R拥有一般现代编程语言中都有的标准控制结构。首先你将看到用于条件执行的结构，接下来是用于循环执行的结构。

为了理解贯穿本节的语法示例，请牢记以下概念：

- 语句（statement）是一条单独的R语句或一组复合语句（包含在花括号{ }中的一组R语句，使用分号分隔）；
- 条件（cond）是一条最终被解析为真（TRUE）或假（FALSE）的表达式；
- 表达式（expr）是一条数值或字符串的求值语句；
- 序列（seq）是一个数值或字符串序列。

在讨论过控制流的构造后，我们将学习如何编写函数。

5.4.1 重复和循环

循环结构重复地执行一个或一系列语句，直到某个条件不为真为止。循环结构包括for和while结构。

1. for结构

for循环重复地执行一个语句，直到某个变量的值不再包含在序列seq中为止。语法为：

```
for (var in seq) statement
```

在下例中：

```
for (i in 1:10)  print("Hello")
```

单词Hello被输出了10次。

2. while结构

while循环重复地执行一个语句，直到条件不为真为止。语法为：

```
while (cond) statement
```

作为第二个例子，代码：

```
i <- 10
while (i > 0) {print("Hello"); i <- i - 1}
```

又将单词Hello输出了10次。请确保括号内while的条件语句能够改变，即让它在某个时刻不再为真——否则循环将永不停止！在上例中，语句：

```
i <- i - 1
```

在每步循环中为对象i减去1，这样在十次循环过后，它就不再大于0了。反之，如果在每步循环都加1的话，R将不停地输出Hello。这也是while循环可能较其他循环结构更危险的原因。

在处理大数据集中的行和列时，R中的循环可能比较低效费时。只要可能，最好联用R中的内建数值/字符处理函数和apply族函数。

5.4.2 条件执行

在条件执行结构中，一条或一组语句仅在满足一个指定条件时执行。条件执行结构包括if-else、ifelse和switch。

1. if-else结构

控制结构if-else在某个给定条件为真时执行语句。也可以同时在条件为假时执行另外的语句。语法为：

```
if (cond) statement
if (cond) statement1 else statement2
```

示例如下：

```
if (is.character(grade)) grade <- as.factor(grade)
if (!is.factor(grade)) grade <- as.factor(grade) else print("Grade already
    is a factor")
```

在第一个实例中，如果grade是一个字符向量，它就会被转换为一个因子。在第二个实例中，两个语句择其一执行。如果grade不是一个因子（注意符号!），它就会被转换为一个因子。如果它是一个因子，就会输出一段信息。

2. ifelse结构

ifelse结构是if-else结构比较紧凑的向量化版本，其语法为：

```
ifelse(cond, statement1, statement2)
```

若*cond*为TRUE，则执行第一个语句；若*cond*为FALSE，则执行第二个语句。示例如下：

```
ifelse(score > 0.5, print("Passed"), print("Failed"))
outcome <- ifelse (score > 0.5, "Passed", "Failed")
```

在程序的行为是二元时，或者希望结构的输入和输出均为向量时，请使用ifelse。

3. switch结构

switch根据一个表达式的值选择语句执行。语法为：

```
switch(expr, ...)
```

其中的...表示与*expr*的各种可能输出值绑定的语句。通过观察代码清单5-7中的代码，可以轻松地理解switch的工作原理。

代码清单5-7　一个switch示例

```
> feelings <- c("sad", "afraid")
> for (i in feelings)
    print(
      switch(i,
        happy  = "I am glad you are happy",
        afraid = "There is nothing to fear",
        sad    = "Cheer up",
        angry  = "Calm down now"
      )
    )

[1] "Cheer up"
[1] "There is nothing to fear"
```

虽然这个例子比较幼稚，但它展示了switch的主要功能。你将在下一节学习如何使用switch编写自己的函数。