

运行程序，结果如图 1.28 所示。

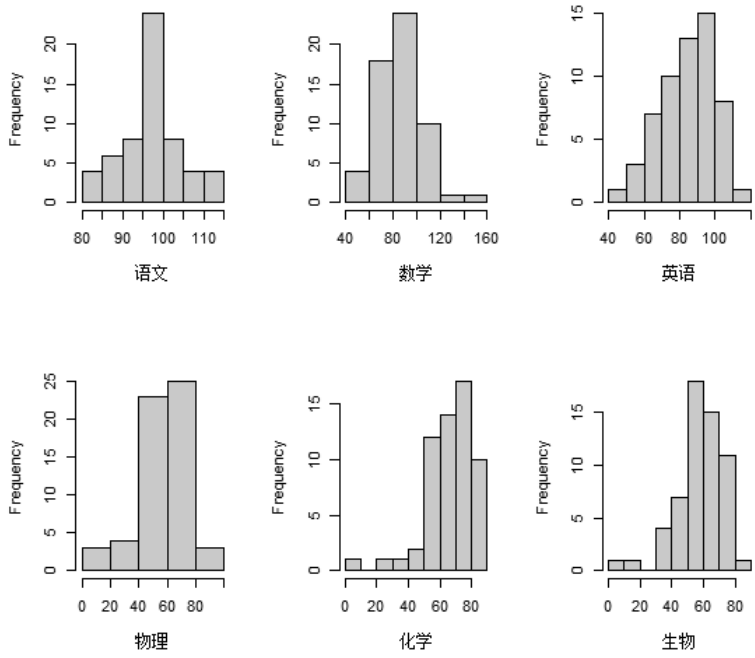


图 1.28 直方图分析各科成绩

（4）通过偏度系数判断偏度，主要使用 timeDate 模块的 skewness()函数进行计算，代码如下：

```
# 计算偏度系数（正值为右偏，负值为左偏）
df1 <- df[,3:8]
```

```
n1 <- timeDate::skewness(df1)
n1
```

运行程序，结果如下：

```
      语文       数学       英语       物理       化学       生物
-0.1335115  0.6028185 -0.4783041 -1.1768642 -1.6404522 -1.4271909
```

从运行结果得知：“数学”成绩呈右偏态分布，即直方图右侧缺失，说明高分段学生缺失，试卷有难度。其他学科呈左偏态分布，即直方图左侧缺失，说明低分段学生较少，试卷难度适中。

说明

负偏态分布，也称为左偏态，指的是在一个不对称或偏斜的次数分布中，次数分布的高峰偏右，而长尾则从右逐渐延伸于左端。这种分布的特点是偏态系数小于零，意味着众数位于较大分数或量数的一侧（右侧），而长尾则位于较小分数或量数的一侧（左侧）。在学生的学业成绩分布中，负偏态分布表明高分人数很多，低分人数很少，这通常意味着测试的难度较低，考试要求低于教学要求，或者学生的基础较好。因此，当学生成绩呈现负偏态分布时，说明测验的整体难度相对偏易。

此外，负偏态分布的原因可能包括试题难度过小，导致部分学生成绩过高，从而使总分的分布呈现负偏态。这种情况下，测验的整体难度被认为是正常的。然而，需要注意的是，负偏态分布并不总是意味着测验难度过低，它也可能反映试题难度介于过大和过小之间，导致一部分学生不能正常发挥。因此，在解释负偏态分布时，需要综合考虑试题的难度设置和学生的实际表现。

1.7.3 箱形图分析各科成绩

通过绘制箱形图分析各科成绩，主要使用 boxplot()函数，实现过程如下（源码位置：资源包\Code\01\box_data.R）。

（1）在项目文件夹下新建一个 R 脚本文件，命名为 box_data.R。

（2）使用 openxlsx 包的 read.xlsx()函数读取 Excel 文件，代码如下：

```
# 加载程序包
library(openxlsx)
# 读取 Excel 文件
df <- read.xlsx("学生成绩统计分析/成绩表 1.xlsx",sheet=1)
```

（3）使用 par()函数创建 2 行 3 列的绘图区域，然后使用 boxplot()函数绘制各科成绩箱形图，代码如下：

```
# 2 行 3 列的绘图区域
par(mfrow = c(2,3))
# 绘制各科成绩箱形图
boxplot(df$语文,main="",xlab="语文")
boxplot(df$数学,main="",xlab="数学")
boxplot(df$英语,main="",xlab="英语")
boxplot(df$物理,main="",xlab="物理")
boxplot(df$化学,main="",xlab="化学")
boxplot(df$生物,main="",xlab="生物")
```

运行程序，结果如图 1.29 所示。

从运行结果得知：语文、英语和物理成绩比较平均。另外，语文、数学、物理、化学和生物均存在异常值。

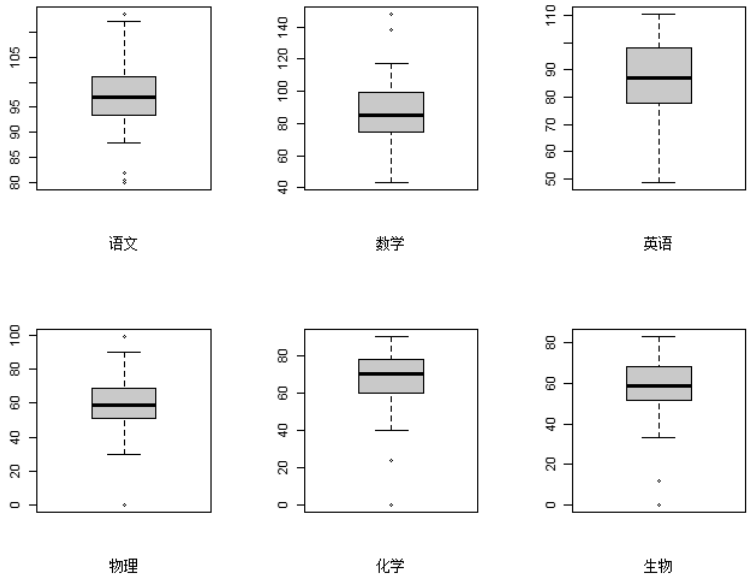


图 1.29　箱形图分析各科成绩

1.7.4　各科最高分和最低分状况分析

对各科最高分和最低分状况进行分析时，首先使用 max()函数和 min()函数分别计算各科成绩的最高分和最低分。需要注意的是，在求得物理、化学和生物 3 科的最小值后，应排除分数为 0 的数据，因为在 1.6.2 节中我们将物理、化学和生物 3 科的缺失值填充为 0 了。然后绘制各科成绩最高分和最低分柱形图。实现过程如下（源码位置：资源包\Code\01\max_min_data.R）。

（1）在项目文件夹下新建一个 R 脚本文件，命名为 max_min_data.R。

（2）使用 openxlsx 包的 read.xlsx()函数读取 Excel 文件，代码如下：

```
# 加载程序包
library(openxlsx)
# 读取 Excel 文件
df <- read.xlsx("学生成绩统计分析/成绩表 1.xlsx",sheet=1)
```

（3）使用 max()函数和 min()函数分别计算各科成绩最高分和最低分，并排除物理、化学和生物 3 科分数为 0 的数据，代码如下：

```
# 计算各科成绩最高分和最低分
y1_max <- max(df$语文)
y1_min <- min(df$语文)
y2_max <- max(df$数学)
y2_min <- min(df$数学)
y3_max <- max(df$英语)
y3_min <- min(df$英语)
y3_max <- max(df$英语)
y3_min <- min(df$英语)
y4_max <- max(df$物理)
# 最低分排除分数为 0 的数据
y4_min <- min(df[df$物理!=0,'物理'])
y5_max <- max(df$化学)
y5_min <- min(df[df$化学!=0,'化学'])
y6_max <- max(df$生物)
y6_min <- min(df[df$生物!=0,'生物'])
```

（4）通过求得的各科成绩的最高分和最低分创建 6*2 的矩阵，代码如下：

```
# 创建 6*2 的矩阵
datas <- matrix(c(y1_max,y2_max,y3_max,y4_max,y5_max,y6_max,
                  y1_min,y2_min,y3_min,y4_min,y5_min,y6_min),6,2,
                dimnames =list(names(df[,3:8])))
# 矩阵行列转置
datas <-t(datas)
datas
```

运行程序，结果如下：

```
     语文 数学   英语 物理  化学 生物
[1,] 113.5  148  110.5   99   90   83
[2,]  80.0   43   48.5   30   24   12
```

（5）使用 barplot()函数绘制各科成绩最高分和最低分柱形图，代码如下：

```
# 绘制柱形图
# 自定义画布大小
par(pin=c(5,4))
# 绘制柱形图
p=barplot(height = datas,
          col = c('darkorange','deepskyblue'), # 每个柱形的颜色
          beside=TRUE,                         # 分组柱形图,即多柱形图
          border=FALSE,                        # 无边框
          ylim=c(0,160))                       # y 轴坐标轴范围
# 添加文本标签
text(p,datas+5,labels=datas)
```

运行程序，结果如图 1.30 所示。

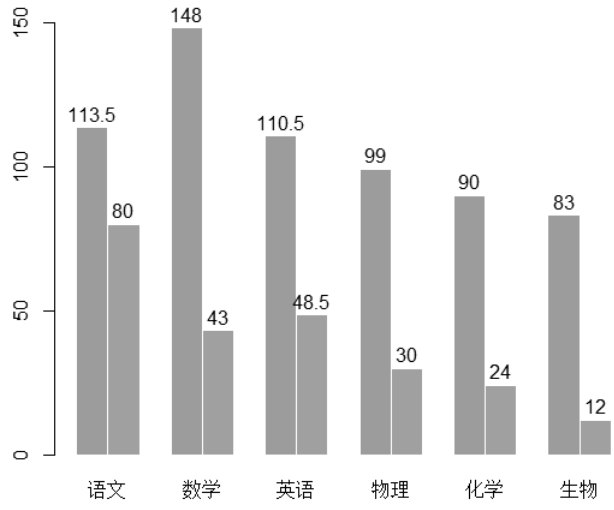


图 1.30　柱形图分析各科最高分和最低分状况

从运行结果得知：除了语文，其他各科成绩最高分和最低分差距还是比较大的。

1.7.5　各科中上等成绩统计分析

各科中上等成绩统计分析，主要统计各科成绩超过平均分的人数。通常来讲，如果学生的成绩在班级里达到或超过了平均分，那么其基本就属于中上等学生，即使是在平均分上下，不是幅度相差太大都可以

算得上是中上等学生。例如，一个班级有 50 个人，那么成绩达到平均分也就是 25 名左右，只要成绩排在 25～30 名都算是中等成绩。成绩高于平均分，只能说明成绩处于中等偏上的水平，属于比较不错的学习成绩。

下面实现各科中上等成绩统计分析。首先使用 apply()函数计算平均分，然后使用 length()函数和 mean()函数统计各科成绩大于平均分的人数，最后使用 barplot()函数绘制柱形图，实现过程如下（源码位置：资源包\Code\01\mean_data.R）。

（1）在项目文件夹下新建一个 R 脚本文件，命名为 mean_data.R。

（2）使用 openxlsx 包的 read.xlsx()函数读取 Excel 文件，然后抽取数据，代码如下：

```
# 加载程序包
library(openxlsx)
# 读取 Excel 文件
df <- read.xlsx("学生成绩统计分析/成绩表 1.xlsx",sheet=1)
# 抽取数据
df1 <- df[,3:8]
```

（3）使用 apply()函数计算平均分，然后使用 length()函数和 mean()函数统计各科成绩大于平均分的人数，代码如下：

```
# 各科成绩平均分
mean1 <- apply(df1, 2, mean)
# 统计各科成绩大于平均分的学生人数
x1=paste(length(df1$语文[df1$语文>mean(df1$语文)]),"人")
x2=paste(length(df1$数学[df1$数学>mean(df1$数学)]),"人")
x3=paste(length(df1$英语[df1$英语>mean(df1$英语)]),"人")
x4=paste(length(df1$物理[df1$物理>mean(df1$物理)]),"人")
x5=paste(length(df1$化学[df1$化学>mean(df1$化学)]),"人")
x6=paste(length(df1$生物[df1$生物>mean(df1$生物)]),"人")
```

（4）使用 barplot()函数绘制柱形图，代码如下：

```
# 绘制柱形图
p=barplot(mean1,                          # 柱子高度
          ylim=c(0,150),                  # y 轴范围
          ylab = "平均分")                # y 轴标签
# 添加文本标签
text(p,mean1+5,labels=c(x1,x2,x3,x4,x5,x6))
```

运行程序，结果如图 1.31 所示。

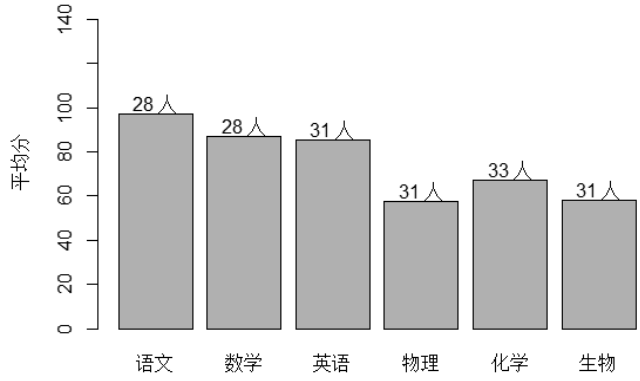


图 1.31　柱形图分析各科中上等学生人数

从运行结果得知：通过查看前面数据得知有 58 名学生，那么基本上各科中上等学生均占到了一半。

1.7.6 语数英成绩等级状况分析

根据某高中班级考试情况，语数英考试成绩按位次由高到低分为 A、B、C、D、E 五个等级。各等级人数所占比例依次为：A 等级 15%，B 等级 30%，C 等级 30%，D、E 等级共 25%，E 等级为不合格。首先通过 quantile()函数获取各个等级的分数，然后使用 cut()函数按分数段分割数据并标记等级，最后按等级统计人数并绘制饼形图。实现过程如下（源码位置：资源包\Code\01\level_data.R）。

（1）在项目文件夹下新建一个 R 脚本文件，命名为 level_data.R。

（2）使用 openxlsx 包的 read.xlsx()函数读取 Excel 文件，代码如下：

```
# 加载程序包
library(openxlsx)
library(dplyr)
# 读取 Excel 文件
df <- read.xlsx("学生成绩统计分析/成绩表 1.xlsx",sheet=1)
```

（3）通过 quantile()函数获取各个等级的分数，代码如下：

```
a <- quantile(df$语文,probs = c(0.15, 0.3,0.75,0.9,1))
a
b <- quantile(df$数学,probs = c(0.15, 0.3,0.75,0.9,1))
b
c <- quantile(df$英语,probs = c(0.15, 0.3,0.75,0.9,1))
c
```

（4）使用 cut()函数按分数段分割数据并标记等级，代码如下：

```
# 按分数段分割数据并标记等级
df$语文等级 <- cut(df$语文,breaks=c(-Inf, 90, 95,101,106,Inf),
                   labels = c("E","D","C","B","A"), right=FALSE)
df$数学等级 <- cut(df$数学,breaks=c(-Inf, 69,77,99, 109,Inf),
                   labels = c("E","D","C","B","A"), right=FALSE)
df$英语等级 <- cut(df$英语,breaks=c(-Inf, 69,79,98, 103, Inf),
                   labels = c("E","D","C","B","A"), right=FALSE)
```

（5）使用 summarise()函数结合 group_by()函数实现按等级统计人数，代码如下：

```
# 统计各等级人数
df1<- summarise(group_by(df,语文等级),人数=length(语文等级))
df2<- summarise(group_by(df,数学等级),人数=length(数学等级))
df3<- summarise(group_by(df,英语等级),人数=length(英语等级))
```

（6）获取人数、设置饼形图颜色和计算百分比，然后创建 1 行 3 列的绘图区域，最后绘制饼形图，代码如下：

```
# 获取人数
x1 = df1$人数
x2 = df2$人数
x3 = df3$人数
# 饼形图颜色
mycolors1 <- topo.colors(5)
# 计算百分比
pct1 <- paste(round(100*x1/sum(x1), 1), "%")
pct2 <- paste(round(100*x2/sum(x2), 1), "%")
pct3 <- paste(round(100*x3/sum(x3), 1), "%")
# 创建 1 行 3 列的绘图区域
par(mfrow = c(1,3))
# 绘制饼形图
pie(x1,labels = paste(df1$语文等级,pct1),col=mycolors1)
```

```
pie(x2,labels = paste(df2$数学等级,pct2),col=mycolors1)
pie(x3,labels = paste(df3$英语等级,pct3),col=mycolors1)
```

运行程序，结果如图 1.32 所示。

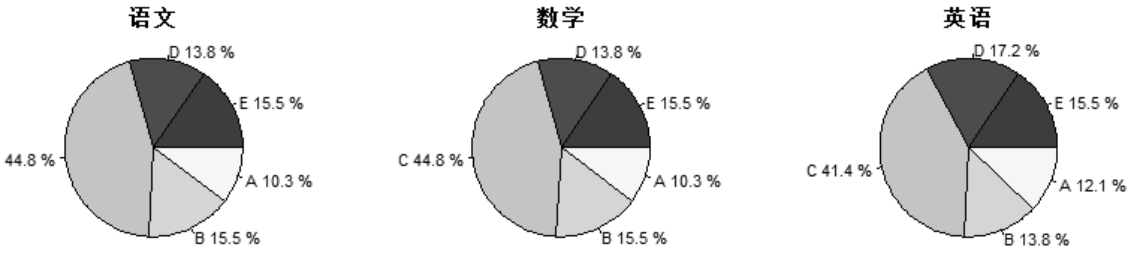


图 1.32　饼形图分析语数英成绩等级状况

1.7.7　成绩波动情况分析

成绩波动情况分析，主要通过学生的 8 次测试成绩排名来分析学生成绩的波动幅度，从而得到成绩从稳定到不稳定的学生名单。首先使用 apply()函数计算 8 次测试成绩的标准差，然后使用 sort()函数按标准差升序排序，实现过程如下（源码位置：资源包\Code\01\fluctuate_data.R）：

（1）在项目文件夹下新建一个 R 脚本文件，命名为 fluctuate_data.R。

（2）使用 openxlsx 包的 read.xlsx()函数读取 Excel 文件，代码如下：

```
# 加载 openxlsx 包
library(openxlsx)
# 读取 Excel 文件
df <- read.xlsx("学生成绩统计分析/测试成绩.xlsx",sheet=1)
```

（3）通过计算 8 次测试成绩的标准差得到波动幅度，然后使用 sort()函数按照标准差升序排序，代码如下：

```
# 波动幅度
df['标准差'] <- apply(df[,4:10],1,sd)
# 按照标准差升序排序
View(df[order(df[,"标准差"]),])
```

运行程序，结果如图 1.33 所示。

	名次	学号	姓名	test1	test2	test3	test4	test5	test6	test7	test8	标准差
9	9	2023010156	学生56	113	81	84	94	74	91	87	60	12.40200
1	1	2023010126	学生26	64	31	47	64	62	44	55	67	12.42118
22	22	2023010141	学生41	110	153	104	121	134	134	119	108	16.65333
12	12	2023010110	学生10	93	102	72	124	77	105	109	74	18.26524
10	10	2023010150	学生50	54	90	100	108	97	111	89	93	18.97116
29	29	2023010129	学生29	123	145	160	161	155	121	171	267	19.38212
56	56	2023010120	学生20	316	310	331	324	376	332	324	343	21.55502
34	34	2023010107	学生7	208	191	152	160	189	211	199	189	22.85774
6	6	2023010121	学生21	50	91	85	54	71	63	116	116	23.30747
28	28	2023010151	学生51	141	137	188	195	186	147	161	142	24.39262
58	58	2023010101	学生1	537	483	502	473	520	506	538	491	25.07892
4	4	2023010154	学生54	77	32	90	104	51	67	95	84	25.64965

图 1.33　按照标准差升序排序

从运行结果得知：标准差越小发挥越稳定，例如排名第 1 和第 9 的学生发挥最为稳定。

1.7.8 个人成绩排名分析

通过 1.7.7 节我们得到了学生成绩的波动情况和名单，接下来看一下排名第 1 的学生 8 次测试成绩排名和升降情况。首先使用 subset()函数检索“名次”等于 1 的数据，然后创建 2 行 1 列的绘图区域，使用 plot()函数分别绘制测试成绩排名折线图和排名升降折线图，其中排名升降折线图应使用 diff()函数计算差分，也就是计算本次排名与上一次排名的差分。实现过程如下（源码位置：资源包\Code\01\TOP1_data.R）。

（1）在项目文件夹下新建一个 R 脚本文件，命名为 TOP1_data.R。

（2）使用 openxlsx 包的 read.xlsx()函数读取 Excel 文件，代码如下：

```
# 加载 openxlsx 包
library(openxlsx)
# 读取 Excel 文件
df <- read.xlsx("学生成绩统计分析/测试成绩.xlsx",sheet=1)
```

（3）使用 subset()函数筛选名次等于 1 的数据并使用 t()函数实现行列转置，代码如下：

```
# 筛选名次等于 1 的数据
df <- subset(df,名次==1)
# 行列转置
df1 <- t(df[,4:11])
```

（4）绘制测试成绩排名折线图，代码如下：

```
# x 轴和 y 轴数据
x1 <- names(df1[,1])
y1 <- df1
# 创建 2 行 1 列的绘图区域
par(mfrow = c(2,1))
# 绘制测试成绩排名折线图
plot(y1,type="l",lwd=2,col="blue",xlab="",ylab="测试成绩排名",xaxt="n",ylim = c(0,100))
# 添加标记
points(y1,pch=21,bg=2)
# 设置 x 轴标签
axis(side=1,at=1:8,labels=x1)
```

（5）绘制排名升降折线图，代码如下：

```
# 使用 diff()函数计算差分
y2=diff(df1)
#x 轴数据
x2 <- names(df1[-1,1])
# 绘制排名升降折线图
plot(y2,type="l",lwd=2,col="red",xlab="",xaxt="n",ylab="排名升降情况")
# 添加标记
points(y2,pch=21,bg=2)
# 设置 x 轴标签
axis(side=1,at=1:7,labels=x2)
```

运行程序，结果如图 1.34 所示。

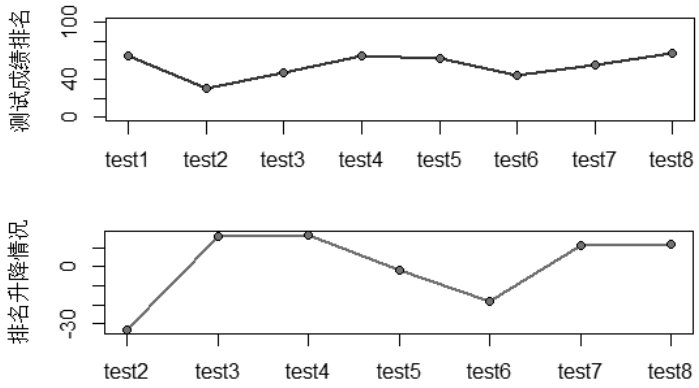


图 1.34　折线图分析 TOP1 学生测试成绩和排名升降情况

1.8　项 目 运 行

通过前述步骤，设计并完成了“学生成绩统计分析”项目的开发。“学生成绩统计分析”项目文件夹中包括 11 个 R 脚本文件和两个 Excel 文件，如图 1.35 所示。

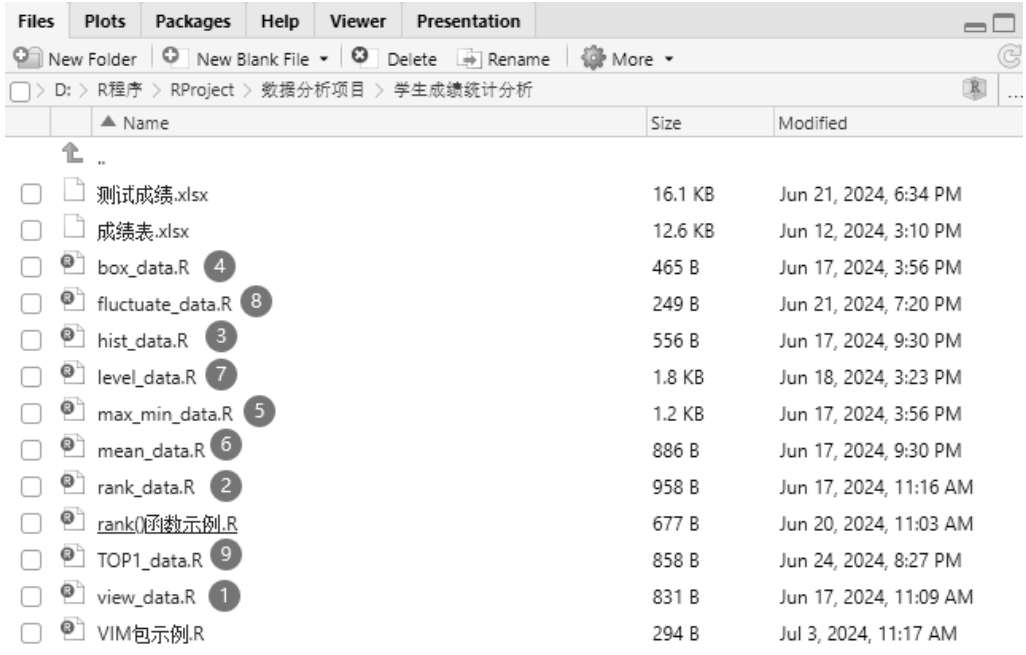


图 1.35　项目文件夹

下面按照开发过程运行脚本文件，检验一下我们的开发成果。例如，运行 view_data.R，首先单击 Files 面板，然后在列表中单击 view_data.R，在代码编辑窗口中单击 Run 按钮，运行光标所在行，如图 1.36 所示，或者单击 Source 按钮，运行所有行。

其他脚本文件按照图 1.35 给出的顺序运行，这里就不再赘述了。

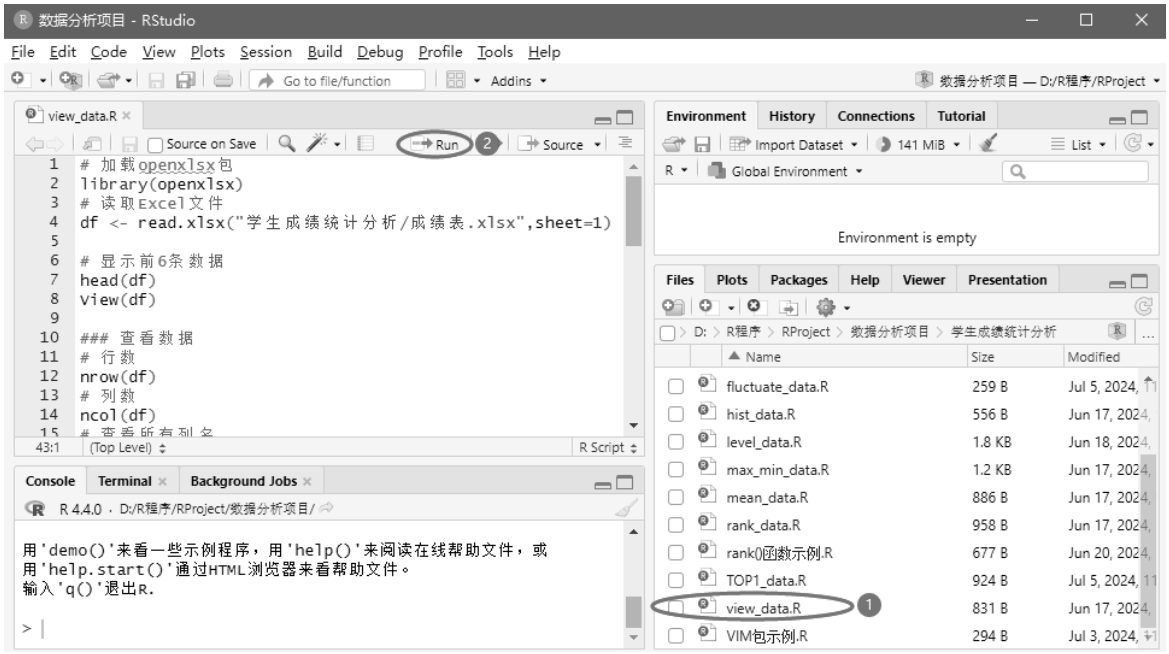


图 1.36　运行 view_data.R

1.9　源 码 下 载

虽然本章详细地讲解了如何通过 openxlsx 包、数据计算、分组统计和基本绘图实现“学生成绩统计分析”项目的各个功能，但给出的代码都是代码片段，而非源码。为了方便读者学习，本书提供了用以下载源码的二维码，扫描右侧二维码即可下载。

源码下载